DUT MMI – IUT de Marne-la-Vallée
28/01/2016
M2202 - Algorithmique

Cours 2
Tris

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

iut Marne-la-Vallée

Philippe Gambette

Résumé de l'épisode précédent

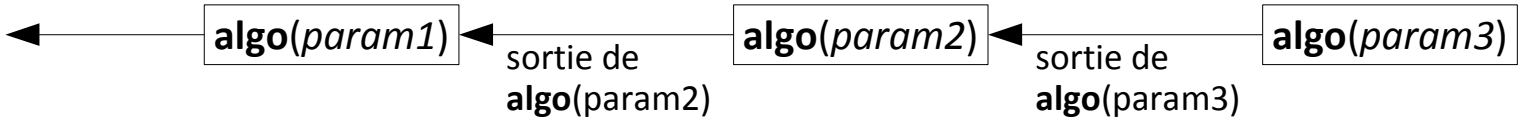
Récursivité

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 par sélection
- Complexité des tris

Plan du cours 2 – Tris

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

Plan du cours 2 – Tris

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

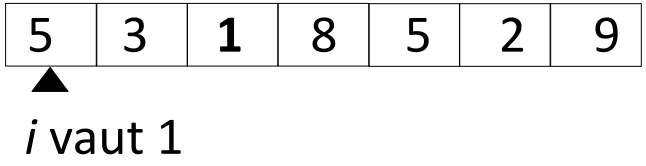
Tri par sélection (ou tri par extraction)

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.

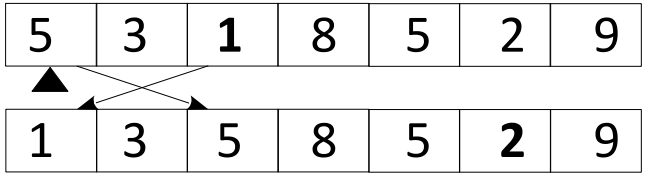
Exemple avec un tableau d'entiers :



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.

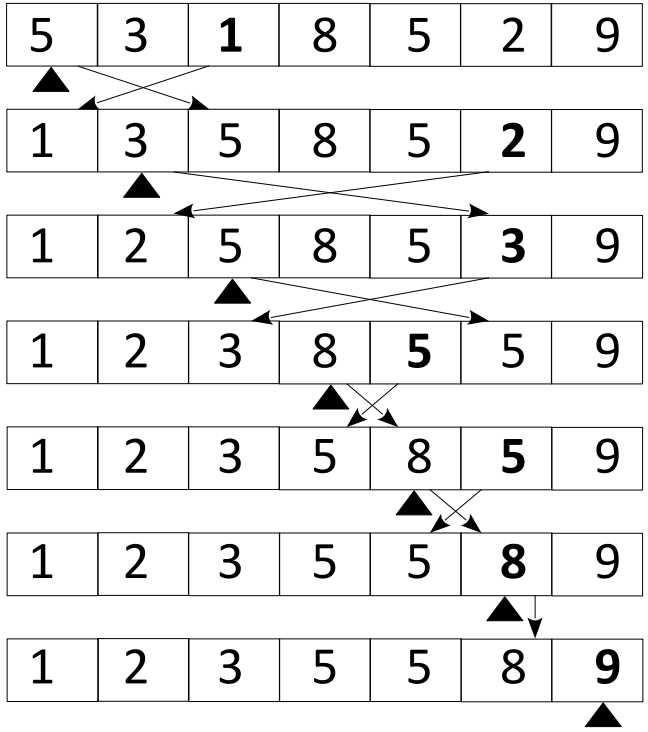
Exemple avec un tableau d'entiers :



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 :

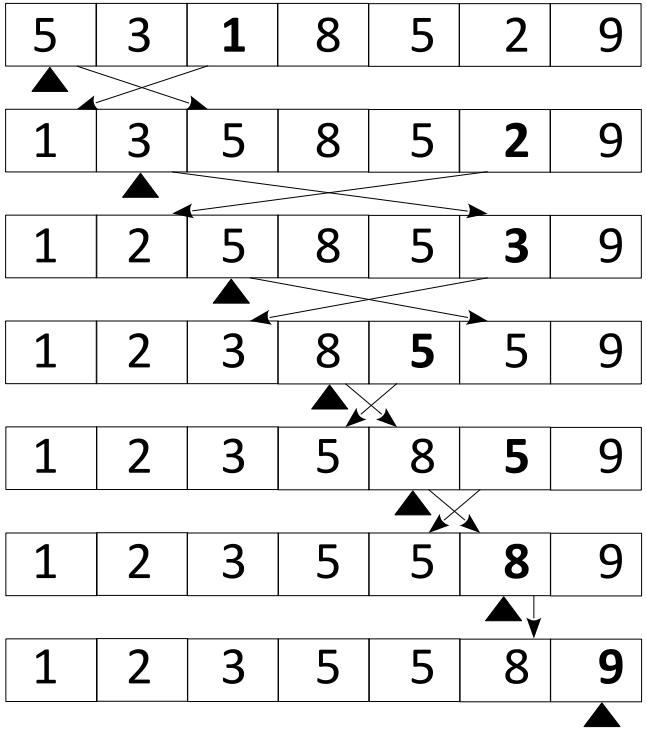


Tri **en place** :
on arrive à trier le tableau sans avoir besoin de créer un nouveau tableau.

