Chapitre 3

Les algorithmes de tri

Objectifs

Acquérir des habilités de résolution de problèmes à travers l'apprentissage de quelques algorithmes de tri.

Plan du chapitre

I- Introduction

II- Tri par insertion

III- Tri Shell

IV- Applications

Retenons

Exercices

Lecture

Chapitre 3

Les algorithmes de tri

I. Introduction

Selon le dictionnaire «Le Petit Larousse», "**trier**" signifie «répartir des objets suivant certains critères». De manière plus restrictive, le terme "**tri**" en algorithmique est souvent attaché au processus de classement d'une suite d'éléments dans un ordre donné. Par exemple, trier M moyennes de type réel dans l'ordre décroissant ou trier N noms dans l'ordre alphabétique croissant. D'une façon générale, tout ensemble muni d'un ordre total peut fournir une suite d'éléments à trier.
Il existe deux catégories de tris :
Les tris internes : Méthodes destinées à des masses de données limitées, stockées dans une structure de données se trouvant dans la mémoire centrale (Exemple : tableaux).
Les tris externes : Méthodes destinées à de grandes masses de données, stockées dans des structures de données telle que les fichiers.

Vous avez utilisé dans le programme de 3ème année les méthodes de tri par sélection et à bulles.

Activité 1

1- Donnez le principe du **tri par sélection**.
2- Déduisez un algorithme de la procédure **SELECTION (n : entier; Var T : Tab)**
3- Avec l'aide de votre enseignant, écrivez un programme en Pascal qui permet de remplir aléatoirement un tableau de n entiers, de le trier en utilisant la méthode de tri par sélection puis de l'afficher.

La méthode de tri par sélection utilise l'algorithme formel suivant :
1- Placer dans l'élément d'indice 1 du tableau T la plus petite valeur présente dans le tableau et mettre à sa place l'ancienne valeur de T[1].
2- Placer dans l'élément d'indice 2 de T la plus petite valeur présente dans la tranche de tableau T[2..N]
3- Placer dans l'élément d'indice 3 de T la plus petite valeur présente dans la tranche de tableau T[3..N]
4- et ainsi de suite jusqu'à l'étape N-1

Activité 2

1- Donnez le principe du tri à bulles.
2- Déduisez un algorithme de la procédure **BULLES (n : Entier; Var T : Tab)**

3- Avec l'aide de votre enseignant, écrivez un programme en Pascal qui permet de remplir aléatoirement un tableau de n entiers, de le trier dans l'ordre croissant en utilisant l'algorithme de tri à bulles puis de l'afficher.

La méthode de tri à bulles utilise l'algorithme formel suivant :

L'algorithme du tri à bulles (**bubble sort** en anglais) consiste à comparer les différentes valeurs adjacentes du tableau T, et à les permuter s'ils ne sont pas dans le bon ordre. L'algorithme se déroule ainsi :

1- Les deux premiers éléments du tableau sont comparés, si le premier élément est supérieur au second, une permutation est effectuée.

2- Ensuite, sont comparées et éventuellement permutées les valeurs 2 et 3, 3 et 4 jusqu'à (n-1) et n.

3- Une fois cette étape achevée, il est certain que le dernier élément du tableau est le plus grand. L'algorithme reprend donc pour classer les (n-1) éléments qui précédent.

4- L'algorithme se termine quand il n'y a plus de permutations possibles.

Remarques :

1- Pour classer les n valeurs du tableau T, il faut, au pire des cas, effectuer l'algorithme n fois.
2- Cette méthode porte le nom de tri à bulles car, petit à petit, les plus grands éléments du tableau remontent, par le jeu des permutations, en fin du tableau.

Les différents algorithmes de tri présentent des performances différentes, en terme de temps d'exécution et en terme d'espace mémoire requis pour leur exécution.

Dans la suite de ce chapitre, vous allez apprendre à utiliser le tri par insertion, le tri Shell ainsi que des applications sur les méthodes de tri.

II. Tri par insertion

II.1 Principe

Le principe du tri par insertion est d'insérer à la n-ième itération, le n-ième élément à la bonne place dans la liste formée par les (n-1) éléments qui le précèdent.

Voici, en pseudo-code, l'algorithme du tri par insertion :

```
PROCEDURE Tri_Insertion (n : Entier; Var T : Tab)
        Pour i de 2 à n Faire
            INSERER T[i] à sa place dans T[1..(i-1)]
FIN
```

Le principe général est le suivant :

- Considérer que les (i-1) premiers éléments de la liste sont triés et placer le $i^{ème}$ élément à sa place parmi les (i) premières places.
- Répéter cette action jusqu'à atteindre la fin de la liste.

Le processus d'insertion consiste à :

- utiliser une variable intermédiaire Tmp pour conserver la valeur à insérer,
- déplacer les éléments T[i-1], T[i-2], ... vers la droite tant que leur valeur est supérieure à celle de tmp.
- affecter alors à l'emplacement laissé libre par ce décalage la valeur de Tmp.

Remarque :

Une variante du tri par insertion consiste à utiliser une recherche dichotomique. En effet, à la n-ième itération, les éléments de 1 à n-1 sont triés, et nous pouvons donc utiliser un algorithme de recherche dichotomique pour trouver la place où insérer l'élément n.

II.2 Exemple

Nous proposons d'utiliser la méthode de tri par insertion pour trier un tableau T d'entiers en ordre croissant.

Considérons le tableau T contenant les 7 éléments suivants :

T	-5	30	0	-2	42	22	9
	1	2	3	4	5	6	7

Commençons par T[2] puisque si le tableau contient un seul élément, il est déjà trié.

