Université Félix Houphouët-Boigny
(UFHB)

Cours d'algorithme avancée
L3-Info

Enseignant :

Dr GBAME Gbede Sylvain

Assistant, enseignant chercheur à l'UFHB

gbamegbedesylvain@gmail.com

MODULE : Algorithme avancée

PLAN

Chapitre 5
Les tris

Chapitre 5 : Les tris

5.1 Notion de tris

1. Définitions

❖ Un tri est une opération qui permet d'organiser les éléments d'un ensemble fini de données selon un ordre déterminé ou selon un ensemble de critères.

❖ Un algorithme de tri est un algorithme qui permet d'ordonner un vecteur homogène d'éléments selon un ordre fixé.

2. Les deux principaux ordres de tri

Les éléments à trier doivent être au préalable homogène afin d'être stockés dans un vecteur.
Les ordres de tri les plus utilisés sont *l'ordre numérique* (qui porte sur les nombres) et *l'ordre lexicographique* (qui porte sur les caractères et les chaînes de caractères).
Les éléments à trier appartiennent à un ensemble E muni *d'une relation d'ordre noté* ≤ (ou ≥).
Un vecteur est dit trié ou ordonné s'il existe une relation d'ordre entre ses éléments.
Il existe *deux principaux ordres* de tri.

Chapitre 5 : Les tris

5.1 Notion de tris

On parle de :

- ***tri croissant*** lorsque l'élément de la cellule d'indice i est inférieur ou égal à l'élément de la cellule d'indice ***i + 1*** ;

On dit alors que les éléments du tableau sont triés dans l'ordre croissant.

- ***tri décroissant*** lorsque l'élément de la cellule d'indice i est supérieur ou égal à l'élément de la cellule d'indice ***i + 1***.

On dit alors que les éléments du tableau sont triés dans l'ordre décroissant

5.1 Notion de tris

3. Algorithmes de tri

Les algorithmes de tri sont nombreux. Trois algorithmes classiques de tri seront présentés : ***le tri par sélection***, ***le tri à bulles*** et ***le tri par insertion***.

Ce sont des ***algorithmes de tri*** en place, c'est-à-dire que le tri s'effectue sans faire intervenir un autre tableau mais plutôt une variable scalaire comme variable temporaire.

Dans un algorithme de tri en place, au départ, on a un vecteur non trié, et à la fin du traitement, ce vecteur est trié.

3.1. le tri par sélection

Le principe est le suivant :

- d'abord rechercher le plus petit élément du vecteur ;
- ensuite le placer dans la première cellule ;
- puis rechercher le second plus petit élément ;
- puis le placer dans la deuxième cellule ;
- et ainsi de suite jusqu'à parcourir tout le vecteur.

5.1 Notion de tris

Chaque fois que l'on déplace le minimum relatif, on procède à un échange de cellule entre le minimum relatif et l'élément qui occupe la cellule où l'on désire stocker le minimum.
De plus, au fil des échanges de positions, la partie du vecteur où la recherche s'effectue "***se réduit***" car on n'effectue pas de recherche de minimum "***en arrière***"

Chapitre 5 : Les tris

5.1 Notion de tris

Exemple : tri croissant par recherche des minima successifs sur le vecteur ci-dessous :

6	5	-1	0	3

Le plus petit élément est dans la cellule 3. On le repositionne en cellule 1, et l'élément de cette cellule vient occuper la cellule 3 :

-1	5	6	0	3

Après le premier passage, on considère le vecteur à partir de sa deuxième cellule. Le plus petit élément de cette partie se trouve dans la cellule 4. Il y a échange de cellule avec l'élément occupant la cellule 2. On obtient :

-1	0	6	5	3

5.1 Notion de tris

On considère maintenant les trois dernières cellules. Il y a échange de cellules entre les éléments des cellules 3 et 5 :

-1	0	3	5	6

On considère maintenant les deux dernières cellules. Il n'y a pas d'échange de cellules entre les éléments des cellules 4 et 5, car l'ordre est respecté :

-1	0	3	5	6

Il reste maintenant la dernière cellule, la cellule 5. L'élément de cette cellule est supérieur ou égal à tous les éléments qui le précèdent dans le vecteur, et qui eux-mêmes sont rangés selon l'ordre fixé. Il n'y aura alors pas d'échange. Donc le vecteur final est :

-1	0	3	5	6

Le vecteur est trié lorsque le nombre d'éléments non triés vaut 1.

Chapitre 5 : Les tris

5.1 Notion de tris

Considérons les variables : **T,** un vecteur de **n** éléments **; i**, l'indice de parcours de **T** ; **j**, l'indice relatif de parcours de **T** ; **indmin,** l'indice minimal relatif de **T** ; **temp**, une variable temporaire. Le bloc d'instructions ci-dessous permet d'effectuer un **tri croissant** par **recherche des minima** successifs sur un vecteur de n éléments :

```
Pour (i ← 1 à (n - 1)) Faire
        indmin ← i
        Pour (j ← (i + 1) à n) Faire
                Si (T[j] < T[indmin])
                Alors indmin ← j
                FinSi
        FinPour j
        Si (i ≠ indmin)
        Alors temp ← T[i]
                T[i] ← T[indmin]
                T[indmin] ← temp
        FinSi
FinPour i
```

Chapitre 5 : Les tris

5.1 Notion de tris

On peut aussi effectuer un tri décroissant en s'inspirant de la méthode du tri croissant.
Si l'on procède par :

- recherche des minima successifs, on stocke le minimum à la dernière cellule.
- recherche des maxima successifs, on stocke le maximum à la première cellule.

