ALGORITHMIQUE II

TRIS ITERATIFS

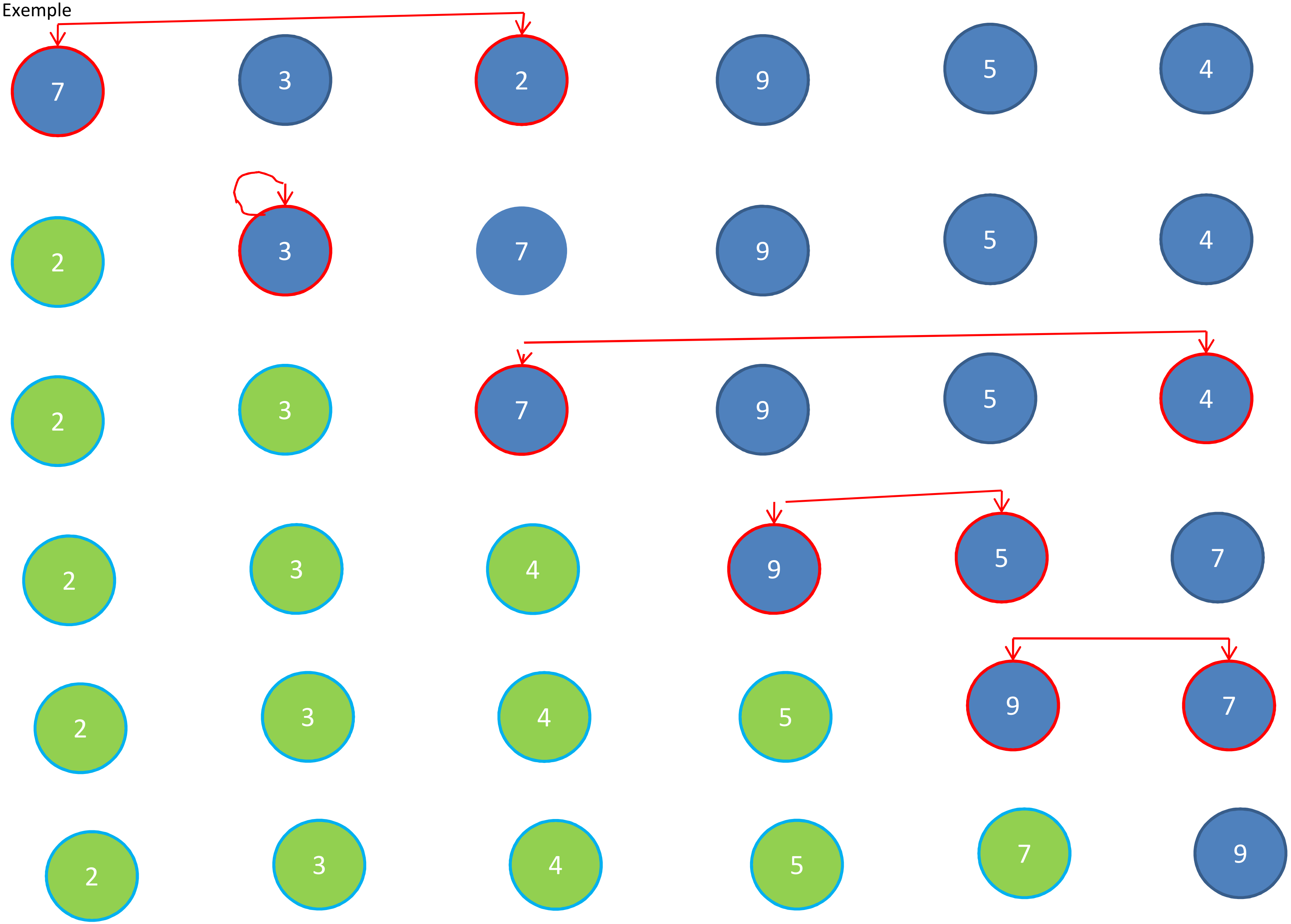
- Le problème:

  Etant donnée une suite de n nombres $(a_1, a_2, \ldots, a_n)$, on cherche une permutation (arrangement) des éléments de cette suite $(a'_1, a'_2, \ldots, a'_n)$ telle que $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.

