DUT SRC – IUT de Marne-la-Vallée
29/01/2014
M2202 - Algorithmique

Cours 2
Tris

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

iut Marne-la-Vallée

Philippe Gambette

Résumé de l'épisode précédent

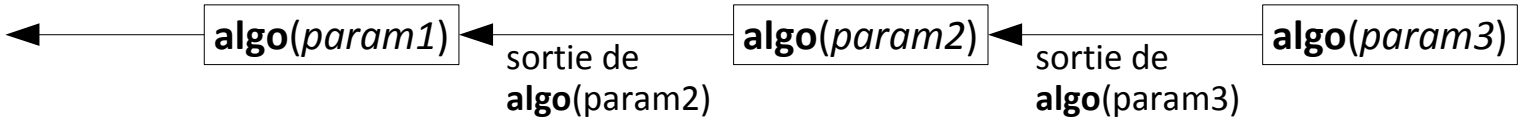
- **Récursivité**

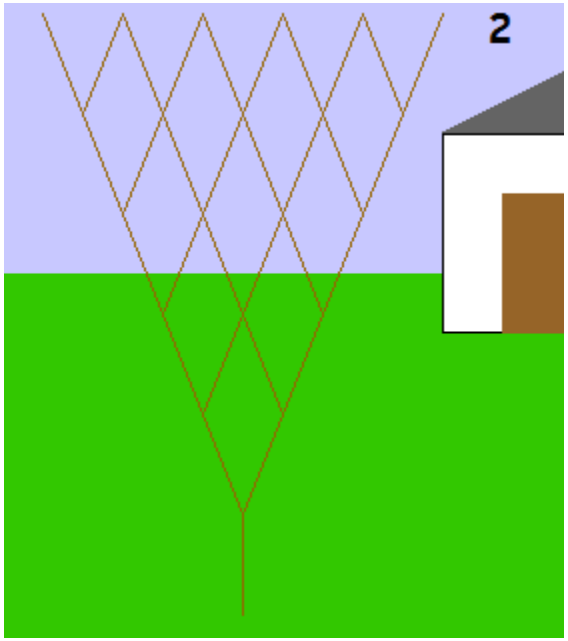
Un algorithme **récursif** = un algorithme qui s'appelle lui-même
On suppose que :

* on sait construire/calculer le premier objet (**initialisation**)

* à partir du (*n*-1)-ième objet on sait construire/calculer le *n*-ième (**hérédité**)

alors on arrive à construire/calculer tous les objets.





Sources

- Cours de Jean-François Berdjugin à l'IUT de Grenoble
http://berdjugin.com/enseignements/inf/inf220/

- Cours de Xavier Heurtebise à l'IUT de Provence
http://x.heurtebise.free.fr

- *Le livre de Java premier langage*, d'A. Tasso

- http://xkcd.com, http://xkcd.free.fr

Plan du cours 2 – Tris

- Les tris
- Le tri à bulles
- Le tri par sélection
- Complexité des tris

Plan du cours 2 – Tris

- Les tris
- Le tri à bulles
- Le tri par sélection
- Complexité des tris

Principe d'un algorithme de tri

Entrée :

Ensemble d'éléments **tous comparables deux à deux** par un ordre **≤**

Sortie :

Éléments **triés selon cet ordre**, du plus petit au plus grand

Ensemble d'éléments : quel type ?

Principe d'un algorithme de tri

Entrée :

Ensemble d'éléments **tous comparables deux à deux** par un ordre **≤**

Sortie :

Éléments **triés selon cet ordre**, du plus petit au plus grand

Ensemble d'éléments : quel type ?

Type pour stocker plusieurs éléments ?

Principe d'un algorithme de tri

Entrée :

Ensemble d'éléments **tous comparables deux à deux** par un ordre **≤**

Sortie :

Éléments **triés selon cet ordre**, du plus petit au plus grand

Ensemble d'éléments : quel type ?

Type pour stocker plusieurs éléments ?

→ un tableau !

→ autres structures de données au prochain cours.

Principe d'un algorithme de tri

Entrée :

Ensemble d'éléments **tous comparables deux à deux** par un ordre **≤**

Sortie :

Éléments **triés selon cet ordre**, du plus petit au plus grand

Ensemble d'éléments	**Ordre**	**Tous comparables deux à deux ?**
tableau d'entiers	plus petit	
tableau de chaînes de caractères	lexicographique (du dictionnaire)	
tableau de cartes à jouer	2 ≤ 3 ≤ … ≤ 10 ≤ V ≤ D ≤ R ≤ 1	
tableau de cartes à jouer	♣ ≤ ♦ ≤ ♥ ≤ ♠ puis 2 ≤ 3 ≤ … ≤ 10 ≤ V ≤ D ≤ R ≤ 1	