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 :



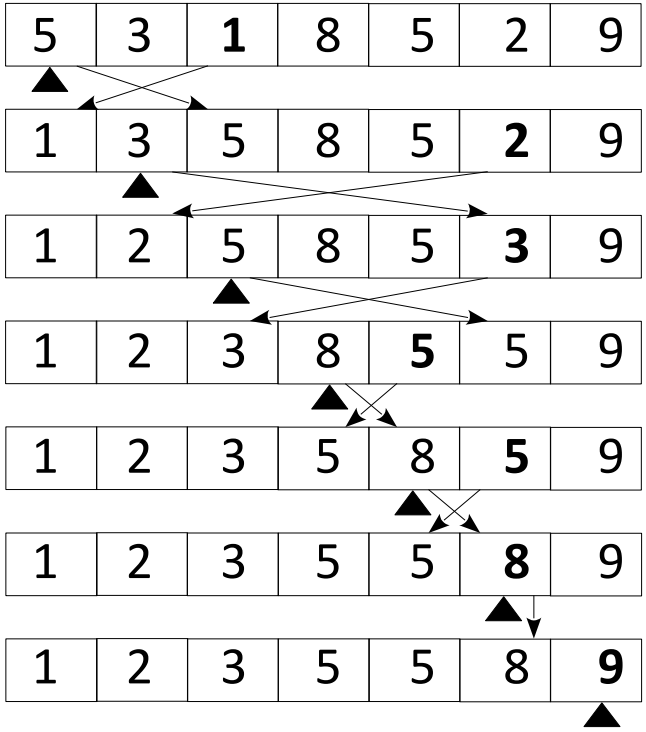
Algorithme **positionMinimum**
Entrée : tableau d'entiers *tab*, entier *debut*
Type de sortie : entier
Variables : entiers *position, i* et *min*
Début

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 :



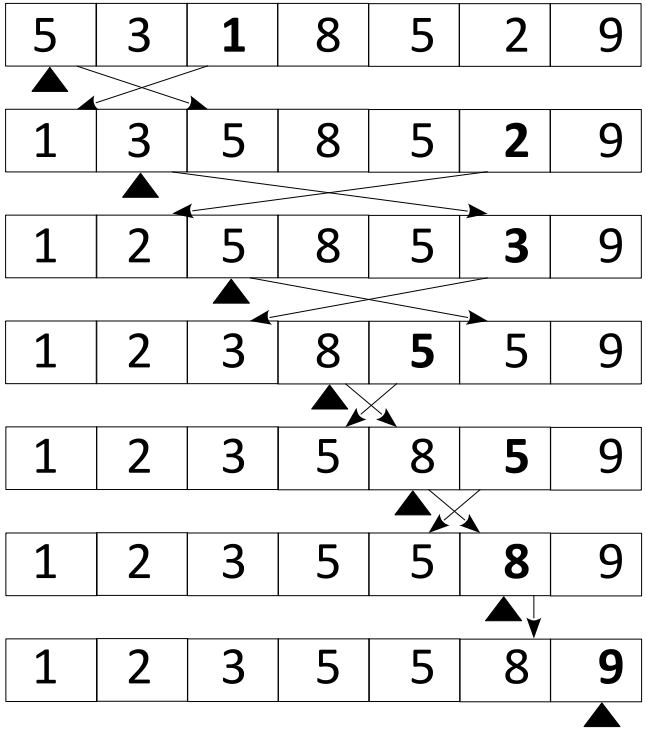
Algorithme **positionMinimum**
Entrée : tableau d'entiers *tab*, entier *debut*
Type de sortie : entier
Variables : entiers *position, i* et *min*
Début
  position ← *debut*
  min ← **Case**(*tab*,*debut*)
  i ← *debut*
  Tant que *i*≤**Longueur**(*tab*) faire :
    Si **Case**(*tab*,*i*)<*min* alors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

version récursive vue en cours !



