



INFO0101

INTRODUCTION À L'ALGORITHMIQUE ET À LA PROGRAMMATION

COURS 4

TABLEAUX



Pierre Delisle, Cyril Rabat, Christophe Jaillet et Jessica Jonquet
Département de Mathématiques et Informatique
Septembre 2020

Plan de la séance

- Tableaux
 - Définition
 - Accès
 - Algorithmes
- Les tableaux en Java
- Algorithmique et tableaux
 - Recherche
 - Tri

Problème : trouver le maximum entre 4 (en général n) nombres

Algorithme Max4

Déclarations

Variables locales

a, b, c, d, max : entier

Début

a ← lire() //idem pour b, c, d

max ← a

Si b > max Alors

max ← b

FinSi

Si c > max Alors

max ← c

FinSi

Si d > max Alors

max ← d

FinSi

écrire(max)

Fin

Remarques

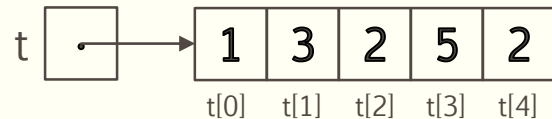
- Dans cet algorithme
 - On recommence plusieurs fois le même test
 - Mais on ne peut pas utiliser de boucle
 - Car les tests portent sur des variables différentes
- Solution
 - On crée une structure commune pour l'ensemble des variables et on en numérote chaque composant
 - Cette structure est appelée un *tableau*. Tout élément de la structure est une variable qui est accessible par son *indice*

Tableau

- Un tableau (ou vecteur) est une suite d'éléments de même type, stockés dans des blocs de mémoire contigus, et désignés sous un nom commun

▪ t :

- tableau d'entiers initialisé à 1, 3, 2, 5, 2



- Les cases d'un tableau sont numérotées
 - À partir de 0
 - Jusqu'à $n - 1$ (n désignant la taille du tableau)

- On peut accéder dans un ordre quelconque à n'importe quel élément : accès direct

- t[1] a pour valeur 3

- t[4] a pour valeur 2

- t[-4] n'est pas défini, de même que t[5]

- t[i] donne l'accès (la valeur) au (i+1)-ième élément du tableau

- Il est possible d'affecter une valeur à t[i]
 - t[3] = 0

Déclaration, allocation et affectation

- Déclaration d'un tableau
 - t : tableau de $\langle \text{type primitif} \rangle$
- On doit toujours préciser la taille d'un tableau et l'allouer avant de lire ou écrire dedans
 - $t \leftarrow \text{allouer}(10)$
- La fonction *allouer* réserve le nombre de cases passé en paramètre et retourne une référence sur le tableau ainsi créé
 - Si t référençait un autre tableau avant l'appel, cette référence est perdue
- Important : l'affectation d'une variable tableau à une autre affecte la référence du tableau, elle ne copie pas le tableau
 - Si on veut copier le contenu d'un tableau dans un autre, il faut copier les éléments un par un

Retour sur l'algorithme max4

- Exemple 1
 - Avec un tableau, on peut contenir tous les nombres dans une seule structure
 - On utilise une seule variable : seul l'indice de l'élément du tableau varie
- Exemple 2
 - Comment améliorer l'algorithme ?
 - État de la mémoire durant l'exécution

Parcours d'un tableau

- La méthode de parcours la plus simple consiste à parcourir le tableau dans l'ordre croissant de ses indices : accès séquentiel
 - Boucle Pour allant de 0 à $\text{taille}(t) - 1$
- Exemple 3 : Procédure afficherTab

Recherche séquentielle dans un tableau

- Principe : on parcourt le tableau jusqu'à ce que l'une des deux propriétés soit vraie :
 - L'élément courant est la valeur recherchée
 - Le tableau a été entièrement parcouru
- Si $t = \{9, 2, 5, 7, 3\}$
 - `rechercher(t, 7)` vaudra 3
 - `rechercher(t, 9)` vaudra 0
 - `rechercher(t, 8)` vaudra -1 (convention)
- On recherche la valeur v dans le tableau
 - Si v est dans le tableau
 - résultat : l'indice où est stocké v
 - Si v ne figure pas dans le tableau
 - résultat = -1 (convention)
- Exemple 4
 - Fonction `rechercher` qui effectue la recherche séquentielle dans un tableau passé en paramètre

Passage de tableau en paramètre et en retour de fonction

Passage en paramètre

- Pas de recopie du tableau
- Seule la référence du tableau est passée en paramètre
 - Les modifications d'un tableau à l'intérieur d'une fonction/procédure sont permanents
- Exemple 5 : échange de 2 éléments d'un tableau
 - État mémoire