Principe d'un algorithme de tri

Entrée :

Ensemble d'éléments **tous comparables deux à deux** par un ordre **≤**

Sortie :

Éléments **triés selon cet ordre**, du plus petit au plus grand

Ensemble d'éléments	**Ordre**	**Tous comparables deux à deux ?**
tableau d'entiers	plus petit	oui : 1 ≤ 2, 2 ≤ 3, 1 ≤ 3...
tableau de chaînes de caractères	lexicographique (du dictionnaire)	oui : a ≤ b, char ≤ charles, char ≤ chat...
tableau de cartes à jouer	2 ≤ 3 ≤ … ≤ 10 ≤ V ≤ D ≤ R ≤ 1	non : 5♣ incomparable avec 5♠
tableau de cartes à jouer	♣ ≤ ♦ ≤ ♥ ≤ ♠ puis 2 ≤ 3 ≤ … ≤ 10 ≤ V ≤ D ≤ R ≤ 1	oui : A♣ ≤ 5♠, 5♠ ≤ 6♠, ...

Principe d'un algorithme de tri

Entrée :

Ensemble d'éléments **tous comparables deux à deux** par un ordre **≤**

Sortie :

Éléments **triés selon cet ordre**, du plus petit au plus grand

Exemple avec un tableau d'entiers :

5	3	1	8	5	2	9

1	2	3	5	5	8	9

En anglais : trié = sorted
trier = to sort

Principe d'un algorithme de tri

Entrée :

Ensemble d'éléments **tous comparables deux à deux** par un ordre **≤**

Sortie :

Éléments **triés selon cet ordre**, du plus petit au plus grand

Exemple avec un tableau d'entiers :

5	3	1	8	5	2	9

1	2	3	5	5	8	9

En anglais : trié = sorted
trier = to sort

Algorithme **inferieurOuEgal**
Entrées : ?
Type de sortie : ?

Exemple : **inferieurOuEgal**(5,8) renvoie ...

Principe d'un algorithme de tri

Entrée :

Ensemble d'éléments **tous comparables deux à deux** par un ordre **≤**

Sortie :

Éléments **triés selon cet ordre**, du plus petit au plus grand

Exemple avec un tableau d'entiers :

5	3	1	8	5	2	9

1	2	3	5	5	8	9

En anglais : trié = sorted
trier = to sort

Algorithme **inferieurOuEgal**
Entrées : deux entiers *a* et *b*
Type de sortie : booléen

Exemple : **inferieurOuEgal**(5,8) renvoie VRAI

Plan du cours 2 – Tris

- Les tris
- Le tri à bulles
- Le tri par sélection
- Complexité des tris

Tri à bulles

La “minute culturelle”

http://www.youtube.com/watch?v=lyZQPjUT5B4

Tri à bulles

Idée : tant que le tableau n'est pas trié, on le parcourt en effectuant l'opération suivante à la *i*-ième case : si elle est supérieure à la suivante, on les échange.

Exemple avec un tableau d'entiers :

Tri à bulles

Idée : tant que le tableau n'est pas trié, on le parcourt en effectuant l'opération suivante à la *i*-ième case : si elle est supérieure à la suivante, on les échange.

Exemple avec un tableau d'entiers :

Étape 1 :