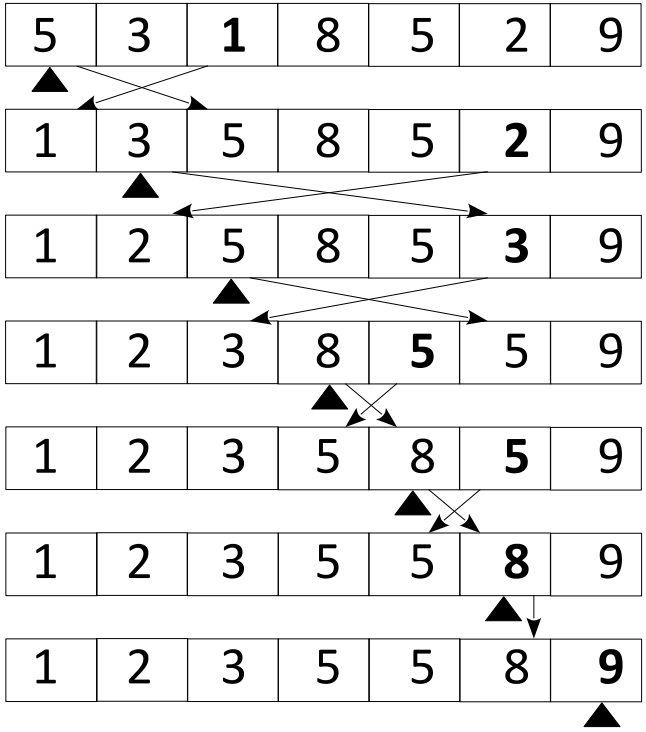

```
Algorithme positionMinimum
Entrée : tableau d'entiers tab, entier debut
Type de sortie : entier
Variables : entier position
Début
  Si debut = Longueur(tab) alors :
    renvoyer debut
  Sinon :
    position ← positionMinimum(tab,debut+1)
    Si Case(tab,debut) < Case(tab,position) alors :
      renvoyer debut
    Sinon :
      renvoyer position
    Fin si
  Fin si
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 :

version récursive plus courte !

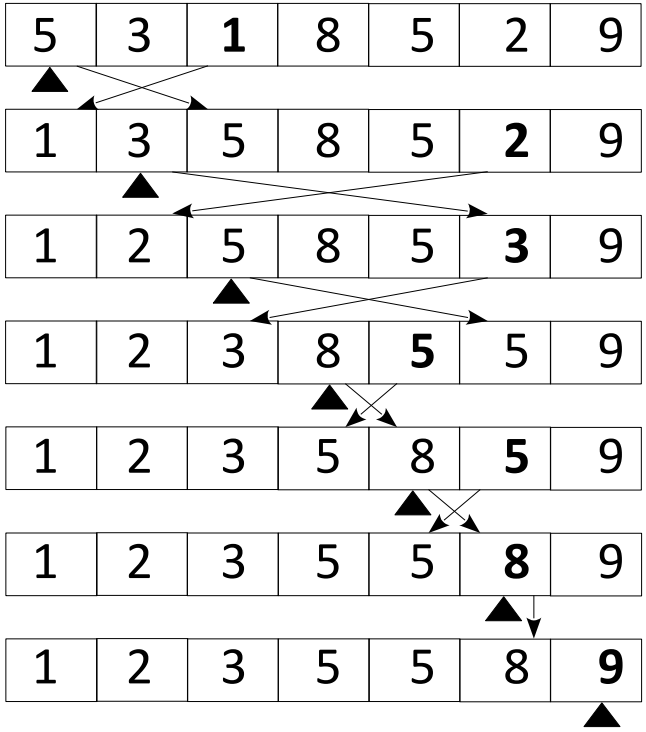


Algorithme **positionMinimum**
Entrée : tableau d'entiers *tab*, entier *debut*
Type de sortie : entier
Variables : entier *position*
Début
  position ← *debut*
  Si *debut* < **Longueur**(*tab*) alors :
    position ← **positionMinimum**(*tab*,*debut*+1)
    Si **Case**(*tab*,*debut*) < **Case**(*tab*,*position*) alors :
      position ← *debut*
    Fin si
  Fin si
  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 :



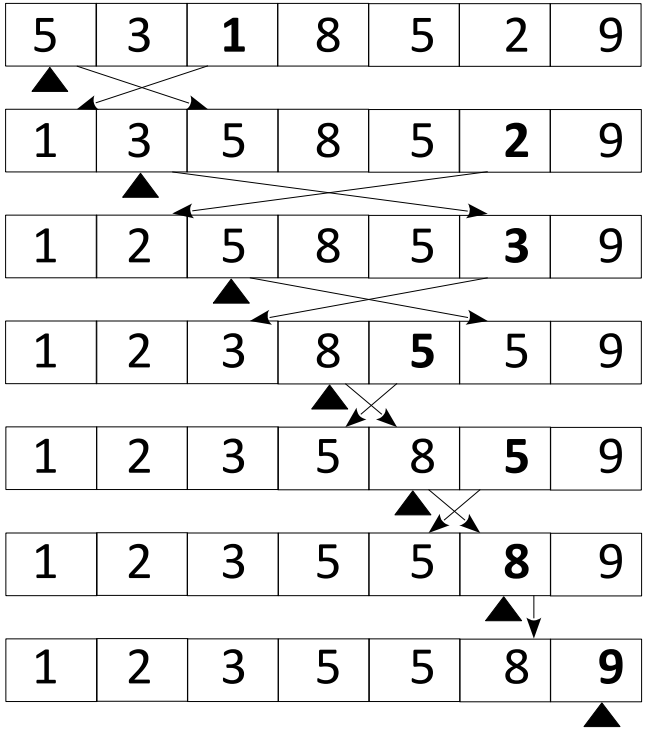
Algorithme **triSelection**
Entrée : tableau d'entiers *tab*
Type de sortie : tableau d'entiers
Variables : entiers *i, temp*, *posMin*
Début