Etape 1

Cherchons la position d'insertion de **T[2]** dans la première partie du tableau T avec le souci de la garder triée.

Puisque 30 > -5, donc les deux premiers éléments sont triés.

T	-5	30	0	-2	42	22	9
	1	2	3	4	5	6	7

Etape 2

Nous cherchons la position d'insertion de **T[3]** dans la première partie du tableau T avec le souci de la garder triée.

- Nous affectons à Tmp la valeur de T[3] = 0
- Nous décalons T[2] à droite, car il est supérieur à 0
- Nous affectons à la dernière case décalée la valeur de Tmp

T	-5	0	30	-2	42	22	9
	1	2	3	4	5	6	7

Etape 3

Cherchons la position d'insertion de **T[4]** dans la première partie du tableau T avec le souci de la garder triée.

- Nous affectons à Tmp la valeur de T[4] = -2
- Nous décalons T[3] et T[2] à droite, car ils sont supérieurs à -2
- Nous affectons à la dernière case décalée la valeur de Tmp

T	-5	-2	0	30	42	22	9
	1	2	3	4	5	6	7

Activité

1- Sur votre cahier, continuez les autres étapes jusqu'à obtenir un tableau trié.
2- Combien faut-il d'étapes pour trier ce tableau?
3- Pouvez-vous donner une réponse dans le cas général en fonction de n (n étant le nombre d'éléments du tableau) ?

II.3 Résolution du problème

Nous utiliserons l'approche de l'analyse modulaire qui consiste à décomposer le problème en modules.

a) Analyse du programme principal

Résultat : L'affichage du tableau trié est la tâche de la procédure Afficher
Traitement :
Nous devons donc trier le tableau, ce qui sera la tâche de la procédure Trier

- *Pour pouvoir trier le tableau, nous devons tout d'abord le remplir, ce qui se fera par la procédure Remplir*
- *Pour remplir le tableau, nous devons connaître le nombre d'éléments à saisir ce qui est la tâche de la procédure Saisir.*

b) Algorithme du programme principal et codification des objets

0) Début tri_INS
1) Proc Saisir (n)
2) Proc Remplir (T, n)
3) Proc Trier (T, n)
4) Proc Afficher (T, n)
5) Fin tri_INS

Tableau de codification des objets globaux

Objets	Type / Nature	Rôle
n	Entier	Taille du tableau à trier
T	Tab	Tableau à trier
Saisir	Procédure	Permet de saisir la taille du tableau à trier
Remplir	Procédure	Permet de remplir le tableau à trier
Trier	Procédure	Permet de trier le tableau en utilisant la méthode du tri par insertion
Afficher	Procédure	Permet d'afficher le tableau trié

Dans ce qui suit, nous allons détailler l'analyse de la procédure Trier. Les autres modules ont été déjà traités lors de la résolution d'autres problèmes.

a) Analyse de la procédure Trier

Résultat : Trier le tableau T

Traitement :

A l'aide de l'algorithme formel de la méthode de tri par insertion et en exécutant manuellement l'exemple, nous pouvons déduire ce qui suit :

Il s'agit d'un traitement répétitif complet de l'élément n° 2 jusqu'au dernier élément du tableau : Pour c de 2 à n Faire

Pour chaque valeur du compteur et si l'élément correspondant n'est pas à sa place, nous réalisons les actions suivantes :

- Ranger la valeur de T[c] dans la variable Tmp
- Décaler vers la droite les valeurs de T[c-1], T[c-2], ... jusqu'à arriver à une valeur qui est inférieure à T[c]. La procédure DECALER permettra de réaliser cette action.
- Affecter au dernier élément décalé la valeur de Tmp

```
Si T[c-1] > T[c]
Alors     Tmp ← T[c]
          Proc DECALER (T, c-1, p)
          T[p+1] ← Tmp
FinSi
```

b) Algorithme de la procédure Trier

```
0) Début Procédure Trier (n : entier; VAR T : Tab)
1) Pour c de 2 à n Faire
     Si T[c-1] > T[c]
     Alors     Tmp ← T[c]
               Proc DECALER (T, c-1, p)
               T[p+1]← Tmp
     FinSi
   Fin Pour
2) Fin Trier
```

Tableau de codification des objets globaux

Objets	Type / Nature	Rôle
c	Entier	Compteur
Tmp	Entier	Variable intermédiaire
p	Entier	Position d'insertion
DECLARER	Procédure	Permettant de décaler des éléments d'un tableau d'un nombre de positions

a) Analyse de la procédure DECLARER

Résultat : Décaler à droite les éléments du tableau T d'indice deb à l'indice fin

Traitement :

Il s'agit d'un traitement répétitif à condition d'arrêt :

Tant que *(fin >= 1) Et (T[fin] > Tmp)* ***Faire***

- Cette boucle est initialisée par l'instruction suivante : fin← deb
- L'action de décalage est une simple affectation : T[fin+1]←T[fin]
- La décrémentation par 1 de la variable fin : fin ← fin - 1

b) Algorithme de la procédure DECLARER

0) Début Procédure DECALER (VAR T : Tab; deb : Entier; Var fin : Entier)
1) fin ← deb
Tant que (fin >= 1) ET (T[fin] > Tmp) **Faire**
T[fin+1] ← T[fin]
fin ← fin - 1
Fin Tant que
2) Fin DECALER

e) Traduction en Pascal de la procédure Trier

```
Procedure Trier (n : Integer; Var T : Tab);
 Var c, Tmp,  p : Integer;
 Procedure Decaler (Var T : Tab; deb : Integer; Var fin : Integer);
   Begin
          fin := deb;
          While (fin >= 1) And (T[fin] > tmp) Do
                Begin
                      T[fin + 1] := T[fin];
                      fin := fin - 1;
                End;
   End;
{Procédure Trier}
 Begin
   For c := 2 To n Do
     If T[c-1] > T[c] Then
      Begin
            Tmp := T[c];
            Decaler (T, c-1, p);
            T[p+1] := Tmp;
      End;
 End;
```

Application

Proposez une solution récursive de la méthode de tri par insertion.