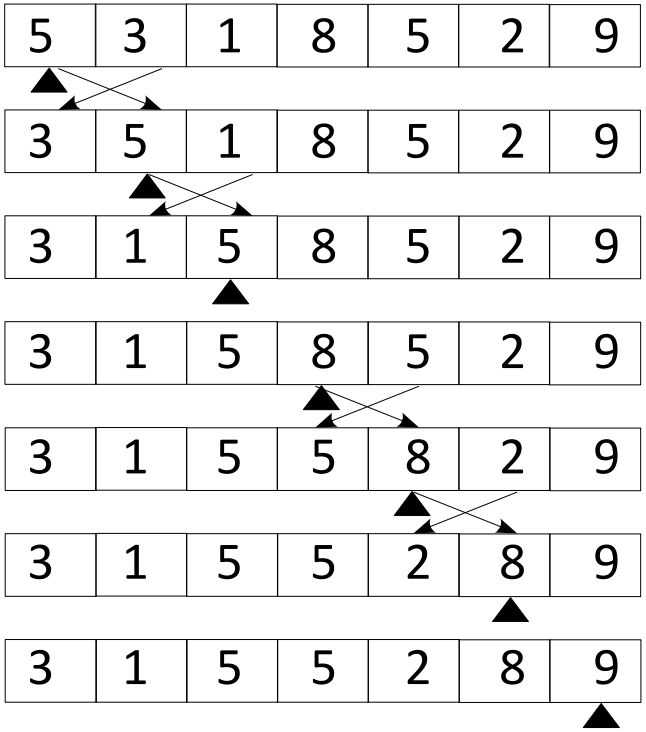


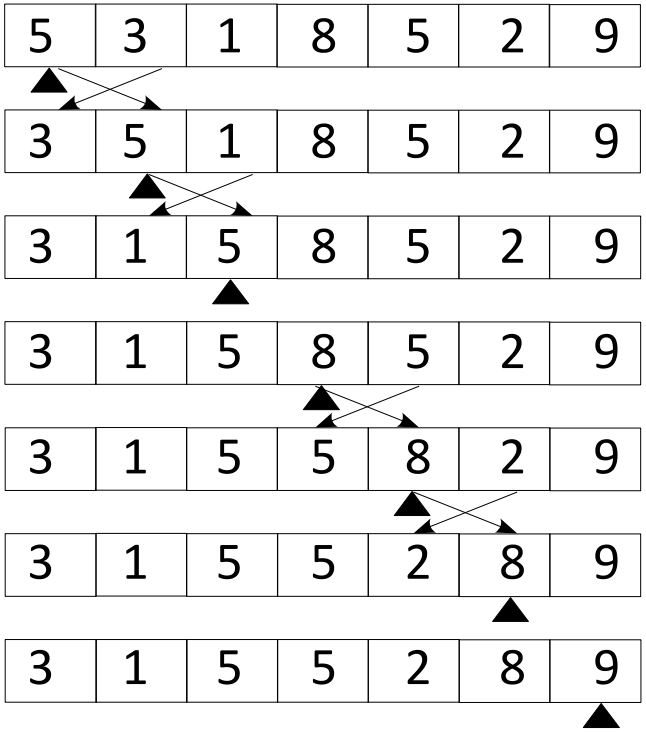
Tableau trié ?

Tri à bulles

Idée : tant que le tableau n'est pas trié, on le parcourt en effectuant l'opération suivante à la *i*-ième case : si elle est supérieure à la suivante, on les échange.

Exemple avec un tableau d'entiers :

Étape 1 :



Étape 2 :

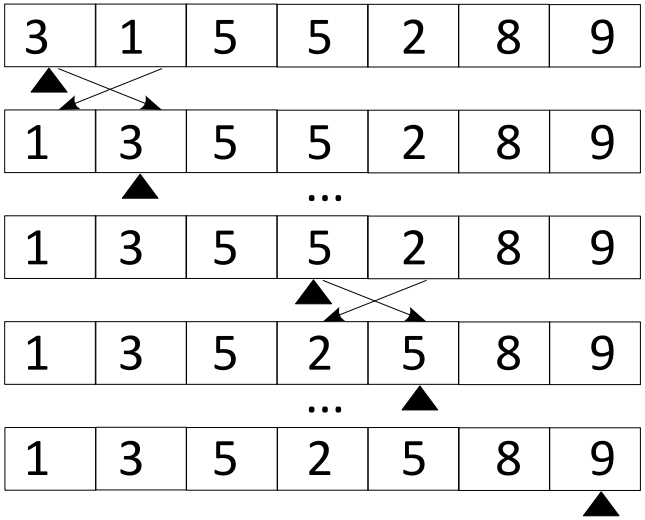


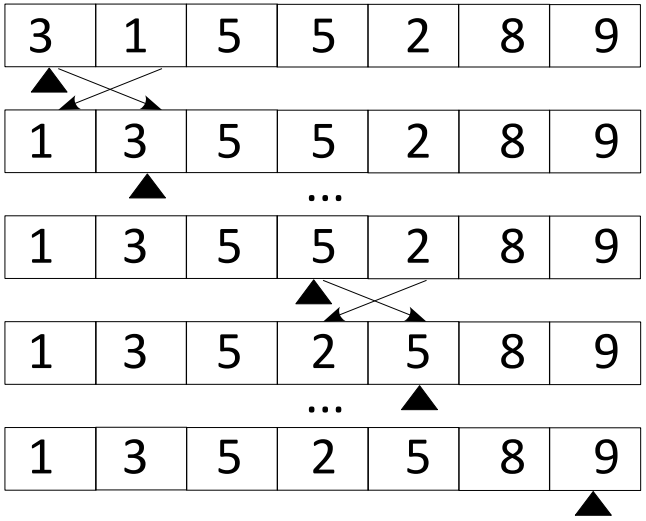
Tableau trié ?

Tri à bulles

Idée : tant que le tableau n'est pas trié, on le parcourt en effectuant l'opération suivante à la *i*-ième case : si elle est supérieure à la suivante, on les échange.

