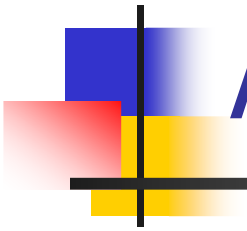
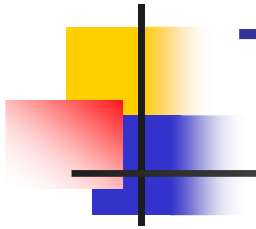


Cours 5 : Algorithmes de tri lents

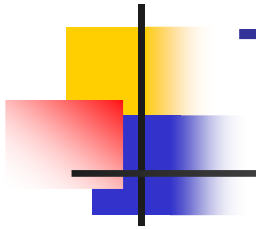
- Tri par sélection
- Tri à bulles
- Tri par insertion



Tri par sélection

- Principe : à chaque étape, on range le plus petit élément vers la gauche.

27	63	1	72	64	58	14	9



Tri par sélection

- Principe : à chaque étape, on range le plus petit élément vers la gauche.

27	63	1	72	64	58	14	9

1	63	**27**	72	64	58	14	9

Tri par sélection

- Principe : à chaque étape, on range le plus petit élément vers la gauche.

27 | 63 | 1 | 72 | 64 | 58 | 14 | 9

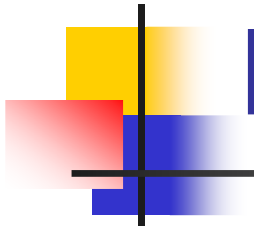
1 | 63 | **27** | 72 | 64 | 58 | 14 | 9

1 | **9** | 27 | 72 | 64 | 58 | 14 | **63**

Tri par sélection

- Principe : à chaque étape, on range le plus petit élément vers la gauche.

27 | 63 | 1 | 72 | 64 | 58 | 14 | 9
1 | 63 | **27** | 72 | 64 | 58 | 14 | 9
1 | **9** | 27 | 72 | 64 | 58 | 14 | **63**
1 | **9** | **14** | 72 | 64 | 58 | **27** | 63

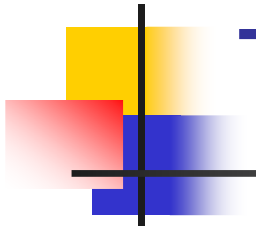


Exercice

Ecrire le pseudo-code du tri par sélection!

Tri par sélection : algorithme

```
Procédure TriSélection (ES A[n] :Tableau de T)
Variables i, j, posmin : Entier, temp : T
Début
    Pour i ← 0 à n-2
        posmin ← i
        Pour j ← i + 1 à n-1
            Si A[j] < A[posmin] Alors
                posmin ← j
            FinSi
        FinPour
        Si posmin ≠ i Alors
            temp ← A[posmin]
            A[posmin] ← A[i]
            A[i] ← temp
        FinSi
Fin
```



Tri par sélection : complexité

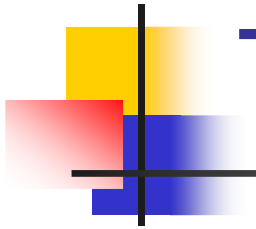
- Meilleur cas (le tableau est déjà trié) :
  - Nombre de comparaisons : (n-1)+(n-2)+…+1 = n*(n-1)/2
  - Nombre d'échanges : 0

  ⇒ Complexité : O(n²)

- Pire cas (le tableau est de cette forme 5 1 2 3 4) :
  - Nombre de comparaisons : n*(n-1)/2
  - Nombre d'échanges : n-1

  ⇒ Complexité : O(n²)

- Moyenne : O(n²)



Tri à bulle

- Principe :
  - tout élément doit être plus petit que celui qui suit
  - compare chaque élément avec l'élément suivant. Si l'ordre n'est pas bon, on permute ces deux éléments. Puis on recommence jusqu'à ce que l'on n'ait plus aucune permutation à effectuer.

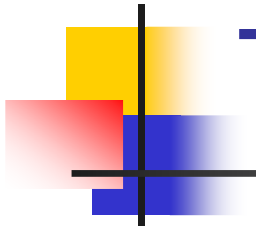
27	63	1	72	64	58	14	9

Tri à bulle

- Principe :
  - tout élément doit être plus petit que celui qui suit
  - compare chaque élément avec l'élément suivant. Si l'ordre n'est pas bon, on permute ces deux éléments. Puis on recommence jusqu'à ce que l'on n'ait plus aucune permutation à effectuer.

27	63	1	72	64	58	14	9

27	1	63	64	58	14	9	**72**



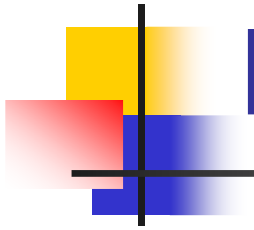
Tri à bulle

- Principe :
  - tout élément doit être plus petit que celui qui suit
  - compare chaque élément avec l'élément suivant. Si l'ordre n'est pas bon, on permute ces deux éléments. Puis on recommence jusqu'à ce que l'on n'ait plus aucune permutation à effectuer.

