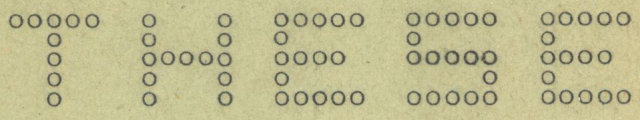
UNIVERSITE DE NANCY FACULTE DES SCIENCES

ETUDE
DE METHODES DE TRI

.°.



pour l'obtention du

DOCTORAT de SPECIALITE MATHEMATIQUES (3ème CYCLE)

Soutenue devant le Jury le 21 décembre 1964

par

Raymond ROMAC

.°.

Jury : Mr J. LEGRAS Président
Mr C. PAIR Examinateurs
Mr M. DEPAIX

Tri, méthode

UNIVERSITE DE NANCY — FACULTE DES SCIENCES

ETUDE DE METHODES DE TRI

par

Raymond ROMAC

UNIVERSITE DE NANCY - FACULTE DES SCIENCES

Doyen : M. AUBRY

Assesseur : M. GAY

Doyens honoraires : MM. CORNUBERT - DELSARTE - URION - ROUBAULT -

Professeurs honoraires : MM. CROZE - RAYBAUD - LAFFITE - LERAY - DLY- LAPORTE - EICHHORN - CAPELLE - GODEMENT - DUBREIL - L. SCHWARTZ EUDONNE - De MALLEMANN - LONGCHAMBON - LETORT - DODE - GAUTHIER - OUDET - OLMER - CORNUBERT - CHAPELLE - GUERIN - CHEVALLIER - WAHL - ERVE -

Maîtres de conférences honoraires : MM. LIENHART - PIERRET -

PROFESSEURS

M.			
RION	Chimie biologique	SCHWARTZ	Exploit. minière
LSARTE	Analyse supérieure	GAYET	Physiologie
UBAULT	Géologie	MANGENOT	Phytopathologie
ILLET	Biologie animale	MALAPRADE	Chimie
HEVIN	Botanique	HADNI	Physique
RRIOL	Chimie théorique	BONVALET	Mécanique physique
ZETTE	Physique	KERN	Minéralogie
ILLIEN	Electronique	BASTICK	Chimie
BERT	Chimie physique	DUCHAUFOUR	Pédologie
GRAS	Mécanique rationnelle	NEEL	Chimie ind. orga.
LFA	Minéralogie	GARNIER	Agronomie
CLAUSE	Chimie	WEPPE	Minéralogie appli.
IVRE	Physique appliquée	BERNARD	Géologie appliquée
BRY	Chimie minérale	CHAMPIER	Physique
VAL	Chimie	REGNIER	Physico-chimie
PPENS	Radiogéologie	GAY	Chimie biologique
UHLING	Physique	WERNER	Botanique
HNER	Physique expérimentale	CONDE	Zoologie
LLY	Géologie	STEPHAN	Zoologie
GOFF	Génie chimique	EYMARD	Cal. Dif. et Int.
IAPON	Chimie biologique	LEVISALLES	Chimie organique
ROLD	Chimie industrielle	N.	Physique
		N.	M. M. P.

MAITRES DE CONFERENCES ET PROFESSEURS SANS CHAIRE

M.			
GSE	Génie chimique	Mme HERVE	Math. (propédeutique)
CCI	Géologie	AUROUZE	Géologie
ILLAUME	Psychophysiologie	MARI	Chimie I. S. I. N.
AN	Mécanique physique	LAFON	Physique I. S. I. N.
e BASTICK	Chimie M. P. C. Epinal	FELDEN	Phy. Théor. & Nucl.
DEFIN	Physique	FLECHON	M. P.C.
RN	Physique	VIGNES	Physique (Mines)
ENTZ	Biologie animale	Melle HUET	Math. (S. P. C. N.)
		JANOT	M. P. C. Epinal

SECRETAIRE PRINCIPAL : C. CARON

Je tiens tout d'abord à exprimer toute ma reconnaissance à Monsieur le Professeur J. LEGRAS, pour l'attention bienveillante qu'il m'a portée au cours de ces années d'études passées au Centre de Calcul Automatique de l'Université de Nancy.

Je dois beaucoup à Monsieur C. PAIR, qui a dirigé mes recherches, et qui n'a pas cessé de me manifester un intérêt constant. Je lui adresse donc, ici, mes plus vifs remerciements.

Je remercie également Monsieur M. DEPAIX, qui m'honore en faisant partie du Jury.

SOMMAIRE

CHAPITRE I

GENERALITES

I - POSITION du PROBLEME

Considérons une suite SI formée de n termes a_1, a_2,... a_n, chaque a_i étant repéré par un indicatif $I(a_i)$, c'est à dire, en général, un nombre.

Trier(ou ordonner) cette suite, consiste à la réarranger de façon à obtenir une suite SR, que j'appellerai "suite résultat" (ou encore "suite finale") elle-même composée de n termes r_1, r_2,... r_n. Cette suite est une permutation de SI, et, si nous avons choisi pour ordre de tri la relation d'ordre $\leqslant$ elle est telle que :

$$I(r_j) \leqslant I(r_{j+1}) \quad \text{pour } j = 1, 2, \ldots, n-1.$$

Chaque r_j provient d'un a_i et d'un seul. Nous appellerons $p(a_i)$ le rang d'un élément a_i dans la suite résultat SR.

La quantité $f_i = i - p(a_i)$ représente donc l'écart d'un élément a_i à sa position définitive dans la suite SR.

II - TERMINOLOGIE

Afin d'alléger la notation dans la suite de cette étude, nous remplacerons l'écriture $I(a_i)$ qui est l'indicatif ("key" en anglais") associé à l'élément a_i par l'élément a_i lui-même.

J'appellerai SI la suite initiale des éléments à trier, suite qui sera composée des éléments a_i.

Un passage dans SI sera une exploration de tous les éléments de SI (ou d'une certaine partie de SI ce que je préciserai alors le cas échéant) afin d'extraire un (ou plusieurs) éléments de SI et de le (les) transférer à une place déterminée (en général, ce sera sa place définitive dans la suite finale SR). Il est clair qu'à l'issue d'un passage, SI va se trouver modifié. Le terme "étape" sera utilisé comme synonyme du mot "passage".

Au début du k° passage, je note s la suite issue de SI et formée des éléments e_i, chaque e_i provenant d'un seul a_j. A la fin de ce k° passage, s se transforme en une suite que je note s', suite formée des éléments e'_j chaque e'_j provenant d'un seul e_i.

Pour qu'un élément e'_j de s' soit "bien rangé", il faut que $j-p(e'_j)=0$ en appelant $p(e'_j)$ le rang de e'_j dans la suite SR.

Comme e'_j provient de e_i $p(e'_j) = p(e_i)$.

L'élément e'_j de s' provenant par transposition de l'élément e_i de s sera dit "mieux rangé" (ou "mieux classé") que e_i si

$$|j-p(e_j)| < |i-p(e_i)|$$

c'est à dire que cet élément s'est rapproché de sa position définitive dans SR. (Dans le cas contraire, les éléments e'_j et e_i seront dits "plus mal classés").

D'autre part, deux éléments e_j et e_k de s seront dits "bien classés (ou bien rangés) l'un par rapport à l'autre" si

$e_j \leqslant e_k$ lorsque $j < k$

De la même façon, deux éléments e_j et e_k de s seront "mal classés" si

$e_j > e_k$ lorsque $j < k$.

Je n'ai envisagé dans cette étude que les méthode de tri interne (Internal sorting). On suppose dans ces méthodes que les n éléments a_i de SI se trouvent dans la mémoire centraledu calculateur et que le nombre de mémoires disponibles est suffisant pour que le tri s'effectue sans avoir recours aux organes de stockage annexes du calculateur tels que les bandes ou disques magnétiques.

III - CLASSIFICATION des METHODES

Une telle classification est délicate pour ne pas dire arbitraire. J'ai retenu la classification suivante :

a) les méthodes de transposition.

b) les méthodes de sélection.

c) les méthodes d'inter-classement.

d) les méthodes d'insertion

e) les méthodes utilisant des piles simples

IV - TEMPS de TRI

Lorsque les éléments a_i de SI sont chargésdans la mémoire centrale ainsi que le programme lui-même, le temps de tri est le temps au bout duquel la suite SI est totalement triée.

Le temps τ est fonction, d'une part du nombre total C(n) de tests à effectuer sur les n éléments a_i (et aussi sur les indices de ces éléments), et d'autre part du nombre total t(n) de transferts nécessaires pour trier ces nombres.

Nous avons $\tau(n) = a\,C(n) + b\,t(n)$.

a et b étant des constantes qui dépendent du calculateur.

Remarque : Nous appelons nombre théorique de tests d'une méthode le nombre de tests à effectuer sur les a_i pour trier SI à l'exclusion des tests sur les indices. Par contre, le nombre théorique de transferts est le nombre de transferts véritables à effectuer pour que tous les a_i soient triés.

Les essais de ces méthodes ont été effectués sur le calculateur IBM 650 du Centre de Calcul Automatique de NANCY.

CHAPITRE II

LES METHODES DE TRANSPOSITION

I - GENERALITES

Soit SI = $a_1, a_2, \ldots, a_n$ la suite initiale des n nombres à trie

A l'issue de k transformations sur SI nous obtenons une suite s formée des éléments $e_1, e_2, \ldots, e_n$.

Les méthodes de transposition consistent à choisir deux éléments déterminés de s, soient e_j et e_k, à les comparer, à les permuter si

$e_j > e_k$ lorsque $j < k$

et nous allons consulter deux autres éléments e_{jl} et e_{kl}.

Si $e_j \leqslant e_k$ nous les laissons en place et nous allons consulter deux autres éléments e_{jl} et e_{kl} de la suite s.

Les différentes méthodes se distinguent par le choix d'une part de e_j et e_k, et également par le choix des éléments suivants : e_{jl} et e_{kl}.

L'intérêt de ces méthodes réside dans le fait que le tri s'effectue en utilisant une seule mémoire auxiliaire de travail : celle qui sert à la transposition.

Citons parmi ces méthodes celle de Bubble, celle de Shell, celle de Nelson Bose, et une méthode issue de celle de Von Neumann.

II - METHODE de BUBBLE

1°) Généralités

La méthode de Bubble [1] est une des plus naturelles qui soit puisqu'elle consiste à extraire le plus grand élément de la suite s et à le transporter en queue de liste en utilisant des transpositions portant sur des éléments consécutifs.

2°) Exposé de la méthode

2. 1. Soient deux éléments a_i et a_{i+1} de SI au cours du premier passage. Si $a_i > a_{i+1}$ alors nous les permutons et nous allons consulter les éléments a'_{i+1} et a_{i+2} où a'_{i+1} est en fait l'ancien a_i.

Si $a_i \leqslant a_{i+1}$ les éléments sont inchangés et nous allons consulter les éléments a_{i+1} et a_{i+2}.

Soit δ le plus grand des éléments de SI.

Si δ n'est pas le dernier élément de SI, c'est à dire s'il n'est pas a_n, il le deviendra : en effet supposons que $\delta = a_k$.

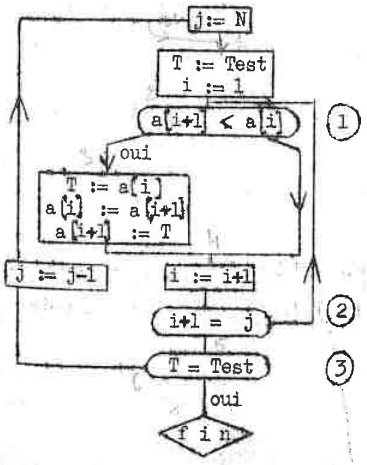
La comparaison de a_k et a_{k+1} va amener la permutation de a_{k+1} et a_k ce qui nous donnera $a'_k = a_{k+1}$ et $a'_{k+1} = \delta$. Au test suivant portant sur a'_{k+1} et a_{k+2}, δ et a_{k+2} vont à nouveau permuter etc... jusqu'à ce que δ occupe la place de a_n, a_n devenant a'_{n-1}.

D'autre part si δ est déjà le dernier élément de SI, il le restera. Nous voyons donc qu'à l'issue de ce premier passage un élément au moins sera classé.

De la même façon chaque nouveau passage dans la suite s classera au moins un élément de sorte qu'au bout de n passages au plus les n éléments de SI seront classés. Il convient de remarquer que puisqu'à chaque passage un élément au moins se trouve rangé définitivement nous pourrons ne plus en tenir compte de sorte qu'au premier passage nous aurons n-1 tests, au second passage n-2 tests, etc...

D'autre part si un passage dans s ne donne lieu à aucune transposition, il en résulte que $e_i \leqslant e_{i+1}$ quel que soit i et les nombres seront donc triés ce qui suggère le test de fin de tri.

2. 2. organigramme et exemple



nous avons posé :

i = indice de l'élément e_i dans la suite s

j = indice du dernier élément de la suite s (s est formé de $e_1, e_2, \ldots, e_j$)

le symbole := est le symbole "affecté à" du langage ALGOL.

Donnons un exemple : chaque ligne représente l'évolution du contenu d'une mémoir au cours du tri.

Notation : - les colonnes repérées par les numéros 1, 2, 3, etc... représentent l'évolution des mémoires au cours du premier passage, deuxième passage, troisième passage, etc...

- les colonnes notées 1B, 2B, 3B, etc... représentent l'état des mémo res à l'issue du premier passage, du deuxième passage, du troisièm passage, etc...

N° d'ordre des mémoires	Etat initial des mémoires.	1	1B	2	2B	3	3B	4	4B	5	5B	6	6B	7	7B	8	8B
1	7	4	4	4	4	0	0	0	0	0	0	0	0	0	0	0	0
2	4	7	7	0	0	4 2	2	2	2	2	2	2	2	2	2	2 1	1
3	9	0	0	7 2	2	4	4	4	4	4	4	4 3	3	3 1	1	2	2
4	0	9 2	2	7 6	6	6 5	5	5	5	5 3	3	4 1	1	3	3	3	3
5	2	9 6	6	7 5	5	6	6	6 3	3	5 1	1	4	4	4	4	4	4
6	6	9 5	5	7	7	7 3	3	6 1	1	5	5	5	5	5	5	5	5
7	5	9 8	8	8 3	3	7 1	1	6	6	6	6	6	6	6	6	6	6
8	8	9 3	3	8 1	1	7	7	7	7	7	7	7	7	7	7	7	7
9	3	9 1	1	8	8	8	8	8	8	8	8	8	8	8	8	8	8
10	1	9	9		9	9	9	9	9	9	9	9	9	9	9	9	9

Nous voyons donc sur cet exemple que ce n'est qu'à l'issue du neuvième passage que nous n'avons plus de transpositions, ce qui est considérable.

Soit $p(e_i)$ le rang de l'élément e_i de s dans la suite finale SR des éléments triés, c'est à dire pour fixer les idées

lorsque s = 1B $\quad e_9 = 1 \quad p(e) = 2 \quad f_9 = |9 - p(e_9)| = 8$

s' = 2B $\quad e'_9$ a donné e'_8

Le tri sera d'autant plus valable qu'il y aura un grand nombr de valeurs i pour lesquelles $|i - p(e_i)| < |j - p(e'_j)|$ (pour les notations, voir l'introduction).

Or comme e'_j provient de e_i par transposition, nous avons $p(e'_j) = p(e_i)$. Posons toujours $\phi = e_k$ le plus grand élément de s (dans l'exemple choisi $\phi = e_7 = 8$ car nous ne tenons plus compte de $e_{10} = 9$). Tous les éléments e_i tels que $k < i < j$ sont décalés d'une mémoire vers :

à la suite du passage dans s, seront "mieux classés" les éléments e_i de s pour lesquels

$$i - p(e_i) > 0 \qquad \text{avec } k < i \leq n.$$

Ce ne sont pas, d'ailleurs, les seuls.

Seront au contraire plus mal classés les éléments e_j tels que : $\quad j - p(e_j) < 0 \qquad \text{avec } k < j \leq n$

et ce ne sont pas non plus les seuls à être plus mal classés.

Nous pouvons donc prévoir que la méthode ne sera pas excellente car elle réalise des opérations qui sont contraires à l'ordre de tri.

2.3. Nombres de tests théoriques nécessaires pour trier n nombres :

a) Supposons tout d'abord que la suite initiale SI soit totalement rangée. Un seul passage dans s est nécessaire et le nombre de tests est donc : $\quad N_a = n - 1$

b) Envisageons à présent la suite SI telle que

$$a_i \leq a_{i+1} \qquad \text{pour } i = 1, 2, \ldots, n-1$$

et a_n soit le plus petit des a_i.

Soit s la suite formée des éléments e_i à l'issue du k° passage : nous avons i = 1,2,... , n-k. Seront rangés les éléments $a_{n-1}, a_{n-2}, \ldots, a_{n-k}$. Par contre, $e_{n-k} = a_n$; a_n ne s'est rapproché de son rang définitif que de k positions. Nous voyons donc qu'il nous faudra n passages dans s pour que a_n soit définitivement rangé ! Le nombre de tests est alors :

$$N_b = \sum_{i=1}^{n-1} i+1 = \frac{n(n-1)}{2} + 1 = \frac{n^2-n+2}{2}$$

Au nombre de transferts près, ce cas est identique à celui d'une suite SI totalement rangée dans l'ordre inverse de tri, c'est à dire telle que

$$a_i \geq a_{i+1} \qquad \text{avec } i = 1, 2, \ldots n.$$

c) Soit une suite initiale SI qui n'est pas totalement rangée il existe alors des indices i tels que $i-p(a_i)$ soit positif. Soit l l'indice qui réalise le maximum de $i-p(a_i)$; posons $m = l - p(a_l)$.

Appelons e_k l'élément provenant de a_l, à l'issue du k° passage.

Si $e_{k-1} \leq e_k$ e_k n'est pas déplacé

$e_{k-1} > e_k$ e_k va se trouver décalé d'une position seulement vers la gauche.

Nous voyons donc que m décroit au plus d'une unité à chaque passage de sorte qu'il faudra au moins m passages pour mettre en place a_l dans SR (c'est ce qui se passait en b, où il fallait n passages pour que a_n soit placé à sa position définitive dans SR).

Le nombre de tests N_c sera tel que :

$$N_c \geq (n-l) + (n-2) + \ldots + (n-m) = n \times m - \frac{m(m+1)}{2}$$

$$= \frac{m(2n-m-1)}{2}$$

Pour étudier cette quantité, évaluons la valeur moyenne de m :

nous avons $i - p(a_i) \leq \max_i (i - p(a_i))$

d'où $E(i-p(a_i)) \leq E(\max_i (i - p(a_i)))$

$\max_i E(i-p(a_i)) \leq E(\max_i (i - p(a_i)))$

lorsque i est fixé nous avons $E(i - p(a_i)) = i - E(p(a_i))$

mais $E(p(a_i)) = \frac{1}{n} \sum_{i=1}^{n} i = \frac{n+1}{2}$

d'où $E(i-p(a_i)) = i - \frac{n+1}{2}$

Cette quantité est maximum pour $i = n$ et est égale à $\frac{n-1}{2}$

d'où $m \geq \frac{n-1}{2}$

Remarque : Afin de remédier à un inconvénient analogue à celui signalé au b, j'ai envisagé une méthode dans laquelle la suite s est explorée dans le sens $e_1, e_2, \ldots, e_j$ et la suite s' dans le sens $e'_{j-1}, e'_{j-2}, \ldots, e'_1$.

Au cours des passages impairs, ce sont donc les plus grands éléments qui sont extraits et, en inversant les bornes du test $e_{j+1} < e_j$, au cours des passages pairs les plus petits le sont à leur tour : les résultats expérimentaux ont été alors améliorés dans le rapport $\frac{3}{4}$ par rapport à la première méthode.

3°) Programme ALGOL

```
PROCEDURE Bubble (A,N) ; VALEUR N; ENTIER N; TABLEAU A ;
    DEBUT ENTIER I, J ; REEL T, Test ;
        POUR J := N PAS - 1 JUSQUA 0 FAIRE
            DEBUT T := Test ;
                POUR I := 1 PAS 1 JUSQUA J - 1 FAIRE
                SI A[I+1] < A[I] ALORS
                DEBUT T := A[I]; A[I] := A[I+1] ; A[I+1] := T FIN;
                SI T = Test ALORS ALLER A Sortie
            FIN ;
Sortie : FIN de la procédure Bubble ;
```

4°) Programme pour IBM 650 écrit en PASØ

cf. au programme n° 1

Ce programme occupe 53 mémoires.

5°) Résultats expérimentaux

Le nombre de tests réels N_r est différent du nombre de tests théoriques N_c (test n° 1 sur l'organigramme) car il comprend en plus pour chaque test (1) un test (2) portant sur l'indice i mais également pour chaque passage un test (3) test de fin. (nous verrons dans d'autres méthodes l'importance que peuvent avoir les tests sur les indices).

Nous avons alors $N_r \simeq 2 N_c + m$.

Les essais sur calculateur ont été faits avec deux types de suite SI afin de déterminer l'influence de la distribution sur le nombre de tests :

SI 1 est une suite de nombres répartis au hasard

SI 2 est une suite de nombres pratiquement triés.

L'équation du temps de passage est de la forme :

$T = k_1 N_r + k_2 M_r + c$

en appelant M_r le nombre de transferts. Elle est donc en n^2.

Nous avons obtenu les résultats suivants :

Temps de passage	n = 128	n = 256	n = 512
avec SI 1	7'20"	29'	115'
avec SI 2		28"	2'30"

III - LA METHODE de SHELL

1°) Généralités

La méthode de Bubble est une méthode très simple et très naturelle, mais, malheureusement, son efficacité est médiocre, et nous sommes amenés à considérer des méthodes beaucoup plus élaborées et beaucoup plus complexes. On s'est aperçu qu'il était toujours très intéressant de diviser la suite initiale en plusieurs sous-suites et de les traiter d'abord séparément. La méthode de Shell [2] [3] utilise cette constatation et prend à son compte les avantages de la méthode de Von Neumann sur laquelle nous reviendrons à propos des méthodes d'interclassement.

Shell l'a mise au point en 1959 et les performances obtenues sont très intéressantes dans le cas d'une suite SI quelconque.