Exemple avec un tableau d'entiers :

Étape 2 :



Étape 3 :

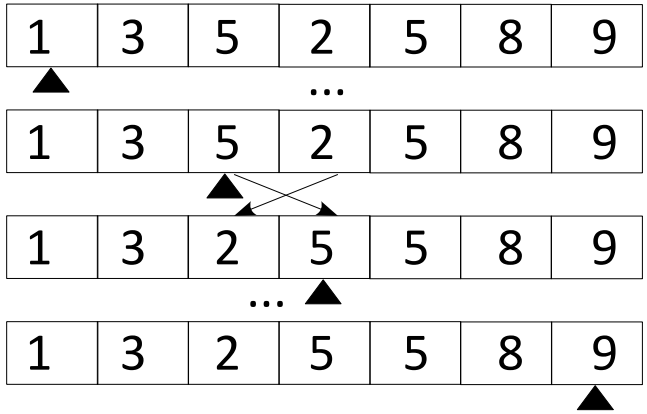


Tableau trié ?

Tri à bulles

Idée : tant que le tableau n'est pas trié, on le parcourt en effectuant l'opération suivante à la *i*-ième case : si elle est supérieure à la suivante, on les échange.

Exemple avec un tableau d'entiers :

Étape 3 :

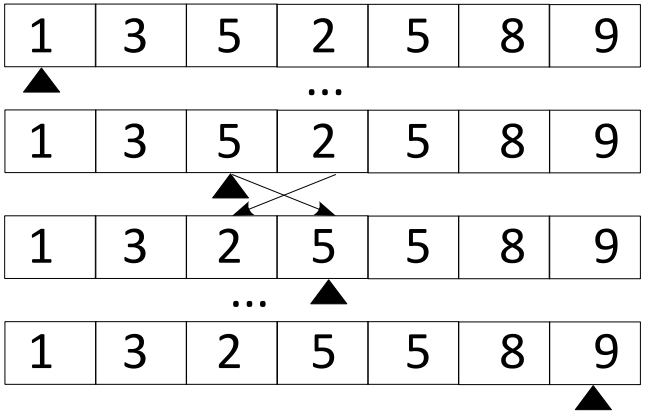


Tableau trié ?

Étape 4 :

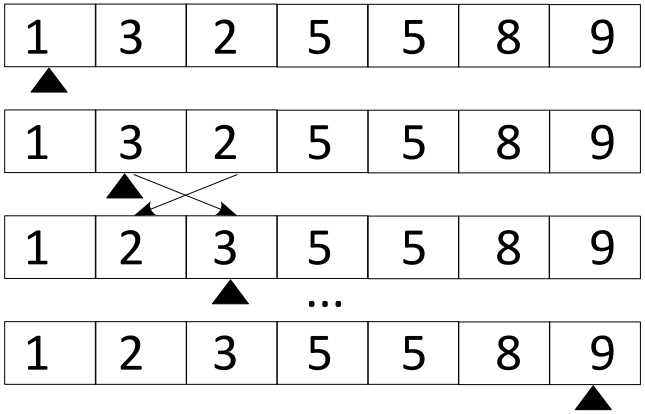


Tableau trié ?

Oui.

Plan du cours 2 – Tris

- Les tris
- Le tri à bulles
- Le tri par sélection
- Complexité des tris

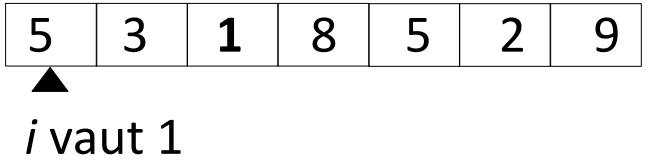
Tri par sélection (ou tri par extraction)

Idée : à la *i*-ième étape, on prend le plus petit élément à partir de la *i*-ième case (comprise) et on l'échange avec l'élément de la *i*-ième case.

Tri par sélection (ou tri par extraction)

Idée : à la *i*-ième étape, on prend le plus petit élément à partir de la *i*-ième case (comprise) et on l'échange avec l'élément de la *i*-ième case.

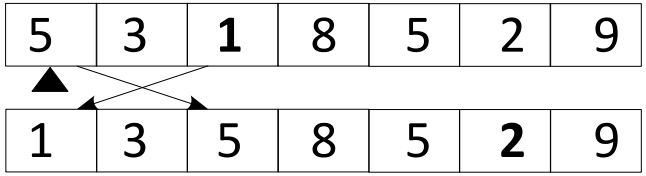
Exemple avec un tableau d'entiers :



Tri par sélection (ou tri par extraction)

Idée : à la *i*-ième étape, on prend le plus petit élément à partir de la *i*-ième case (comprise) et on l'échange avec l'élément de la *i*-ième case.