Retour

- Seule la référence du tableau est retournée au programme appelant
- Le tableau sera généralement déclaré, alloué et initialisé dans la fonction
- Si un tableau est passé en paramètre et modifié, on n'a pas à le retourner
- Exemple 6 : Création d'un tableau inversé



LES TABLEAUX EN JAVA

Les tableaux en Java

- Occupent une position intermédiaire entre les données *primitives* (entiers, ...) et les *objets*
- À tout type de valeurs correspond un type de tableaux, obtenu en suffixant [] à son nom
- Exemples
 - `int []` déclare un tableau d'entiers
 - `boolean []` déclare un tableau de booléens
- Déclaration d'un tableau
- Exemple

```
type_des_valeurs [ ] nom_du_tableau;
```

```
int [ ] t;
```

Création de tableau en Java

- Important : La création d'un tableau en Java se fait en 2 étapes

- Déclaration du tableau
- Allocation du tableau

- Déclaration d'un tableau t d'entiers

```
int [ ] t;
```

- Aucun objet n'est créé : une référence de nom t est créée pour désigner le tableau
- Pour créer le tableau, il faut utiliser l'opérateur new, qui alloue le tableau et retourne une référence vers celui-ci

```
t = new int [10];
```

- Une fois le tableau déclaré et alloué, il est nécessaire d'initialiser ce tableau

```
t[0] = 8;  
t[1] = 5;  
...  
t[9] = 3;
```

- On peut regrouper les étapes de déclaration, d'allocation et d'initialisation

```
boolean [] tab;           //déclaration sans allocation  
int [] u = new int [3];    //déclaration et allocation  
int [] v = {1, 3, 2};      //déclaration, allocation et initialisation  
                           //équivalent à int [] v; v= new int[3];  
                           //v[0]=1; v[1]=3; v[2]=2;
```

Utilisation des tableaux en Java

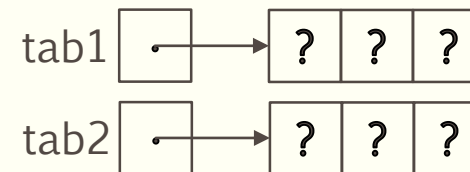
- La taille d'un tableau (le nombre d'éléments) est fixée lors de sa création et ne peut être modifiée ultérieurement (en principe !)
 - Obtention de la taille par l'appel au champ *length*
- Les tableaux sont indexés à partir de 0
 - Le dernier élément du tableau *t* est donc accessible en position *t.length - 1*

```
int [] t = {1, 3, 2};  
System.out.println ("Longueur de t : " + t.length);
```

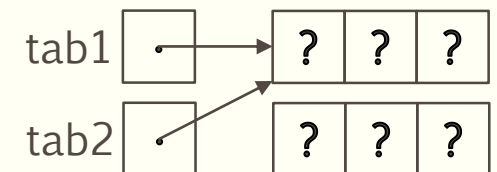
Les tableaux et les fonctions en Java

- Une fonction en Java reçoit une copie de la valeur des paramètres qui lui sont transmis
 - On parle de passage des paramètres par *valeur*
 - Ces paramètres se comportent comme des variables locales
- La zone mémoire associée à une variable d'un type de base contient la valeur de la variable
- Si un paramètre est d'un type de base, la modification de sa valeur n'a de portée qu'à l'intérieur du corps de la fonction
- Les tableaux en Java sont considérés comme des *objets*
- Comme pour tous les objets, les variables associées à des tableaux ne contiennent pas directement les valeurs du tableau, mais une référence vers les éléments du tableau

```
int [] tab1 = new int [3];  
int [] tab2 = new int [3];
```



```
tab2 = tab1;
```

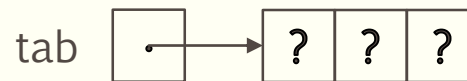


Les tableaux et les fonctions en Java

- Comme pour les données primitives, quand un tableau est transmis à une fonction, la fonction reçoit une copie de la valeur du paramètre
- Pour les tableaux, la fonction reçoit donc une *copie de la référence du tableau*
 - Et non une copie de chaque valeur du tableau
- Si un paramètre est un tableau, on peut alors modifier le contenu de ce tableau

```
int [] tab;  
tab = new int [3];  
remplirTab(tab);
```