  ♦ A partir de la suite (7,1,2,6), un algorithme de tri donne comme résultat la suite (1,2,6,7).

- On se limite aux nombres entiers rangés dans un tableau A à n éléments.
- Dans le cas où les éléments sont des collections de données (enregistrement), on trie le tableau suivant une **clé** (champ de l'enregistrement).

- Exemple



Tri par sélection:

<u>Algorithme</u>:

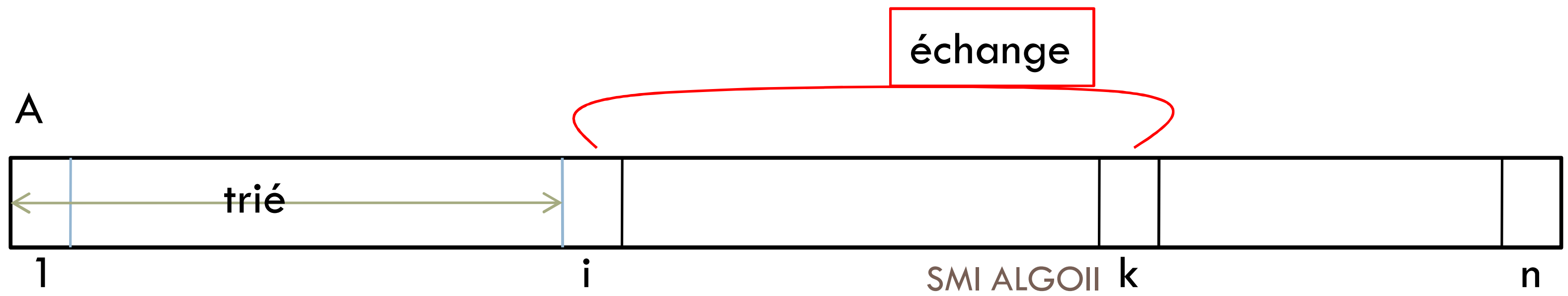
pour i=1 à n-1 faire

-chercher le 1ème minimum, A_k , de $\{A_{i+1},\ldots,A_n\}$

(K est l'indice de min$\{A_{i+1},\ldots,A_n\}$ dans le tableau A)

- échanger A_i et A_k

L'algorithme fonctionne selon le schéma suivant:



Analyse du tri par sélection

- Algorithme:

Tri_selection(A,n)

début

```
pour i := 1 à n-1 faire
    //Recherche de min{A_i, …, A_n} = A_k
    k := i;
    pour j:=i+1 à n faire
        si A[j] < A[k] alors k:=j;
        fsi;
    fpour
    // échange de A_k et A_i
    temp := A[k];
    A[k] := A[i];
    A[i] := temp;
fpour;
fin
```

- La boucle j détermine le $i^{è}$ minimum; elle tourne n-i fois (au maximum) pour faire (n-i) tests d'éléments.
- Les échanges de A[k] et A[i] demandent 3 opérations.
- La complexité du corps de la boucle j est de la forme

  a(n-i)+b, donc en O(n-i) (i=1, …, n-1).
- La complexité de l'algorithme est de l'ordre de:

$$\sum_{i=1}^{n-1}(n-i) = \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

T(n) = O(n^2)