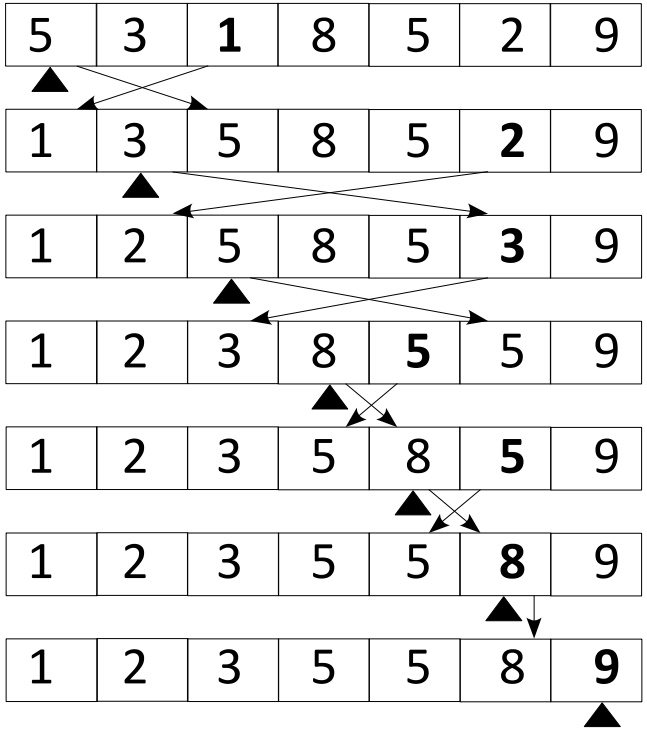
Exemple avec un tableau d'entiers :



Tri par sélection (ou tri par extraction)

Idée : à la *i*-ième étape, on prend le plus petit élément à partir de la *i*-ième case (comprise) et on l'échange avec l'élément de la *i*-ième case.

Exemple avec un tableau d'entiers :

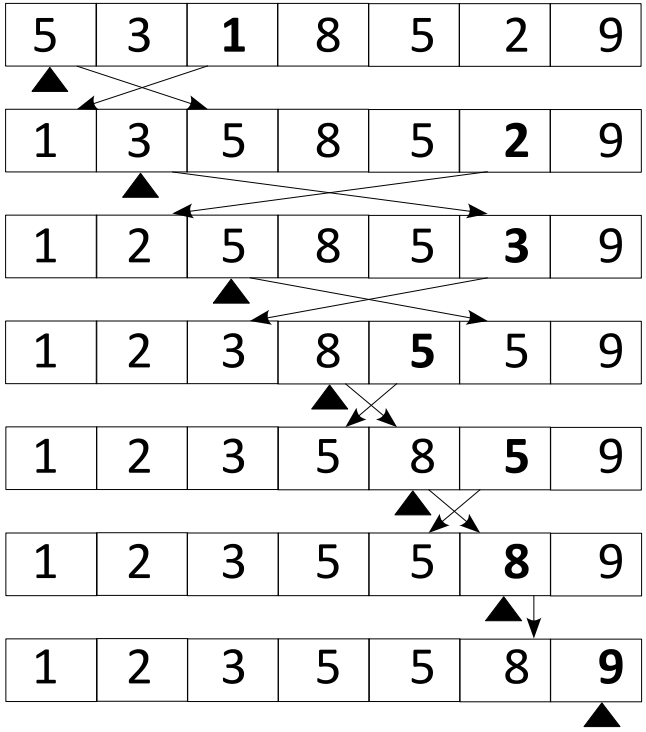


Tri **en place** :
on arrive à trier le tableau sans avoir besoin de créer un nouveau tableau.

Tri par sélection (ou tri par extraction)

Idée : à la *i*-ième étape, on prend le plus petit élément à partir de la *i*-ième case (comprise) et on l'échange avec l'élément de la *i*-ième case.

Exemple avec un tableau d'entiers :