2°) Exposé de la méthode

Pour rendre plus clair cet exposé, supposons que vous avez $n = 2^p$ nombres à trier (la méthode elle-même n'est pas aussi restrictive).

a) Soit donc SI la suite initiale des nombres à trier. Divisons la en $\frac{n}{2} = 2^{p-1} = m$ sous-suites formées de 2 éléments seulement qui ne sont pas consécutifs. Appelons f_j^1 ces sous-suites avec $1 \leqslant j \leqslant \frac{n}{2}$.

f_1^1 est formé des éléments a_1 et $a_{\frac{n}{2}+1}$

f_2^1 est formé des éléments a_2 et $a_{\frac{n}{2}+2}$

f_j^1 est formé des éléments a_j et $a_{\frac{n}{2}+j}$

⋮

$f_{\frac{n}{2}}^1$ est formé des éléments $a_{\frac{n}{2}}$ et a_n

A l'issue du premier passage dans la suite SI nou obtenons donc une suite s_1 formée de $\frac{n}{2}$ sous-suites f'^1_j ordonnées comprenant 2 éléments. Appelons s^1_j les n éléments de s_1 ils sont tels que

$$s^1_k \leqslant s^1_{k+\frac{n}{2}} \qquad \text{avec } 1 \leqslant k \leqslant \frac{n}{2}$$

b) La suite s_1 est à présent divisée en $\frac{n}{4}$ sous-suites formée de 4 éléments et que nous appellerons :

f_1^2 formée des éléments s^1_1 , $s^1_{\frac{n}{4}+1}$, $s^1_{\frac{2n}{4}+1}$, $s^1_{\frac{3n}{4}+1}$

f_2^2 formée des éléments s^1_2 , $s^1_{\frac{n}{4}+2}$, $s^1_{\frac{2n}{4}+2}$, $s^1_{\frac{3n}{4}+2}$

⋮

$f^2_{\frac{n}{4}}$ formée des éléments $s^1_{\frac{n}{4}}$, $s^1_{\frac{2n}{4}}$, $s^1_{\frac{3n}{4}}$, s^1_n

Chaque sous-suite f_j^2 est triée par transposition. Shell avait le choix entre : d'une part, trier f_1^2 puis, f_1^2 étant totalement rangée, passer au tri de f_2^2 , puis à celui de f_3^2 etc... , d'autre part, imbriquer les tris des f_j^2 les uns dans les autres.

Afin de simplifier les tests sur les indices, il a préféré adopter la deuxième solution : trier les deux premiers éléments de f_1^2 , puis les deux premiers éléments de f_2^2, etc. et enfin les 2 1^{ers} éléments de $f^2_{\frac{n}{4}}$, ensuite trier les 3 1^{ers} éléments de f_1^2, puis les 3 1^{ers} éléments de f_2^2, et enfin les 3 1^{ers} éléments de $f^2_{\frac{n}{4}}$ etc.

A l'issue de ce second passage, la suite s_1 est transposée en une suite s_2 formée de $\frac{n}{4}$ sous-suites f'^2_j comprenant 4 éléments qui sont rangés entre eux.

c) La suite s_2 est à présent divisée en $\frac{n}{8}$ sous-suites f^3_j non ordonnées formées de 8 éléments et qui par transposition donneront $\frac{n}{8}$ sous suites ordonnées f'^3_j avec $1 \leqslant j \leqslant \frac{n}{8}$.

Nous voyons donc qu'à l'issue du p° passage nous obtenons une suite s_p formée d'une seule sous-suite f'^p et dans laquelle les éléments s^p_j sont tels que $s^p_j \leqslant s^p_{j+1}$ avec $1 \leqslant j \leqslant n-1$.

Remarques :

Nous avions supposé pour simplifier l'exposé que $n = 2^p$. En fait, la méthode est valable quel que soit la valeur de n. En effet, soit au k° passage m = partie entière de $\left(\frac{n}{2^k}\right)$. Nous formerons alors la sous-suite f_1^k avec les éléments s_i^k tels que :

f_1^k sera formé de s_1^k ; s_{m+1}^k ; s_{2m+1}^k ; ..... ; s_{pxm+1}^k avec $pxm+1 \leq$

f_2^k sera formé de s_2^k ; s_{m+2}^k ; s_{2m+2}^k ; ..... ; s_{qxm+2}^k avec $qxm+2 \leq$

etc...

Le nombre d'éléments dans chaque sous-suite est supérieur ou égal à 2^k. Le tri sera terminé lorsque nous aurons une seule sous-suite, c'est à dire lorsque m = 1 (ou encore partie entière de $\frac{m}{2} = 0$).

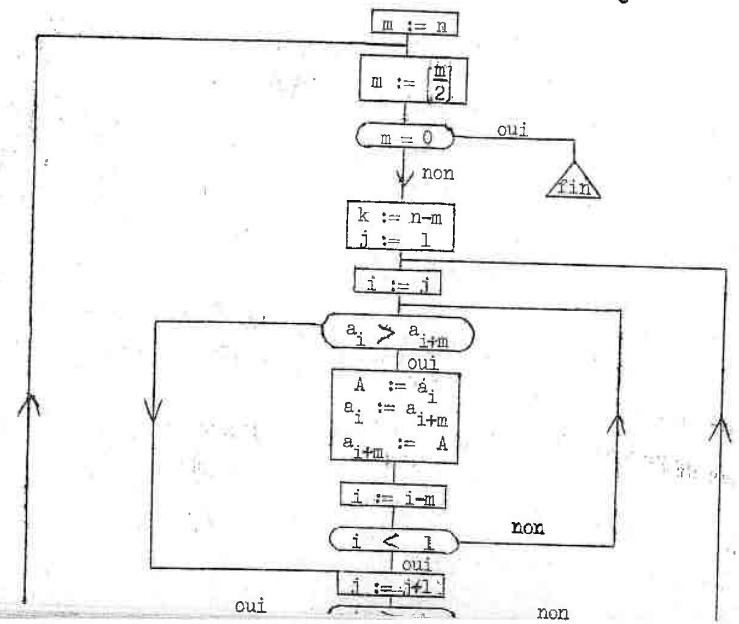
3°) Organigramme :

L'organigramme ci-dessous permet de bien comprendre le processus.

Les éléments à trier a_i sont rangés dans n mémoires consécutives.

i = indice de l'élément s_i^k dans la sous-suite f_ℓ^k

j = indice qui appartient à f_ℓ^k puis à $f_{\ell+1}^k$ etc... et qui réalise la simultanéité du tri des f_ℓ^k.



Donnons également un exemple qui fixera les idées, soit, par exemple, n = 11.

n° des mémoires	SI		s_1			s_2		v	
1	0		0			0		0	
2	10	1	1			1		1	
3	9	5	5	2		2		2	
4	6		6	3		3		3	
5	8	2	2	5	4	4		4	
6	3		3	6		6	5	5	
7	1	10	10	7	4 5	5	6	6	
8	5	9	9	8		8	7	7	
9	7		7	10	4 7	7	8	8	
10	2	8	8	9		9		9	
11	4		4	10		10		10	

Nous avons dans cet exemple $\left[\frac{m}{2}\right] = 5$ au premier tour

f_1^1 formée de $a_1 = 0$, $a_6 = 3$, $a_{11} = 4$

f_2^1 formée de $a_2 = 10$, $a_7 = 1$.

Les flèches ↗ symbolisent la permutation des éléments a_j et $a_{\left[\frac{m}{2}\right]+j}$ lorsque celle-ci s'avère nécessaire; par exemple dans le cas présent a_7 permute avec a_2.

A l'issue du premier passage, nous obtenons les f'^1_j qui sont

f'^1_1 formé de $s_1^1 = 0$; $s_6^1 = 3$, $s_{11}^1 = 4$;

f'^1_2 formé de $s_2^1 = 1$; $s_7^1 = 10$;

Dans la suite s_1 nous obtenons les sous-suites f_j^2 avec $\left[\frac{m}{2}\right] = 2$

f_1^2 formée de $s_1^1 = 0$; $s_3^1 = 5$; $s_5^1 = 2$; $s_7^1 = 10$; $s_9^1 = 7$; $s_{11}^1 = 4$.

A l'issue du second passage s_1 est transposée en s_2, les f^2_j donnant des sous-suites f'^2_j ordonnées :

par exemple : f'^2_1 est formée de $s^2_1 = 0$; $s^2_3 = 2$; $s^2_5 = 4$; $s^2_7 = 5$; $s^2_9 = 7$; $s^2_{11} = 10$

et f'^2_2 est formé de $s^2_2 = 1$; $s^2_4 = 3$; $s^2_6 = 6$; $s^2_8 = 8$; $s^2_{10} =$

Le troisième passage, $\left[\frac{m}{2}\right] = 1$, nous donne une seule suite, où tous les nombres sont triés, c'est ce que nous appelons la suite résultat R.

4°) Programme ALGOL

```
PROCEDURE Shell (A,N) ; VALEUR N ; TABLEAU A ; ENTIER N ;
    DEBUT ENTIER M,I,J,K ; REEL A 1 ;
    M := N ;
    B1 : M := Entière (M/2) ; SI M=0 ALORS ALLER A Sortie SINON K:=
    POUR J := 1 PAS 1 TANT QUE J ≤ K FAIRE
    POUR I := J PAS -M TANT QUE I ≥ 1 ∧ A[I] > A[I+M] FAIRE
        DEBUT A1= A[I] ; A[I] := A[I+M] ; A[I+M] := A1
        FIN ;
    ALLER A B1 ;
Sortie : FIN de la procédure Shell ;
```

5°) Programme IBM 650 écrit en PAS Ø

cf. programme n° 2

Il n'occupe que 66 mémoires et il est très simple à mettre en œuvre.

6°) Résultats expérimentaux

Shell a donné comme résultats expérimentaux un temps de tri proportionnel à $n^{1,226}$ et Hibbard [6] la relation $n\left[A(\log_2 n)^2 + B\log_2 n + \right.$

En ce qui nous concerne, nous avons obtenu :

Temps de passage	128 Nb	256 Nb	512 Nb
avec S 1	T_1 = 1'30"	T_2 = 4'30"	T_3 = 14'45"
avec S 2		T'_2 = 1'43"	T_3 = 3'42"

En faisant l'hypothèse que le temps de tri T est proportionnel à n^x, on trouve $x \simeq 1,6$.

IV - METHODE ISSUE de la METHODE de VON NEUMANN

1°) Généralités

Partant de la même idée que Shell, j'ai essayé dans cette méthode d'associer les qualités de la méthode de Von Neumann que je vais exposer ci-dessous à celles des méthodes de transposition. En effet, le gros avantage de la méthode de Von Neumann est sa rapidité relative, mais par contre, elle utilise, pour trier n nombres, n mémoires auxiliaires, les méthodes de transposition n'en utilisant qu'une seule.

2°) Exposé du principe de la méthode de Von Neumann

2. 1. Nous ne ferons qu'un exposé succinct de cette méthode de Von Neumann, car nous y reviendrons longuement au chapitre V.

Supposons toujours que le nombre d'éléments à trier soit

$$n = 2^p$$

La méthode de Von Neumann diffère de la méthode de Shell précédemment décrite en ce sens qu'elle utilise des sous-suites formées d'éléments consécutifs et non distants de $\left[\frac{n}{2}\right]$ comme dans la méthode de Shell.

Au cours du premier passage, les éléments a_i de SI sont comparés de la façon suivante : soit a_i tel que $i = 1, 3, 5, \ldots, n-1$; si $a_i > a_{i+1}$ nous permutons a_i et a_{i+1}.

A l'issue de ce premier passage, nous obtenons $\frac{n}{2}$ sous suites ordonnées, formées de deux éléments consécutifs ; soient f^i_1 ces sous suites où $i = 1, 2, \ldots, \frac{n}{2}$.

Au cours du deuxième passage, nous fusionnons deux à deux ces sous-suites, afin d'obtenir $\frac{n}{4}$ sous-suites ordonnées, formées de quatre éléments, ce que nous symboliserons par :

$$f^1_1 + f^2_1 \longrightarrow f^1_2$$
$$f^3_1 + f^4_1 \longrightarrow f^2_2$$
$$\vdots$$
$$f^{\frac{n}{2}-1}_1 + f^{\frac{n}{2}}_1 \longrightarrow f^{\frac{n}{4}}_2$$

A l'issue du p° passage, nous obtiendrons $\frac{n}{2^p} = 1$, sous suite unique ordonnée ; c'est donc la suite finale SR formée de n éléments.

Le tri est alors terminé.

Remarque : Le fusionnement des deux sous-suites f_i^j et f_i^{j+1} est obtenu par une méthode d'insertion (cf. chapitre VI).

Donnons un exemple pour fixer les idées :

N° des mémoires	SI	1	2	3	4
1	3	2	1	1	1
2	2	3	2	2	2
3	1	1	3	3	3
4	4	4	4	4	4
5	16	8	5	5	5
6	8	16	8	8	6
7	9	5	9	9	7
8	5	9	16	16	8
9	15	6	6	6	9
10	6	15	7	7	10
11	7	7	11	10	11
12	11	11	15	11	12
13	12	12	10	12	13
14	14	14	12	13	14
15	13	10	13	14	15
16	10	13	14	15	16

Nous avons symbolisé chaque sous-suite f_i^k par un rectangle vertical :

2. 2. Nombre de tests théoriques dans la méthode de Von Neumann :

Dans la méthode de Von Neumann, le nombre de passages dans la suite est égal à p si $n = 2^p$. D'autre part, un test permet de classer au moins un élément, de sorte que le nombre de tests N est

$$N \leqslant n \times p \qquad \text{avec } p = \log_2 n.$$

Nous verrons dans les résultats que nous sommes loin de ce résultat.

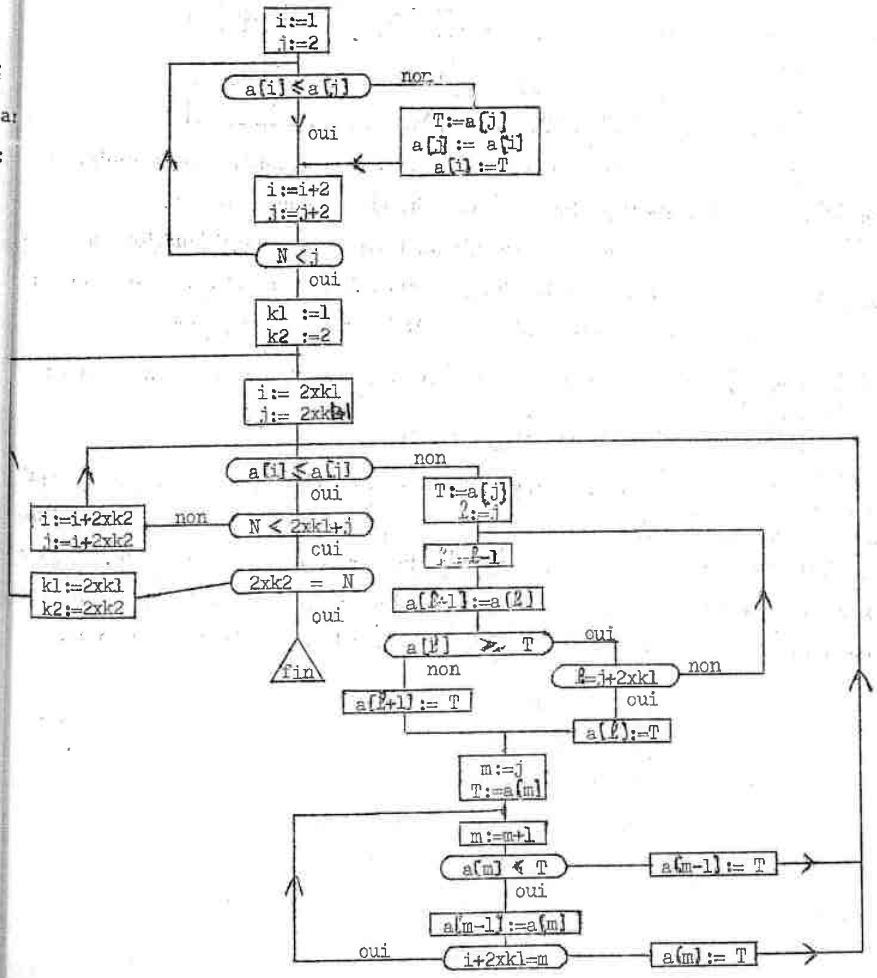
3°) Organigramme

Cet organigramme a été conçu pour un programme écrit dans le langage PASØ IBM.

Soit f_{k-1}^{ℓ} et $f_{k-1}^{\ell+1}$ les deux sous-suites consécutives, avant de commencer le k° passage.

i représente l'indice du dernier élément de f_{k-1}^{ℓ} et dans la suite du tri, il sera remplacé par 1, lequel sera décroissant.

j représente l'indice du premier élément de f_{k-1}^{i+1}, et il sera croissant.



4°) Programme PASØ

C'est le programme n° 6. Il occupe 147 mémoires.

C'est le seul programme à ne pas avoir été écrit sous forme de sous-programme en raison, d'une part de son encombrement, et, d'autre part, des résultats expérimentaux décevants obtenus.

5°) Résultats expérimentaux

	256 Nb	512 Nb
avec SI 1	16 '	68 '
avec SI 2	32 "	3 '

Ces résultats sont assez décevants, tout du moins en fonction des résultats théoriques de la méthode de Von Neumann.

En effet, en faisant l'hypothèse que l'équation du temps de passage est de la forme $T = k\, n \log_2 n$ avec 256 nombres, nous obtenons :

$$T_1 = k \times 256 \times 8 = 16' \Longrightarrow k = \frac{16}{256 \times 8}$$

avec 512 nombres, nous devrions obtenir un temps de passage T2 tel que :

$$T_2 = \frac{16}{8 \times 256} \times 512 \times 9 = 36'$$

Or, le résultat obtenu est très supérieur ; cela s'explique par le fait que nous fusionnons les f_1^e et f_i^{e+1} en utilisant la méthode d'insertion et qu'un seul test ne permet pas de classer au moins un élément. Nous verrons que pour la méthode de tri par insertion, l'équation du temps de passage est en n^2, et c'est ce que nous constatons dans le cas présent également.

CHAPITRE III

LES METHODES DE SELECTION

I - GENERALITES

Les méthodes de transposition ont l'avantage de ne demander qu'une seule mémoire de travail pour trier la suite initiale SI formée de n éléments $a_1, a_2, \ldots, a_n$. Par contre, elles ne sont pas rapides.

Les méthodes de sélection ont des performances nettement meilleures ; par contre elles nécessitent un nombre de mémoires de travail dépendant de n. Elles consistent à extraire, d'après certains critères, un (ou plusieurs) éléments e_i de la suite s et à le (ou les) transférer à une (ou des) place déterminée(s).

II - METHODE de SELECTION ORDINAIRE

1°) Généralités

L'élément e_i extrait sera le plus petit (ou le plus grand) des e_j avec $j = 1, 2, \ldots, n$. Cet élément est alors transféré dans une zone de sortie, que nous noterons r_i où il sera mis à sa place définitive dans la suite SR.

2°) Exposé de la méthode

Au premier passage, plaçons a_1 dans la mémoire T, et comparons T à a_j avec $j = 2, \ldots, n$.

Si $T \leqslant a_j$, nous allons consulter l'élément suivant a_{j+1}.

Si $T \geqslant a_j$, on T nous plaçons a_j et nous allons consulter l'élément a_{j+1}.

Après avoir examiné tous les a_j, l'élément a_i^1, tel que $a_i^1 \leqslant a_j$ pour $j = 1, 2, \ldots, n$ va se trouver dans T, et nous aurons bien sélectionné le plus petit de tous les a_j que nous transférerons en r_1.

Pour sélectionner à l'issue du second passage un nouvel élément a_i^2 (le second plus petit des a_j), il faut remplacer a_i^1 par un élément M tel que $M \geqslant a_j$ quel que soit $j = 1, 2, \ldots, n$.

Ce transfert nous oblige à conserver l'adresse de a_i^1.

Sous cette forme, la méthode est donc très lourde et très lente, car il nous faut n passages dans SI pour classer les n éléments, d'où un nombre de tests $N = n(n-1)$, ce qui est beaucoup. De plus, nous avons besoin, pour trier n nombres, de n mémoires auxiliaires, servant de zone de transfert pour les éléments sélectionnés.

Nous éliminons ce dernier inconvénient en combinant la zone de stockage des éléments à trier (initialement SI) avec la zone de sortie pour les éléments sélectionnés. Pour cela, il suffit de libérer une mémoire au début de chaque passage, et d'y transférer alors l'élément sélectionné.

Or, au cours du premier passage, en plaçant a_1 dans T, nous libérons la mémoire m_1. Soit a_k le premier des éléments a_j suivants a_1, tel que $a_k < T$ (nous avons $a_j \geqslant a_1$ pour $j = 1, 2, \ldots, k-1$). Nous permutons alors a_k et a_1, c'est à dire que a_k devient $a'_k = a_1$, et que T contient a_k. Lorsque les éléments a_j ($j = 1, 2, \ldots, n$) auront été examinés, T contiendra le plus petit de tous les a_i soit a_i^1. La mémoire ayant contenu a_1 étant libre, nous y transférerons alors l'élément a_i^1.

Au second passage, nous placerons alors a_2 dans T, au troisième : a_3 etc,..

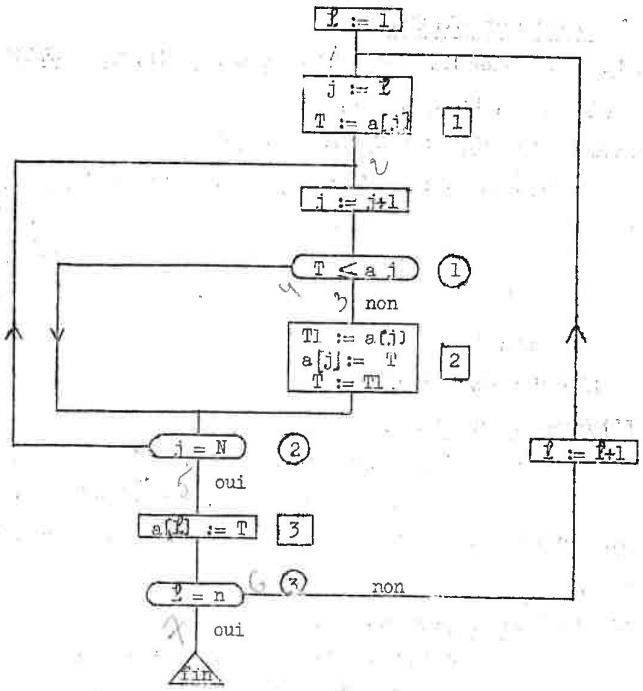
A l'issue de n passages, tous les éléments seront classés.

2°) Organigramme

ℓ est le numéro du ℓ° passage

Les éléments a [1], a [2], ... , a [ℓ-1] sont donc les (ℓ-1) plus petits éléments de SI rangés dans l'ordre a [1] $\leqslant$ a [2] ... $\leqslant$ a [ℓ-1]

j est tel que $1 \leqslant j \leqslant n$ le ℓ° plus petit élément de SI est donc un des a_j.



Remarquons que sous cette forme, cette méthode aurait pu être classée parmi les méthodes de transpositions, car elle n'utilise que deux mémoires auxiliaires de travail.

3°) Nombres de tests théoriques et de transferts

Ceux-ci sont indépendants de la distribution initiale des éléments a_i dans SI. Soit N le nombre de tests, nous avons

$$N = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

Le nombre de transferts T, par contre, dépend de l'arrangement initial T ; nous avons $T = 2n + 3N_n$ en appelant N_n le nombre de tests (1) répondant non.