III. Tri Shell

III.1 Insuffisances de la méthode de tri par insertion

En analysant l'algorithme de la méthode de tri par insertion, nous pouvons remarquer qu'en moyenne, il s'agit d'un algorithme d'ordre n^2 puisqu'il comporte deux boucles imbriquées d'ordre n chacune. Il est toutefois évident que si le vecteur est initialement presque trié dans le bon ordre, le nombre d'opérations sera beaucoup plus réduit.
Si la méthode de tri par insertion est efficace quand la liste est à peu près triée, elle est inefficace en moyenne car elle ne change les valeurs que d'une position par instruction. En effet, cette méthode traite un à un les éléments de la liste à trier et réalise une rotation des éléments précédents jusqu'à avoir la position d'insertion de l'élément en cours.

III.2 Principe

Donald L. Shell proposa, en 1959, une variante du tri par insertion. Ce tri consiste à trier séparément des sous-tableaux du tableau initial objet du tri, formés par les éléments répartis de p éléments.

N.B.

Le tri Shell est une amélioration du tri par insertion. Au lieu de faire une rotation de tous les éléments, nous ferons une rotation par pas de p ce qui permet d'affiner le tri du tableau et de faire moins de déplacements d'éléments.
Nous commençons donc par un pas assez élevé et nous le diminuons au fur et à mesure jusqu'à arriver à un pas de 1. Ceci permet d'éliminer les plus grands désordres pour abréger le travail aux étapes suivantes. Le pas est diminué à l'aide d'une suite, mais il faut construire cette suite de manière à ce que les valeurs ne soient pas multiples entre elles sinon nous allons traiter des éléments et pas d'autres.

Shell propose la suite d'incréments vérifiant $p_1 = 1$, $p_{n+1} = 3p_n+1$ en réalisant les tris du plus grand incrément possible vers le plus petit.
D'autre part, puisque au dernier stade, p_0 est égal à 1, nous déterminerons le pas maximal par récurrence en inversant la relation donnée ci-dessus :

$$p_{k+1} = 3 * p_k + 1$$

et en s'assurant qu'il y a encore un nombre suffisant de composantes dans les sous-vecteurs considérés.

Remarque :

Le tri Shell trie chaque liste d'éléments séparés de p positions chacun avec le tri par insertion. L'algorithme effectue plusieurs fois cette opération en diminuant p jusqu'à p égal à 1 ce qui équivaut à trier tous les éléments ensemble

III.3 Exemple

Nous proposons d'utiliser la méthode du tri Shell pour trier un tableau T d'entiers en ordre croissant.

Considérons le tableau T contenant les 15 éléments suivants :

T	-5	30	0	-2	42	22	9	-4	10	15	40	-9	12	28	14
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Etape 1

La première action à faire consiste à déterminer la valeur maximale du pas.

Etape 2

N.B. En utilisant l'expression p ← p Div 3, nous pouvons déduire que les valeurs que prendra le pas p sont 13, 4 et 1.

Etape 2-1

Pour p=13, nous appliquons le tri par insertion pour chacun des sous-vecteurs de pas 13.

- Trions par insertion linéaire le sous-vecteur qui regroupe les composantes 1 et 14 et qui ont pour valeurs respectives –5 et 28.
  Le tableau ne subira pas de modifications car –5 < 28, d'où les valeurs garderont leurs places.
- Trions par insertion linéaire le sous-vecteur qui regroupe les composantes 2 et 15 qui ont pour valeurs respectives 30 et 14.

Nous allons obtenir le tableau suivant :

T	-5	14	0	-2	42	22	9	-4	10	15	40	-9	12	28	30
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Etape 2-2

Pour p=4, nous appliquons le tri par insertion pour chacun des sous vecteurs de pas 4 en commençant par l'élément d'indice 1, puis l'élément d'indice 2, etc.

- Trions par insertion linéaire le sous vecteur qui regroupe les composantes 1, 5, 9 et 13 et qui ont pour valeurs respectives : -5, 42, 10, 12.

Nous allons obtenir le tableau suivant :

T	-5	14	0	-2	10	22	9	-4	12	15	40	-9	42	28	30
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Remarquez que les valeurs des cases d'indices 1, 5, 9 et 13 sont triées par ordre croissant.

- Trions par insertion linéaire le sous vecteur qui regroupe les composantes 2, 6, 10 et 14 et qui ont pour valeurs respectives : 14, 22, 15 et 28.

Nous allons obtenir le tableau suivant :

T	-5	14	0	-2	10	15	9	-4	12	22	40	-9	42	28	30
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Remarquez que les valeurs des cases d'indices 2, 6, 10 et 14 sont triées en ordre croissant

- Quels sont les indices des éléments à trier en 3ème lieu ? Donnez le contenu du tableau T après avoir réalisé cette étape.
- Même question pour la 4ème opération.

Etape 2-3

Pour p=1, nous appliquons le tri par insertion pour chacun des sous-vecteurs de pas 1. Ce qui revient à appliquer le tri par insertion simple à tout le tableau.