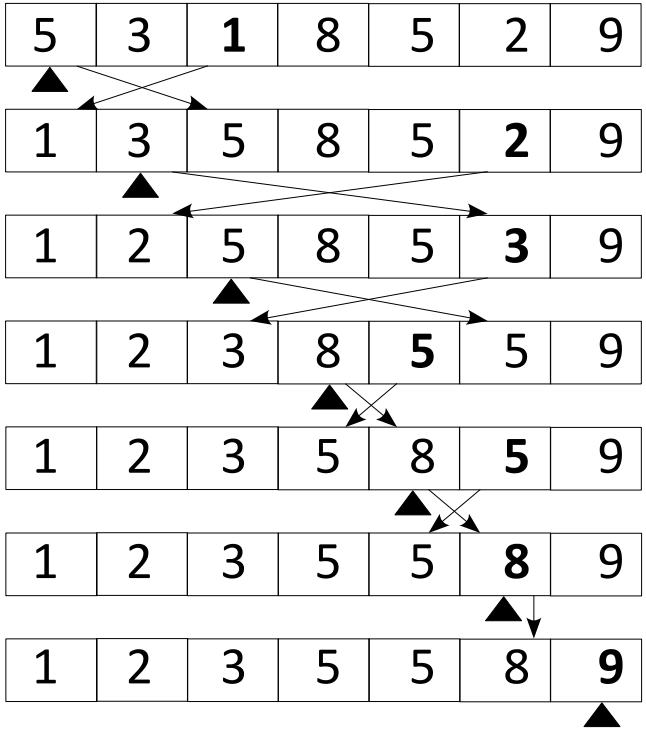
Algorithme **positionMinimum**
Entrée : tableau d'entiers *tab*, entier *debut*
Type de sortie : entier
Variables : entiers *position, i* et *min*
Début

Fin

Tri par sélection (ou tri par extraction)

Idée : à la *i*-ième étape, on prend le plus petit élément à partir de la *i*-ième case (comprise) et on l'échange avec l'élément de la *i*-ième case.

Exemple avec un tableau d'entiers :

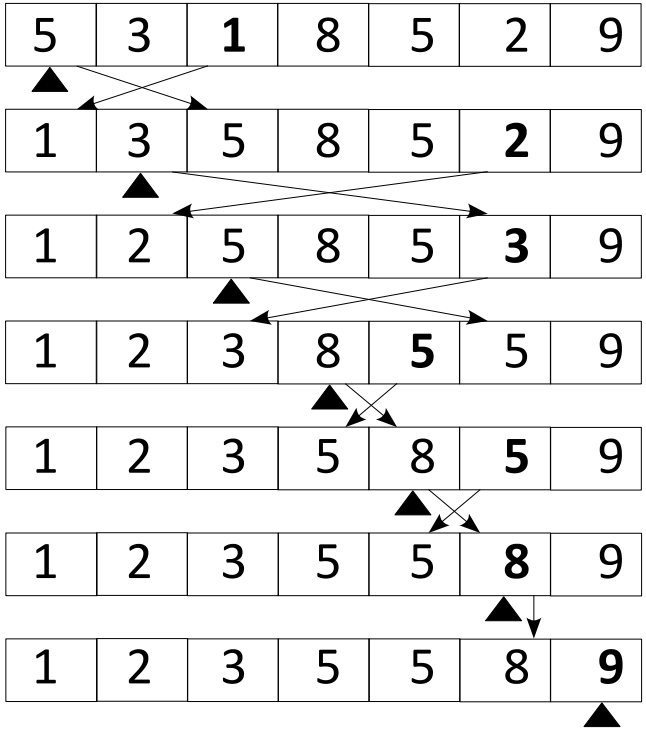


Algorithme **positionMinimum**
Entrée : tableau d'entiers *tab*, entier *debut*
Type de sortie : entier
Variables : entiers *position, i* et *min*
Début
  position ← *debut*
  min ← **Case**(*tab*,*debut*)
  i ← *debut*
  Tant que *i*≤**Longueur**(*tab*) faire :
    Si **Case**(*tab*,*i*)<*min* alors :
      position ← *i*
      min ← **Case**(*tab*,*i*)
    Fin si
    i ← *i*+1
  Fin Tant que
  renvoyer *position*
Fin

Tri par sélection (ou tri par extraction)

Idée : à la *i*-ième étape, on prend le plus petit élément à partir de la *i*-ième case (comprise) et on l'échange avec l'élément de la *i*-ième case.

Exemple avec un tableau d'entiers :



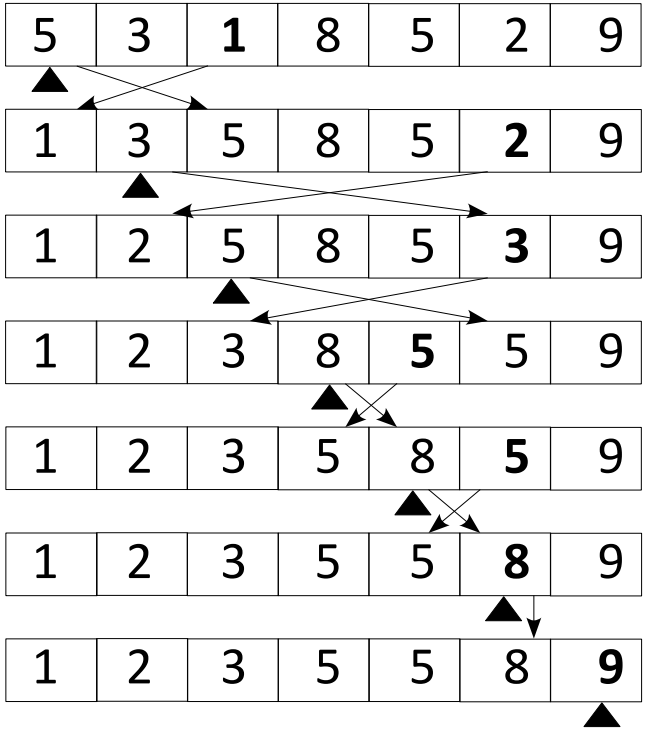
Algorithme **triSelection**
Entrée : tableau d'entiers *tab*
Type de sortie : tableau d'entiers
Variables : entiers *i, temp*, *posMin*
Début