4°) Programme ALGOL

```
PROCEDURE Sélection linéaire (A,N) ; VALEUR N ; TABLEAU A ; ENTIER
DEBUT ENTIER J, l ; REEL T, T1 ;
    POUR ℓ := 1 PAS 1 JUSQUA N FAIRE
        DEBUT T := A[ℓ] ; POUR J := ℓ+1 PAS 1 JUSQUA N FAIRE
            SI T > A[J] ALORS DEBUT T 1 := A[J] ;
                                    A[J] := T ;
                                    T := T1 FIN ;
        FIN ;
FIN     Sélection linéaire ;
```

5°) Programme PASØ

C'est le programme n° 7.

Il occupe 49 mémoires. La mémoire NSELE contient le nombre n d'éléments à trier, lesquels sont rangés en séquence en A0001, ..., A000 n.

6°) Résultats expérimentaux

J'ai obtenu les résultats suivants :

	256 Nb	512 Nb
avec SI1	21'	81'
avec SI2	16'	62'

Ces résultats confirment les résultats théoriques et donnent une équation du temps de passage f(t) de la forme

$$f(t) = k\,n^2.$$

Les résultats obtenus avec SI2 sont assez proches de ceux obtenus avec SI1, et les performances sont donc très peu améliorées lorsque la suite initiale est déjà plus ou moins triée. Les méthodes précédentes étaient sensiblement plus rapides, avec des suites déjà arrangées.

III - SELECTION QUADRATIQUE

Nous avons vu à propos de la méthode de Shell que le fait de diviser la suite initiale SI en plusieurs sous-suites rendait cette méthode beaucoup plus rapide.

Partageons donc la suite initiale SI, formée de n éléments a_i, en m sous-suites, formées de $\frac{n}{m}$ éléments et soient $s_1, s_2, \ldots, s_m$ ces sous suites.

Pendant une première phase de tri, nous allons extraire le plus petit élément de s_1, soit s_1^1, puis le plus petit élément de s_2, soit s_2^1, etc... et enfin le plus petit élément de s_m, soit s_m^1.

Nous obtenons alors une sous-suite s, comprenant les m éléments définis ci-dessus. Il nous suffira d'extraire le plus petit de ces m éléments pour avoir le plus petit élément de SI, que nous appellerons r_1. Supposons qu'il appartient à la sous-suite s_j ; c'est donc l'élément que nous avons appelé s_j^1.

Au cours du second passage, pour obtenir le deuxième plus petit élément de SI, c'est à dire r_2, il suffit d'extraire le deuxième plus petit élément de s_j, soit s_j^2 afin qu'il vienne remplacer s_j^1 dans la sous-suite s. Le nouveau plus petit élément de s sera le deuxième plus petit élément de SI, c'est à dire r_2.

A l'issue du k° passage, nous avons donc sélectionné les k premiers plus petits éléments de SI, soit $r_1, r_2, \ldots, r_k$. Supposons que r_k est le ℓ° plus petit élément de la sous-suite s_j, c'est à dire s_j^ℓ d'après nos notations. Pour obtenir r_{k+1}, il faudra sélectionner $s_j^{\ell+1}$, le transférer dans s, à la place de s_j^1, et extraire le plus petit élément de s, c'est à dire r_{k+1}.

Cet élément sera donc obtenu à l'issue de $(\frac{n}{m} + m)$ tests.

Soit $f(m) = \frac{n}{m} + m$ le nombre de tests sera minimum pour $\frac{\partial f}{\partial n} = 0$, c'est à dire $m = \sqrt{n}$.

Nous obtenons alors le nombre de tests N pour trier n éléments $N = n + 2(n-1)\sqrt{n}$, c'est à dire n tests pour obtenir le plus petit et $2\sqrt{n} \times (n-1)$ pour obtenir les (n-1) autres éléments.

Ces résultats sont nettement supérieurs à ceux de la méthode de sélection ordinaire. Néanmoins, la sélection quadratique présente un grave défaut, de sorte qu'elle est peu utilisée.

En effet, il n'est plus possible d'imbriquer la zone initialement occupée par SI avec la zone de sortie des éléments s_i sélectionnés définitivement. Pour constituer la suite s, il nous faut également $\sqrt{n}$ mémoires auxiliaires, celles-ci contenant, soit les éléments eux-mêmes, soit leurs adresses. Les éléments s^1_j des suites s_j peuvent être sélectionnés, soit par la méthode de sélection ordinaire dans sa première version (s^1_j sera alors transféré dans la zone s, et remplacé dans s_j par M défini plus haut), ou dans sa seconde version (les $\sqrt{n}$ mémoires auxiliaires contenant les adresses des s^1_j, la sous-suite s est formée de telle façon que j° élément soit inclu dans la j° sous-suite s_j).

Il nous faut donc au total $n + \sqrt{n}$ mémoires auxiliaires et c'est un très gros handicap pour cette méthode.

Je n'ai pas personnellement essayé cette méthode sur calculateur.

IV - METHODE de THOMAS N. HIBBARD

1°) Généralités

La méthode de Thomas N. Hibbard [6] est une des approches du problème du tri les plus récentes. A une grande rapidité du tri, elle allie un nombre réduit de mémoires auxiliaires de travail.

2°) Exposé de la méthode

La suite initiale SI de premier élément a_1 est scindée en deux sous-suites I_1 et S_1. I_1 est formée à partir des éléments $a_i \leqslant a_1$, que nous notons $i_1, i_2, \ldots, i_k$; S_1 est formée à partir des éléments $a_j \geqslant a_1$, et nous la notons $s_1, s_2, \ldots, s_m$; a_1 lui-même n'appartenant ni à I_1, ni à S_1. Cette partition est effectuée de telle façon qu'à l'issue du premier passage i_k et s_1 soient séparées par une position de mémoires, cette (k+1)° mémoire va recevoir a_1. Or, (k+1) est, par définition, le rang de a_1 dans SR, a_1 est donc mis en place définitivement.

Le processus va alors s'appliquer à I_1 et S_1 séparément et non à leur réunion, car $i_j \leqslant s_i$ quels que soient j et i.

Si, par exemple, nous choisissons de placer i_1 à son rang définitif dans SR, nous scinderons I_1 en deux nouvelles sous-suites I_2 et S_2, mais cela impose de conserver les adresses des bornes de S_1. Nous allons donc constituer une pile (*) dans laquelle nous mettrons à chaque étape les adresses des bornes des sous-suites S_j laissées pour compte. Une sous-suite I_k sera totalement triée au cours d'un passage, lorsqu'elle ne comportera plus que deux éléments. Il ne restera plus qu'à repartir en sens inverse en allant chercher dans le dernier étage de la pile les bornes de la sous-suite à scinder.

Pour constituer les deux sous-suites I_1 et S_1, et libérer la (k+1)° mémoire devant recevoir a_1, nous opérons de la façon suivante :

a) a_1 est transféré en T, ce qui libère la 1ère mémoire

b) nous commençons l'exploration des a_j à partir de a_n puis a_{n-1} etc... soit a_ℓ le premier de ces éléments inférieur à T ($a_j \geqslant a_\ell$ pour $\ell+1 \leqslant j \leqslant n$) ; a_ℓ est transféré dans la première mémoire, la ℓ° mémoire devenant libre, nous recommençons l'exploration des a_i à partir de a_2, a_3 etc..

Soit alors a_m le premier élément supérieur à T ($a_i \leqslant$ T pour $2 \leqslant i \leqslant m-1$) ; a_m est alors transféré dans la ℓ° mémoire, ce qui libère la m° mémoire etc... Chaque fois qu'un transfert s'avère nécessaire nous inversons le sens d'exploration des éléments de la suite.

Soient alors i l'indice de ces éléments lorsque nous les explorons par indice croissant, et j l'indice décroissant : lorsque i = j tous les éléments ont été examinés, et éventuellement transférés et T sera transféré dans cette i° mémoire.

---

* La notion de pile a été essentiellement utilisée en compilation des langages [9]. Il s'agit en gros d'un mode de stockage analogue à une pile d'objets on ne peut à chaque instant qu'enlever le sommet de la pile, ou placer un nouvel élément en son sommet.

Donnons un exemple : chaque colonne 1, 2, 3, ...représente le contenu des mémoires à l'issue du 1er, 2ème, 3ème ... passage. La colonne SI représente la suite initiale et la colonne O le numéro des mémoires. La ligne T représente le contenu de T (c'est à dire l'élément qui sert à partager chaque suite). Dans la pile, les bornes sont notées (e,k), l'évolution d'un étage de la pile se lisant de gauche à droite.

O	SI	1	2	3	4	5	6	7	8	9	SR
1	7	1	0	[0]							0
2	4	4	[1]	1							1
3	9	3	3		2	[2]					2
4	0	0	4		[3]	3					3
5	2	2	2		4		[4]				4
6	6	6	6		5		5	[5]			5
7	5	5	5		6		6	6			6
8	8	[7]									7
9	3	8							[8]		8
10	1	9							9		9
T		[7]	1	0	3	2	4	5	8		

⋮	
(3,7) ; (5,7) ; (6,7)	2ème étage
(9 , 10)	1er étage
Evolution de la pile	

Dans les colonnes, je n'ai indiqué que le contenu des mémoires sur lesquelles je travaille effectivement au cours du passage correspondant.

Le tri est terminé lorsque la pile est vide, après avoir trié une sous-suite, formée seulement de deux éléments.

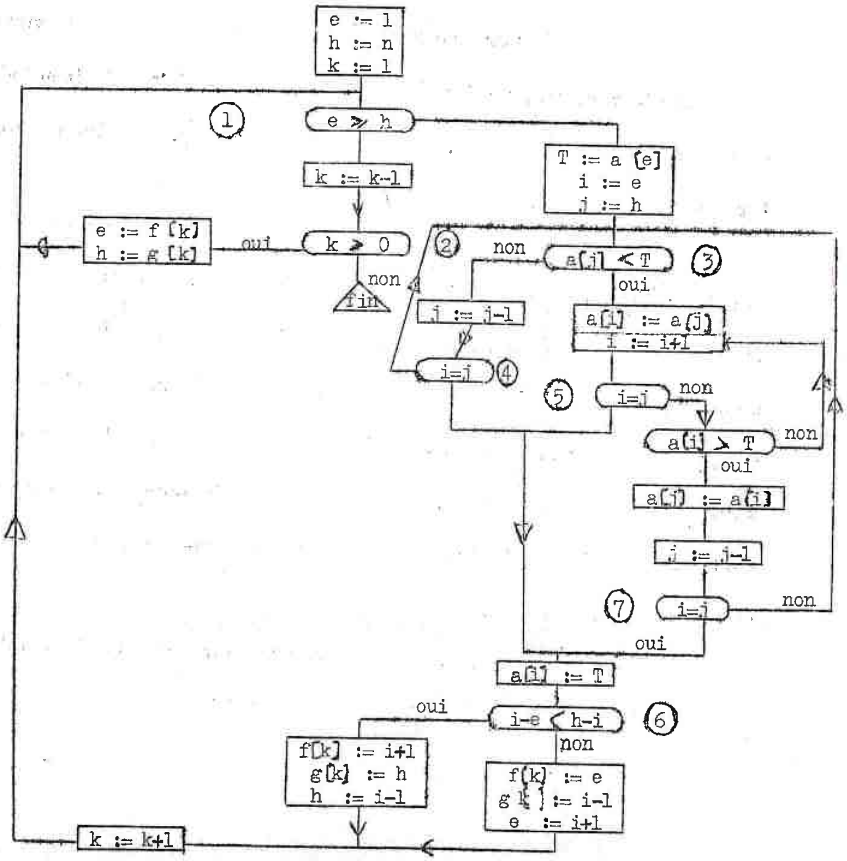
3°) Organigramme

e et h sont les bornes inférieures et supérieures de chaque sous-suite à scinder.

i correspond aux indices qui varient en croissant à partir de e

j correspond aux indices qui varient en décroissant à partir de h

k est le sommet de la pile f et g qui contient e et h.



Remarque : La méthode telle qu'elle est exposée en 2°) nous obligerait à réserver pour la pile n mémoires auxiliaires (la pile atteindrait la hauteur n dans le cas d'une suite SI rangée par exemple dans l'ordre de tri).

Montrons comment le test (6) : i-e < h-i qui, parmi les deux sous-suites I_j et S_j qui viennent d'être obtenues, partage à nouveau la plus petite des deux, permet de réduire la hauteur maximum de la pile de n à $\log_2 n$.

Par récurrence :

Cette propriété est vérifiée pour n = 1.

Supposons qu'elle le soit pour n = 1, 2, ..., (m-1).

Lorsque le programme trie une suite de longueur m, il détermine le rang du premier élément a_1 de cette suite, puis il scinde la suite en deux, et trie une sous-suite de longueur $m_1 \leqslant \frac{m}{2}$. La pile a comme hauteur lorsque débute ce tri.

D'après notre hypothèse, il n'y a pas plus de $\log_2 m_1 \leqslant \log_2 m-1$ nouvelles entrées dans la pile, entrées nécessitées par le tri de cette première sous-suite, c'est à dire pas plus de $\log_2 m$ entrées au total pour trier la sous-suite de longueur m_1. Lorsque débute le tri de la seconde sous-suite de longueur $m_2 = m-m_1$, la pile est vide, car nous n'avons plus qu'une seule suite de longueur m_2 à trier. Or, $m_2 < m-1$ l'hypothèse, selon laquelle la hauteur de la pile est inférieure ou égale à $\log_2(m-1)$ pour une suite de longueur m-1, est encore valable.

Donc, en aucun cas la hauteur de la pile ne dépasse $\log_2$

4°) Nombre de tests théoriques

Hibbard [6] donne pour une suite de nombres répartis au hasard un nombre de tests $N_H = 1{,}4\ n \log_2 n$.

Nous verrons que pour une suite SI, déjà totalement ordonnée, le nombre de tests est très supérieur à N_H.

5°) Programme ALGOL

Ce programme est extrait de l'article de Hibbard [6].

```
PROCEDURE P(a,n) ; VALEUR n ; TABLEAU a ; ENTIER n ;
DEBUT ENTIER e,h,i,j,k ; ENTIER TABLEAU f,g(1 : log2n) ; REEL T ;
      e := 1 ; h := n ; k := 1 ;
B : SI e ⩾ h ALORS ALLER A C ; T := a[e] ; i := e ; j := h ;
E : SI a[j] ⩽ T
      ALORS DEBUT a[i] := a[j] ; i := i+1 ; SI i=j ALORS ALLER A D FIN
      SINON DEBUT j := j-1 ; SI i=j ALORS ALLER A D ; ALLER A E FIN ;
F : SI a[i] > T
      ALORS DEBUT a[j] := a[i] ; j := j-1 ; SI i=J ALORS ALLER A D ;
                  ALLER A E FIN ;
      SINON DEBUT i := i+1 ; SI i=j ALORS ALLER A D ; ALLER A F FIN ;
D : a[i] := T ;
      SI i-e < h-i
      ALORS DEBUT f[k] := i+1 ; g[k] := h ; h := i-1 FIN ;
      SINON DEBUT f[k] := e ; f[k] := i-1 ; e := i+1 FIN ;
      k := k+1 ; ALLER A B ;
C : k := k-1 ;
      SI k ⩾ 0 ALORS DEBUT e := f[k] ; h := g[k] ; ALLER A B FIN ;
FIN de la procédure P(a,n) ;
```

6°) Programme PASØ

cf. au programme n° 8.

Ce programme occupe 124 mémoires.

7°) Résultats expérimentaux

n	128	256	512
SI1	32"	2'00"	4'15"
SI2	5'20"	23'	86'

Si nous comparons ces résultats avec ceux des méthod précédentes, nous voyons qu'il sont pratiquement inversés.

En effet, la méthode est extrêmement rapide dans le d d'une suite initiale d'éléments répartis au hasard. Dès que la suite devient plus ou moins arrangée, le temps de tri augmente de plus en plus, pour dev nir aussi médiocre, dans le cas d'une suite totalement ordonnée, que, par exemple, celui de la méthode de sélection ordinaire, dans le cas d'une suite d'éléments répartis au hasard.

a) Montrons, par exemple, ce qui se passe dans le ca d'une suite totalement ordonnée.

Soit SI = 1,2,3,4,5,6,7,8,9.

Au cours du premier passage e=1, h=9, et T contient SI est alors partagée en :

$$\text{et} \begin{cases} I_1 = 1 & \text{à l'issue de n-1 tests} \\ S_1 = 2,3,4,5,6,7,8,9. & \text{(dans cet ordre).} \end{cases}$$

Au second passage, e=1, h=1 ; I_1 est totalement trié, nous allons trier S_1. Au cours du troisième passage : e=2, h=9, T contient 2 et nous obtenons

$$I_2 = 2$$
$$S_2 = 3,4,5,6,7,8,9.$$

Pour trier n nombres totalement ordonnés, nous auron alors un nombre N de tests tel que

$$N = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

A chaque passage, la suite ne subit aucune modificatio et le nombre de transfert est donc nul.

b) Envisageons à présent le cas d'une suite totalement rangée dans l'ordre inverse de tri.

Soit SI = 9,8,7,6,5,4,3,2,1.

Au premier passage, T contient 9.

Nous obtenons :

$$\underbrace{1,8,7,6,5,4,3,2}_{I_1} \quad , \quad \underbrace{9}_{S_1}.$$

A l'issue du second passage, S_1 est triée totalement.

A l'issue du troisième passage, T contenait 1, et nous obtenons :

$$\underbrace{1}_{I_3} \quad , \quad \underbrace{8,7,6,5,4,3,2}_{S_3}$$

La suite ne subit aucune modification.

Nous obtenons le même nombre de tests que dans le cas a) avec, en plus, n transferts.

Ces seuls exemples suffisent à montrer que la méthod de Hibbard est sensible à la distribution des éléments dans la suite SI.

Afin d'éviter cet inconvénient, signalons la méthode de Hoare (Algorithmes n° 63 et 64 : Communications of ACM - avril 1961).

Cette méthode, dans son principe, est la même que celle de Hibbard, mais au lieu de prendre systématiquement le premier élément de chaque suite s pour la scinder, elle choisit un élément quelconque de la suite, soit a[q] et où q est un nombre choisi au hasard, tel que $e \leqslant q \leqslant h$ e et h étant les indices des bornes inférieures et supérieures de s.

CHAPITRE IV

LES METHODES D'INTERCLASSEMENT

I - GENERALITES

Considérons deux suites s et s' déjà ordonnées, et soient $e_1, e_2, \ldots, e_n$ les éléments de s et $e'_1, e'_2, \ldots, e'_k$ ceux de s'.

Interclasser ces deux suites consiste à les fusionner de façon à obtenir une seule suite SR également ordonnée et formée de ces n+k éléments, que nous noterons r_i.

Nous voulons, d'autre part, que la comparaison d'un élément e_i de s avec un élément e'_j de s' permette de placer définitivement dans SR l'un ou l'autre de ces éléments au moyen d'un seul test et d'un seul transfert.

Supposons que nous avons obtenu les p premiers éléments de SR ; le (p+1)° sera alors, soit l'élément e_i de s, soit l'élément e'_j de s'. (i et j sont tels que i+j = p+1).

- si $e_i \leqslant e'_j$ alors $r_{p+1} = e_i$
  et nous comparerons ensuite e_{i+1} à e'_j
- si $e_i > e'_j$ alors $r_{p+1} = e'_j$
  et nous comparerons ensuite e'_{j+1} à e_i

Nous devrons donc explorer s et s' par indice croissant, et borner s et s' par deux éléments $e_{n+1} = F$ tels que $F > e_i$ avec $1 \leqslant i \leqslant n$
$e'_{k+1} = F$ tels que $F > e'_j$ avec $1 \leqslant j \leqslant k$.

Pour obtenir S, nous aurons besoin de n+k+2 mémoires auxiliaires.

Enfin, à l'issue de n+k+1 tests, tous les éléments de s et s' seront classés.

Bien entendu, cette méthode se généralise à un nombre quelconque ℓ de suites $s_1, s_2, \ldots, s_\ell$ à fusionner en une seule suite SR.

Nous voyons tout de suite les limitations qui apparaissent avec ces méthodes : il nous faut d'une part avoir des sous-suites s et s' déjà ordonnées, et d'autre part, utiliser un nombre de mémoires auxiliaires au moins égal au nombre total d'éléments à trier.

Néanmoins, elles seront intéressantes chaque fo
la suite initiale SI sera formée d'un nombre suffisamment grand de so
suites adjacentes triées.

II - METHODE de VON NEUMANN

1°) Généralités

Au cours de chaque passage, cette méthode const
un nombre donné P des sous-suites triées et adjacentes, de longueur ℓ,
interclasse pour donner $\frac{P}{2}$ sous-suites adjacentes et triées, de longue
En itérant le processus, à l'issue d'un nombre fini de passages, nous n'
nons plus qu'une seule sous-suite SR totalement triée, et formée de n é

2°) Exposé de la méthode

Le problème est donc de construire puis de fusion
les sous-suites précédentes. Supposons, pour simplifier l'exposé, que

Choisissons dans SI n sous-suites adjacentes for
d'un seul élément. Nous les noterons s_i^1 et nous avons $s_i^1 = a_i$ pour $i = 1,2$
Elles sont toutes forcément triées puisqu'elles ne comportent qu'un séu
ment. Interclassons alors s_1^1 et s_2^1 nous obtenons une seule sous-suite t
formée de deux éléments et nous la notons s_1^2 (la notation s_j^i représen
donc la j° sous-suite au début du i° passage).

Nous interclassons donc au cours du premier pas

s_1^1 et s_2^1 ce qui nous donne la sous-suite s_1^2
s_3^1 et s_4^1 ce qui nous donne la sous-suite s_2^1
⋮ ⋮
s_{2p-1}^1 et s_{2p}^1 " s_p^2
s_{n-1}^1 et s_n^1 " $s_{\frac{n}{2}}^2$

A l'issue de ce premier passage, nous obtenons d
s_i^2 sous-suites adjacentes formées de 2 éléments ($1 \leqslant i \leqslant \frac{n}{2}$) (lesquelle
trouvent en fait dans les mémoires auxiliaires).

Au cours du second passage, nous interclassons les sous-suites s_1^2 et s_2^2, pour obtenir la sous-suite s_1^3.

c'est à dire que s_1^2 et s_2^2 nous donne s_1^3
⋮ ⋮
" s_{2p-1}^2 et s_{2p}^2 " s_2^3
⋮ ⋮
" $s_{\frac{n}{2}-1}^2$ et $s_{\frac{n}{2}}^2$ " $s_{\frac{n}{4}}^3$

Nous obtenons donc à présent $\frac{n}{4}$ sous-suites triées notées s_i^3 ($1 \leqslant i \leqslant \frac{n}{4}$) et formée de 4 éléments.