TRI par ISERTION

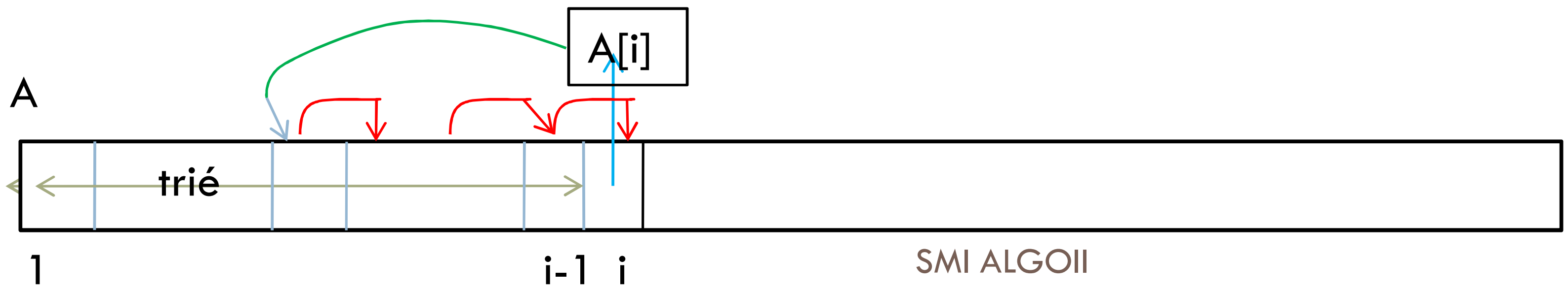
- Algorithme:

Pour i := 2 à n faire

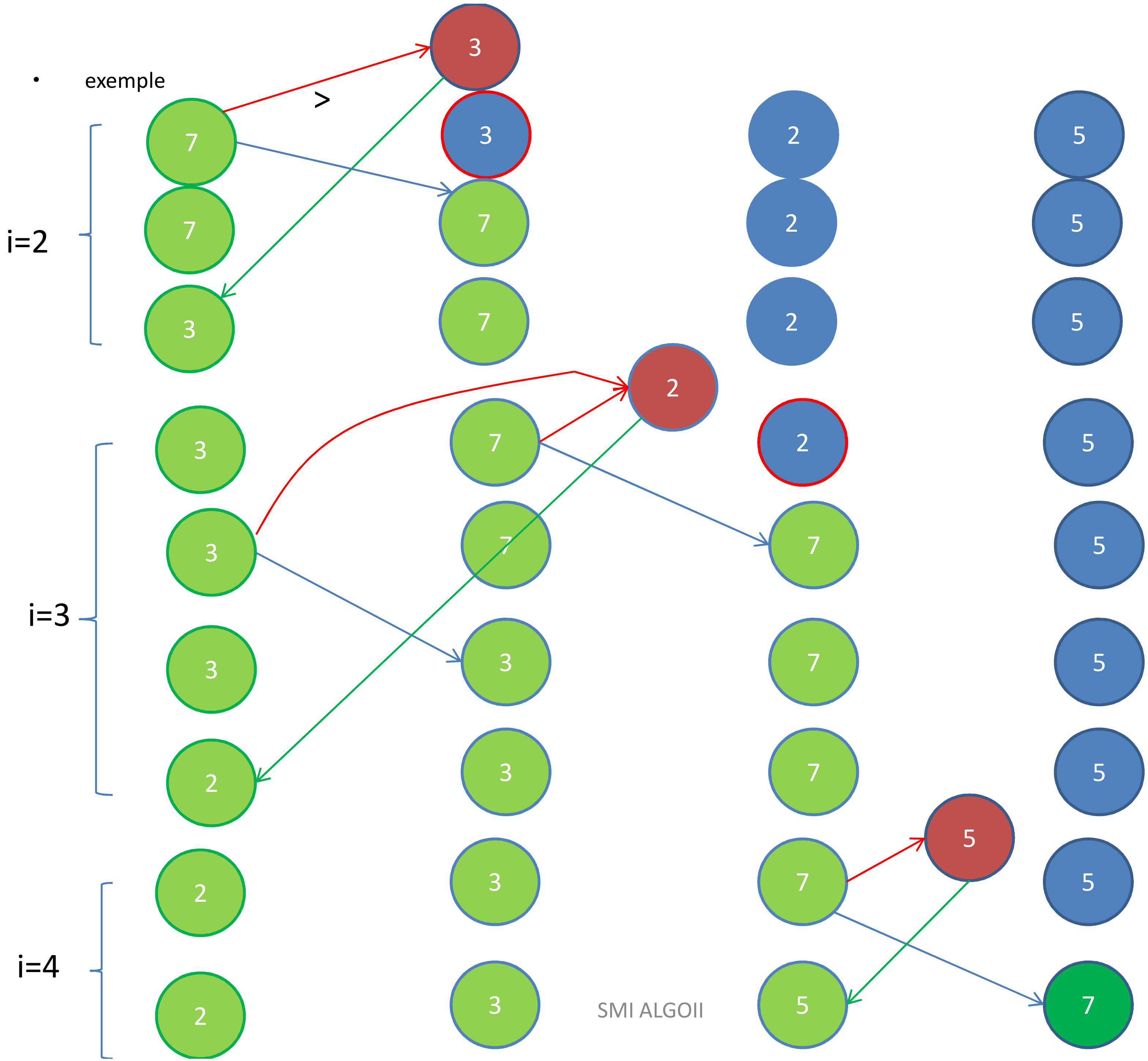
- on insère A[i], à sa place, dans le sous-tableau A[1..i-1]