Constatations :

Il est bien évident que la dernière étape aurait pu suffire, puisqu'il s'agit d'un tri par insertion linéaire simple. Toutefois, dans le contexte où nous nous sommes progressivement placés, nous avons fait déplacer les grandes valeurs vers la fin du tableau et les petites vers le début. Le nombre de comparaisons dans le dernier passage sera très réduit et nous devrons déplacer moins souvent les éléments considérés.

III.4 Résolution du problème

Nous proposons de trier un tableau T en utilisant la méthode de tri Shell.
1- Analysez ce problème
2- Analysez chaque module proposé
3- Déduisez les algorithmes correspondants.

a) Analyse du programme principal :

Résultat : L'affichage du tableau trié est la tâche de la procédure **Afficher**.
Traitement :

- Nous devons donc trier le tableau ce qui sera la tâche de la procédure **Trier**.
- Pour pouvoir trier le tableau, nous devons tout d'abord le remplir ce qui se fera par la procédure **Remplir**.
- Pour remplir le tableau, nous devons connaître le nombre d'éléments à saisir ce qui est la tâche de la procédure **Saisir**.

b) Algorithme du programme principal

0) Début Tri_Shell
1) Proc Saisir (n)
2) Proc Remplir (T, n)
3) Proc Trier (T, n)
4) Proc Afficher (T, n)
5) Fin Tri_Shell

Dans ce qui suit, nous allons détailler l'analyse de la procédure **Trier**. Les autres modules ont été déjà traités lors de la résolution d'autres problèmes.

a) Analyse de la procédure Trier :

Résultat : Trier le tableau T.
Traitement :

- **L'action du tri :**

« Etant une variante du tri par insertion, ce tri consiste à trier séparément des sous tableaux du tableau initial objet du tri, formés par les éléments répartis de p éléments.»

Nous pouvons déduire de ce qui précède qu'il s'agit d'un traitement répétitif pour chacun des sous tableaux formés à partir d'une valeur du pas : **Tant que** (p ≠ 0) **Faire**

➢ **Pour** i de p à n **Faire** :
- valeur ← T[i]
- Décaler de la valeur du pas, vers la droite, les valeurs de T[c-p], T[c-2*p], etc.
- Affecter au dernier élément décalé le contenu de la variable Valeur

```
Tant que p ≠ 0 Faire
        p ← p div 3
        Pour i de p + 1 à n Faire
            valeur ← T[i]
            j ← i
            Tant que (j > p) ET (T[j-p] > valeur) Faire
                    T[j] ← T[j-p]
                        j ← j-p
            Fin Tant que

            T[j] ← valeur
        Fin pour
    Fin Tant que
```

- **Détermination de la valeur maximale du pas :**

Pour définir les valeurs successives du pas (que nous allons nommer p), Shell propose la relation suivante : $\mathbf{p_{k+1} = 3 * p_k + 1}$ avec 1 comme valeur de départ de p.

Pour avoir la valeur maximale du pas, nous allons utiliser une boucle à condition d'arrêt. Il est évident que la valeur du pas doit être plus petite que le nombre des éléments du tableau T. Nous aurons donc :

```
p ← 0
Tant que (p < n) Faire
   p ← 3*p + 1
Fin Tant que
```

b) Algorithme du programme principal

```
0) Début Procédure Trier (n : entier; VAR T : Tab)
1) p ← 0
Tant que (p < n) Faire
   p ← 3*p + 1
Fin Tant que
```

```
2) Tant que (p ≠ 0) Faire
        p ← p Div 3
        Pour i de p + 1 à n Faire
            valeur ← T[i]
            j ← i
            Tant que (j > p) ET (T[j-p] > valeur) Faire
                    T[j] ← T[j-p]
                    j ← j-p
            Fin Tant que
            T[j] ← valeur
        Fin pour
    Fin Tant que
3) fin Trier
```

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
p	Entier	Représente le pas
i, j	Entier	Compteurs
valeur	Entier	Variable intermédiaire

Activité

Traduisez la solution en un programme Pascal. Enregistrez votre travail sous le nom **Tri_Shel**.

IV. Applications

IV.1 Application 1 : Tri par fusion

IV.1.1 Principe

Il s'agit d'un tri suivant le paradigme "Diviser pour régner". Le principe est le suivant :

1- Nous divisons le tableau à trier en deux sous tableaux (en prenant par exemple, un élément sur deux pour chacun des sous tableaux).
2- Nous trions chacun d'entre eux.
3- Nous fusionnons les deux tableaux obtenus pour reconstituer le tableau trié.

Le principe est donc simple, en effet, pour trier une liste, nous la coupons d'abord en deux parties (de préférence de tailles égales ou proches). Ces deux parties sont triées puis fusionnées. La fusion construit une liste triée à partir des deux listes triées.

En effet, étant donnés deux tableaux d'éléments triés, de longueurs respectives L1 et L2, il est facile d'obtenir un troisième tableau d'éléments triés de longueur L1+L2, par

« interclassement » (ou fusion) des deux précédents tableaux.
Dans la suite de cette partie, nous allons nous intéresser à la présentation de l'action de fusion.

Remarque :

Nous constatons que la méthode de tri par fusion nécessite un tableau intermédiaire aussi grand que le tableau initial à trier et c'est là où réside le principal inconvénient car cela peut se révéler handicapant dans les situations où la place mémoire est restreinte.

IV.1.2 Exemple

Nous proposons d'utiliser la méthode de tri par fusion pour fusionner les deux tableaux d'entiers triés T1 et T2 de longueurs respectives 7 et 8.

T1	-5	-2	0	4	6	22	40
	1	2	3	4	5	6	7

T2	-7	0	8	8	12	28	35	39
	1	2	3	4	5	6	7	8

Le résultat sera rangé dans le tableau T.
Combien d'éléments devons-nous allouer pour le tableau T ? Justifiez votre réponse.