3.2. le tri à bulles

On parcourt le vecteur en comparant les éléments deux à deux consécutifs.
Lorsque deux éléments successifs ne sont pas ordonnés, on les échange.

Si au moins un échange a eu lieu pendant le parcours du vecteur, on effectue un nouveau parcours. On recommence jusqu'à ce qu'il n'y ait plus d'échange. S'il n'y a pas eu d'échange pendant un parcours, cela signifie que le vecteur est trié. On effectue donc des itérations.

5.1 Notion de tris

Après le premier passage, le plus grand élément sera placé dans la dernière cellule du tableau, dans la bonne cellule. Pour le passage suivant, on n'effectue pas de comparaison avec le dernier élément. Ainsi à chaque passage, le nombre de valeurs à comparer diminue de 1.
Pour savoir s'il y a eu échange ou pas, on peut utiliser une variable booléenne qui changera de valeur s'il n'y a pas d'échange.

Exemple : trier la collection d'entiers (5, 1, 4, 2, 0).

Les éléments à comparer sont en gras.

Première étape ou premier passage :

(**5 1** 4 2 0) → (**1 5** 4 2 0) : on compare 5 et 1, puisque 5 > 1 alors échange
(1 **5 4** 2 0) → (1 **4 5** 2 0) : on compare 5 et 4, puisque 5 > 4 alors échange
(1 4 **5 2** 0) → (1 4 **2 5** 0) : on compare 5 et 2, puisque 5 > 2 alors échange
(1 4 2 **5 0**) → (1 4 2 **0 5**) : on compare 5 et 0, puisque 5 > 0 alors échange

5.1 Notion de tris

Il y a eu au moins un échange, alors on recommence

Deuxième étape ou deuxième passage :

(**1 4** 2 0 5) → (**1 4** 2 0 5) : puisque 1 < 4 alors pas d'échange

(1 **4 2** 0 5) → (1 **2 4** 0 5) : on compare 4 et 2, puisque 4 > 2 alors échange

(1 2 **4 0** 5) → (1 2 **0 4** 5) : on compare 4 et 0, puisque 4 > 0 alors échange

Il y a eu au moins un échange, alors on recommence

Troisième étape ou troisième passage :

(**1 2** 0 4 5) → (**1 2** 0 4 5) : puisque 1 < 2 alors pas d'échange

(1 **2 0** 4 5) → (1 **0 2** 4 5) : puisque 2 > 0 alors échange

Il y a eu au moins un échange, on recommence

5.1 Notion de tris

Quatrième étape ou quatrième passage

(1 0 2 4 5) → (0 1 2 4 5) : puisque 1 > 0 alors échange

À l'issue de ce passage, le vecteur est trié, mais pour le montrer, on effectue un passage de contrôle. On constate que les éléments sont deux à deux ordonnés, il n'y a alors aucun échange. On arrête le traitement d'où le résultat : (5, 1, 4, 2, 0) → (**0, 1, 2, 4, 5**).

Considérons les variables : **T**, un vecteur de n éléments ; **i,** l'indice de parcours de **T** ; **j**, l'indice de parcours relatif de **T** ; temp, une variable temporaire.
Le bloc d'instructions ci-dessous permet d'effectuer ***un tri croissant*** sur un vecteur de n éléments

5.1 Notion de tris

```
Pour (i ← 1 à (n-1)) Faire
        j ← n
        TantQue (j ≠ i)
                Si (T[j] < T[j-1])
                Alors temp ← T[j]
                        T[j] ← T[j-1]
                        T[j-1] ← temp
                FinSi
                j ← j-1
        FinTantQue
FinPour
```

5.1 Notion de tris

3.3. le tri par insertion

On parcourt le vecteur du début à la fin en insérant chaque élément rencontré (ou élément courant) à la bonne position sans écraser l'élément initialement stocké à cette position. Le décalage s'effectue vers la droite jusqu'à trouver le point d'insertion. Ainsi, tous les éléments qui précèdent l'élément courant sont triés. Pour insérer l'élément courant à la bonne place, on le compare à ses précédents, puis on décale les éléments afin d'effectuer l'insertion.

Exemple : trier la collection d'entiers (9, 6, 1, 4, 8).

Au départ, l'élément courant est le deuxième élément de la collection. Pour la suite de l'exercice, l'élément courant est en gras, et les éléments déjà triés sont soulignés.