État de la mémoire à
l'appel de la procédure



Mémoire locale à la
fonction remplissage



Après l'appel de la
procédure





ALGORITHMES SUR LES TABLEAUX

Recherche, tri

Rappel : Recherche séquentielle dans un tableau

- Principe : on parcourt le tableau jusqu'à ce qu'une des deux propriétés soit vraie :
 - L'élément courant est la valeur recherchée
 - Le tableau a été entièrement parcouru
- Si $t = \{9, 2, 5, 7, 3\}$
 - `rechercher(t, 7)` vaudra 3
 - `rechercher(t, 9)` vaudra 0
 - `rechercher(t, 8)` vaudra -1 (convention)
- On recherche la valeur v dans le tableau
- Si v est dans le tableau
 - résultat : l'indice où est stocké v
- Si v ne figure pas dans le tableau
 - résultat = -1 (convention)
- Observation : si v ne figure pas dans le tableau, il faudra parcourir tout le tableau pour s'en rendre compte

Recherche dans un tableau trié

- Un tableau $t[0..n-1]$ est trié par ordre croissant si
 - Pour $i, j \in [0, n-1]$ tels que $i < j < n$, $t[i] < t[j]$
- Si le tableau est trié, il est possible d'améliorer la recherche séquentielle d'un élément
 - On recherche v dans un tableau trié $t[0..n-1]$:
 - Si $v \notin t[0..i]$ et $t[i] > v$, inutile de parcourir le reste du tableau
- Exemple 7 : Fonction rechercheTriée
- Évaluation du coût de la recherche pour la première solution (tableau non trié)
 - Si $v \in t[0..n-1]$, on réalise en moyenne $n/2$ accès au tableau
 - Sinon, n accès dans tous les cas
- Évaluation du coût de la recherche pour la deuxième solution (tableau trié)
 - Si $v \in t[0..n-1]$, on réalise en moyenne $n/2$ accès au tableau
 - Sinon, n accès dans le pire des cas (v est un majorant du tableau t)

Recherche dichotomique

- Principe
 - On considère un élément pivot du tableau : p
- On compare $t[p]$ à v , 3 cas sont possibles
 - $t[p] = v \rightarrow$ l'algorithme termine
 - $t[p] > v \rightarrow$ ce qui signifie que $v \notin t[p+1..n-1]$
 - $t[p] < v \rightarrow$ ce qui signifie que $v \notin t[0..p-1]$
- Le principe est appliqué à nouveau sur la partie du tableau qui peut contenir v
 - Arrêt : trouvé ou $\text{borneInf} > \text{borneSup}$

Exemple d'exécution 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26
↑							↑							↑	
binf							p							bsup	

- Valeur recherchée : 20 (ici, présente)
- Au début :
 - $\text{binf} = 0$ et $\text{bsup} = 15$
 - $p = (\text{binf} + \text{bsup}) / 2 = 7$

Exemple d'exécution 1

<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>	<i>11</i>	<i>12</i>	<i>13</i>	<i>14</i>	<i>15</i>
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26
							p	binf						bsup	

- $t[p] = 11$: ce n'est pas la valeur recherchée
 - Mise-à-jour de l'intervalle de recherche
- $t[p] < 20$: décalage de binf
 - $binf = p + 1$

Exemple d'exécution 1

<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>	<i>11</i>	<i>12</i>	<i>13</i>	<i>14</i>	<i>15</i>
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26

↑ ↑ ↑
binf p bsup

- L'intervalle de recherche est maintenant :
 - $\text{binf} = 8$
 - $\text{bsup} = 15$
 - $p = (\text{binf} + \text{bsup}) / 2 = 11$

Exemple d'exécution 1

<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>	<i>11</i>	<i>12</i>	<i>13</i>	<i>14</i>	<i>15</i>
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26

↑ ↑ ↑
p binf bsup

- $t[p] = 19$: ce n'est pas la valeur recherchée
 - Mise-à-jour de l'intervalle de recherche
- $t[p] < 20$: décalage de binf
 - $\text{binf} = p + 1$

Exemple d'exécution 1

<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>	<i>11</i>	<i>12</i>	<i>13</i>	<i>14</i>	<i>15</i>
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26

↑ ↑ ↑
binf p bsup

- L'intervalle de recherche est maintenant :
 - $\text{binf} = 12$
 - $\text{bsup} = 15$
 - $p = (\text{binf} + \text{bsup}) / 2 = 13$

Exemple d'exécution 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26

↑ ↑
b_{inf} p
b_{sup}

- $t[p] = 21$: ce n'est pas la valeur recherchée
 - Mise-à-jour de l'intervalle de recherche
- $t[p] = 21 > 20$: décalage de b_{sup}
 - $b_{sup} = p - 1$