Bien sûr, nous allons déclarer un tableau T de 15 entiers (15 étant la somme des nombres d'éléments de T1 et de T2).

Etape 1

Nous allons commencer par :

- comparer le premier élément de chacun des deux tableaux T1 et T2
  Le plus petit est T2 [1] = -7
- placer -7 dans T [1] et se pointer à l'élément n°2 de T2
- se pointer à l'élément n°2 de T

T1	-5	-2	0	4	6	22	40
	1	2	3	4	5	6	7

T2	-7	0	8	8	12	28	35	39
	1	2	3	4	5	6	7	8

T	-7														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Remarquez que nous sommes toujours à la première position de T1.

Etape 2

Dans cette étape, nous allons :

- comparer le premier élément de T1 et le deuxième élément de T2
  Le plus petit est T1 [1] = -5
- placer -5 dans T [2] et se pointer à l'élément n°2 de T1
- se pointer à l'élément n°3 de T

T1	-5	-2	0	4	6	22	40
	1	2	3	4	5	6	7

T2	-7	0	8	8	12	28	35	39
	1	2	3	4	5	6	7	8

T	-7	-5													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Etape 3

Dans cette étape, nous allons :

- comparer T1 [2] et T2 [2]
  Le plus petit est T1 [2] = -2
- placer -2 dans T [3] et se pointer à l'élément n°3 de T1
- se pointer à l'élément n°4 de T

T1	-5	-2	0	4	6	22	40
	1	2	3	4	5	6	7

T2	-7	0	8	8	12	28	35	39
	1	2	3	4	5	6	7	8

T	-7	-5	-2												
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

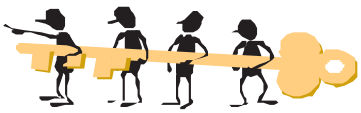
Activité

Réalisez manuellement la suite des étapes jusqu'à avoir placé tous les éléments de T1 et de T2 dans le tableau T.

IV.1.3 Résolution du problème

Nous proposons de trier un tableau T de b entiers ($5 \leq b \leq 100$) en utilisant la méthode de tri par fusion.

1- Analysez ce problème
2- Analysez chaque module proposé
3- Déduisez les algorithmes correspondants
4- Traduisez la solution en un programme Pascal.



a) Analyse du programme principal :

Résultat : L'affichage du tableau fusionné et trié T est la tâche de la procédure **Afficher.**

Traitement :

- Nous allons fusionner les deux tableaux triés T1 et T2 en appelant la procédure **Fusionner**
- Le remplissage des deux tableaux se fera suite à l'appel deux fois de la procédure **Remplir** qui réalisera :
  1- Le remplissage à tour de rôle des tableaux T1 et T2. Nous ferons de façon à ce que cette action soit effectuée en même temps que l'action de tri.
  2- La saisie du nombre d'éléments de chaque tableau.

b) Algorithme du programme principal

0) Début Tri_Fusion
1) Proc Remplir (T1, n1)
2) Proc Remplir (T2, n2)
3) Proc Fusionner (T1, n1, T2, n2, T)
4) Proc Afficher (T, n1+n2)
5) Fin Tri_Fusion

Dans ce qui suit, nous allons détailler l'analyse de la procédure **Remplir**

a) Analyse de la procédure Remplir :

Résultat : Remplir le tableau A de m entiers
Traitement :

- Le remplissage du tableau A par m entiers est une itération complète.
  Pour c de 2 à m Faire
- Pour chaque valeur du compteur, nous devons prévoir la saisie d'un entier qui doit être supérieur ou égal à son prédécesseur. Ceci nous amène à prendre en considération les contraintes suivantes :
  1- Commencer par saisir le premier élément A[1] avant d'entrer dans l'itération
  2- Pour chacun des éléments qui restent (de l'élément n°2 à l'élément n°m),
  Répéter
  Saisir A[c]
  Jusqu'à A[c] ≥ A[c-1]
- Saisir le nombre d'éléments du tableau est un traitement itératif à condition d'arrê
  Répéter
  Saisir m
  Jusqu'à m Dans [5,100]

Remarque :

Cette procédure sera appelée à deux reprises, une première fois pour remplir le tableau T1 par n1 entiers et une deuxième fois pour remplir le tableau T2 par n2 entiers.

b) Algorithme de la procédure Remplir :

0) Début Procédure Remplir (Var A : Tab ; Var m : entier)
1) **Répéter**
Lire (m)
Jusqu'à (m dans [5..100])
2) Lire (A[1])
Pour c de 2 à m **Faire**
Répéter
Lire (A[c])
Jusqu'à (A[c] ≥ A[c-1])
FinPour
3) Fin Remplir

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
c	Entier	Compteur

Détaillons maintenant l'analyse de la procédure **Fusionner.**

c) Analyse de la procédure Fusionner :

Résultat : Trier en fusionnant deux tableaux T1 et T2 de n1 et n2 entiers

Traitement :

- Le tri par fusion des deux tableaux T1 de n1 éléments et T2 de n2 éléments est une itération complète :

  Pour c de 1 à (n1+n2) Faire

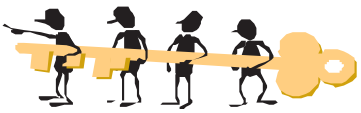
Pour chaque valeur du compteur, nous devons spécifier :
- Quel élément devons-nous ranger dans le tableau T ?
- A quel tableau appartient-il ?