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
- 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*-1 étapes, *n*-1 comparaisons par étape
donc ≤ (*n*-1) 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*-1 étapes, *n*-1 comparaisons par étape
donc ≤ (*n*-1) 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)

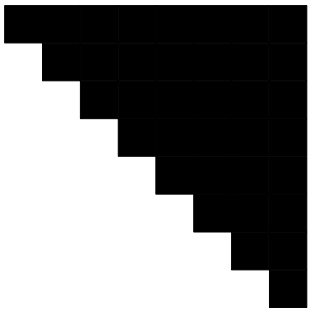
Sur un exemple ou dans le pire des cas, pour *n* éléments.

Tri à bulles

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

Complexité des tris

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

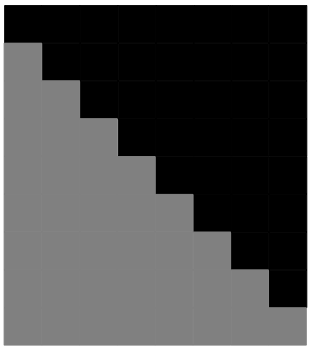
Sur un exemple ou dans le pire des cas, pour *n* éléments.

Tri à bulles

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

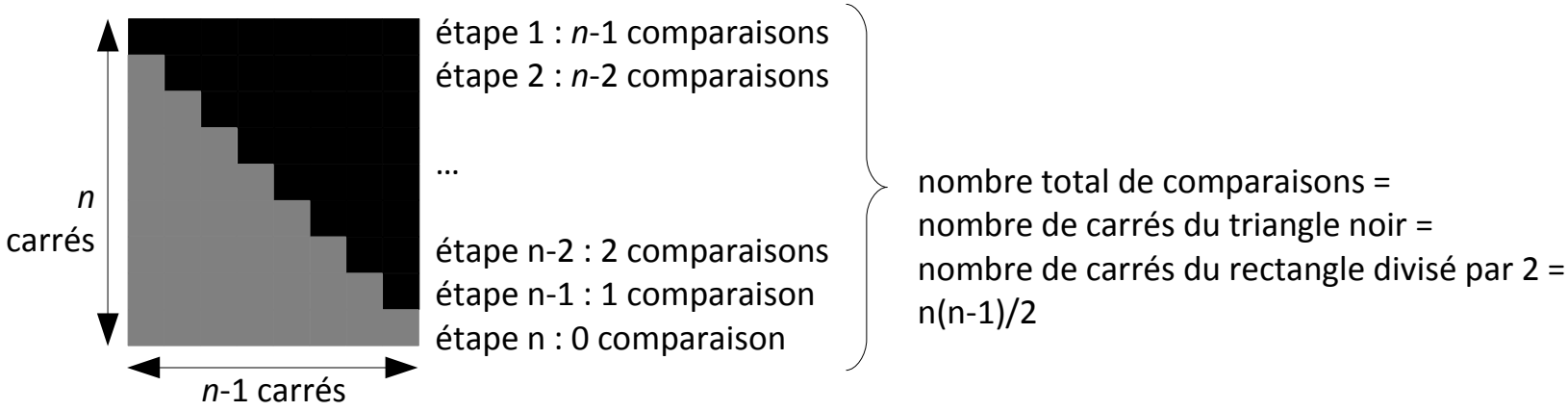
Sur un exemple ou dans le pire des cas, pour *n* éléments.

Tri à bulles

Dans tous les cas : ≤ *n*-1 étapes, *n*-1 comparaisons par étape
donc ≤ (*n*-1) 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 en sens inverse) : (*n*-1) x (*n*-1) comparaisons
Dans le meilleur cas (tableau trié) : *n*-1 comparaisons

Tri par sélection

Dans tous les cas : *n* x (*n*-1)/2 comparaisons