Étape 1 : (9 **6** 1 4 8) → (6 9 1 4 8)
Étape 2 : (6 9 **1** 4 8) → (1 6 9 4 8)
Étape 3 : (1 6 9 **4** 8) → (1 4 6 9 8)
Étape 4 : (1 4 6 9 **8**) → (1 4 6 8 9)

Chapitre 5 : Les tris

5.1 Notion de tris

À l'issue de ce passage, le tableau est trié. D'où le résultat : (9, 6, 1, 4, 8) → (1, 4, 6, 8, 9).

Considérons les variables : T, un vecteur de n éléments ; i, l'indice de parcours de T ; j, l'indice de parcours de la partie non triée de V ; temp, la variable contenant l'élément courant à insérer.

Le bloc d'instructions ci-dessous permet d'effectuer un tri croissant sur un vecteur de n éléments :

```
Pour (i ← 2 à n) Faire
        temp ← T[i]     // mémorisation de l'élément courant (élément à insérer)
        // décalage des éléments jusqu'à trouver le point d'insertion
        j ← i + 1
        TantQue ((j ≥ 1) ET (T[j] > temp)) Faire
                T[i + i] ← T[i]
                j ← j - 1
        FinTantQue
        T [j + i] ← temp  // insertion de l'élément courant à sa place
FinPour
```

Chapitre 5 : Les tris

5.1 Notion de tris

Le bloc d'instructions pour effectuer un tri décroissant :

```
Pour (i ← 2 à n) Faire
        temp ← T[i]
        j ← i + 1
        TantQue ((j ≥ 1) ET (T[j] < temp)) Faire
                T[i + i] ← T[i]
                j ← j - 1
        FinTantQue
        T [j + i] ← temp
FinPour
```

5.1 Notion de tris

4. Algorithmes de recherche dans un vecteur

On dispose de deux méthodes de recherche d'un élément dans un vecteur : la recherche séquentielle et la recherche dichotomique.

4.1. la recherche séquentielle

Supposons que l'on recherche une seule occurrence d'un élément d'un vecteur. Cette méthode consiste à effectuer la recherche, à l'aide de la boucle Tant Que, en allant de la première cellule du vecteur à la dernière. La recherche s'arrête lorsque l'élément est trouvé ou lorsque l'on a atteint la dernière cellule du vecteur.

Si le vecteur est de grande taille et que de plus, ses éléments sont triés dans l'ordre croissant ou dans l'ordre décroissant, cette méthode se révèle inefficace. On la qualifie alors de méthode naïve ou même de méthode bestiale.

Lorsque le vecteur est trié, on peut considérablement améliorer la recherche en effectuant une recherche par dichotomie ou une recherche dichotomique.

5.1 Notion de tris

4.2. la recherche dichotomique

La ***recherche dichotomique*** s'applique sur un vecteur trié. C'est un procédé itératif au cours duquel, à chaque boucle, on divise le vecteur en deux parties égales ou presque, et on recherche l'élément dans la partie appropriée.

La méthode de la recherche dichotomique appliquée sur un vecteur trié dont l'indice minimal est ***borninf***, et l'indice maximal est ***bornsup*** :

Chapitre 5 : Les tris

5.1 Notion de tris

1. rechercher l'élément stocké à l'indice médian du vecteur
(on obtient l'indice médian du vecteur en effectuant le calcul (*borninf + bornsup*) DIV 2)

2. comparer l'objet recherché et l'objet médian. Trois possibilités se présentent :

a. l'élément recherché est égal à l'objet médian : on arrête la recherche

b. l'élément recherché est supérieur à l'objet médian : s'il existe, l'élément recherché est stocké dans une cellule de la partie droite du vecteur. Appliquer de nouveau le **point 1**, borninf change de valeur et vaut indice médian **+ 1**. Ensuite appliquer le **point 2**.

c. **l'élément recherché est inférieur à l'élément médian** : s'il existe, l'élément recherché est stocké dans une cellule de la partie gauche du vecteur. Appliquer de nouveau **le point 1**, *bornsup* change de valeur et vaut *indice médian* **– 1**. Ensuite appliquer **le point 2**.

Chapitre 5 : Les tris

5.1 Notion de tris

Le bloc d'instructions ci-dessous permet d'effectuer une recherche dichotomique sur le vecteur T [1:8] trié dans l'ordre croissant :

```
borninf ← 1
bornsup ← 8
trouve ← Faux
écrire ("Donnez l'élément à rechercher : ")
lire (valeur)
Répéter
        indmedian ← (borninf + bornsup) DIV 2
        Si (T[indmedian] = valeur)
        Alors   trouve ← Vrai
        Sinon   Si (valeur  >  T[indmedian])
                Alors borninf ← indmedian  + 1
                Sinon bornsup ← indmedian  - 1
                FinSi
        FinSi
Jusqu'à ((trouve  =  Vrai) OU (borninf  >  bornsup))
Si (trouve = Vrai)
Alors écrire ("L'élément ", valeur, " est au rang ", indmedian)
Sinon écrire ("L'élément ", valeur, " n'est pas dans le tableau")
FinSi
```

FIN DU COURS