- Il s'agit donc de l'action conditionnelle suivante :

Si T1[c1] < T2[c2] **alors**
ranger T1[c1] dans T[c]
incrémenter c1 de 1
Sinon
ranger T2[c2] dans T[c]
incrémenter c2 de 1

Activité

Que se passe-t-il si, par exemple, tous les éléments de T1 ont été rangés dans le tableau T alors que ceux de T2 ne le sont pas encore ?



En effet, la solution présentée précédemment ne prévoit pas ce cas, d'où la nécessité d'apporter la rectification suivante : Il s'agit toujours d'un traitement répétitif que nous devons décomposer en deux parties.

Partie1 : Ranger les éléments des tableaux T1 et T2 jusqu'à la fin de l'un des tableaux T1 ou T2.

```
[c ← 0, c1← 1, c2 ← 1]
        Répéter
                c ← c+1
                Si (T1 [c1] < T2 [c2]) Alors
                        T[c] ← T1 [c1]
                        c1 ← c1 + 1
                Sinon
                        T[c] ← T2 [c2]
                        c2 ← c2 + 1
        Jusqu'à (c1 > n1) OU (c2 > n2)
```

Partie 2 : Ranger le reste des éléments du tableau T1 ou T2 (il s'agit d'une action de copie)

```
Si (c1 > n1) Alors                {tous les éléments de T1 ont été rangés dans
                                   T, il reste à  copier les éléments de T2}

Pour i de c2 à n2 Faire
      T[c] ← T2[i]
      c ← c + 1
       Sinon
Pour i de c1 à n1 Faire           {copier le reste des éléments de T2}
      T[c] ← T1[i]
      c ← c + 1
```

Remarque :

Il est inutile de prévoir un paramètre pour définir le nombre d'éléments du tableau résultat T. En effet, la valeur de ce paramètre est la somme de n1 et n2.

d) Algorithme de la procédure Fusionner :

```
0) Début Procédure Fusionner (T1 : Tab; n1 : entier; T2 : Tab; n2 : entier; Var T : Tab)
1) c ← 0, c1 ← 1, c2 ← 1
2) Répéter
      c ← c + 1
      Si (T1[c1] < T2[c2]) Alors
         T[c] ← T1[c1]
         c1 ← c1 + 1
      Sinon
         T[c] ← T2[c2]
         c2 ← c2 + 1
      Fin si
   Jusqu'à (c1 > n1) OU (c2 > n2)

3) Si (c1 > n1) Alors
      Pour i de c2 à n2 Faire
         c ← c + 1
         T[c] ← T2[i]
      Fin Pour
   Sinon
            Pour i de c1 à n1 Faire
              c ← c + 1
              T[c]← T1[i]
            Fin Pour
   Fin Si
4) Fin Fusionner
```

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
i, c, c1, c2, i	Entier	Compteurs

Activité

Traduisez la solution en un programme Pascal. Enregistrez votre travail sous le nom **Tri_Fus1**.

Remarque :

Pour réduire le nombre de paramètres de la procédure Fusionner, nous allons donner la solution suivante qui interclasse deux suites d'éléments placés dans un tableau T, respectivement entre les indices **début** et **mil** et entre les indices **mil + 1** et **fin**. La solution fait appel à un tableau intermédiaire que nous appellerons Temp.

```
0) Début Procédure Fusionner (début, mil, fin : entier ; Var T : Tab)
1) i ← début, j ← mil + 1
2) Pour k de début à fin Faire
     Si ((j > fin) ou ((i ≤ mil) ET (T[i] < T[j]))) Alors
          Temp[k] ←T[i]
          i ← i + 1
     Sinon
          Temp[k] ← T[j]
          j ← j + 1
     Fin Si
   Fin Pour
3) Pour k de début à fin Faire   {copier le contenu de Temp dans T}
        T[k] ←Temp[k]
  Fin pour
4) Fin Fusionner
```

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
i, j, k	Entier	Compteurs
Temps	Tab	Tableau intérmédiaire

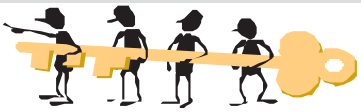
Activité

Traduisez la solution en un programme Pascal. Enregistrez votre travail sous le nom **Tri_Fus2**.

IV.2 Application 2 : Classement

Nous disposons des noms et des moyennes de 30 élèves d'une classe 4SI et nous proposons d'afficher sous forme de tableau le rang de chaque élève en plus des données déjà citées.

1. Définissez les structures de données à utiliser
2. Donnez l'analyse de ce problème.



1) Structure de données :

Comme structure de données, nous avons le choix d'utiliser :

1- un tableau d'enregistrements appelé ELEVE ayant la structure suivante :

ELEVE est un enregistrement de

- Nom de type chaîne de caractères
- Moyenne de type réel
- Rang de type entier

2- trois tableaux à une dimension pour ranger respectivement les noms, les moyennes et les rangs.

Dans la suite de la résolution, nous allons opter pour l'utilisation du tableau d'enregistrements appelé **TAB_EL**.

2) Analyse du programme principal :

Résultat : L'affichage du tableau TAB_EL sera effectué par la procédure **Afficher.**

Traitement :

- La détermination des rangs des élèves sera réalisée en appelant la procédure **Rechercher_Rang**
- La saisie des noms et des moyennes des 30 élèves se fera suite à l'appel de la procédure **Saisir**.

Algorithme du programme principal :

0) Début Traitement_EL
1) Proc Saisir (TAB_EL)
2) Proc Rechercher_Rang (TAB_EL)
3) Proc Afficher (TAB_EL)
4) Fin Traitement_EL

Tableau de codification des objets globaux

Objets	Type / Nature	Rôle
Tab_el	Tab	Tableau d'élèves
Saisir	Procédure	Permet de remplir un tableau de 30 élèves
Rechercher_Rang	Procédure	Permet de déterminer le rang de chaque élève
Afficher	Procédure	Permet d'afficher le tableau résultat

Dans ce qui suit, nous allons détailler l'analyse de chacune de ces procédures.

a) Analyse de la procédure Saisir :

Résultat : Remplir le tableau TAB_EL par les 30 enregistrements relatifs aux élèves de la classe 4SI.