Fin

Tri par sélection (ou tri par extraction)

Idée : à la *i*-ième étape, on prend le plus petit élément à partir de la *i*-ième case (comprise) et on l'échange avec l'élément de la *i*-ième case.

Exemple avec un tableau d'entiers :



Algorithme **triSelection**
Entrée : tableau d'entiers *tab*
Type de sortie : tableau d'entiers
Variables : entiers *i, temp*, *posMin*
Début
  i ← 1
  Tant que *i*<**Longueur**(*tab*) faire :
    posMin ← **positionMinimum**(*tab*,*i*)
    //*posMin* peut être égal à *i*
    temp ← **Case**(*tab*,*i*)
    Case(*tab*,*i*) ← **Case**(*tab*,*posMin*)
    Case(*tab*,*posMin*) ← *temp*
    i ← *i*+1
  Fin Tant que
  renvoyer *tab*
Fin

Plan du cours 2 – Tris

- Les tris
- Le tri par sélection
- Le tri à bulles
- Complexité des tris

Complexité des tris

Combien de comparaisons ? (lié au **temps d'exécution** de l'algorithme)

Sur un exemple ou dans le pire des cas, pour *n* éléments.

Tri à bulles

Tri par sélection

Complexité des tris

Combien de comparaisons ? (lié au **temps d'exécution** de l'algorithme)

Sur un exemple ou dans le pire des cas, pour *n* éléments.

Tri à bulles

Dans tous les cas : ≤ *n* étapes, *n*-1 comparaisons par étape
donc ≤ *n* x (*n*-1) comparaisons au total

Tri par sélection

Pour *n* éléments : *n* étapes, *n-i* comparaisons par étape pour trouver le min

Complexité des tris

Combien de comparaisons ? (lié au **temps d'exécution** de l'algorithme)

Sur un exemple ou dans le pire des cas, pour *n* éléments.

Tri à bulles

Dans tous les cas : ≤ *n* étapes, *n*-1 comparaisons par étape
donc ≤ *n* x (*n*-1) comparaisons au total

Tri par sélection

Pour *n* éléments : *n* étapes, *n-i* comparaisons par étape pour trouver le min

étape 1 : *n*-1 comparaisons
étape 2 : *n*-2 comparaisons

…

étape n-2 : 2 comparaisons
étape n-1 : 1 comparaison
étape n : 0 comparaison

Complexité des tris

Combien de comparaisons ? (lié au **temps d'exécution** de l'algorithme)

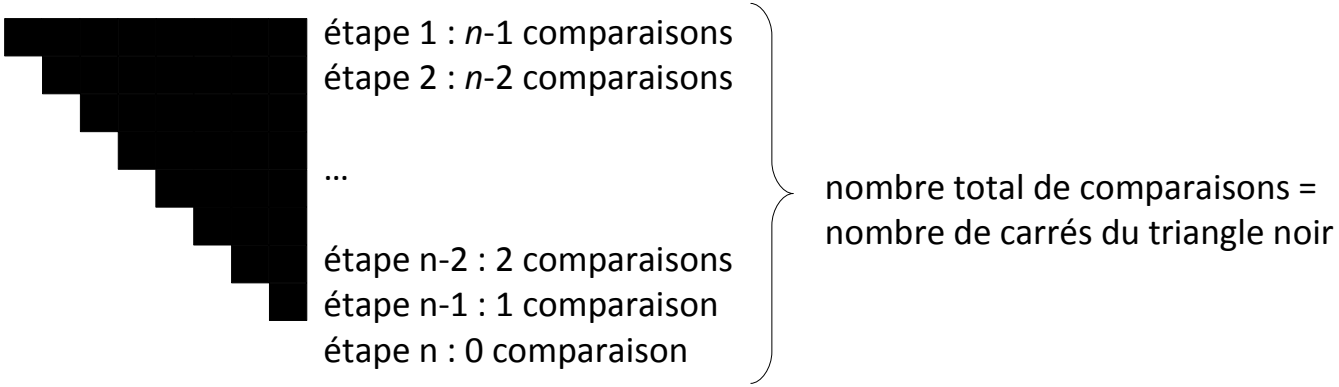
Sur un exemple ou dans le pire des cas, pour *n* éléments.