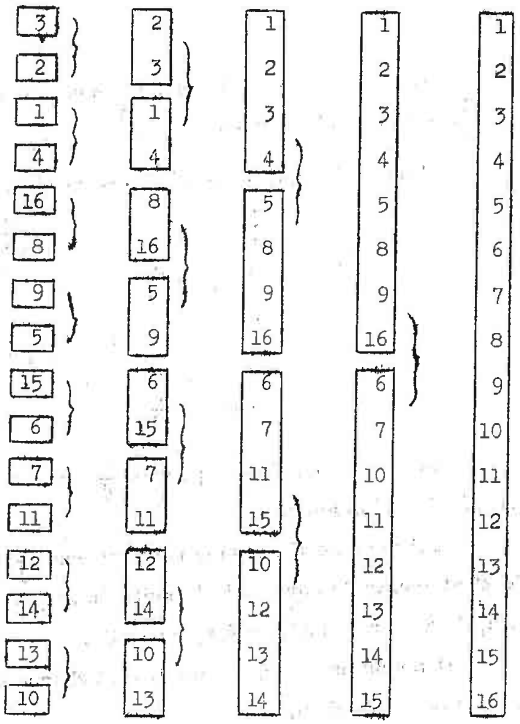
A l'issue du j° passage nous aurons $\frac{n}{2^j}$ sous-suites triées formées de 2^j éléments de sorte qu'à la suite du k° passage ($n=2^k$) tous les éléments a_i de SI seront triés et formeront SR.

Donnons un exemple dans lequel chaque sous-suite est encadrée dans un rectangle vertical.

L'accolade symbolise l'interclassement des deux sous suites qu'elle relie.

D'autre part, si k est pair, la suite SR se trouve dans les mémoires qui contenaient initialement SI.

Si k est impair, la suite SR se trouve dans les mémoires auxiliaires.



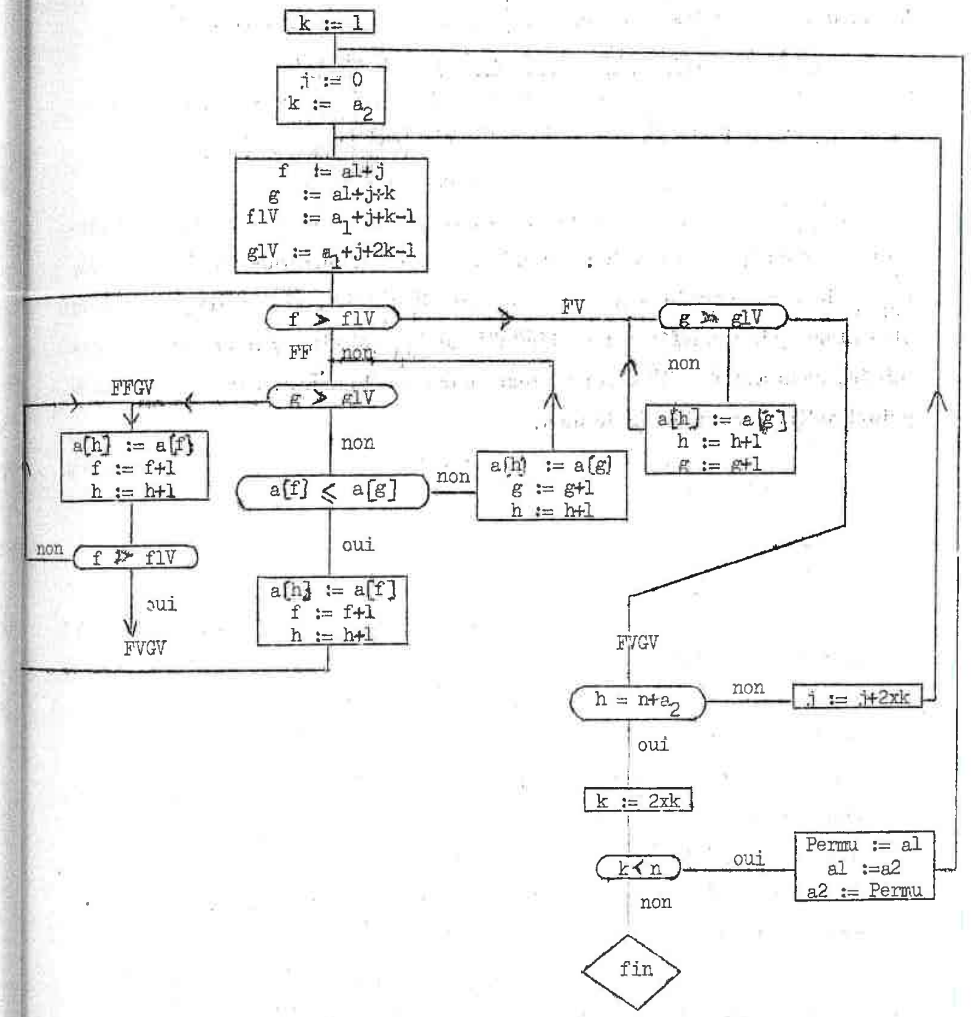
3°) Organigramme

Nous ne pouvons plus, ainsi que nous l'avions fait dans les généralités sur les méthodes d'interclassement, borner chaque sous suite s^i_j par l'élément F que nous avions introduit alors. Cet élément était simplement destiné dans le cas où $e_n < e'_k$ (cf paragraphe I) à permettre le transfert total jusqu'en e'_k de s' dans la suite SR. Dans le cas présent, nous le remplacerons par un calcul sur les indices n et k de e_n et e'_k, indices que nous noterons respectivement f1V et g1V.

L'indice des éléments dans la sous-suite s^i_{2p-1} est repéré par f, celui des éléments dans la sous suite s^i_{2p} par g.

La suite SI est initialement stockée dans les n mémoires consécutives commençant en a_1, les n mémoires auxiliaires commençant en a_2.

L'organigramme est alors le suivant :



j est l'indice qui, au cours du i° passage, permet de passer de l'interclassement des sous-suites s^i_{2p-1} et s^i_{2p} à l'interclassement des sous suites s^i_{2p+1} et s^i_{2p+2}

k est la longueur des sous-suites s^i_m au cours du i° passage ($k=2^i$)

4°) Nombre de tests et de transferts théoriques

Le nombre de transferts au cours d'un passage est égal à n, d'où le nombre de transferts $T = n \log_2 n$.

Au cours d'un passage, le nombre de tests sur les éléments eux-mêmes est inférieur à n, par suite de la suppression de l'élément F de chaque sous-suite. En effet, supposons que tous les éléments de s^i_{2p-1} aient été examinés (ce que nous savons par le test f > flV), il ne reste plus alors qu'à transférer les éléments de s^i_{2p} qui n'ont pas encore été examinés, sans avoir à effectuer de tests sur ces éléments, car nous savons à priori qu'ils sont triés également.

D'où le nombre total de tests à l'issue du k° passage :

$N \leqslant n \times k$

$\leqslant n \log_2 n$.

5°) Programme ALGOL

```
PROCEDURE Interclassement (a,al,a2,n) ; VALEUR n,al,a2 ; TABLEAU a ;
   ENTIER  al,a2,n ;
DEBUT ENTIER j,k,f,g,h,flv,glv ,Permu ; k := 1 ; al := 1 ; a2 := n+1 ;
   B1 : j := 0 ; h := a2 ;
   B2 : f := al+j ; g := al+j+k ; flv := g-1 ; glv := flv+k ;
   F : Si f > flv ALORS ALLER A FV ;
   FF : Si g > glv ALORS ALLER A FFGV ;
      SI a[f] ≤ a[g] ALORS DEBUT a[h] := a[f] ; f := f+1 ; h := h+1 ;
         ALLER A F FIN ;
                  SINON DEBUT a[h] := a[g] ; g := g+1 ; h := h+1 ;
         ALLER A F F FIN ;
   FFGV : a[h] := a[f] ; f := f+1 ; h := h+1 ;
         ALLER A SI f > flv ALORS FVGV SINON FFGV ;
```



```
   FV : SI g > glv ALORS ALLER A FVGV ;
      a[h] := a[g] ; g := g+1 ; h := h+1 ; ALLER A FV ;
   FVGV : SI h = a2+n-1 ALORS ALLER A Modik ; j := j+2xk ; ALLER A B2 ;
   Modik : k := k x 2 ;
      SI k < n ALORS DEBUT Permu := al ; al := a2 ; a2 := Permu ; ALLER A B1
                  FIN ;
   FIN Procédure Interclassement ;
```

Cette procédure correspond à l'organigramme du paragraphe 3. Elle est relativement longue, car, sous cette forme, elle reprend plusieurs fois des éléments du programme qui existaient déjà ; ceci est simplement destiné à éviter des tests, soit sur g (soit sur f) dont on connait à priori la réponse, du fait que l'indice g (ou f) n'a pas été modifié.

Donnons un second programme ALGOL, dont l'écriture est plus condensée, mais qui fait intervenir un plus grand nombre de tests.

```
PROCEDURE Interclassement (a,al,a2,n) ; VALEUR n,al,a2 ;
   ENTIER n,al,a2 ; TABLEAU a ;
   DEBUT ENTIER j,f,g,h,flv,glv,Permu ; k:=1 ; al := 1 ; a2 := n+1 ;
      B1 := j := 0 ; h := a2 ;
      B2 : f := al+j ; g := al+j+k ; flv := g-1 ; glv := flv+k ;
      F : SI f > flv ALORS ALLER A SI g > glv ALORS FVGV SINON M ;
         SI g > glv ALORS N : DEBUT a[h] := a[f] ; f := f+1 ; ALLER A R FIN ;
                  SINON ALLER A SI a[f] ≤ a[g] ALORS N SINON M ;
                  M : a[h] := a[g] ; g := g+1 ;
                  R : h := h+1 ; ALLER A F ;
      FVGV : SI h = a2+n-1 ALORS ALLER A Modik ; j:=j+2xk ; ALLER A B2
      Modik : k := k x 2 ;
         SI k < n ALORS DEBUT Permu := al ; al := a2 ; a2 := Permu ;
                  ALLER A B1 FIN ;
   FIN ;
```

6°) Programme PASØ

C'est le programme n° 9. Il occupe 147 mémoires.

7°) Résultats expérimentaux

Les essais ont été faits avec une méthode ne permettant de trier n nombres que si $n = 2^k$. Nous verrons comment il est possible de classer n nombres lorsque n n'est pas égal à 2^k.

	128 Nb	256 Nb	512 Nb
SI1	T_1=1'2"	T_2=2'17"	T_3=5'5"
SI2		2'	4'28"

Nous voyons donc que la méthode est très rapide et très intéressante. De plus, le temps de tri, comparé, par exemple, à la méthode de Monsieur Thomas N. Hibbard, au lieu d'être allongé lorsque la suite est déjà plus ou moins arrangée, est, au contraire, légèrement plus court. Cela provient de la réduction du nombre de tests sur les a_i : en effet, dans le cas d'une suite SI totalement rangée lorsque tous les éléments de s^i_{2p-1} sont transférés dans les mémoires auxiliaires (c'est à dire à la suite du k° test sur f et sur a_i, si k est la longueur de s^i_{2p-1}), il n'y a plus un seul test sur les a_i, mais uniquement un test sur g et un transfert.

Cas où n est différent d'une puissance entière de 2

Posons alors $n = 2^{k-1} + \alpha$ où $\alpha < 2^{k-1}$

c'est à dire que nous avons $2^{k-1} < n < 2^k$.

a) on peut, soit compléter SI avec des éléments M tels que $M > a_i$ quel que soit i.

Le nombre de tests sera alors au plus égal à $2^k \times k$.

L'inconvénient de ce procédé est de nécessiter un nombre de mémoires auxiliaires supérieur à n.

b) on peut diviser SI en deux sous-suites s_1 et s_2, telles que s_1 soit formée des 2^{k-1} premiers éléments de SI et s_2 des α derniers éléments de SI.

Nous trierons alors s_1 par la méthode de Von Neumann. Soit sR1 la suite obtenue de sR1, nous extrayons les α premiers éléments, nous

appelons sR1' la sous-suite obtenue, et nous formons une seconde sous-suite avec les $2^{k-1} - \alpha$ éléments restants auxquels nous ajoutons les α éléments de s_2. Nous obtenons alors une nouvelle sous-suite s'_2 formée de 2^{k-1} éléments, et que nous trions par la méthode de Von Neumann. Nous obtenons alors une seconde sous-suite ordonnée sR2.

Il ne nous reste plus qu'à interclasser sR1' et sR2 (en utilisant $2^{k-1} + \alpha$ = n mémoires auxiliaires) pour obtenir la suite finale SR, issue de n éléments de SI.

Le nombre de tests est alors au plus égal à :

$2^{k-1} \times (k-1)$ pour trier s_1

$2^{k-1} \times (k-1)$ pour trier s'_2

$2^{k-1} + \alpha$ pour interclasser sR1' et sR2.

d'où $N \leq 2^k \times k + \alpha$

Il est supérieur à celui obtenu en a), néanmoins, cette seconde solution est intéressante car elle n'utilise que n mémoires auxiliaires.

c) on peut également trier les α derniers éléments par une méthode différente de la méthode de Von Neumann.

Soit sR2' la sous-suite obtenue, il ne reste plus alors qu'à interclasser sR1 et sR2'.

III - IDEE d'UNE AUTRE METHODE de TRI par INTERCLASSEMENT

On recherche dans SI les sous-suites s_i maximales formées d'éléments consécutifs, et déjà ordonnées.

Par exemple, soit SI = 1,2,3,7,9,5,2,4,8,6.

Nous avons dans SI les sous-suites ordonnées suivantes :

1,2, - 1,2,3 - 1,2,3,7, - 1,2,3,7,9 - 5 - 2,4, - 2,4,8 - 6 -

Sont maximales les sous-suites suivantes :

1,2,3,7,9, = s_1

5 = s_2

2,4,8 = s_3

6 = s_4

Soit m le nombre de sous-suites maximales incluses dans SI. Pour trier SI, nous pourrons alors procéder comme dans la méthode de Von Neumann, à cette différence près, que les sous-suites obtenues au cours d'un passage n'auront pas une longueur constante. Au cours du premier passage, nous interclasserons s_1 et s_2, puis s_3 et s_4 etc... Nous obtiendrons $\left[\frac{m}{2}\right]+1$ sous suites à l'issue de ce premier passage. En itérant le processus, la suite SI sera totalement triée à l'issue du k'^{o} passage, k' étant défini par la relation

$$2^{k'-1} < m \leq 2^{k'}$$

Il nous faut alors procéder de la façon suivante :

a) déterminer les extrémités de deux sous-suites consécutives s_{2p-1} et s_{2p}. (un test de la forme $a_i \leqslant a_{i+1}$ permet de déterminer ces extrémités.) Soient alors ℓ et p les longueurs respectives de s_{2p} et s_{2p-1} :

ℓ + p tests seront nécessaires pour obtenir ces extrémités de sorte que n tests sur a_i, au cours d'un passage, permettront de déterminer les extrémités de toutes les sous-suites maximales incluses dans la suite obtenue à partir de SI.

Pour les k' passages, nous obtenons un nombre N' de tests sur les a_i, tel que :

$$N' = k' \times n = \left(\left[\log_2 m\right] + 1\right) \times n$$

b) interclasser s_{2p-1} et s_{2p} en ℓ + p tests au plus sur les a_i

Pour les k' passages, cela nous donne un nombre N'' de tests sur les a_i, tel que :

$$N'' \leq k' \times n$$
$$\leq \left(\left[\log_2 m\right] + 1\right) \times n$$

Le nombre total N de tests à effectuer sur les a_i est donc :

$$N = N' + N'' \leq 2n\left(\left[\log_2 m\right] + 1\right)$$

Quant aux nombres de transferts T, il est encore égal à $\left(\left[\log_2 m\right] + 1\right) \times n$.

Comparons ces résultats à ceux obtenus avec la méthode de Von Neumann :

α) supposons pour cela que $n = 2^k$ et $m = 2^{k'}$

Posons N_1 = nombre total de tests sur les a_i avec la méthode de Von Neumann

N_2 = " avec cette nouvelle méthode.

Déterminons m pour que cette méthode soit plus intéressante que la précédente, c'est à dire que $N_2 \leqslant N_1$.

$2n \log_2 m \leqslant n \log_2 n$ c'est à dire $m \leqslant \sqrt{n}$

β) Dans le cas général, c'est à dire

$n = 2^{k-1} + \alpha$ et $n \neq 2^{k'}$

Nous avons :

$$N_1 = 2 \times \left(2^{k-1}\right)\left(\left[\log_2 n\right] + 1\right) + \alpha$$
$$N_2 = 2\left(2^{k-1} + \alpha\right) \times \left(\left[\log_2 m\right] + 1\right)$$

La valeur de m pour laquelle $N_2 \leqslant N_1$ est alors fonction de α et, de toute façon, elle est supérieure à celle obtenue dans le cas α), de sorte que cette deuxième méthode peut devenir très intéressante.

CHAPITRE V

LES METHODES D'INSERTION

I - GENERALITES

Soit toujours SI la suite initiale formée des n éléments a_i à trier. A l'issue de la j° étape, la sous-suite s_j obtenue à partir des j premiers éléments de SI est triée. On place alors a_{j+1} par rapport aux éléments e_i de s_j, c'est à dire que nous allons "insérer" a_{j+1} dans s_j de façon à obtenir une sous-suite s_{j+1}, également triée, et composée des (j+1) premiers éléments de SI (nous noterons les éléments de s_{j+1} e'_i pour les distinguer des e_i).

Il s'agit donc de déterminer l'élément e_m de s_j tel que

$$e_m \leqslant a_{j+1}$$
$$e_{m+1} > a_{j+1}$$

Cet élément e_m étant déterminé, il suffira de déplacer $e_{m+1}, e_{m+2}, \ldots, e_j$, d'une position vers la droite, et de transférer a_{j+1} à la place de e_{m+1}.

La suite s_{j+1} est alors construite, et nous pouvons alors placer a_{j+2} dans s_{j+1}.

Le tri sera totalement terminé lorsque nous aurons placé a_n dans la suite s_{n-1}, et nous avons alors SR = s_n.

II - La METHODE d'INSERTION ORDINAIRE

1°) Généralités

Le classement de a_{j+1} est fait, théoriquement, par un produit de transposition. Mais nous verrons dans l'exposé de la méthode que ces transpositions ne sont pas complétement effectuées. Ces transpositions incomplètes permettent de déplacer les e_i d'une position vers la droite, et nous allons donc comparer a_{j+1} successivement à e_j, e_{j-1}, e_{j-2}... etc... jusqu'à e_m.

2°) Exposé de la méthode

a) Plaçons a_{j+1} dans une mémoire de travail T.

Nous comparons alors T à e_j.

— si $T < e_j$, e_j est décalé d'une position vers la droite, c'est à dire qu'il devient e'_{j+1}. Puis T est comparé à e_{j-1}, e_{j-2}, etc... jusqu'à ce que $e_m \leqslant T$ soit déterminé.

Nous voyons que nous n'avons pas une transposition complète, puisque le test $T < e_\ell$ répondant oui amène un transfert de e_ℓ en $\ell+1$, mais pas de transfert de T en ℓ, lequel est inutile tant que e_m n'est pas déterminé. e_m étant déterminé, T est transféré en m+1. La sous-suite s_{j+1} est alors construite et nous pouvons alors recommencer le processus avec a_{j+2}.

Si aucun e_m ne peut être déterminé, alors a_{j+1} est placé dans la première mémoire, les e_i devenant des e'_i tel que $e'_{i+1} = e_i$.

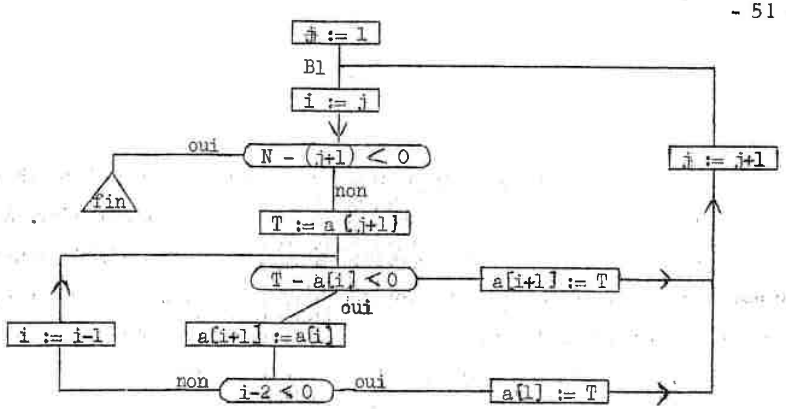
Remarque : Ce résultat peut être obtenu par un test sur l'indice i de e_i, et c'est la méthode que j'ai utilisée ou d'une façon plus astucieuse en bornant à gauche SI par un élément a_o (qui n'appartiendra donc ni à SI ni à s_j) et tel que a_o soit inférieur à tous les a_i, de sorte que le test $T \geqslant a_o$ répondra toujours oui.

— si $T \geqslant e_j$, alors a_{j+1} reste en place, et la sous-suite s_{j+1} est construite. Nous pouvons alors aller recommencer le processus avec l'élément a_{j+2} etc... jusqu'à ce que a_n soit inséré à sa bonne place.

b) Organigramme :

j+1 est le même indice que ci-dessus

i = j , j - 1 , . . . , m.



Donnons un exemple : chaque colonne représente l'état des mémoires à la fin de chaque étape, l'élément encadré étant l'élément a_{j+1} de l'étape suivante, les flèches verticales symbolisant les transferts.

SI	1ère étape	2ème	3ème	4ème	5ème	6ème	7ème	8ème	9ème
7	4	4	0	0	0	0	0	0	0
(4)	7	7	4	2	2	2	2	2	1
9	(9)	9	7	4	4	4	4	3	2
0		(0)	9	7	6	5	5	4	3
2			(2)	9	7	6	6	5	4
6				(6)	9	7	7	6	5
5					(5)	9	8	7	6
8						(8)	9	8	7
3							(3)	9	8
1								(1)	9

c) Nombre de tests et nombres de transferts théoriques :

Soit SI la suite initiale des n éléments à trier. Soit, au j' passage, N_j le nombre de tests nécessaires pour placer le (j+1)° élément de SI, le nombre de transferts t_j correspondant est égal à N_j-1. Envisageons les différents cas qui peuvent se présenter (le nombre de passages dans SI est toujours égal à n-1).

- SI est une suite totalement rangée.

Nous avons, en appelant N le nombre de tests totaux, et T le nombre de transferts associés :

$N = n-1$

$T = (n-1) - (n-1) = 0$

- SI est une suite totalement rangée dans l'ordre inverse de tri.