Traitement :

- Le remplissage du tableau TAB_EL par 30 enregistrements est un traitement répétitif qui nécessite une itération complète :

Pour c de 1 à 30 Faire

Chaque valeur du compteur correspond au remplissage d'un enregistrement d'un élève donné qui comprend la saisie :

1- du nom
2- de la moyenne comme étant un réel compris entre 0 et 20.

N.B. : Puisqu'il s'agit d'un tableau d'enregistrements, la saisie se fait de la façon suivante :

1- remplir les champs nom et moyenne d'une variable temporaire de type enregistrement
2- copier la valeur de l'enregistrement dans le tableau TAB_EL.

b) Algorithme de la procédure Saisir :

```
0) Début Procédure Saisir (Var TAB_EL : Tab)
1) Pour c de 1 à 30 Faire
      Lire (Enreg.nom)
      Répéter
            Lire (Enreg.moy)
      Jusqu'à (Enreg.moy ≥ 0) et (Enreg.moy ≤ 20)
      TAB_EL[c] ← Enreg
   Fin Pour
2) Fin Saisir
```

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
c	Entier	Compteur
Enreg	Elève	Enregistrement intérmédiaire

Détaillons maintenant l'analyse de la procédure Afficher.

a) Analyse de la procédure Rechercher_Rang :

Résultat : Compléter le remplissage du tableau TAB_EL par les 30 rangs relatifs aux élèves de la classe 4SI.

Traitement :

➢ Le remplissage du tableau TAB_EL par les 30 rangs est une itération complète :

Pour i de 1 à 30 Faire

Pour chaque élément du tableau, nous allons compter le nombre d'enregistrement ayant une moyenne supérieure à la sienne. Cela nécessite un autre traitement répétitif sous la forme d'une itération complète :

Pour j de 1 à 30 Faire

Ce traitement répétitif comporte une action conditionnelle :

Si (Enreg2.Moy > Enreg1.Moy)
Alors Enreg1.Rang ← Enreg1.Rang + 1
FinSi

Une initialisation du rang à 1 doit être prévue avant cette itération complète :

Enreg1.Rang ← 1

b) Algorithme de la procédure Rechercher_Rang :

```
0) Début Procédure Rechercher_Rang (Var TAB_EL : Tab)
1)Pour i de 1 à 30 Faire
        Enreg1 ← Tab_el[i]
        Enreg1.Rang ← 1
        Pour j de 1 à 30 Faire
                Enreg2 ← Tab_el[j]
                Si (Enreg1.Moy > Enreg1.Moy)
                Alors Enreg1.Rang ← Enreg1.Rang + 1
                FinSi
        FinPour
   FinPour
2)Fin Rechercher_Rang
```

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
i, j	Entier	Compteurs
Enreg1, Enreg2	Elève	Enregistrements intérmédiaires

Détaillons maintenant l'analyse de la procédure Rechercher_Rang.

a) Analyse de la procédure Afficher :

Résultat : Afficher le tableau TAB_EL relatif aux élèves de la classe 4SI.

Traitement :

➢ L'affichage TAB_EL constitué de 30 enregistrements d'élèves est une itération complète : Pour i de 1 à 30 Faire

Chaque valeur du compteur correspond à l'affichage d'un enregistrement d'un élève qui comprend :

1- le nom
2- la moyenne
3- le rang

b) Algorithme de la procédure Afficher

0) Début Procédure Afficher (TAB_EL : Tab)
1) **Pour** i de 1 à 30 **Faire**
    Enreg ← Tab_el[i]
    Ecrire (“Nom de l'élève : ”, Enreg.Nom)
    Ecrire (“Moyenne de l'élève : ”, Enreg.Moy)
    Ecrire (“Rang de l'élève : ”, Enreg.Rang)
  FinPour
2) Fin Afficher

Tableau de codification des objets locaux

Objets	Type / Nature	Rôle
i, j	Entier	Compteurs
Enreg	Elève	Enregistrement intérmédiaire

Retenons

- **Tri par insertion :** Cet algorithme consiste à traiter une à une les valeurs du tableau et à les insérer, au bon endroit, dans le tableau trié constitué des valeurs précédemment traitées et triées.
  Les valeurs sont traitées dans l'ordre où elles apparaissent dans le tableau.

- **Tri Shell :** Ce tri, proposé en 1959 par Donald L. Shell, constitue une variante optimisée du tri par insertion.
  Le tri par insertion provoquait le décalage de tous les éléments plus grands que l'élément à insérer.

Dans le tri Shell, les éléments ne sont pas décalés d'un élément à la fois, mais de plusieurs éléments, dont la différence d'indice est appelée "pas". Ainsi, à chaque étape, le tri est dégrossit puis le pas est réduit. Chaque réduction de pas provoque un affinage du tri.

- **Tri fusion :** La méthode "diviser pour régner" est tout à fait applicable au problème de tri par fusion. Plutôt que de trier le tableau complet, il est possible de trier deux sous tableaux de taille égale, puis de fusionner les résultats.

- **D'autres méthodes de tri** existent telles que le tri par comptage, tri par création, tri radix, tri rapide, tri par permutation, etc.