Exemple d'exécution 1

<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>	<i>11</i>	<i>12</i>	<i>13</i>	<i>14</i>	<i>15</i>
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26

↑
binf
bsup
p

- L'intervalle de recherche est maintenant :
 - $\text{binf} = \text{bsup} = 12$
 - $p = (\text{binf} + \text{bsup}) / 2 = 12$
- $t[p] = 20$: valeur trouvée
 - L'algorithme est terminé

Exemple d'exécution 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26
↑							↑								↑
binf							p								bsup

- Valeur recherchée : 7 (ici, non présente)
- Au début :
 - $\text{binf} = 0$
 - $\text{bsup} = 15$
 - $p = (\text{binf} + \text{bsup}) / 2 = 7$

Exemple d'exécution 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26

↑
binf

↑
bsup

↑
p

- $t[p] = 11$: ce n'est pas la valeur recherchée
 - Mise-à-jour de l'intervalle de recherche
- $t[p] > 7$: décalage de bsup
 - $bsup = p - 1$

Exemple d'exécution 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26
↑			↑			↑									
binf			p			bsup									

- L'intervalle de recherche est maintenant :
 - $\text{binf} = 0$
 - $\text{bsup} = 6$
 - $p = (\text{binf} + \text{bsup}) / 2 = 3$

Exemple d'exécution 2

<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>	<i>11</i>	<i>12</i>	<i>13</i>	<i>14</i>	<i>15</i>
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26

 ↑ ↑ ↑
 p binf bsup

- $t[p] = 5$: ce n'est pas la valeur recherchée
 - Mise-à-jour de l'intervalle de recherche
- $t[p] < 7$: décalage de binf
 - $\text{binf} = p + 1$

Exemple d'exécution 2

<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>	<i>11</i>	<i>12</i>	<i>13</i>	<i>14</i>	<i>15</i>
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26

↑ ↑ ↑
binf **p** bsup

- L'intervalle de recherche est maintenant :
 - $\text{binf} = 4$
 - $\text{bsup} = 6$
 - $p = (\text{binf} + \text{bsup}) / 2 = 5$

Exemple d'exécution 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26

↑ ↑
binf p
bsup

- $t[p] = 8$: ce n'est pas la valeur recherchée
 - Mise-à-jour de l'intervalle de recherche
- $t[p] > 7$: décalage de bsup
 - $bsup = p - 1$

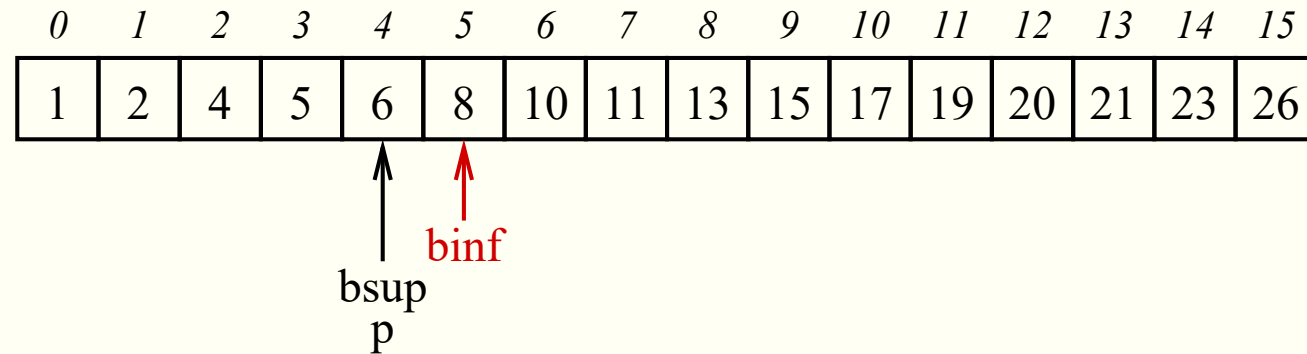
Exemple d'exécution 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	4	5	6	8	10	11	13	15	17	19	20	21	23	26

↑
binf
bsup
p

- L'intervalle de recherche est maintenant :
 - $\text{binf} = 4$
 - $\text{bsup} = 4$
 - $p = (\text{binf} + \text{bsup}) / 2 = 4$

Exemple d'exécution 2



- $t[p] = 6$: ce n'est pas la valeur recherchée
 - Mise-à-jour de l'intervalle de recherche
- $t[p] < 7$: décalage de binf
 - $\text{binf} = p + 1$
- $\text{binf} > \text{bsup}$: valeur non trouvée !
- Exemple 8 : Fonction rechercheDichotomique