Il faut 1 test pour trier le 2ème élément

2 tests pour trier le 3ème élément

⋮

n-1 tests pour trier le $n^{ème}$ élément

d'où $N = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ et $T = \frac{n^2 - n - 2(n-1)}{2} = \frac{n^2 - 3n + 2}{2}$

- SI est une suite quelconque de nombres :

Soit f(j) le nombre total de tests nécessaires pour trier tous les j ! arrangements possibles des j premiers éléments de SI. Pour classer a_{j+1}, il y a j+1 places possibles dans la sous-suite s'_{j+1} correspondant à $e'_1, e'_2 \ldots e'_{j+1}$. Pour chacun des j ! arrangements possibles, ces j+1 places possibles nécessitent respectivement j tests, j-1 tests, ... , 1 test. D'où le nombre total f(j+1) de tests correspondant aux j+1 possibilités de placement de a_{j+1} parmi les j ! arrangements possibles des j premiers éléments de SI.

$$f(j+1) = (j+1)\, f(j) + j! \sum_{i=1}^{n} i$$

$$= (j+1)\, f(j) + (j+1)! \times \left(\frac{j}{2}\right)$$

La fonction $f(j) = \frac{j!\,(j-1)\,j}{4}$ est solution de cette équation, et elle correspond aux j ! arrangements possibles. D'où le nombre de tests N_j correspondant à un seul d'entre eux

$$N_j = \frac{f(j)}{j!} = \frac{j(j-1)}{4}$$

D'où, pour n éléments à trier, nous obtenons le nombre de tests

$$N = \frac{n(n-1)}{4} \quad \text{et} \quad T = \frac{n^2 - 5n + 4}{4}$$

3°) Programme ALGOL

```
PROCEDURE Insertion (A,N) ; VALEUR N ; TABLEAU A ; ENTIER N ;
    DEBUT ENTIER I,J ; REEL T ;
        POUR J := 1 PAS 1 JUSQUA N-1 FAIRE
            DEBUT T := A[J+1] ;
                POUR I := J PAS -1 JUSQUA 1 FAIRE
                    DEBUT SI T < A[I] ALORS A[I+1] := A[I]
                        SINON DEBUT A[I+1] := T ;
                        ALLER A B FIN ;
                    FIN ; A[1] := T ;
        B : FIN ;
FIN de la procédure d'insertion ;
```

4°) Programme PASØ pour IBM 650

Le programme n'occupe que 49 mémoires.

5°) Résultats expérimentaux

J'ai obtenu les résultats expérimentaux suivants :

Temps de passage avec	128 Nb	256 Nb	512 Nb
SI 1	2' 55"	12 '	44 '
SI 2		20 "	2' 30"

Ces résultats nous montrent que la méthode d'insertion est sensible à la distribution, et que ses performances sont très nettement améliorées lorsque la suite SI est déjà "un peu arrangée". Ceci est en accord avec le calcul précédent du nombre de tests théoriques.

D'autre part, en comparant cette méthode à la méthode de Bubble, il n'existe pas dans le cas présent d'éléments a_ℓ (cf. au chapitre II, II §) tel que cet élément puisse imposer un nombre de passages $m \geq a_\ell - p$ (a_ℓ comme cela se passait avec cette méthode. La méthode d'insertion sera dans tous les cas meilleure que celle de Bubble.

III - METHODE d'INSERTION DICHOTOMIQUE

1°) Généralités

Cette méthode diffère de la précédente dans la façon de déterminer la place de a_{j+1} dans la sous-suite s_j. En effet, le nombre de tests nécessaires pour déterminer l'élément e_m (tel que $e_m \leq a_{j+1}$ et $e_{m+1} > a_{j+1}$) sera rendu plus faible si, au lieu de comparer a_{j+1} aux éléments de s_j dans l'ordre décroissant systématiquement, nous divisons s_j en intervalles partiels successifs de longueur sensiblement égale à $\frac{j}{2}$, $\frac{j}{4}$, ..., 1, et si nous déterminons à chaque fois dans quel intervalle a_{j+1} viendra se placer.

2°) Exposé de la méthode

Bornons nous dans cet exposé à donner un organigramme possible de la méthode.

j est l'indice de l'élément à insérer

i = j-1, j-2, ..., p

p et q sont les bornes de chaque intervalle partiel

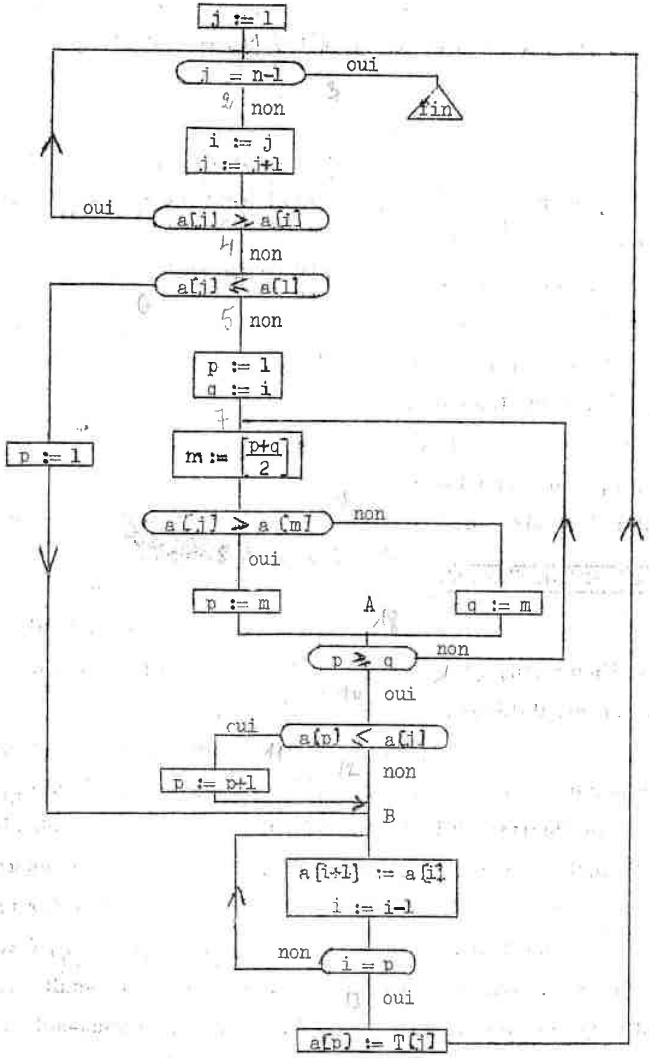
lorsque p = q l'intervalle ne comprend plus un seul seul élément :

il reste alors à déterminer si

$a[p] \leq a[j]$ si oui p=m

si non p=m+1

$\left[\frac{p+q}{2}\right]$ signifie partie entière de $\frac{p+q}{2}$



Nombre de tests et de transferts théoriques :

Le nombre de transferts est bien entendu le même dans cette méthode que dans la méthode d'insertion ordinaire, c'est à dire :

$$T = \frac{n^2 - n}{4}$$

Iverson ([1] page 236) a donné un nombre de tests égal à $n \log_2 n$. Malheureusement, chaque test fait intervenir un calcul d'indice compliqué $m = \left\lceil \frac{p+q}{2} \right\rceil$ ce qui vient beaucoup alourdir la méthode. D'autre part, il convient de remarquer que lorsque e_m est déterminé, il reste encore à effectuer le décalage de tous les e_i tels que $m+1 \leqslant i \leqslant j$. Ces transferts dans la méthode d'insertion ordinaire se font très facilement, juste après les tests, de sorte que les calculs d'indice correspondant sont très rapides.

Je n'ai pas personnellement essayé cette méthode, et personne, à ma connaissance, n'a obtenu de résultats très intéressants susceptibles de venir concurrencés ceux de la méthode de Hibbard (cf. chapitre IV).

IV - METHODE d'INSERTION et METHODE de SHELL

Nous avons supposé, dans les généralités sur les méthodes d'insertion, que la suite s était formée d'éléments consécutifs e_1, e_2, ... , e_j et que l'élément à insérer était a_{j+1}.

Nous pouvons concevoir une méthode dans laquelle les éléments sont e_{h1}, e_{h2}, ... , e_{hj}, l'élément à insérer étant e_{hj+1}, l'indice hj symbolisant "certains" indices particuliers et non le produit d'un indice h par un autre indice j et tel que $hk+f = hk+1$, f étant un entier constant.

Nous pourrons par cette méthode trier une première sous-suite issue de SI, et commençant en a_1, a_{1+f_1}, a_{1+2f_1}, f ayant une certaine valeur f_1, nous obtenons alors s_1^1, première sous-suite ordonnée, puis f ayant toujours cette même valeur f_1, une seconde sous-suite, commençant en a_2, ce qui nous donnera une seconde sous-suite ordonnée s_2^1 etc... c'est à dire que nous obtiendrons f_1 sous-suites ordonnées s_i^1 avec $1 \leqslant i \leqslant f_1$, tous les éléments a_i de SI ayant été examinés.

Nous pouvons alors recommencer le processus avec une nouvelle valeur de f, soit $f_2 < f_1$, pour obtenir f_2 sous-suites ordonnées s_i^2.

Le tri sera alors totalement terminé lorsque f sera égal à 1, la sous-suite ordonnée obtenue ayant n éléments.

Ceci est exactement le principe de la méthode de Shell dans laquelle nous avons f_1 = partière de$\left(\frac{n}{2}\right)$ et f_i = partie entière de $\left(\frac{f_{i-1}}{2}\right)$

Les résultats expérimentaux obtenus montrent que dans le cas d'une suite initiale SI de nombres répartis au hasard, la méthode de Shell est meilleure que la méthode d'insertion ordinaire.

CHAPITRE VI

METHODES UTILISANT DES PILES SIMPLES

I - GENERALITES

C. PAIR [7], [8] a introduit la notion de "pile simple" et montré qu'une pile simple fait passer d'une permutation donnée d'un ensemble fini E (permutation d'entrée) à une autre permutation de E (permutation de sortie). Soient P et S les ordres totaux de E définis respectivement par les permutations d'entrée et de sortie. Nous dirons que P et S sont l'ordre d'entrée et l'ordre de sortie de la pile.

On associe à la pile simple l'ordre partiel R de E tel que

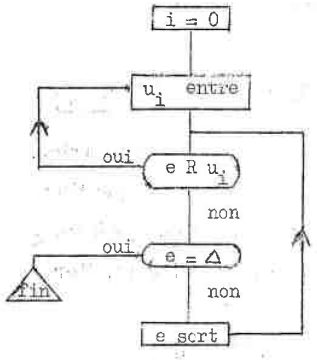
$\alpha R \beta \iff \alpha P \beta$ et $\beta S \alpha$

On déduit de cette définition :

$\alpha P \beta \iff \alpha R \beta$ ou ($\alpha P \beta$ et $\beta \overline{R} \alpha$) (1)

$\alpha S \beta \iff \beta R \alpha$ ou ($\alpha P \beta$ et $\alpha \overline{R} \beta$)

Le passage de la permutation d'entrée à la permutation de sortie est décrit par l'organigramme ci-dessous :



u_i est le premier élément susceptible d'entrer dans la pile

"u_i entre" veut dire que l'élément u_i entre dans la pile, puis que i augmente d'une unité

e est le sommet de la pile

$\Delta = u_o$ et il est tel que $\Delta R u_i$ quel que soit i

"e sort" veut dire que le sommet de la pile sort, puis que la hauteur de la pile décroit d'une unité.

Nous dirons que u est un R-successeur de e lorsque d'une part e R u, et, d'autre part e R v R u entraîne e=v ou v=u.

(1) $\overline{R}$ ...

Enfin, on montre :

Théorème 1 :

Pour qu'il existe une pile simple sur E dont un ordre total donné P soit un ordre d'entrée et un ordre donné R soit l'ordre associé, il faut et il suffit que pour tout α appartenant à E, $R(\alpha)$ [2] soit un P-intervalle de premier élément α.

Théorème 2 :

Pour qu'il existe une pile simple sur E dont un ordre total donné S soit un ordre de sortie et un ordre donné R soit l'ordre associé, il faut et il suffit que pour tout α appartenant à E, $R(\alpha)$ soit un S-intervalle de dernier élément α.

Il n'existe pas en général de pile simple faisant passer de SI, permutation d'entrée, à SR, permutation de sortie. Mais on peut :

1 - Soit s'arranger pour que SR soit la permutation de sortie. Alors, en général, les éléments n'entrent pas dans l'ordre où ils sont donnés dans SI ;

2 - Soit choisir SI pour permutation d'entrée. Alors, en général, SR ne sera pas la permutation de sortie, mais on montrera qu'on peut parvenir à SR en itérant le processus.

I - PREMIERE METHODE : l'ordre de tri est l'ordre de sortie S.

Soit une suite $u_1, u_2, \ldots, u_n$ d'éléments à trier.

Nous pouvons construire une pile telle que l'ordre de sortie S soit l'ordre de tri cherché, c'est à dire dans le cas présent $\leqslant$.

On démontrerait que dans ces conditions, l'ordre R est tel que $\alpha R \beta \Longleftrightarrow \alpha P \beta$ et $\beta S \alpha$ et

$(\forall \gamma \in]\beta, \alpha[_s, \alpha P \gamma)$

où la notation $]\beta, \alpha[_s$ désigne un intervalle pour l'ordre S.

L'ordre d'entrée associé n'est pas l'ordre P précédent, mais un autre ordre P_1. Il est tel que

$\alpha P_1 \beta \Longleftrightarrow \alpha R \beta$ ou $(\alpha S \beta$ et $\beta \overline{R} \alpha)$ ①

(2) $R(\alpha)$ est l'ensemble des α tels que $\alpha R \beta$

1°) Soit e le sommet de la pile : le problème est de savoir si e sort ou si un élément u entre, et lequel :

Or, u entre $\Longrightarrow$ e R u et u pas entré

$\Longrightarrow$ e P u et u S e et u pas encore entré

S'il n'existe pas un tel u, e sort.

2°) Supposons qu'un tel u entre : nous avons :

e P u et u S e et u pas encore entré.

Or, pour que e R u, il faut et il suffit que quel que soit $\gamma \in]u, e[_s$, e P γ.

Soit u S γ S e. Si e $\overline{P}$ γ alors γ P e,

mais γ P e et γ S e $\Longrightarrow$ e $\overline{R}$ γ $\Longrightarrow$ γ P_1 e d'après ①

γ entre avant e ; donc γ est sorti quand e est au sommet de la pile et par suite u est sorti car u S e ce qui est impossible car u n'est pas encore entré, d'où e R u $\Longleftrightarrow$ e P u et u S e et u pas encore entré.

Donc, s'il existe un u tel que

e P u et u S e et u pas encore entré, e ne sort pas. L'un des u entre : reste à savoir lequel.

3°) Ordre dans lequel entrent les successeurs de e :

Les successeurs de e ne sont pas comparables par R.

D'après ①, les successeurs de e entrent donc dans l'ordre S.

Soit β un successeur de e et α le premier des successeurs de e qui entre après β.

Montrons que $\beta P \alpha$; sinon $\alpha P \beta$ mais $\beta S \alpha$ et $\alpha R \beta$.

D'après la définition de R, il reste γ tel que

$\beta S \gamma S \alpha$ et $\gamma P \alpha P \beta \Longrightarrow \alpha \overline{R} \gamma$ et $\gamma R \beta$

De plus, e R β et e R α $\Longrightarrow$ e R γ

Il existe donc δ successeur de e, tel que e R δ R γ et $\delta \neq \alpha$

$\Longrightarrow \beta S \gamma S \delta S \alpha$

ce qui est impossible, car α par hypothèse est le premier successeur de e qui suit β dans l'ordre S.

Donc, $\beta P \alpha$, ce qui montre que les successeurs de e entrent dans l'ordre P.

4°) Soit u l'élément qui entre et λ tel que :

$e\,P\,\lambda$ et $\lambda\,S\,e$ et λ non entré

Nous avons vu en 2°) que $e\,R\,\lambda$; $e \neq \lambda$ car λ n'est pas entré.

Il existe δ successeur de e tel que $e\,R\,\delta\,R\,\lambda \Longrightarrow \delta P \lambda$

Or, si u, l'élément qui entre, est le premier des successeurs de e non entrés, dans l'ordre P, δ n'est pas entré, sinon il serait sorti et λ serait sorti aussi, donc $uP\delta$ et $uP\lambda$.

En résumé, l'élément u qui entre est le premier, dans l'ordre P, des λ tels que

$e\,P\,\lambda$ et $\lambda\,S\,e$ λ non entré.

5°) Organigramme :

Soit SI la suite initiale des nombres à trier formée des éléments a_i où i = 1,2,..., n. Nous l'encadrons par deux éléments d et f tels que :

$\left.\begin{array}{l} a_i\ S\ d \\ a_i\ \overline{R}\ f \end{array}\right\}$ quel que soit i $\qquad$ $d = a_o$ $\qquad$ $f = a_{n+1}$

Soient toujours :

e[j] sommet de la pile ;

a[i] élément susceptible d'entrer ;

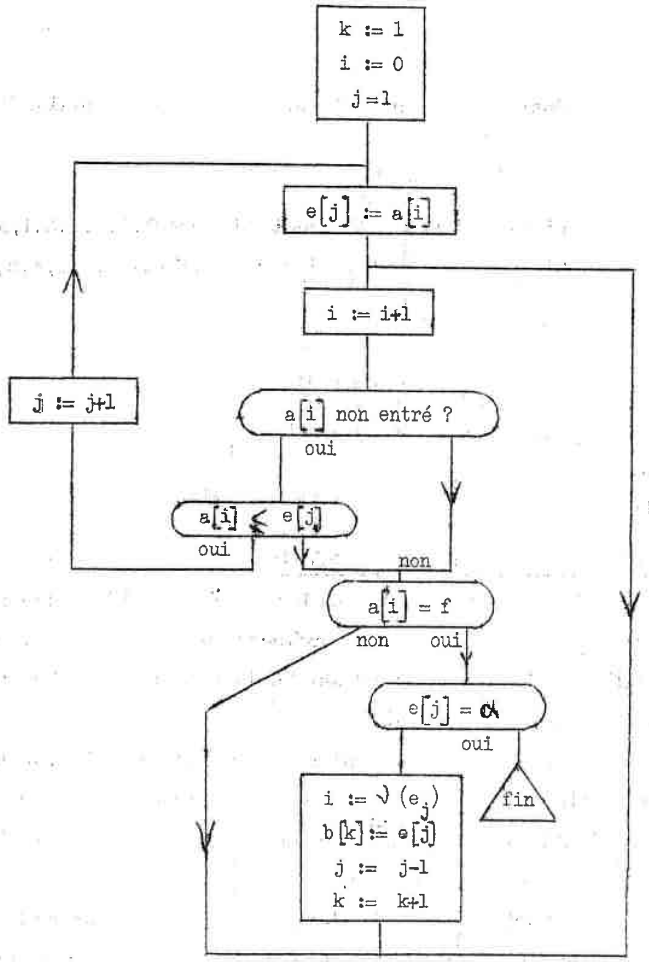
$\nu(e)$ indice de e dans la suite SI ;

Lorsque e[j] sort, il est transféré en b[k]

S la relation $\leqslant$

P est tel que $\nu(e) < i$

Ceci nous conduit à l'organigramme ci-contre.



Donnons un exemple : soit SI : 6,4,2,0,5,7,9,1,8,3, nous l'encadrons par D et F :

			0		1														
		2	2	2	2	2		3											
	4	4	4	4	4	4	4	4	4		5						8		
6	6	6	6	6	6	6	6	6	6	6	6	6		7		9	9	9	
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	I	D	D	D	D

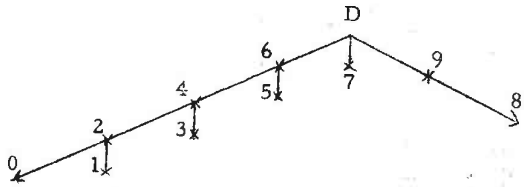
Evolution de la pile pendant le tri

Donnons également l'arbre de SI pour la relation R défini par :

$aRb \iff aPb$ et $b \leqslant a$

L'ordre d'entrée Pl dans la pile est D,6,4,2,0,1,3,5,7,9

L'ordre de sortie est l'ordre de tri 0,1,2,3,4,5,6,7,8,9



Nombres de tests et de transferts théoriques

Pour nous placer dans les mêmes conditions de calculs du nombre de tests théoriques que pour les méthodes précédentes (c'est à dire que nous négligeons les tests portant sur les indices) nous ne tiendrons pas compte des tests $a_i = f$ et $e = d$.

Par contre, en fonction du calculateur utilisé, nous sommes obligés de faire intervenir deux sortes de tests : les tests de la forme "a_i non entré" que nous noterons N, et les tests sur les a_i eux-mêmes de la forme "$a_i \leqslant e$" que nous noterons N'.

a) Soit une suite totalement rangée dans l'ordre inverse de tri ($a_i \geq a_{i+1}$ quel que soit i) et n le nombre d'éléments à trier ; au premier passage, tous les a_i rentrent dans la pile qui atteint alors la hauteur n+1.

Nous avons alors

$$N'_a = n$$
$$N_a = \sum_{i=1}^{n-1} = \frac{n(n-1)}{2}$$

b) Soit une suite totalement rangée dans l'ordre de tri ; la pile à chaque étape ne comporte qu'un seul étage (un seul élément entre, puis sort avant qu'un autre ne rentre dans la pile)

Nous avons alors

$$N'_b = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$
$$N_b = N'_b$$

c) Soit une suite SI d'éléments répartis au hasard :

- nombre de tests "a_i est-il entré ?" : chaque élément α quand il est au sommet de la pile, est comparée dans le test "a_i est-il entré ?" une fois et une seule à tous les éléments qui le suivent puisque si β vient au dessus de α dans la pile nous repartons, après la sortie de β, à l'élément qui suit β dans l'ordre P. Le nombre de tests N_c "a_i est-il entré ?" est donc :

$$N_e = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

- nombre de tests "a_iSe" : il est égal au nombre de tests "a_i est-il entré ? " qui répondent non.

Le nombre de tests "a_i Se" qui répondent oui est évidemment égal au nombre d'entrées dans la pile, c'est à dire n.

Le nombre de tests "a_i Se" qui répondent non correspond aux couples d'éléments tels que ePa_i et eSa_i. D'où le nombre de tests "a_i Se" à réponse non est égal au nombre de combinaisons (a_i, e) tels que ePa_i et eSa_i.

Calculons le nombre moyen de combinaisons tels que ePa_i et eSa_i : supposons tous les ordres P "également possibles". Il y a autant d'ordres P tels que ePa_i et eSa_i que d'ordres tels que ePa_i et a_iSe.

La probabilité pour un couple donné d'être dans P ordonné dans l'ordre S est $\frac{1}{2}$, d'où le nombre moyen de tests a_iSe à réponse non est égal à $\frac{n(n-1)}{4}$.

Le nombre moyen total de tests "a_iSe" est donc :

$$N'_e = \frac{n(n-1)}{4} + n = \frac{n(n+3)}{4}$$

Dans tous les cas, le nombre de transferts est constant et égal à 2n (n transferts pour transférer les éléments dans la pile et n autres pour les en sortir).

Longueur de la zone de travail :

Nous avons besoin de conserver intégralement tous les a_i. Comme, d'autre part, la pile est susceptible d'avoir une hauteur égale à n+1, il nous faut une zone de travail de n+1+2 = n+3 mémoires en plus des

n mémoires contenant les a_i. Pour construire la suite résultat SR (lorsque e sort), il nous faudrait en plus n mémoires, mais celles-ci sont obtenues en imbriquant la zone réservée à la pile et la zone de rangement B lorsque les e sortent (l'extrémité de l'une prise avec un indice croissant, étant égale à l'origine de l'autre, prise avec un indice décroissant).

