Lycée Secondaire Raccada

Chapitre 4

Les traitements avancés

Leçon 1

Les méthodes de tri

4ème Sc

Enseignant: Sami MEJRI

2015-2016

Introduction

Un algorithme de tri permet de mettre dans un ordre croissant ou décroissant des valeurs stockées dans une structure de donnée comme un tableau en mémoire centrale. Ces valeurs peuvent être de type numérique ou caractères. On recense une dizaine de méthodes de tri.

I- Tri par sélection

1- Activité

Faire l'analyse d'un programme qui permet de trier un tableau T de n éléments dans l'ordre croissant. La valeur maximum de n est 50.

2- principe

1- Comparer tous les éléments de la liste non triée afin de sélectionner le plus petit.
2- Permuter le plus petit élément avec le premier élément de la liste non triée.
3- Répéter les étapes 1 et 2 un nombre de fois égal à n-1, en considérant toujours la liste non triée uniquement.

Animation

a. Analyse du programme principal

Résultat = PROC affichage (T, n)
T = PROC tri-sélection (T, n)
T = PROC remplir (T, n)
n = PROC saisie (n)
Fin tri

T.D.N.T

Type
Tab = tableau de 50 entiers

Algorithme

0) Début tri
1) PROC saisie (n)
2) PROC remplir (T, n)
3) PROC tri-sélection (T, n)
4) PROC affichage (T, n)
5) Fin tri

T.D.O.G

Objet	*Type / nature*	*Rôle*
n	Entier	Taille du tableau
T	Tab	Tableau d'entiers
affichage	Procédure	Affichage du tableau trié
tri	Procédure	Trier le tableau
remplir	Procédure	Remplissage de tableau
saisie	Procédure	Saisie contrôlé de n

b. Analyse de la procédure tri-sélection (Ordre croissant)

```
DEF PROC tri-sélection (var T : tab, n : entier)
Résultat = T
T = [ ]
Pour i de 1 à n-1 faire
     pmin← FN recherche_pmin (T, n, i)
    Si i< > pmin alors
    PROC permuter (T[i], T[pmin])
    Fin Si
Fin pour
i = compteur
```

Algorithme

```
0) DEF PROC tri-sélection(var T : tab, n : entier)
1) Pour i de 1 à n-1 faire
   pmin← FN recherche_pmin (T, n, i)
   Si i< > pmin alors
   PROC permuter (T[i], T[pmin])
   Fin Si
    Fin pour
2) Fin tri
```

T.D.O.Locaux

Objet	*T/ N*	*Rôle*
i	Entier	Compteur
permuter	Procédure	Permuter deux entiers
recherche_ pmin	Fonction	recherche position de min
pmin	Entier	Position de minimum

c. Analyse de la fonction recherche_pmin

```
DEF FN recherche_pmin (T: tab, n: entier, i: entier): entier
Résultat  =  recherche_pmin← p

      p = [p← i] pour j de i+1 à n faire
Si T[p] > T[j] alors
p← j
Fin si
Fin pour
j = compteur
```

Algorithme

```
0) DEF FN recherche_pmin (T: tab, n: entier, i: entier): entier
1) p← i
Pour j de i+1 à n faire
    Si T[p] > T[j]
       alors p← j
   Fin si
Fin pour
2) recherche_pmin← p
3) Fin recherche_pmin
```

T.D.O.Locaux

Objet	*Type / nature*	*Rôle*
j	Entier	Compteur
p	Entier	Position de minimum

d. Analyse de la procédure permuter

DEF PROC permuter (var a : entier, var b : entier)

Résultat = (a, b)

p← a
a← b
b← p

Algorithme

0) DEF PROC permuter (var a : entier, var b : entier)
1) p← a
2) a← b
3) b← p
4) Fin permuter

T.D.O.Locaux

Objet	*Type / nature*	*Rôle*
p	Entier	Variable tampon

Algorithme de procédure tri-séléection

```
0) DEF PROC tri-sélection (var T:tab, n:entier)
1)  Pour i de 1 à n-1 faire
    min← i
        Pour j de i+1 à n faire
            si (T[j] < T[min]) alors
                min← j
            Fin Si
        Fin Pour
   Si (i< > min) alors
        aux← T[i]
        T[i]← T[min]
        T[min] ← aux
   Fin Si
 Fin Pour
2) Fin tri-sélection
```

e. Analyse de la procédure affichage

DEF PROC affichage (T : tab, n : entier)
Résultat = Affichage
Affichage = [ ] pour c de 1 à n faire
Ecrire (T[c])
Fin pour
c = compteur