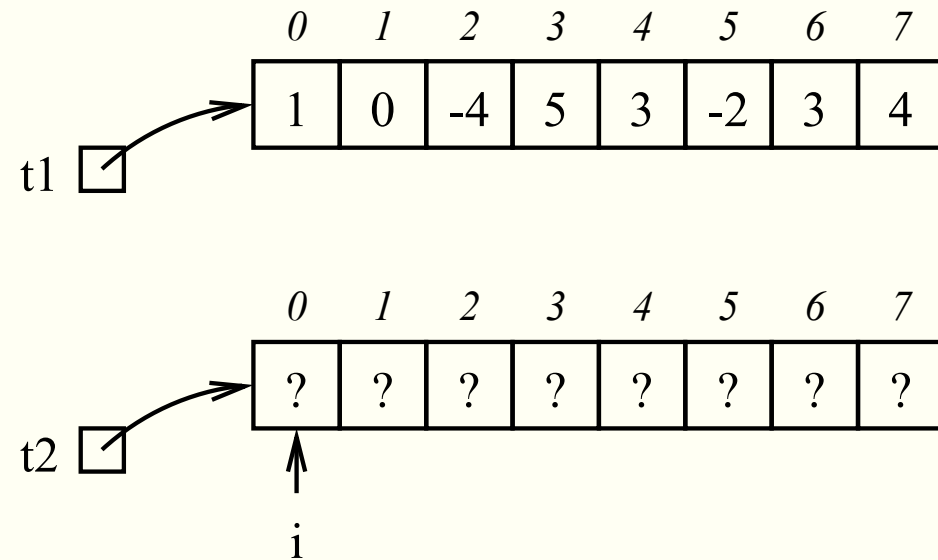
Coût de la recherche dichotomique

- Sensiblement inférieur à celui des 2 algorithmes de recherche précédents
- On élimine, à chaque étape, la moitié du tableau
- On utilise au mieux les deux aspects suivants
 - Le fait que le tableau soit trié
 - L'accès direct offert par les tableaux
- Dans le pire des cas, la recherche d'une valeur dans un tableau de taille n nécessite $\log n$ accès

Tri d'un tableau

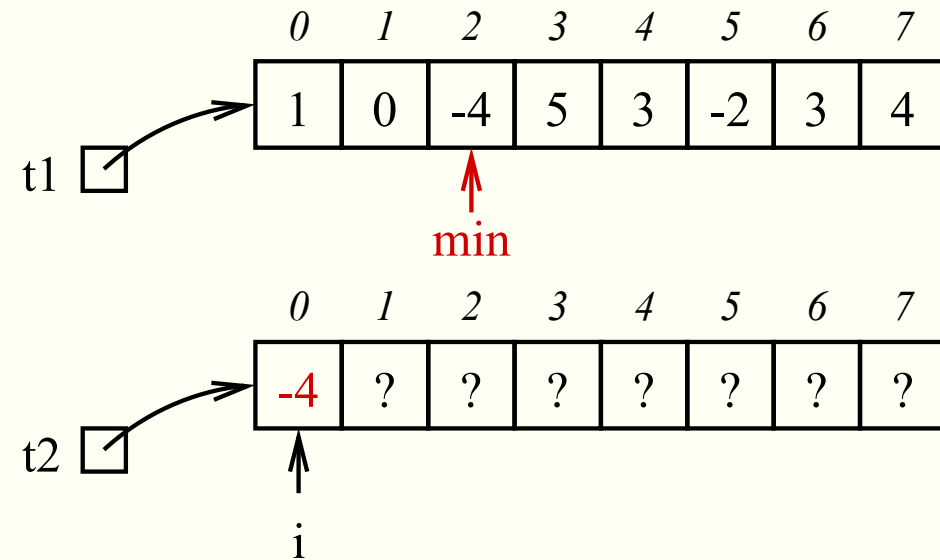
- Il existe plusieurs méthodes de tri d'efficacités différentes
- Le choix d'une méthode plutôt qu'une autre dépend de plusieurs critères
 - Temps d'exécution dans le meilleur des cas
 - Temps d'exécution dans le pire des cas
 - Temps d'exécution moyen
 - Espace mémoire nécessaire
- Tri par sélection
 - Rechercher le minimum du tableau à trier
 - Placer le minimum au début d'un nouveau tableau
 - Remplacer cet élément par un majorant du tableau
 - Répéter ces trois étapes pour tous les éléments restants