(On cherche le 1er élément $\leq$ A[i] parmi {A[i-1],…,A[1]} en décalant d'une position à droite)

L'algorithme fonctionne selon le schéma suivant:



- exemple



Analyse du tri par insertion

- <u>Algorithme</u>:

Tri_Insersion(A,n)

début

pour i := 2 **à** n **faire**

cle := A[i];

j := i-1;

Tantque (j≥1) et (cle<A[j]) **faire**

A[j+1] := A[j];

j := j-1;

ftantque

A[j+1] = cle;

fpour

fin

- La boucle j tourne, dans le pire des cas, (i-1) fois

  (i-1 comparaisons et i-1 décalages)

- La complexité du corps de la boucle i est de la forme

  a(i-1)+b, pour i=2,…,n.

- La complexité de l'algorithme:

$$\sum_{i=2}^{n} a(i-1)+b = \sum_{k=1}^{n-1} ak+b = a\frac{n(n-1)}{2} + b(n-1)$$

$$\boxed{\mathbf{T(n) = O(n^2)}}$$

Tri à Bulles

- On dit qu'on a une inversion s'il existe (i,j) tels que i<j et $a_i > a_j$
- $(a_1,\ldots,a_i , a_{i+1},\ldots,a_n) \longrightarrow (a_1,\ldots,a_{i+1},a_i,\ldots,a_n)$
- Un tableau est trié s'il n'a aucune inversion. La complexité du tri est proportionnelle au nombre d'inversions qui est de l'ordre de C_n^2 (nombre de couple(i,j) tels que i<j).
- L'algorithme consiste à parcourir le tableau à trier en examinant si chaque couple d'éléments consécutifs (a_i, a_{i+1}) est dans le bon ordre ou non, si ce couple n'est pas dans le bon ordre on échange ses éléments et ce processus est répété tant qu'il reste des inversions à faire.

Tri à Bulles

- Algorithme1:

Bulles1(A,n)

Début

fini := faux;

Tant que *non fini* ***faire***

i : = 1 ;

tant que *(i < n) et (A[i] ≤ A[i+1])* ***faire***

i := i + 1 ;

ftantque*;*

si *i < n* ***alors***

échanger (A[i], A[i+1]) ;

fini := faux ; }

sinon

fini := vrai ;

fsi*;*

ftantque*;*

fin

- On remarque que les transpositions successives font pousser le maximum à la dernière position du tableau si on fait un balayage de gauche à droite et le minimum à la 1ère position si on fait un balayage de droite à gauche.
- Algorithme2:

Bulles2(A,n)

Début **i := 1;**

tantque i ≤ n-1 faire *der_ech := n;*

pour *j := n* ***à*** *i+1 pas -1* ***faire***

si *A[j-1] > A[j]* ***alors***

échanger(A[j], A[j-1]); der_ech := j;

fsi*;*

fpour*; i := der_ech;*

ftantque*;*

Fin

Le nombre de comparaison d'éléments de A ≤ $\frac{n(n-1)}{2}$

Le nombre d'échanges ≤ $\frac{n(n-1)}{2}$

$$\boxed{T(n) = O(n^2)}$$

- Exemple