Tri à bulles

Dans tous les cas : ≤ *n* étapes, *n*-1 comparaisons par étape
donc ≤ *n* x (*n*-1) comparaisons au total

Tri par sélection

Pour *n* éléments : *n* étapes, *n-i* comparaisons par étape pour trouver le min



Complexité des tris

Combien de comparaisons ? (lié au **temps d'exécution** de l'algorithme)

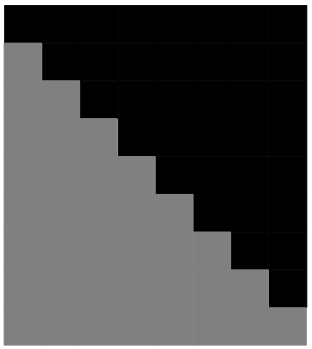
Sur un exemple ou dans le pire des cas, pour *n* éléments.

Tri à bulles

Dans tous les cas : ≤ *n* étapes, *n*-1 comparaisons par étape
donc ≤ *n* x (*n*-1) comparaisons au total

Tri par sélection

Pour *n* éléments : *n* étapes, *n-i* comparaisons par étape pour trouver le min



étape 1 : *n*-1 comparaisons
étape 2 : *n*-2 comparaisons

…

étape n-2 : 2 comparaisons
étape n-1 : 1 comparaison
étape n : 0 comparaison

nombre total de comparaisons =
nombre de carrés du triangle noir =
nombre de carrés du rectangle divisé par 2 =

Complexité des tris

Combien de comparaisons ? (lié au **temps d'exécution** de l'algorithme)

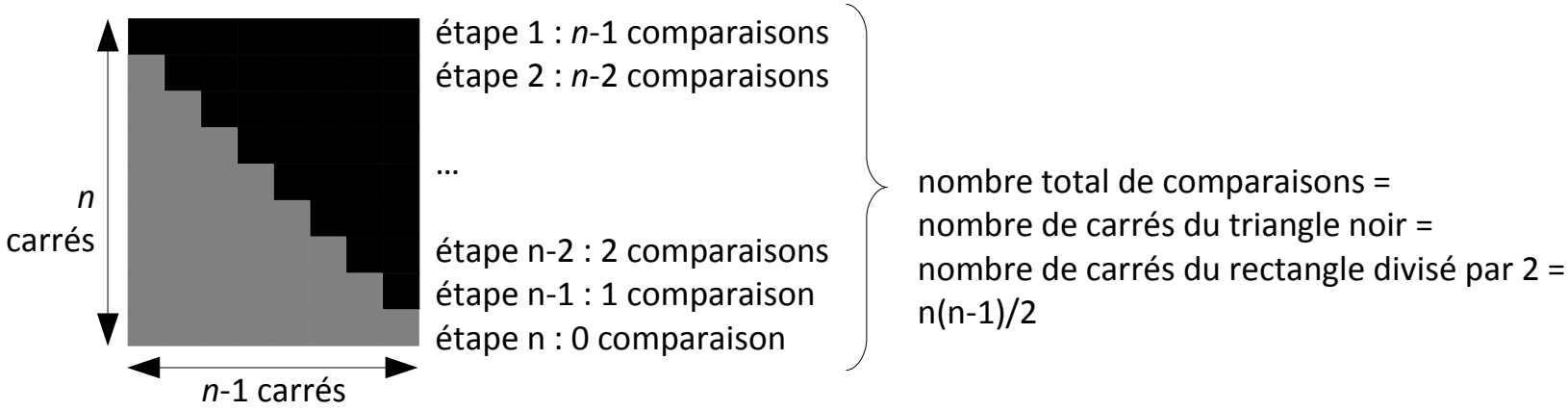
Sur un exemple ou dans le pire des cas, pour *n* éléments.

Tri à bulles

Dans tous les cas : ≤ *n* étapes, *n*-1 comparaisons par étape
donc ≤ *n* x (*n*-1) comparaisons au total

Tri par sélection

Pour *n* éléments : *n* étapes, *n-i* comparaisons par étape pour trouver le min



Complexité des tris

Combien de comparaisons ? (lié au **temps d'exécution** de l'algorithme)

Sur un exemple ou dans le pire des cas, pour *n* éléments.

Tri à bulles

Dans le pire des cas (tableau trié dans le sens inverse) : $n \times (n-1)$ comparaisons
Dans le meilleur cas (tableau trié) : $n-1$ comparaisons

Tri par sélection

Dans tous les cas : $n \times (n-1)/2$ comparaisons