Tri par sélection : exemple



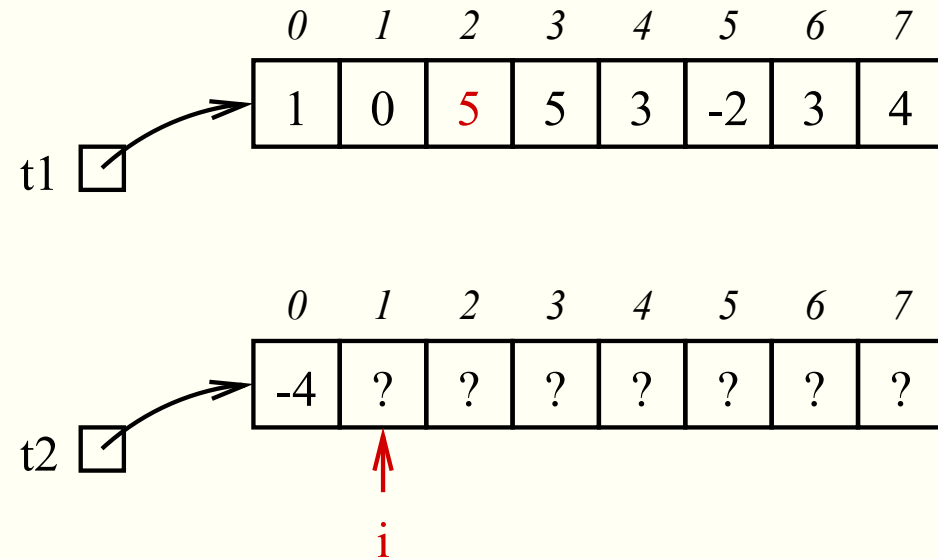
- Tableau à trier : $t1$
- Tableau résultat : $t2$

Tri par sélection : exemple



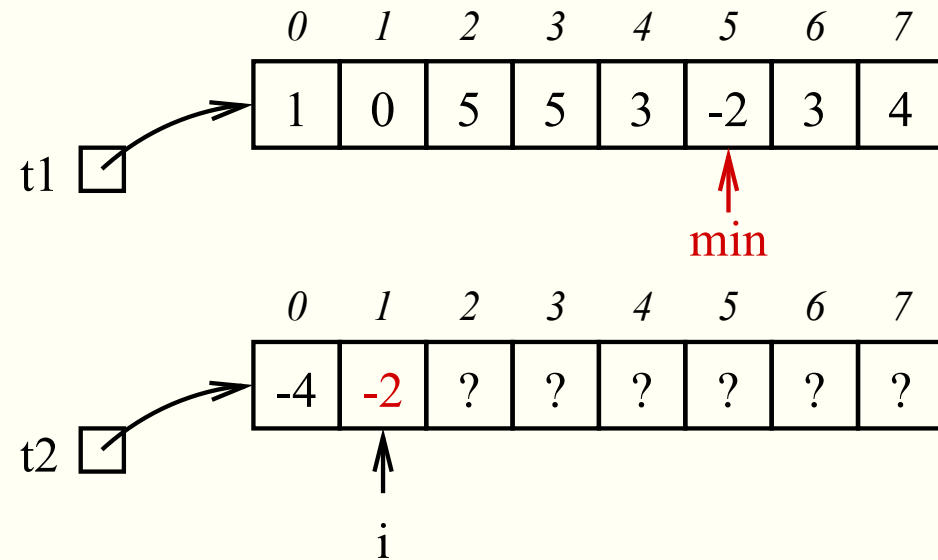
- Recherche de l'indice de la case contenant le minimum : indice = 2
- Minimum placé dans le tableau résultat