27	63	1	72	64	58	14	9

27	1	63	64	58	14	9	**72**

1	27	63	58	14	9	**64**	72

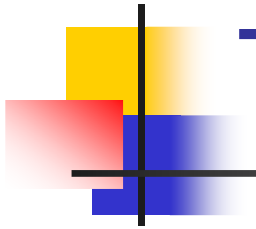


Exercice

Ecrire le pseudo-code du tri à bulle!

Tri à bulle : algorithme

```
Procédure TriBulle (ES A[n] :Tableau de T)
Variables désordre : Booléen, i : Entier, temp : T
Début
    désordre ← vrai
    Tantque désordre
        désordre ← faux
        Pour i ← 0 à n-2
            Si A[i] > A[i+1] Alors
                    désordre ← vrai
                    temp ← A[i]
                    A[i] ← A[i+1]
                    A[i+1] ← temp
            FinSi
        FinPour
    FinTantQue
Fin
```



Tri à bulle : complexité

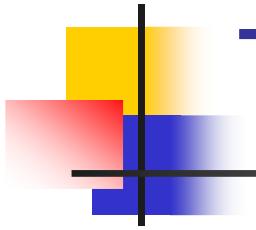
- Meilleur cas (le tableau est déjà trié) :
  - Nombre de comparaisons : n-1
  - Nombre d'échanges : 0

  ⇒ Complexité : O(n)

- Pire cas (le tableau est trié en ordre inverse) :
  - Nombre de comparaisons : n*(n-1)
  - Nombre d'échanges : n*(n-1)/2

  ⇒ Complexité : $O(n^2)$

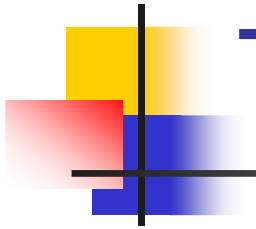
- Moyenne : $O(n^2)$



Tri par insertion

- Principe :
  - On prend les éléments dans l'ordre : on compare chaque élément avec les éléments précédents jusqu'à trouver la place de l'élément que l'on considère.
  - Il ne reste qu'à décaler les éléments du tableau pour insérer l'élément considéré à sa place dans la partie déjà triée.

27	63	1	72	64	58	14	9



Tri par insertion

- Principe :
  - On prend les éléments dans l'ordre : on compare chaque élément avec les éléments précédents jusqu'à trouver la place de l'élément que l'on considère.
  - Il ne reste qu'à décaler les éléments du tableau pour insérer l'élément considéré à sa place dans la partie déjà triée.

27	63	1	72	64	58	14	9

1	27	63	72	64	58	14	9

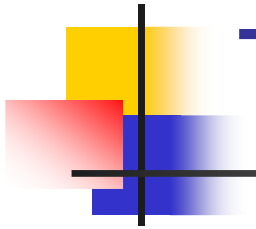
Tri par insertion

- Principe :
  - On prend les éléments dans l'ordre : on compare chaque élément avec les éléments précédents jusqu'à trouver la place de l'élément que l'on considère.
  - Il ne reste qu'à décaler les éléments du tableau pour insérer l'élément considéré à sa place dans la partie déjà triée.

27	63	1	72	64	58	14	9

1	27	63	72	64	58	14	9

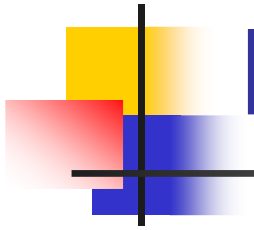
1	27	63	64	72	58	14	9



Tri par insertion

- Principe :
  - On prend les éléments dans l'ordre : on compare chaque élément avec les éléments précédents jusqu'à trouver la place de l'élément que l'on considère.
  - Il ne reste qu'à décaler les éléments du tableau pour insérer l'élément considéré à sa place dans la partie déjà triée.

27 | 63 | 1 | 72 | 64 | 58 | 14 | 9
1 | 27 | 63 | 72 | 64 | 58 | 14 | 9
1 | 27 | 63 | 64 | 72 | 58 | 14 | 9
1 | 27 | 58 | 63 | 64 | 72 | 14 | 9

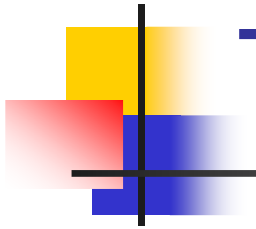


Exercice

Ecrire le pseudo-code du tri par insertion!

Algorithme tri par insertion

```
Procédure TriInsertion (ES A[n] :Tableau de T)
Variables i, j : Entier, temp : T
Début
   Pour i ← 1 à n-1
        j ← i
        Tantque (j > 0) et (A[j] < A[j-1])
                temp ← A[j]
                A[j] ← A[j-1]
                A[j-1] ← temp
                j ← j-1
        FinTantque
   FinPour
Fin
```



Tri par insertion: complexité

- Meilleur cas (le tableau est déjà trié) :
  - Nombre de comparaisons : n-1
  - Nombre d'échanges : 0

  ⇒ Complexité : O(n)

- Pire cas (le tableau est trié en ordre inverse) :
  - Nombre de comparaisons : n*(n-1)/2
  - Nombre d'échanges : n*(n-1)/2

  ⇒ Complexité : $O(n^2)$

- Moyenne : $O(n^2)$

Comparaison des tri lents