Programme PASØ pour IBM 650

Cf. au programme n° 4

Nous avons placé les éléments a_i à trier de u0002 à u000N+1. u0001 = D, u000N+2 = F.

Le fait d'imbriquer la zone réservée à la pile par numéro de mémoires croissant et la zone de sortie des nombres triés par numéro de mémoires décroissant nous donne en fait une suite SR rangée dans l'ordre inverse de tri.

Le programme ainsi conçu utilise 116 mémoires.

Résultats expérimentaux

Pour réaliser le tes "a_i est-il entré ?", j'ai utilisé sur le calculateur IBM 650 du Centre de Calcul de Nancy, un test SD0, c'est à dire que ce test donne également deux sorties "oui, non" mais il est plus rapide qu'un test de la forme $u_i \leqslant e$.

Voici les résultats que j'ai obtenus :

	256 Nb	512 Nb
avec SII	22'30"	88'

II - SECONDE METHODE : l'ordre de tri est l'ordre d'entrée P dans la pile

Construisons une pile dans laquelle l'ordre d'entrée est l'ordre P. On démontre qu'il faut alors choisir R de telle sorte que :

$$\alpha R \beta \Longleftrightarrow \alpha P \beta \quad \text{et} \quad \beta S \alpha \quad \text{et}$$
$$(\forall \gamma \in]\alpha, \beta[_P, \gamma S \alpha)$$

où $]\alpha, \beta[_P$ désigne un intervalle pour l'ordre P.

L'ordre de sortie n'est plus S, mais un ordre S_1 tel que :

$$\alpha S_1 \beta \Longleftrightarrow \beta R \alpha \text{ ou } (\alpha P \beta \text{ et } \alpha \overline{R} \beta) \quad (2)$$

1°) Soit un couple déjà rangé à l'entrée dans l'ordre S, c'est à dire tel que $\alpha P \beta$ et $\alpha S \beta$.

Le problème est de savoir s'il le restera dans l'ordre S_1 à la sortie ? La réponse est oui : en effet :

$$\alpha P \beta \text{ et } \alpha S \beta \Longrightarrow \alpha \overline{R} \beta .$$

Comme $\alpha P \beta \Longrightarrow \alpha S_1 \beta$ d'après (2).

2°) A quelles conditions un couple non encore rangé peut-il le devenir à la sortie ?

Nous avons donc $\alpha P \beta$ et $\beta S \alpha$.

Or $\beta S_1 \alpha \Longleftrightarrow \alpha R \beta$ ou ($\beta P \alpha$ et $\beta \overline{R} \alpha$)

$\Longleftrightarrow \alpha R \beta$ car $\alpha P \beta$

mais $\alpha R \beta \Longleftrightarrow \alpha P \beta$ et $\beta S \alpha$ et ($\forall \gamma \in]\alpha, \beta[_P, \gamma S \alpha$)

Pour qu'à la sortie le couple (α, β) devienne bien rangé dans l'ordre S_1, il faut et il suffit qu'aucun des éléments γ de l'intervalle $]\alpha, \beta[_P$ ne soit bien rangé avec α à l'entrée.

Autrement dit, si γ est le premier élément bien rangé avec α suivant α dans l'ordre P, tous les éléments compris entre α et γ seront bien rangés avec α dans l'ordre S_1.

Les conséquences sont nombreuses :

a) Le dernier élément δ de E pour S sera aussi le dernier pour l'ordre S_1. En effet,

- si $\alpha P \delta$, α et δ restent bien rangés dans l'ordre S_1
- si $\delta P \beta$ entre δ et β aucun élément n'est bien rangé avec δ donc $\beta S_1 \delta$.

b) Si δ est déjà le dernier élément dans l'ordre P, il le restera. Mais alors, si δ' est l'avant dernier élément de E pour S, il restera bien rangé avec δ et le deviendra par rapport à tous les α tels que $\alpha P \delta'$ et à tous les β tels que $\delta' P \beta$.

c) Plus généralement, si les q derniers éléments de E, pour l'ordre S, sont déjà bien rangés, c'est le (q-1)° qui sera rangé.

Nous voyons donc qu'il y a toujours au moins un élément de plus qui vient se placer au bon endroit en queue de liste. Par conséquent, si nous reprenons l'ensemble E, avec pour nouvel ordre P, l'ordre S_1 précédent, nous obtenons un nouvel ordre de sortie S_2 etc... et il est clair qu'au bout d'un nombre fini d'itérations (au plus égal à n, si E comporte n éléments) tous les éléments de E seront bien rangés dans l'ordre S.

3°) Test d'entrée dans la pile :

Soient e le sommet de la pile, et u le premier élément de E susceptible d'y entrer :

$e\ R\ u \Longrightarrow u\ S\ e$

Réciproquement, supposons $u\ S\ e$; or, $e\ P\ u$ puisque e est entré avant u ; soit alors $e\ P\ \gamma$ montrons que $\gamma\ S\ e$:

γ est entré puisque e est encore dans la pile, mais comme e est le sommet de la pile, γ est sorti, d'où $\gamma\ S\ e$.

Donc, le test $e\ R\ u$ est équivalent à $u\ S\ e$.

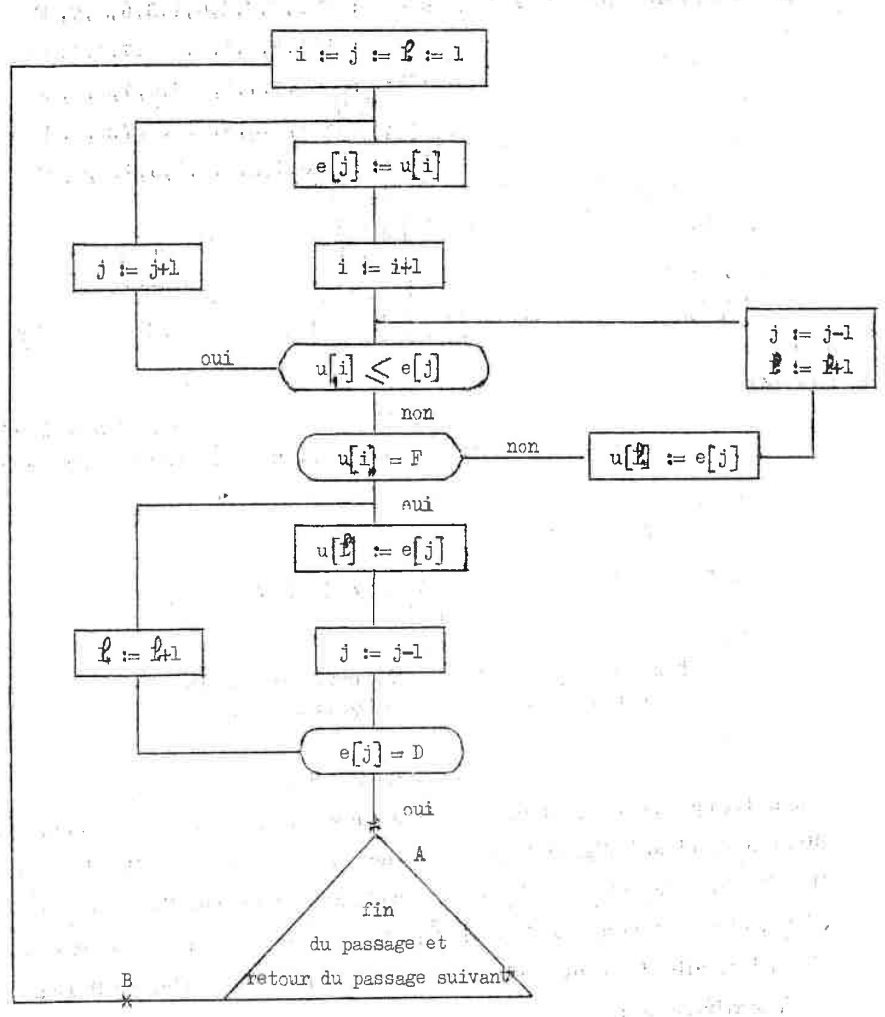
4°) Organigramme :

Donnons pour l'instant l'organigramme qui permet le passage d'un ordre P_i au suivant P_{i+1}.

L'ensemble E est composé des éléments $u[i]$, la pile est désignée par $e[j]$, à leur sortie de la pile, les éléments $e[j]$ sont transférés en $u[\ell]$; ils ne peuvent pas venir se substituer à des $u[i]$ non encore examinés, car nous avons :

$$i \geq j + \ell$$

(La suite est toujours encadrée par $u[1] = D$ et $u[N+2] = F$.



Il nous manque pour l'instant un test de fin : voyons sur un exemple quelle forme nous allons lui donner.

Soit SI la suite des nombres à trier, telle que

7, 5, 4, 9, 1, 3, 0, 2, 8, 6.

Nous l'encadrons par D et F, tels que $F \geq D$, et

$\forall \alpha$, $D > \alpha$ et $D\ R\ \alpha$ et $\alpha\ \bar{R}\ F$.

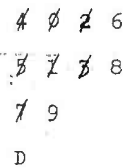
Etat de la suite après le 1er passage : D 4,5,7,1,0,2,3,6,8,9, F

2ème : D 4,5,0,1,2,3,6,7,8,9, F

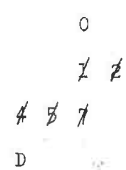
3ème : D 4,0,1,2,3,5,6,7,8,9, F

4ème : D 0,1,2,3,4,5,6,7,8,9, F

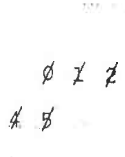
5ème : D 0,1,2,3,4,5,6,7,8,9, F



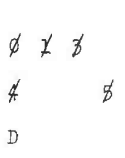
Evolution de la pile au cours du 1er passage



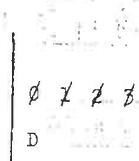
Evolution de la pile au cours du 2ème passage



Evolution de la pile au cours de 3ème passage

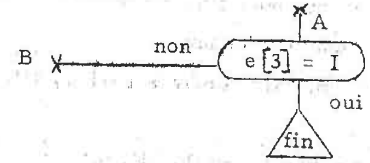


Evolution de la pile au cours du 4ème passage



Evolution de la pile au cours du 5ème passage

Nous voyons donc que l'ensemble E sera totalement ordonné lorsque tous ses éléments seront entrés dans la pile, puis sortis immédiatement, c'est à dire qu'à aucun moment la hauteur de la pile n'excède deux : il suffit donc de mettre un indicatif I quelconque au début d'un passage dans e[3], c'est à dire après i := j := ℓ := 1, et de garder à la fin de ce passage si cet indicatif s'y trouve encore : donnons la transformation de l'organigramme à partir de A :



5°) Reprenons plus en détail l'étude du paragraphe 2

Nous constatons que, moins il y a de couples bien rangés au début d'un passage, et plus il y en aura de bien rangés à l'issue de ce passage, de sorte qu'au passage suivant, nous rangerons moins de couples que pendant ce même passage.

Il va donc se produire un freinage d'autant plus efficace qu'il y a plus de couples rangés.

Pour éviter cet inconvénient, nous pouvons repartir à chaque fois dans l'ordre inverse de sortie, c'est à dire que P_i (ordre P au i° passage) est tel que

$$P_i = \left[S_{i-1}\right]^{-1} \quad \text{ou encore} \quad \left[P_i\right]^{-1} = S_{i-1}$$

Montrons qu'ici encore, si les q S-derniers éléments de E sont déjà à leur place dans l'ordre $\left[P_i\right]^{-1}$, il le seront également dans l'ordre S .

ordre P_i ←

$\ldots d \ldots \delta_q \ldots \delta_2 \delta_1$

→ ordre $\left[P_i\right]^{-1}$

Soient $\delta_1 \ldots \delta_q$ ses éléments, nous avons :

$\delta_q S \ldots S \delta_2 S \delta_1$

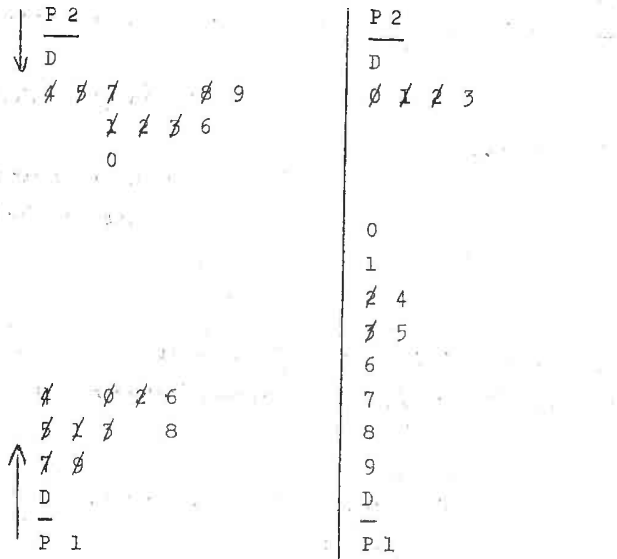
Pour $\left[P_i\right]^{-1}$ tout élément δ_j est mal rangé avec tout ordre, donc, d'après les résultats du paragraphe 2°) il deviendra bien rangé avec tout autre à la sortie S. , et avec le précédent δ_{q+1} également. Le processus convergera donc également, et permettra de ranger tous les éléments au bout d'un nombre fini d'itérations.

Longueur de la zone de travail

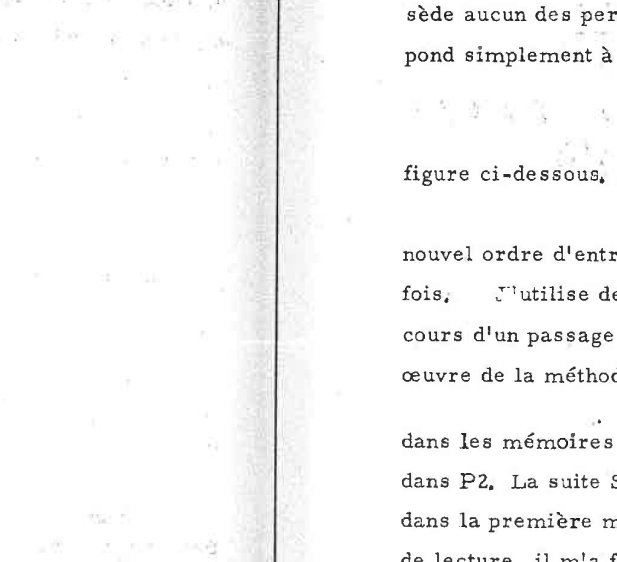
Pour mettre en oeuvre cette méthode, il nous faut réserver n+3 mémoires de travail pour ranger n éléments : n+1 mémoires pour la pile elle-même, et 2 mémoires pour encadrer SI par D et F.

D'autre part, afin d'économiser des transferts, nous avons été conduits à envisager une seconde pile P2 imbriquée dans la première, son extrémité coïncidant avec le second étage de la pile P1. Elle est telle que l'ordre de sortie de la première pile P1 est l'ordre d'entrée dans la seconde pile P2.

Exemple : Soit SI : 7,5,4,9,1,3,0,2,8,6 que nous encadrons par D et F.



Evolution de P1 et P2 au cours du 1er passage



Evolution de P1 et P2 au cours du 2ème passage

Etat de la suite s : initialement $\xrightarrow{P}$ D,7,5,4,9,1,3,0,2,8,6F

: après le 1er passage $\xleftarrow{P}$ D,4,5,0,1,2,3,6,7,8,9F

: après le 2ème passage $\xrightarrow{P}$ D,9,8,7,6,5,4,3,2,1,0F

Test d'élimination des éléments déjà rangés :

Tous les éléments u_i qui, à partir du dernier élément sorti de P2, sont entrés dans P2, puis sortis immédiatement, sont bien rangés.

Ces éléments sont tous supérieurs au dernier élément v entré dans le 3ème étage de P2, et le test $u_i > v$ permet donc de les éliminer.

Programme PASØ

Afin de tenir compte de tous les résultats de l'étude précédente, j'ai été amené à écrire plusieurs programmes, dont voici les principaux :

a) Le programme n° 5 (dénommé P2) et qui ne possède aucun des perfectionnements mentionnés dans cette étude. Il correspond simplement à l'organigramme du paragraphe 4°).

Ce programme occupe 95 mémoires.

b) Le programme n° 5-bis, dont l'organigramme figure ci-dessous. Il occupe 177 mémoires.

Dans ce programme, l'ordre de sortie S_{i-1} est le nouvel ordre d'entrée P_i ; nous inversons donc le sens de lecture à chaque fois. J'utilise deux piles P1 et P2. Les $u_i > v$ ne sont éliminés qu'au cours d'un passage sur deux pour des raisons de commodité de mise en œuvre de la méthode.

A leur sortie de P2, les éléments sont transférés dans les mémoires libérées par les u_i qui sont entrés, soit dans P1, soit dans P2. La suite SI est encadrée par deux éléments D et F (définis comme dans la première méthode), mais, comme j'inverse à chaque fois le sens de lecture, il m'a fallu prendre D = F.

Donnons un exemple :

SI $\longrightarrow$ F,6,4,2,0,5,7,2,9,1,8,3,F

Etat de SI à l'issue du 1er passage $\longleftarrow$ F,0,2,4,5,2,6,1,3,F,7,8,9

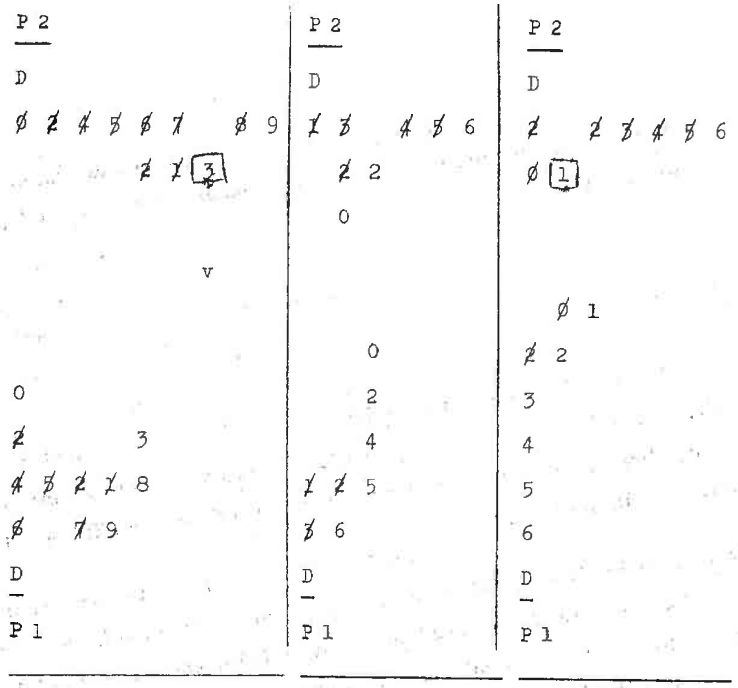
2ème $\longrightarrow$ F,6,5,4,3,2,2,0,1,F

3ème $\longleftarrow$ F,0,1,F,2,2,3,4,5,6

4ème $\longrightarrow$ F,1,0,F,

5ème $\longleftarrow$ FF,0,1,

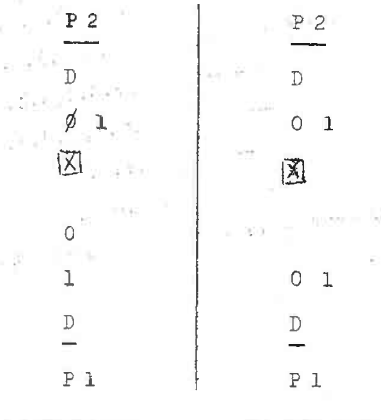
NB : La flèche du j° passage représente le sens de lecture des éléments pour le passage suivant.



Evolution des 2 piles au cours du 1er passage

2ème passage

3ème passage



4ème passage

5ème passage

L'organigramme est le suivant :

u_i est l'élément susceptible d'entrer dans P1

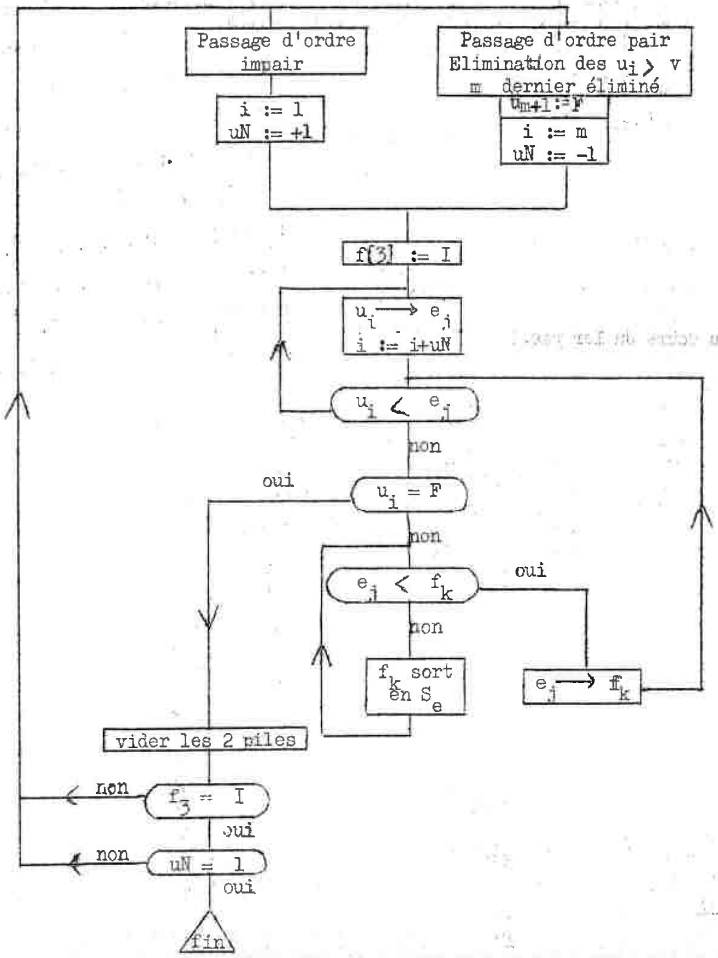
e_j est le sommet de P 1

f_k est le sommet de P 2

$u_i \longrightarrow e_j$ veut dire : u_i vient au sommet de P1, la hauteur de P1 augmente d'une unité

$e_j \longrightarrow f_k$ veut dire : e_j vient en f_k, la hauteur de P1 diminue d'une unité, celle de P2 augmente d'une unité

"f_k sort en S_e" f_k est transféré en S_e et la hauteur de P2 diminue d'une unité



c) Le programme n° 5-ter et qui occupe 223 mémoires :

J'ai encore utilisé les deux piles P1 et P2. Le sens de lecture est inversé à chaque passage, et les u_i sont éliminés à chaque passage également. Pour obtenir ce résultat, j'ai été obligé de déplacer à chaque passage, soit le symbole F supérieur (à la fin des passages impairs), soit le symbole F inférieur (à la fin de chaque passage pair).