Algorithme

0) DEF PROC affichage (T : tab, n : entier)
1) pour c de 1 à n faire
Ecrire (T[c])
Fin pour
2) Fin affichage

T.D.O.Locaux

Objet	*Type / nature*	*Rôle*
c	Entier	compteur

II- Tri à bulles

1- Activité 2

Faire l'analyse d'un programme qui permet de trier un tableau T de n éléments dans l'ordre croissant. La valeur maximum de n est 50.

2- Principe

1- Comparer le contenu des cases n°1 et 2; s'ils ne sont pas dans le bon ordre, les inverser.
2- Faire de même pour les cases n°2 et 3, 3 et 4 et ainsi de suite jusqu'à (n-1 et n).
3- Si au moins une inversion a été effectuée, recommencer au début du tableau.
Le processus s'arrête lorsqu'au cours du balayage complet du tableau, aucune inversion n'a été effectuée (on effectue donc toujours un tour pour rien)

Animation

Analyse de la procédure tri_bulles :

DEF PROC tri_bulles (var T : tab, n : entier)

Résultat = T
T = [ ] Répéter
[changer← faux]
pour i de 1 à n-1 faire
Si T [i] > T [i+1] alors
PROC permuter (T[i], T [i+1])
changer← vrai
Fin si
Fin pour
Jusqu'à (change = faux)
i = compteur

T.D.O.Locaux

Objet	*Type / nature*	*Rôle*
Changer	Booléen	Test l'état de permutation
i	Compteur	
permuter	Procédure	

Algorithme

0) DEF PROC tri_bulles (var T : tab, n : entier)
1) Répéter
changer← faux
pour i de 1 à n-1 faire
Si T [i] > T [i+1] alors
PROC permuter (T[i], T [i+1])
changer← vrai
Fin si
Fin pour
Jusqu'à (change = faux)
2) Fin tri_bulles

III- Tri par Insertion

1- Activité

Faire l'analyse d'un programme qui permet de trier un tableau T de n éléments dans l'ordre croissant. La valeur maximum de n est 50.

2- Principe

on suppose que les i-1 premiers éléments sont triés

- **On chercher la position du $i^{ème}$ élément dans la partie du tableau commençant de 1 à i.**
- **Si cette position est i, l'élément est donc à sa bonne place.**
- **Sinon, supposant que cette position est j; ce j est forcement entre 1 et i-1.**

(1) on affecte T[i] à une variable auxiliaire tmp,

(2) On décale d'un pas vers l'avant (à droite) tous les éléments de j à i-1

(3) puis on insère l'élément d'indice i (précédemment sauvegardé dans tmp) à la position j.

On commence ce procédé à partir du 2ème éléments i=2 jusqu'à atteindre la fin du tableau.

Remarque:

On fait au maximum (n-1) + (n-2)+ (n-3)+…..+2+1= n(n-1)/2 comparaisons et autant de décalages.

Animation

Analyse de la procédure tri_insertion :

```
DEF PROC tri_insertion (var T : tab, n : entier)
Résultat = T
T = [ ]
pour i de 2 à n faire
    [p← T [i], j← i]
Tant que (j – 1 ≥ 1) et (p < T[ j – 1]) faire
    T [j] ← T [j – 1]
    j← j–1
Fin Tant que
    T[j]← p
Fin pour
i = compteur
```

T.D.O.Locaux

Objet	Type / nature	Rôle
i	Entier	Compteur
j	Entier	Compteur
p	Entier	Entier provisoire

Algorithme

```
0) DEF PROC tri_insertion (var T :
tab, n : entier)
1) Pour i de 2 à n faire
    [p← T [i], j← i]
Tant que (j – 1 ≥ 1) et (p < T[ j – 1])
faire
    T [j] ← T [j – 1]
    j← j–1
Fin Tant que
T[j] ← p
Fin pour
2) Fin tri_ insertion
```

Merci pour votre attention