Citation du Chapitre

« L'ordre conduit à toutes les vertus mais qu'est-ce qui conduit à l'ordre »

Georg Christoph Lichtenberg

Exercices

Exercice 1 (Bac TP 2005)

L'algorithme suivant est celui d'une fonction permettant de retourner la position du plus petit élément dans un tableau A de k éléments à partir d'une position p.

```
0) Deffn pos_min (A : tab ; p, k : entier): entier
1)pm  ←  p
  Pour i de p+1 à k Répéter
        Si (A[i] < A[pm])
          Alors pm  ←  i
        Finsi
  Fin pour
2) pos_min  ←  pm
3) Fin Pos_min
```

Utiliser la fonction **Pos_min** ci-dessus pour écrire un programme Pascal permettant de saisir un tableau **T** de **n** réels, le trier dans l'ordre croissant par la méthode de "tri par sélection" puis de l'afficher.

Exercice 2 (Bac TP 2005)

On propose ci-dessous l'algorithme d'une procédure de tri à Bulles :

```
0) DefProc TRI_Bulles (Var T : Tab; n : entier)
1) Pour i de 1 à n-1 Répéter
          Pour j de 1 à n-i Répéter
                Si  (T[j] < T[j+1])
               Alors Proc Permut (T[j], T[j+1])
               Fin si
          Fin Pour
    Fin Pour
2) Fin TRI_Bulles
```

Remarque :

Le module **Permut** (a, b) permute le contenu de deux entiers a et b.

Questions

1- Ecrire un programme Pascal intitulé **Tri** permettant de saisir **p** éléments entiers dans un tableau **V** et de faire appel au module **TRI_Bulles** ci-dessus.
2- Sous forme de commentaire, déterminer l'ordre du tri (croissant ou décroissant) accompli par le programme.
3- Dans le cas où le tableau **V** est déjà trié dès la saisie, les parcours effectués par le module **TRI_Bulles** s'avèrent inutiles. En effet, a**ucune permutation n'aura lieu au sein de ce module** dans ce cas.
Modifier la procédure TRI_Bulles pour tenir compte de cette contrainte.

Exercice 3 (Bac TP 2005)

Ecrire un programme Pascal intitulé Tri permettant de trier un tableau **T** de **N** entiers distincts (5 < N < 20) selon le principe suivant :
Pour chaque élément du tableau **T** :
- Déterminer le nombre d'éléments qui lui sont inférieurs.
- En déduire sa position au sein d'un autre tableau résultat appelé R.

Exemple : Pour un tableau T de 10 éléments :

6	2	0	5	12	25	13	8	14	3
1	2	3	4	5	6	7	8	9	10

Quatre valeurs sont inférieures au premier élément du tableau T. Cet élément sera donc placé à la position 5 du tableau **R.**

N.B : - Le candidat n'est pas appelé à vérifier que les éléments du tableau T sont distincts.

Exercice 4 (Bac TP 2005)

Ecrire un programme Pascal intitulé **Tri_permut** permettant de trier, dans l'ordre croissant, un tableau **T** de **N** éléments (5 < N < 20) selon le principe suivant :
- Effectuer toutes les permutations possibles sur les éléments du tableau T.
- Avant chaque permutation, faire appel à une fonction qui déterminera si la disposition des éléments de T est ordonnée ou non.
- Arrêter le traitement ci-dessus dès que le tableau T est trié.

Dans ce qui suit, l'algorithme de la procédure **Combinaisons** qui permet d'effectuer toutes les permutations possibles sur les N éléments du tableau **T**. Les procédures **Permut** et **Affiche** permettent respectivement de permuter deux éléments du tableau **T** et d'afficher le contenu de ce dernier.

0) Def Proc **Combinaisons** (N : entier; var T : Tab)
1) **Pour** i de 1 à N **Répéter**
    Pour j de 1 à N-1 **Répéter**
        Proc Permut(T[j], T[j+1])
        Proc Affiche(T, N)
    Fin pour
  Fin pour
2) Fin Combinaisons

Exercice 5

Tri par rangement ou par création

Soit un vecteur de n entiers.

Nous cherchons à trier ce vecteur par ordre croissant par une méthode dite du rangement. Cette méthode est décrite par les étapes suivantes :

1- Trouver le minimum local de la partie restant à trier
2- Mettre le minimum dans un vecteur résultat
3- Recommencer jusqu'à avoir rangé tous les éléments

Questions

Proposez une analyse modulaire au problème dans chacun des cas suivants :

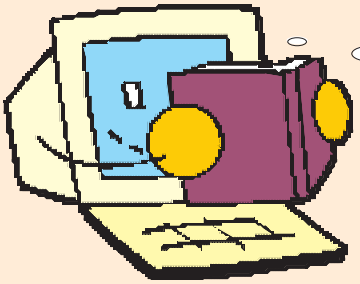
- Le vecteur initial peut être modifié
- Le vecteur initial reste inchangé

Exercice 6

a) Proposez une analyse modulaire en utilisant un procédé récursif, au problème permettant de trier un vecteur d'entiers par la méthode de tri **'Par fusion'**,

b) Déduisez les algorithmes correspondants,

c) Traduisez et testez la solution obtenue. Enregistrez votre programme sous le nom **Fus_Rec**.



Lecture

LES TRIS

TRIS INTERNES

Tri par insertion.

- *Tri par insertion linéaire*
- *Tri par insertion en double sens*
- *Tri Shell*
- *Tri par arbre binaire*

Tri par échange.

- *Tri à bulle*
- *Tri cocktail shaker*
- *Tri par la méthode de K.E.Batcher*
- *Quicksort*

Tri par sélection, par fusion, par distribution

TRIS EXTERNES

- *Tri polyphase*

www.chez.com/algo/tri/