Nous voyons alors qu'il n'est plus possible de transférer les u_i définitivement triés dans la zone qui initialement contenait SI, car nous obtiendrions plusieurs suites rangées en ordre croissant ou décroissant, et séparées les unes des autres par le symbole F. Nous avons choisi de transférer ces éléments dans la zone occupée par la première pile, l'origine de cette pile étant alors décalée à chaque passage.

Reprenons l'exemple donné en b) :

SI : ⟶ F,6,4,2,0,5,7,2,9,1,8,3,F

Etat de SI à l'issue du 1er passage : ⟵ F,0,2,4,5,2,6,1,3,F,89

2ème : ⟶ F,6,5,4,F2,2,0,1F

3ème : ⟵ F0,1,2,2,F

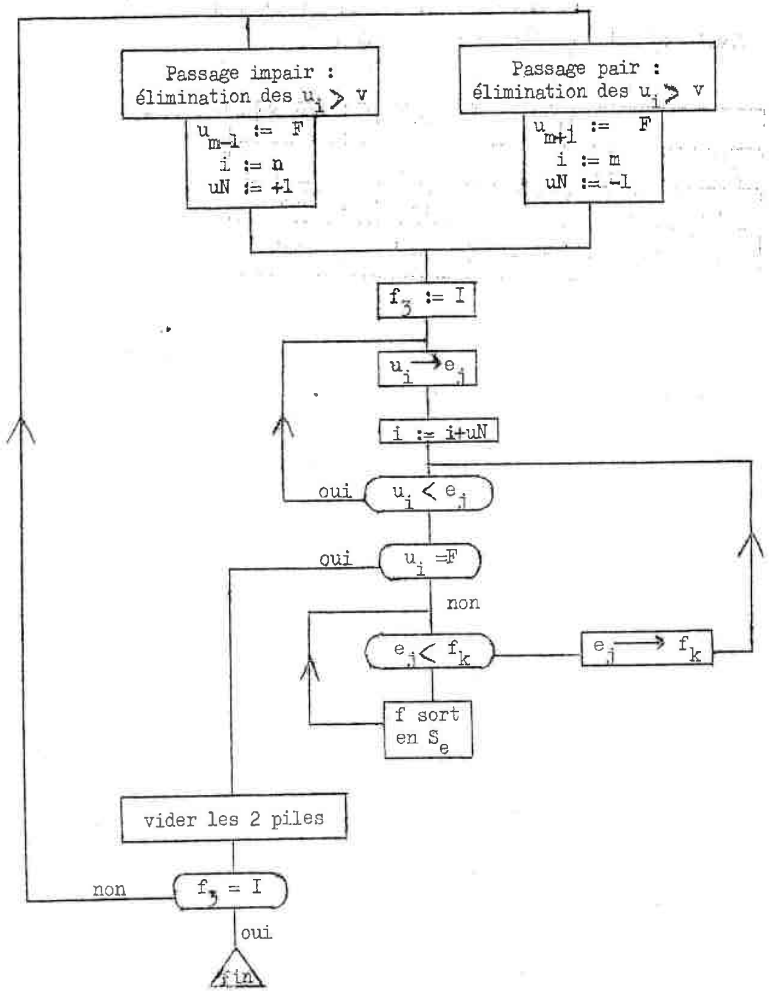
Evolution des piles	2ème passage	3ème pass.	
P 2	P 2	P 2	
D 2	D 2	D 2	
∅ ∄ ∄ 5 6 7 8 9	∄ ∄ ∄ ∄ 6	∅ ∄ ∄ 2	
2 1 [3]	∄ [2]	[x]	
↑ v	0 ↑ v	↑ v	
		∅ 1	0
		2	1
	0	2	2
	2	D1	2
	4	3	3
	∄ ∄ 5	4	4
0	∄ 6	5	5
∄ ∄	D1	6	6
∄ ∄ ∄ ∄ 8	7	7	7
6 7 9	8	8	8
D1	9	9	9
P1	P1	P1	P1

L'organigramme est le suivant :

n = indice du dernier élément éliminé au cours d'un passage impair

m = indice du dernier élément éliminé au cours du passage pair

Les autres notations sont les mêmes que dans l'organigramme précédent.



Résultats expérimentaux :

Avec SI 1, j'ai obtenu :

	256 Nb	512 Nb
Programme n° 5	65'	
Programme n° 5-bis	20'45"	79'
Programme n° 5-ter	20'	75'

CHAPITRE VII

COMPARAISON DES METHODES

La comparaison peut porter par exemple :

- sur le nombre total de tests
- sur le nombre total de transferts
- sur le nombre de mémoires auxiliaires nécessaires
- sur la complexité des calculs d'indice

La suite SI étant donnée, il s'agit de déterminer la méthode qui nous donnera la suite SR dans le minimum de temps et en tenant compte, évidemment, du nombre n d'éléments à trier et des particularités éventuelles de SI.

Nous pourrions penser qu'il suffit de choisir la méthode qui nous a donné les meilleurs résultats expérimentaux. En fait, il n'en est rien. En effet, le temps $\mathcal{C}(n)$ nécessaire pour trier les n éléments dépend d'une part du nombre total C(n) de tests à effectuer sur les éléments a_i, et du nombre total t(n) de transferts.

Nous avons : $\mathcal{C}(n) = a\,C(n) + b\,t(n)$

a et b étant des constantes, fonction du calculateur. La connaissance de C(n) et t(n) est donc du plus grand intérêt.

D'autre part, il nous faut tenir compte du nombre n d'éléments à trier. En effet, soit N le nombre de mémoires disponibles dans le calculateur, et soit N1 le nombre de mémoires nécessaire à l'utilisation d'une méthode donnée pour trier les n nombres.

Supposons que, dans une première analyse, ce soit la méthode que nous appellerons (1) qui est la plus rapide. Et que, d'autre part, cette méthode nécessite N1 mémoires, mais que N1 soit supérieur à N. Pour trier SI totalement, il nous faudra alors avoir recours à des organes annexes de stockage, ce qui risque d'augmenter considérablement le temps de tri. Il sera alors peut être possible de trouver une méthode (2) à priori moins rapide que la méthode (1) mais nécessitant un nombre N2 de mémoires tel que $N2 < N$ et il est possible que dans ces conditions, le temps de tri total obtenu soit inférieur à celui de la méthode (1).

Avantages et inconvénients de chaque méthode

1°) Méthode de Bubble

La méthode est simple et n'utilise pas de mémoires auxiliaires de travail.

Malheureusement, elle est très lente, par suite du nombre élevé de tests à effectuer pour trier SI. La rapidité est très améloirée lorsque SI est arrangée, mais l'influence de l'élément a_1, dont nous avons parlé au cours de l'exposé sur la méthode, ne permet pas à cette amélioration d'être vraiment importante. La méthode d'insertion est supérieure à cette méthode, quelle que soit la suite initiale SI.

2°) Méthode de Shell

Cette méthode est relativement rapide et n'utilise aucune mémoire de travail. De plus, ces performances sont très nettement améliorées lorsque la suite SI est déjà plus ou moins triée. La méthode d'inter-classement et la méthode de Thomas N Hibbard sont plus rapides, mais font usage de mémoires auxiliaires. Si le nombre N de mémoires disponibles sur le calculateur est tel qu'il permet de trier les n éléments de SI par la méthode de Shell sans avoir à faire appel aux organes annexes de stockage, alors que les deux autres méthodes y auraient recours, la méthode de Shell peut devenir très intéressante.

3°) Méthode déduite de celle de Von Neumann

Je ne la cite que pour mémoire, car elle présente peu d'intérêt.

4°) Méthode de sélection linéaire

Elle est très simple, et n'utilise pas de mémoires auxiliaires. Néanmoins, elle est tellement lente, que son emploi ne présente aucun intérêt.

5°) Méthode de Hibbard

Dans le cas d'une suite d'éléments répartis au hasard, c'est une des meilleures méthodes disponibles actuellement, en dépit de ses deux défauts : d'une part elle utilise $\log_2 n$ mémoires auxiliaires de travail, d'autre part, et cela est plus grave, elle est sensible à la distribution des éléments dans la suite SI. Ainsi, pour une suite SI totalement rangée initialement, le temps de tri est considérablement augmenté, à tel point que la méthode perd tout intérêt.

6°) Méthode d'insertion

Si la suite SI est déjà plus ou moins rangée, cette méthode est vraiment rapide. D'autre part, elle n'utilise aucune mémoire auxiliaire de travail, de sorte que chaque fois qu'on aura des raisons de penser que la suite est un peu rangée, cette méthode sera intéressante.

7°) Méthode d'insertion dichotomique

Tous les résultats que nous venons d'énoncer à propos de la méthode d'insertion ordinaire sont valables dans le cas présent. Le nombre de tests théoriques est inférieur à ceux de la méthode précédente, et elle devrait donc à priori être meilleure. En fait, ce n'est pas certain, car les calculs d'indices sont longs et compliqués, car ils font intervenir l'opérateur

$$\left[\frac{i+j}{2}\right]$$

8°) Méthode d'inter-classement de Von Neumann

Elle est au moins aussi rapide que la méthode de Monsieur Hibbard, et, de plus, contrairement à celle-ci, ses performances sont améliorées lorsque la suite SI est déjà un peu arrangée. Toutefois, elle présente l'inconvénient de nécessiter n mémoires auxiliaires pour trier n nombres, de sorte que, pour n suffisamment grand, il faudra avoir recours aux organes annexes de stockage, et cela ralentit beaucoup le processus de tri.

9°) Méthode utilisant une pile simple, dans laquelle l'ordre de sortie est l'ordre de tri

Cette méthode est lente et utilise n mémoires auxiliaires. Le temps de tri diminue très peu lorsque la suite est plus ou moins arrangée, de sorte que l' utilisation de cette méthode est à déconseiller.

10°) Méthode utilisant une pile simple dans laquelle l'ordre d'entrée est l'ordre de tri

Elle est beaucoup plus intéressante que la précédente, bien qu'elle utilise également n mémoires auxiliaires. En effet, ses résultats sont bien meilleurs lorsque la suite SI est déjà un peu arrangée. De plus, elle permet d'obtenir de nombreuses sous-suites ordonnées, et les résultats pourront être très bons en combinant cette méthode avec une méthode d'inter classement telle que celle exposée au chapitre IV § III.

Néanmoisn le nombre important de mémoires occupées par le programme correspondant (c'est le programme le plus volumineux de tous ceux qui ont été essayés), est un obstacle à une telle conception de l'usage de cette méthode, car le nombre de mémoires utilisables se trouve très réduit.

Tous ces résultats se trouvent résumés dans le tableau ci-joint.

En conclusion, nous pouvons dire que la méthode d'inter classement de Von Neumann sera intéressante chaque fois que le nombre de mémoires disponibles sera suffisant. Si ce n'est pas le cas, nous pourrons alors envisager l'utilisation de la méthode de T. N. Hibbard, ou, à la rigueur, celle de Shell. Finalement, lorsque nous aurons de sérieuses raisons de penser que la suite initiale est plus ou moins triée, nous pourrons avoir recours à la méthode d'insertion.

Programme n° 1 : Méthode de Bubble

```
BUBLE   TDI   BUBSO
        AAD   CTES2
        TRD   BTES2   BB1
BB1     EDI   BTEST
        TDI   BAI
        EDI   CORD1
        TDI   BORD1
        EDI   CORD2
        TDI   BORD2
        EDI   CORD3
        TDI   BORD3
        EDI   CORD4
        TDI   BORD4   BORD1
TAI1    TDI   PAI1    BORD2
BTES1   ANG           BCOMU
        TDI   BAI     BORD4
BSUI4   EDI   PAI1    BORD3
BCOMU   RAD   BORD2
        AAD   1COL5
        TRD   BORD2
        EDI   BORD3
        SBF   BORD3
        RAD   BORD1
        AAD   1COL5
        TRD   BORD1
        EDI   BORD4
        SBF   BORD4
        SAD   BTES2
        ANG   BORD1
        RAD   BAI
        SAD   BTEST
        ANN           BUBSO
        RAD   BTES2
        SAD   1COL5
        TRD   BTES2   BB1
CTES2   RAD   A0001   TAI1
1COL5    00    0001    000
CORD4   TDI   A0002   BSUI4
CORD3   TDI   A0001   BCOMU
CORD2   SAD   A0001   BTES1
BTEST    99    9999    9999
CORD1   RAD   A0002   TAI1
```

Programme n° 2 : Méthode de Shell

```
SHELL   TDI   SHLS0
        TRD   MEM
        TDI   MEN      SHEL1
SHEL1   RAD   MEM
        DRR   2COL5
        DAG     0004
        TRD   MEM
        ANN           SHLS0
        RSD     8002
        AAD   MEN
        TRD   MEK
        EDI   1COL5
        TDI   MEJ      IJ
IJ      RAD   MEJ
        AAD   AO
        EDI   CSAI
        SBF   SAI
        EDI   CTAI
        SBF   TAI
        AAD   MEM
        EDI   CRAIM
        SBF   RAIM
        EDI   CTAIM
        SBF   TAIM     RAIM
TRANS   TDI   AIM      SAI
TEST    ANG   TAIM     JPLU1
SUI1    EDI   AIM      TAI
SUI2    RAD   SAI
        SAD   MEM
        SAD   CSAI
        ANG   JPLU1
        AAD   CSAI
        TRD   SAI
        EDI   TAI
        SBF   TAI
        AAD   MEM
        EDI   RAIM
        SBF   RAIM
        EDI   TAIM
        SBF   TAIM     RAIM
CTAIM   TDI     0000   SUI1
CTAI    TDI     0000   SUI2
AO        00    A0000
1COL5     00      0001   000
2COL5     00      0002   000
CSAI    SAD   A0001   TEST
CRAIM   RAD   A0001   TRANS
JPLU1   RAD   MEJ
        AAD   1COL5
        TRD   MEJ
        RSD     8002
        AAD   MEK
        ANG   SHEL1    IJ
```


Programme n° 3 : Méthode d'insertion

```
INSER   TDI   ISORT
        TRG   NINSR
        AAG   IC1
        TDI   ORD1
        SAG   UN
        TRG   TESTI    B1
B1      RAD   ORD1
        EDI   IC2
        SBF   ORD2
        AAD   UN
        TRD   ORD1
        EDI   IC3
        SBF   ORD3
        RAD   TESTI
        SAD   ORD1
        ANG   ISORT    ORD1
IIS1    TDI   AI       ORD2
IIS2    ANG   ORD3
        RAD   IC4
        EDI   ORD3
        SBI   ORD4
        EDI   AI       ORD4
TEST3   RAD   ORD2
        SAD   J2
        ANG   OUI
        RAD   ORD2
        EDI   ORD3
        SBF   ORD3
        SAD   UN
        TRD   ORD2
        RAD   AI       ORD2
OUI     EDI   AI
        TDI   A0001    B1
J2      SAD   A0002    IIS2
UN        00    0001     00
IC1     RAD   A0001    IIS1
IC2     SAD     0000   IIS2
IC3     TDI     0000   TEST3
IC4       00      000  B1
```

Programme n° 4 : Première méthode des piles : ordre de tri = ordre de sortie de la pile.

```
P1      TDI  P1SOR
        TRG  CNK
        AAG  CSTF
        TRG  TRANF
        RAD  C8
        TDI  ORD8
        AAD  CNK
        TRD  CI8
        EDI  C9
        TDI  ORD9    INIT
INIT    RAD  CI8
        SAD  ORD8
        ANG  DEBUT   ORD8
I8      AAD  8       ORD9
I9      RAD  ORD8
        AAD  UN
        TRD  ORD8
        EDI  ORD9
        SBF  ORD9    INIT
DEBUT   EDI  F       TRANF
IF      EDI  D
        TDI  U0001
        RAD  CNK
        AAD  C7
        TRD  ORD7
        EDI  C1
        TDI  ORD1
        EDI  C2
        TDI  ORD2
        EDI  C3
        TDI  ORD3
        EDI  ED
        TDI  NENT
        EDI  U0001
        TDI  UI      B1
B1      EDI  ORD3    ORD2
I2      RAD  UI
        AAD  10      ORD3
B2      RAD  ORD3
        AAD  UN
        TRD  ORD3
        EDI  C5
        SBF  ORD5    ORD5
I5      SDO  HUIT    ENTRE
HUIT    TDI  UI      NENT
SUIT    EDI  C4
        SBF  ORD4     8001
```



```
I4      AAD  10
        AAD  UI
        ANG          B2
        RAD  ORD2
        AAD  UN
        TRD  ORD2
        EDI  NENT
        SBF  NENT    B1
ENTRE   SAD  F
        ANN  P2
        RAD  NENT
        SAD  ED
        ANN          P1SOR
        RAD  C10
        EDI  NENT
        SBI  ORD11    8001
I10     EDI  C11
        SBF  ORD11    8001
SUIT1   TDI  P0001
        PFO  P0001   SUIT2
SUIT2   RAD  ORD7
        SAD  UN
        TRD  ORD7
        RAD  ORD2
        SAD  UN
        TRD  ORD2
        EDI  NENT
        SBF  NENT
        RAD  ORD11
        AAD  UN
        EDI  ORD3
        SBF  ORD3
        EDI  ORD5
        SBF  ORD5     8001
C2      TDI  E0001   I2
C3      TRD  U0001   B2
C4      RSD   0000   I4
C5      RAD   0000   I5
C7      TDI  E0003   SUIT1
C8      RAD  U0002   I8
C9      TRD  U0002   I9
C10      0           I10
C11     EDI   0000   ORD7
D        88   9999    9999
F        99   9999    9999
CSTF    TDI  U0002   IF
ED      RAD  E0001   SUIT
8        80   00
UN       00   0001    000
10       10   00
```

Programme n° 5 : Tri par pile : l'ordre d'entrée est l'ordre de tri.

```
P2      TDI  P2SOR
        TRG  CNK
        AAG  CSTF
        TRG  TRANF
        TRD  P0001
        EDI  F       TRANF
IF      EDI  D
        TDI  U0001   DEB
DEB     EDI  C1
        TDI  ORD1
        EDI  F
        TDI  E0003
        EDI  C2
        TDI  ORD2
        EDI  C5
        TDI  ORD5
        EDI  C4
        TDI  ORD4    ORD1
I2      RAD  ORD1
        AAD  UN
        TRD  ORD1
        EDI  C3
        SBF  ORD3    ORD3
I3      TDI  UI      ORD4
I4      TDI  EJ
        ANG          NON1
        RAD  ORD2
        AAD  UN
        TRD  ORD2
        EDI  ORD4
        SBF  ORD4    ORD1
NON1    RAD  UI
        SAD  F
        ANN          NON2
        EDI  EJ      ORD5
I5      RAD  ORD2
        SAD  UN
        TRD  ORD2
        EDI  ORD4
        SBF  ORD4
        RAD  ORD5
        AAD  UN
        TRD  ORD5    ORD3
NON2    RAD  C6
        EDI  ORD4
        SBI  ORD6
        RAD  C7
        EDI  ORD5
        SBI  ORD7    B1
```



```
B1      EDI  EJ      ORD7
I7      RAD  ORD6
        SAD  UN
        TRD  ORD6
        RAD  E0001   ORD6
I6      TDI  EJ
        ANN          NON3
        RAD  ORD7
        AAD  UN
        TRD  ORD7    B1
NON3    RAD  E0003
        SAD  F
        ANN  DEB2    P2SOR
C1      EDI  U0001   ORD2
C2      TDI  E0001   I2
C3      RAD    0000  I3
C4      SAD  E0001   I4
C5      TDI  S0001   I5
C6        0          I6
C7        0          I7
E2      TDI  E0002   I2
D         89   9999    9999
F         99   9999    9999
CSTF    TDI  U0002   IF
UN        00   0001    000
IS      ART    1111    1111
N                        7
SO                       1
```

Programme n° 5 bis : Méthode de tri par pile : l'ordre de tri est l'ordre d'entrée dans la pile,

```
P29     TDI  P2SOR
        TRG  CNK
        AAG  CTD2
        TRG  TD2
        RAD  CNK
        AAD  CELIM
        TRD  ELIME
        AAD  1COL5
        EDI  CRANG
        SBF  RANGE
        EDI  CSTF
        SBF  TRANF
        RAD  CNK
        AAD  CNEUF
        TRD  NEUF
        EDI  CTRAF
        SBF  TRAF3
        EDI  CTEST
        SBF  TEST8
        EDI  D1
        TDI  E0001
        EDI  D2       TD2
ALORS   EDI  F        TRANF
IF      TDI  U0001
        EDI  8COL1
        TDI  TEST     B4
B4      EDI  TEST
        SD1  HUIT     NEUF
HUIT    EDI  1COL5
        TDI  UN
        EDI  C1
        TDI  ORD1
        EDI  9COL1
        TDI  TEST
        EDI  U0002    COMUN
DANN    ANG  RANGE    NON
SUIT    TDI  P0001
        PFO  P0001
        RAD  ELIME
        EDI  RANGE
        SBF  RANGE
        SAD  1COL5
        TRD  ELIME    NEUF
NON     RAD  SUI
        EDI  RANGE
        SBI  TRFER
        EDI  F        TRFER
ENSUI   EDI  8COL1
        TDI  TEST
        RSD  1COL5
        TRD  UN
        RAD  ELIME
        EDI  C1
        SBF  ORD1
        EDI  CINIT
        SBF  T        8001
```



```
COMUN   TDI  UI
        RAD  ORD1
        EDI  C4
        SBF  ORD4
        EDI  F        TRAF3
SUI3    EDI  C2
        TDI  ORD2
        EDI  C3
        TDI  ORD3
        RAD  C7
        AAD  CNK
        TRD  ORD7     B1
B1      EDI  UI       ORD2
PROGI   RAD  ORD1
        AAD  UN
        TRD  ORD1     8002
TUI     TDI  UI       ORD3
TEST3   ANG  TEST4
        RAD  ORD2
        AAD  1COL5
        TRD  ORD2
        EDI  ORD3
        SBF  ORD3     B1
TEST4   TDI  EJ
        RAD  UI
        SAD  F
        ANN  B5       NON4
B5      RAD  EJ       ORD7
TEST5   ANG           ORD4
        RAD  ORD7
        SAD  1COL5
        TRD  ORD7
        EDI  CTEJ
        SBF  TEJ
        EDI  EJ       TEJ
SUIT9   RAD  ORD2
        SAD  1COL5
        TRD  ORD2
        EDI  ORD3     BCLE
BCLE    SBF  ORD3     ORD1
SUIT4   RAD  ORD7
        AAD  1COL5
        TRD  ORD7
        RAD  ORD4
        AAD  UN
        TRD  ORD4     B5
NON4    RAD  ORD3
        EDI  C5
        SBF  ORD5
        RAD  ORD7
        EDI  C8
        SBF  ORD8
        RAD  ORD4
        EDI  C6
        SBF  ORD6     ORD5
```

```
SUIT5   TDI  EJ       ORD8
TEST6   ANG            TEST7
        RAD  ORD8
        SAD  1COL5
        TRD  ORD8
        EDI  CTEJP
        SBF  TEJP
        EDI  EJ       TEJP
IFP     RAD  ORD5
        SAD  1COL5
        TRD  ORD5      8002
TEST7   TDI  FK
        RAD    8001
        SAD  D2
        ANN            TEST8
        EDI  FK       ORD6
IF6     RAD  ORD8
        AAD  1COL5
        TRD  ORD8
        RAD  ORD6
        AAD  UN
        TRD  ORD6     ORD5
ENFIN   SAD  F
        ANN  B4
        RAD  UN
        ANG  B4       P2SOR
CTEJ    TDI  E0002    SUIT9
C7      SAD  E0002    TEST5
C2      TDI  E0002    PROGI
C3      AAD  E0002    TEST3
C4      TDI    0000   SUIT4
CINIT   EDI    0000   COMUN
9COL1     00                9
SUI       00     0000  ENSUI
C1      RSD  U0002    TUI
1COL5     00     0001    000
8COL1     00     0              8
F         99     9999    9999
P2SOR   ART    1234    1234
CNK       00     0016    000
D1        89     9999    9999
CSTF    TDI  U0002    IF
CELIM   SAD  U0001    DANN
CNEUF   RAD  E0000    ELIME
CRANG   TDI  U0002    SUIT
CTD2    TDI  E0002    ALORS
CTRAF   TDI  E0000    SUI3
CTEST   RAD  E0000    ENFIN
C6      TDI    0000   IF6
D2        88     9999    9999
CTEJP   TDI    0000   IFP
C8      SAD    0000   TEST6
C5      RAD    0000   SUIT5
```