Tri par sélection : exemple



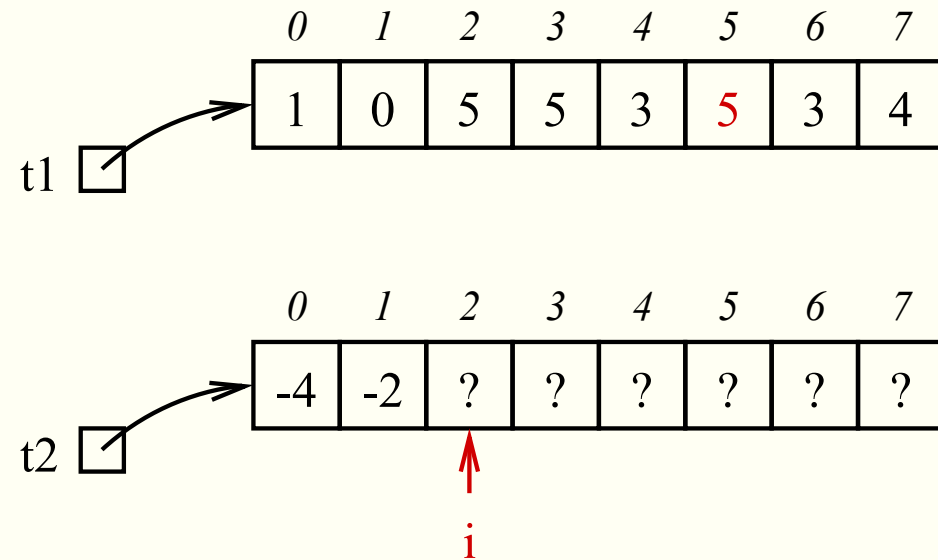
- Remplacement de la valeur minimale par la valeur maximale
- Déplacement de l'indice dans le tableau résultat

Tri par sélection : exemple



- Recherche de l'indice de la case contenant le minimum : indice = 5
- Minimum placé dans le tableau résultat

Tri par sélection : exemple

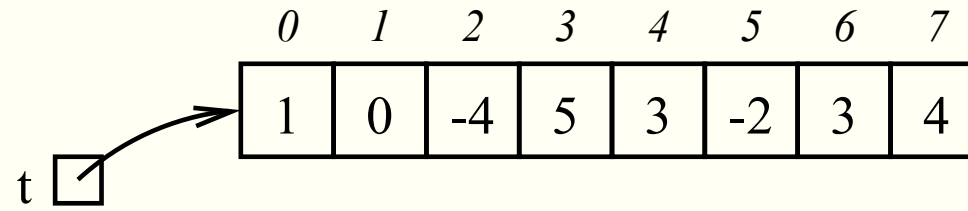


- Remplacement de la valeur minimale par la valeur maximale
- Déplacement de l'indice dans le tableau résultat, etc.
- Exemple 9 : Procédure triSélection

Tri par sélection/échange - remarques

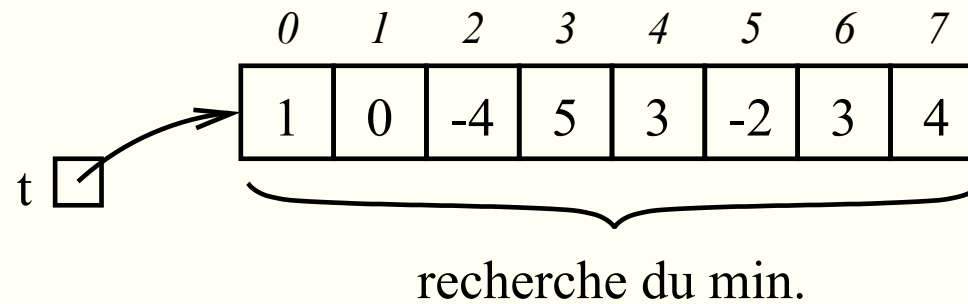
- Le tableau trié est contenu dans le tableau t_2
 - Il est possible de n'utiliser qu'un seul tableau
- Il suffit alors d'inverser deux éléments du tableau
 - La valeur minimum et la valeur à la position i (au i ème tour)
 - La recherche du minimum s'effectue sur la partie non triée du tableau, c'est-à-dire sur $t[i..n-1]$