Programme n° 5 ter : Méthode de tri par piles : l'ordre de tri est l'ordre d'entrée dans la pile
Nombre de mémoires occupées : 226.

```
P29BI   TDI  P2SOR
        TRG  CNK
        AAG  CTD2
        TRG  TD2
        RAD  CSTF
        AAD  CNK
        TRD  UOM
        AAD  1COL5
        TRD  TRANF
        EDI  CUON
        TDI  UON
        RAD  CNK
        AAD  CSF3
        TRD  SF3
        EDI  CFIN9
        SBF  FIN9
        EDI  CTRAF
        SBF  TRAF3
        EDI  CTRF
        SBF  TRF3
        EDI  CRANG
        TDI  RANGE
        EDI  D2       TD2
ALORS   EDI  F        TRANF
IF      TDI  U0001    TRF3
INIT8   EDI  D2
        TDI  TEST     B4
F         99    9999    9999
D2        88    9999    9999
CRANG   TDI  E0001    TPF0
CUON      00  U0002    000
CFIN9   RAD    0000   SUIT9
CTRAF   TDI    0000   SUIT2
CTRF    TDI    0000   INIT8
CSF3    RAD  E0003    ELIME
CSTF    TDI  U0001    IF
1COL5     00    0001    000
CELIM   SAD    0000   TEST2
CTD2    TDI  E0005    ALORS
P2SOR   ART    1234    1234
CNK       00    0256    000
B4      EDI  TEST
        SDO  HUIT     NEUF
HUIT    RAD  UON
        EDI  CELIM
        SBF  ELIME
        EDI  1COL5
        TDI  UN
        RAD  CUONP
        EDI  CORD
        SBF  ORD
        EDI  F
        TDI  TEST     SF3
```

```
NEUF    RAD   UOM
        EDI   CELIM
        SBF   ELIME
        RSD   1COL5
        TRD   UN
        RAD   CUOMP
        EDI   CORD
        SBF   ORD
        EDI   D2
        TDI   TEST     SF3
TEST2   ANG   RANGE    NON2
TPFO    TDI     1977
        PFO     1977
        RAD   RANGE
        AAD   1COL5
        TRD   RANGE
        RAD   ELIME
        AAD   UN
        TRD   ELIME    SF3
NON2    TDI   UI
        RAD   ELIME
        SAD   UN
        EDI   CTFER
        SBF   TRFER
        EDI   F        TRFER
REINI   RAD   ELIME    ORD
SUIT1   EDI   CORD1
        SBF   ORD1
        EDI   CORD4
        SBF   ORD4
        RAD   RANGE
        EDI   CORD2
        SBF   ORD2
        EDI   CORD3
        SBF   ORD3
CORD3   AAD     0000   TEST4
CORD2   TDI     0000   PROGI
CORD1   RSD     0000   TUI
CORD    TDI     0000   SUIT1
CUOMP     00    UOM      0
CUONP     00    UON      0
CTFER   TDI     0000   REINI
        EDI   CTD1
        SBF   TD1
        RAD   CNK
        AAD   CORD7
        TRD   ORD7
        EDI   F        TRAF3
SUIT2   EDI   D1       TD1
B1      RAD   ORD2
        AAD   1COL5
        TRD   ORD2
        EDI   ORD3
        SBF   ORD3
        EDI   UI       ORD2
PROGI   RAD   ORD1
```



```
        AAD   UN
        TRD   ORD1     8002
TUI     TDI   UI       ORD3
TEST4   ANG            B1
        TDI   EJ
        RAD   UI
        SAD   F
        ANN   B5       NON5
B5      RAD   EJ       ORD7
TEST7   ANG            ORD4
        RAD   ORD7
        SAD   1COL5
        TRD   ORD7
        EDI   CTEJ
        SBF   TEJ
        EDI   EJ       TEJ
SUIT3   RAD   ORD2
        SAD   1COL5
        TRD   ORD2
        EDI   ORD3
        SBF   ORD3     ORD1
SUIT4   RAD   ORD7
        AAD   1COL5
        TRD   ORD7
CORD4   TDI     0000   SUIT4
CTEJ    TDI     0000   SUIT3
D1        89    9999     9999
CTD1    TDI     0000   B1
CORD7   SAD   E0005    TEST7
        RAD   ORD4
        AAD   UN
        TRD   ORD4     B5
NON5    RAD   ORD3
        EDI   CORD5
        SBF   ORD5
        RAD   ORD7
        EDI   CORD8
        SBF   ORD8
        EDI   CTEJP
        SBF   TEJP
        RAD   ORD4
        EDI   CORD6
        SBF   ORD6     ORD5
```

```
SUIT5   TDI   EJ        ORD8
TEST6   ANG             TFK
        RAD   ORD5
        SAD   1COL5
        TRD   ORD5
        RAD   TEJP
        SAD   1COL5
        EDI   EJ          8002
TFK     TDI   FK
        RAD    8001
        SAD   D2
        ANN             FIN9
        RAD   ORD8
        AAD   1COL5
        TRD   ORD8
        EDI   TEJP
        SBF   TEJP
        EDI   FK        ORD6
SUIT6   RAD   ORD6
        AAD   UN
        TRD   ORD6      ORD5
SUIT9   SAD   F
        ANN   B4
        EDI   TEST
        SDO   HUITP     NEUFP
HUITP   RAD   UON
        EDI   CFIN
        SBF   FIN
        EDI   1COL5
        TDI   UN        COMUN
NEUFP   RAD   UOM
        EDI   CFIN
        SBF   FIN
        RSD   1COL5
        TRD   UN        COMUN
CTUJ    TDI    0000     TES11
CORD6   TDI    0000     SUIT6
CORD8   SAD    0000     TEST6
CORD5   RAD    0000     SUIT5
COMUN   RAD   RANGE
        EDI   CTUJ
        SBF   TUJ       FIN
TES11   SAD   F
        ANN             P2SOR
        RAD   TUJ
        AAD   1COL5
        TRD   TUJ
        RAD   FIN
        AAD   UN
        TRD   FIN        8002
CTEJP   TDI    0000     ORD6
CFIN    RAD    0000     TUJ
```


Programme n° 6 : Méthode issue de celle de Von Neumann

```
DEBUT   EDI   C3
        TDI   ODRE3
        EDI   C4
        TDI   ODRE4     ODRE3
IS3     TDI   T2        ODRE4
IS4     TDI   T1
        ANG   OUI
        ANN             OUI
        RAD   ODRE3
        EDI   C5
        SBF   ODRE5
        AAD   CST1
        EDI   C6
        SBF   ODRE6
        EDI   T1        ODRE5
IS5     EDI   T2        ODRE6
OUI     RAD   ODRE3
        AAD   C2
        TRD   ODRE3
        RAD   ODRE4
        AAD   C2
        TRD   ODRE4
        RSD    8001
        AAD   CJ
        ANG   DEB       ODRE3
C2       00    0002      000
C3      RAD   A0001     IS3
C4      SAD   A0002     IS4
C5      TDI    0000     IS5
C6      TDI    0000     OUI
CJ      SAD   A1024     IS4
DEB     RAG   C2
        TDI   2K1
        MUL   CST2
        TRD   2K2       B1
B1      RAD   CAO
        AAD   2K1
        EDI   CST3
        SBF   ORD3
        AAD   CST1
        EDI   CST4
        SBF   ORD4      ORD3
AIS4    ANG   OUI1
        ANN   OUI2      OUI1
OUI1    RAD   CNJ
        SAD   2K1
        SAD   ORD4
        ANG   OUI3
        RAD   ORD3
        AAD   2K2
        TRD   ORD3
        RAD   ORD4
        AAD   2K2
        TRD   ORD4      ORD3
```

```
OUI3    RAG  2K2
        SAG  CNK
        GNN            PFO
        RAG  2K1
        MUL  CST2
        TRD  2K1
        RAG  2K2
        MUL  CST2
        TRD  2K2       B1
OUI2    TDI  T
        RAD  ORD4
        EDI  CST5
        SBF  ORD5
        AAD  CST1
        EDI  CST6
        SBF  ORD6      B2
B2      RAD  ORD5
        SAD  CST1
        TRD  ORD5
        RAD  ORD6
        SAD  CST1
        TRD  ORD6      ORD5
AIS6    RAD  8001
        SAD  T
        ANG  OUI4
        ANN            AIS7
        RAD  ORD4
        SAD  2K1
        EDI  ORD5
        SBF  T1
        RSG  8001
        AAG  ORD5
        GNN  B2
        RAD  ORD5
        EDI  CST7
        SBF  ORD7
        EDI  T         ORD7
OUI4    RAD  CST7
        EDI  ORD6
        SBI  ORD8
        EDI  T         ORD8
AIS7    RAD  ORD4
        EDI  CST9
        SBF  ORD9
        EDI  CST10
        SBF  T1        8001
AIS10   TDI  T         B3
B3      RAD  ORD9
        AAD  CST1
        TRD  ORD9      ORD9
```



```
AIS9    TDI  T2
        SAD  T
        ANG            NON6
        RAD  ORD9
        SAD  CST1
        EDI  CST11
        SBF  ORD11
        EDI  T2        ORD11
AIS11   RAD  ORD3
        AAD  2K1
        EDI  ORD9
        SBF  T2
        RAG  8001
        SAG  ORD9
        GNN  B3        COMUN
COMUN   AAD  ORD9
        EDI  CST12
        SBF  ORD12
        EDI  T         ORD12
NON6    RSD  CST1      COMUN
CST1     00   0001     00
CST2     00                 2
CST3    RAD   0000     ORD4
CST4    SAD   0000     AIS4
CST5    EDI   0000     ORD6
CST6    TDI   0000     AIS6
CST7    TDI   0000     AIS7
CST9    RAD   0000     AIS9
CST10   EDI   0000     AIS10
CST11   TDI   0000     AIS11
CST12   TDI   0000     ORD3
CAO      00   A0000
CNJ     SAD   A1024    AIS4
CNK      00   A1024     000
```

Programme n° 7 - Sélection ordinaire

```
SELEC   TDI   SSORT
        TRG   NSELE
        SAG   UN
        AAG   SC2
        TDI   SORD2
        TRG   SCNJ
        SAG   SC2
        SAG   UN
        AAG   SC1
        TRG   SCNI
        EDI   SC4
        TDI   SORD4   SB1
SB1     RAD   SORD2
        EDI   SC1
        SBF   SORD1   SORD2
SI2     TDI   AJ      SB2
SB2     RAG   SORD1
        AAG   UN
        TRG   SORD1
        RAD   AJ      SORD1
SI1     ANG   TEST2
        TDI   AI
        RAD   SORD1
        EDI   SC3
        SBF   SORD3
        EDI   AJ      SORD3
SI3     EDI   AI
        TDI   AJ      TEST2
TEST2   RAG   SCNI
        SAG   SORD1
        ANG           SB2
        EDI   AJ      SORD4
SI4     RAD   SCNJ
        SAD   SORD2
        ANG   SSORT
        RAD   SORD2
        AAD   UN
        TRD   SORD2
        EDI   SORD4
        SBF   SORD4   SB1
SC1     SAD   A0001   SI1
SC2     EDI   A0001   SI2
SC3     TDI   A0001   SI3
SC4     TDI   A0001   SI4
UN       00    0001    000
AO       00   A0001   A0001
```


Programme n° 8 - Tri par la méthode de Hibbard

```
HIBAR   TDI   HIBSO
        TRD   H
        EDI   UN5
        TDI   E
        TDI   K       BB
BB      RAD   E
        SAD   H
        ANG           MODIK
        RAD   E
        AAD   CTRAD    8002
TRAD    TDI   D
        SAD   CTRAD
        EDI   AFECI
        SBF   AFECI
        EDI   TESAI
        SBF   TESAI
        RAD   H
        EDI   AFECJ
        SBF   AFECJ
        EDI   TESAJ
        SBF   TESAJ   BE
BE      RSD   D       TESAJ
TEST    ANG   AFECI   REGRJ
REGRJ   RAD   TESAJ
        SAD   UN5
        TRD   TESAJ
        EDI   AFECJ
        SBF   AFECJ   IEGJ
IEGJ    RAD   CREGR
        EDI   AFECI
        SBI   QCQ
        RAG    8001
        SAG   AFECJ
        GNN   PE      BD
CREGR    00    0000   REGRJ
TESAJ   AAD    0000   TEST
AFECJ   TDI    0000   REGRJ
TESAI   SAD    0000   TESTP
AFECI   TDI    0000   PROGI
CTRAD   EDI   A0000   TRAD
UN5      00    0001    000
PROGI   RAD   TESAI
        AAD   UN5
        TRD   TESAI
        EDI   AFECI
        SBF   AFECI   JEGI
JEGI    RAD   CREGR
        EDI   AFECI
        SBI   QCQ
        RAG    8001
        SAG   AFECJ
        GNN   BF      BD
```

```
BF      RAD  D        TESAI
TESTP   ANG  AFECJ    PROGI
BD      RAD  TESAI
        EDI  CAID
        SBF  CAID
        EDI  D        CAID
SUI     EDI  N
        SBF  I
        RAD    8001
        AAD    8001
        SAD  E
        SAD  H
        ANG  OUI
        RAD  K
        AAD  CFKNO
        TRD  FKNON
        SAD  CFKNO
        AAD  CGKNO
        TRD  GKNON
        EDI  E        FKNON
SUIN    RAD  I
        SAD  UN5      GKNON
SUINP   AAD  DEUX5
        TRD  E        COMUN
COMUN   RAD  K
        AAD  UN5
        TRD  K        BB
DEUX5     00    0002    000
CGKNO   TRD  G0000    SUINP
CFKNO   TDI  F0000    SUIN
CAID    TDI    0000   SUI
OUI     RAD  K
        AAD  CFKOU
        TRD  FKOUI
        SAD  CFKOU
        AAD  CGKOU
        TRD  GKOUI
        RAD  I
        AAD  UN5      FKOUI
SUIO    EDI  H        GKOUI
SUIOP   SAD  DEUX5
        TRD  H        COMUN
MODIK   RAD  K
        SAD  UN5
        TRD  K
        ANG  HIBSO
        ANN           HIBSO
        AAD  CFK        8002
TDIE    TDI  E
        SAD  CFK
        AAD  CGK        8002
TDIH    TDI  H        BB
CGK     EDI  G0000    TDIH
CFK     EDI  F0000    TDIE
CGKOU   TDI  G0000    SUIOP
CFKOU   TRD  F0000    SUIO
```


Programme n° 9 - Interclassement par la méthode de Von Neumann

```
NEUMA   TDI  NEUS0
        TRD  N
        AAD  CA1
        TDI  A1
        TRD  AN1
        EDI  UN5
        TDI  K
        EDI  DEUX5
        TDI  2K       INIT
INIT    RAD  AN1
        AAD  CH1
        TRD  H1
        AAD  N
        SAD  DEUX5
        TRD  HN
        EDI  A1
        TDI  A1J      BCLE
BCLE    RAD  A1J
        AAD  CF1
        TRD  F1
        AAD  K
        EDI  CG1
        SBF  G1
        SAD  UN5
        TRD  F1V
        AAD  K
        EDI  CG1
        SBF  G1V      F
F       RAD  F1V
        SAD  F1
        ANG  FVGF     FF
FF      RAD  G1V
        SAD  G1
        ANG  FFGV
        RAD    8001
        EDI  CSADA
        SBF  SADAG
        RAD  F1
        EDI  CRADA
        SBF  RADAF    8001
TEST    ANG           G1
        RAD  H1
        EDI  CH1P
        SBF  H1P
        AAD  UN5
        TRD  H1       F1
```

```
SUI1P   RAD   F1
        AAD   UN5
        TRD   F1        F
SUI1    RAD   G1
        AAD   UN5
        TRD   G1
        RAD   H1
        AAD   UN5
        TRD   H1        FF
FFGV    RAD   F1
        EDI   CF2
        SBF   F2
        AAD   UN5
        TRD   F1
        RAD   H1
        EDI   CH2P
        SBF   H2P
        AAD   UN5
        TRD   H1        F2
SUI2P   RAD   F1V
        SAD   F1
        ANG   FVGV      FFGV
FVGF    RAD   G1V
        SAD   G1
        ANG   FVGV
        RAD   G1
        EDI   CG2
        SBF   G2
        AAD   UN5
        TRD   G1
        RAD   H1
        EDI   CH2
        SBF   H2
        AAD   UN5
        TRD   H1        G2
FVGV    RAD   HN
        SAD   H1
        ANG   OUI
        RAD   A1J
        AAD   2K
        TRD   A1J       BCLE
OUI     RAD   2K
        TDI   K
        AAD     8001
        TRD   2K
        SAD   K
        SAD   N
        ANG             NON
        EDI   A1
        TDI   PERMU
        EDI   AN1
        TDI   A1
        EDI   PERMU
        TDI   AN1       INIT
```



```
NON     RAD   AN1
        AAD   N
        SAD   UN5
        DAD     0004
        AAD   AN1
        TRD   NTRIE     NEUSO
CH2     TDI     0000    FVGF
CG2     EDI     0000    H2
CH2P    TDI     0000    SUI2P
CF2     EDI     0000    H2P
CH1P    TDI     0000    SUI1P
CRADA   RAD     0000    SADAG
CSADA   SAD     0000    TEST
CG1     EDI     0000    H1
CF1     EDI     0000    H1P
CH1     TDI     0000    SUI1
DEUX5    00     0002     000
UN5      00     0001     000
CA1      00    A0001     000
```

D'APRES TABLEAUX [1] pages 438 à 460 [Ed] 1-08

METHODE	Nombre de mémoires auxiliaires	Nombre de Tests sur les n			Nombre de transferts			Temps de Tri 256 Mots		Temps de Tri 512 Mots		Mémoires occupées par le Programme
		MOYEN	MINI	MAXI	MOYEN	MIN.	MAXI	SI 1	SI 2	SI 1	SI 2	
BULLE	0	$> \frac{n^2-n}{4}$	$n-1$	$\frac{n^2-n}{2}$	$> \frac{n^2-n}{4}$	0	$\frac{n^2-n}{2}$	29'	28''	115'	2'30''	
SHELL	0	$n^{1,226}$	$n \log_2 n$	$\frac{n^2-n}{2}$	$\frac{n^2-n}{4}$	0	$\frac{n^2-n}{2}$	1'30''	1'43''	14'45''	3'42''	
DEDUITE de VON NEUMANN	0							16'	32''	68'	3'	
SELECTION LINEAIRE	0	$\frac{n^2-n}{2}$	$\frac{n^2-n}{2}$	$\frac{n^2-n}{2}$	$> \frac{n^2-n}{4}$	0	$\frac{n^2-n}{2}$	21'	16'	81'	62'	
SELECTION QUADRATIQUE	$N + \sqrt{N}$	$n+2(n-1)\sqrt{n}$	$n+2\sqrt{n}$	$n+2(n-1)\sqrt{n}$								
de HIBBARD	$\log_2 N$			$\frac{n^2+n}{2}$		0	$\frac{n^2+n}{2}$	2'	23'	4'45''	86'	
INSERTION	0											
INSERTION DICHOTOMIQUE	0	$n \log_2 n$	$n-1$	$n \log_2 n$	$\frac{n^2-n}{4}$	0	$\frac{n^2-n}{2}$					
INTERCLASSEMENT VON NEUMANN		$n \log_2 n$	$n \log_2 n$	$n \log_2 n$	$n \log_2 n$	0	$n \log_2 n$	2'17''	2'	5'5''	4'28''	
PILE N° 1								21'30''		98''		
PILE N° 2								65'				
N° 2 bis								20'45''		79'		
N° 2 ter								20'		75'		

BIBLIOGRAPHIE

[1] IVERSON et KENNETH "A programming language" John Wiley and Sons 1962 - pages 176 - 246

[2] SHELL D. L. "A high speed sorting procedure" Communications ACM 2 (1959) - pages 30-32

[3] FRANK R. M. ; LAZARUS R. B. "A high speed sorting procedure" Communication ACM 3 (1960) - pages 20-22

[4] PICARD C. Théorie des questionnaires - CNRS - Institut Blaise Pascal (publication n° 19, 3, 3/BI) page 147

[5] BOTTENBRUCH Structure and use of ALGOL 60 - ORNL - Report 3148 - AskRidge (1961)

[6] HIBBARD T. N. Some combinationial properties of certain trees with applications to scareling and sorting - Journal of ACM - (1962) - pagés 25-28

[7] PAIR C. Arbres, piles et compilation - Revue française de traitement de l'information - Chiffres (Vol, 7, 1964, pages 199-216)

[8] PAIR C. Séminaire de Programmation 1963-1964 - Centre de Calcul Automatique de Nancy

[9] SAMELSON et BAUER Sequential formula of translation - Communications of ACM - n° 3 - (1963) - pages 76-86

[10] HALL M. M. A method of comparing the time requirements of sorting methods - Communications of ACM -n° 5 (mai 1963) - pages 259-263

Vu et Approuvé
Nancy, le 15/12/64
Le Doyen de la Faculté
des Sciences de NANCY

M. AUBRY

Vu et Permis d'imprimer
Nancy, le 17/12/64.
le Recteur :
Président du Conseil de
l'Université
P. IMBS