Tri par sélection/échange : exemple



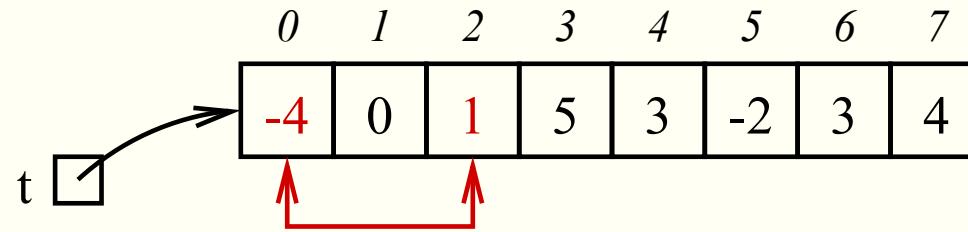
- Tableau à trier : t

Tri par sélection/échange : exemple



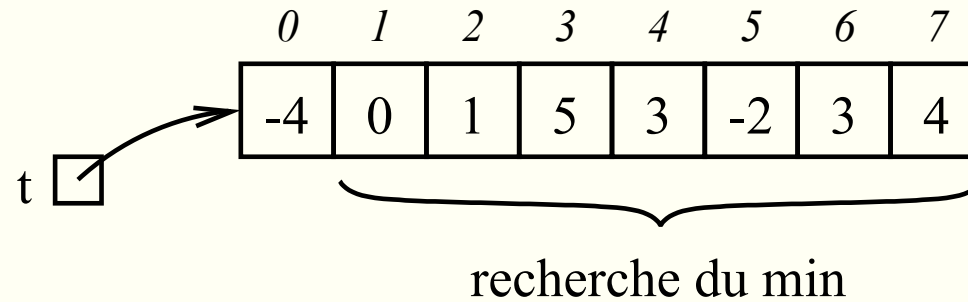
- Recherche de l'indice du minimum
 - Indice = 2

Tri par sélection/échange : exemple



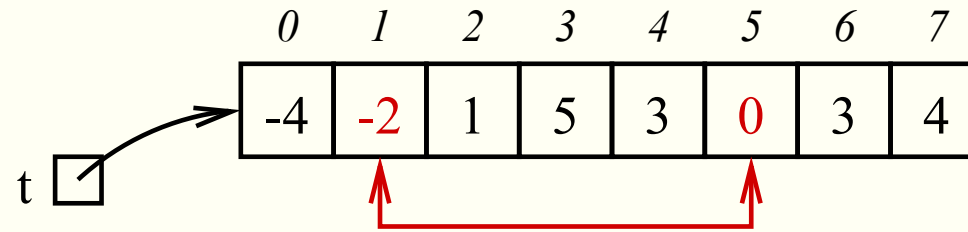
- Échange des valeurs des deux cases

Tri par sélection/échange : exemple



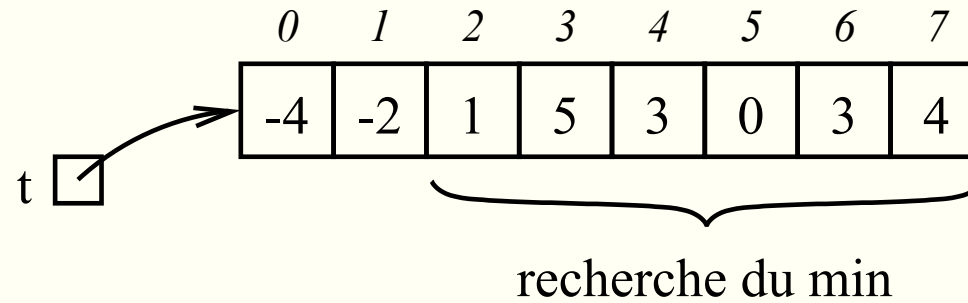
- Recherche de l'indice du minimum
 - Indice = 5

Tri par sélection/échange : exemple



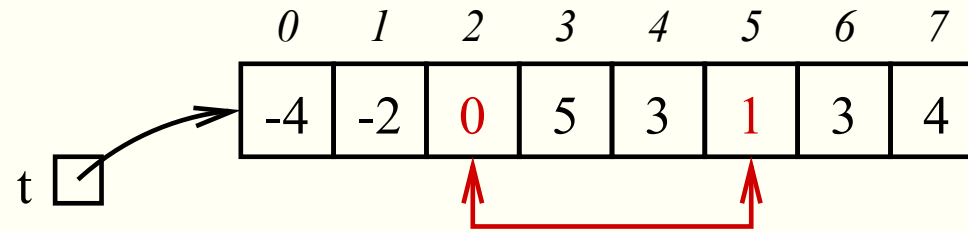
- Échange des valeurs des deux cases

Tri par sélection/échange : exemple

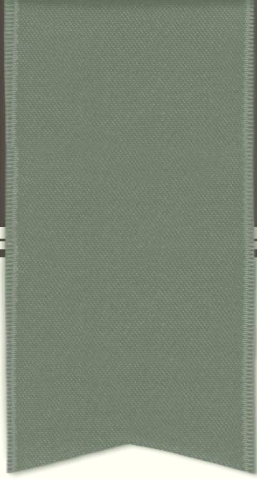


- Recherche de l'indice du minimum
 - Indice = 5

Tri par sélection/échange : exemple



- Échange des valeurs des deux cases
- etc.
- Exemple 10 : Procédure triSélectionEchange



PROCHAIN COURS : CHAÎNES DE CARACTÈRES