1 Tri par insertion et sélection

1.1 Tri par sélection

Dans le prolongement de la recherche d'un maximum (ou d'un minimum dans une liste on va étudier un premier algorithme de tri, le **tri par sélection**

Regardons sur un exemple, la liste T = [L,I,G,N,E] comment fonctionne le tri par sélection :

i	min	0	1	2	3	4
0	4	L	I	G	N	E
1	2	E	I	G	N	L
2	2	E	G	I	N	L
3	4	E	G	I	N	L
4	4	E	G	I	L	N

1. D'abord on parcourt toute la liste pour **sélectionner la première position du plus petit élément de la liste**, dans notre exemple la position 4 pour la lettre E (en rouge)
2. Ensuite on échange de place cet élément avec le premier élément de la liste.
3. A la deuxième itération numérotée 1, la lettre E (en vert) est bien placée et on recommence ce que l'on a fait précedemment sur le reste du tableau de la position 1 à 4

D'où l'algorithme

Algorithme 1 : Tri par sélection

Données : Un tableau T ayant $n \geqslant 2$ éléments comparables
Résultat : T est trié dans l'ordre croissant

```
1  début
2      pour debut ∈ ⟦0; n − 1⟦ faire
3          //Sélection de la position du plus petit élément dans l'intervalle
             ⟦debut; n⟦
4          min ← debut
5          pour i ∈ ⟦debut + 1; n⟦ faire
6              si T[i] <T[min] alors
7                  min ← i
8              fin
9          fin
10         //On échange les valeurs en position min et deb
11         temp ← T[debut]
12         T[debut] ← T[min]
13         T[min] ← temp
14     fin
15 fin
```

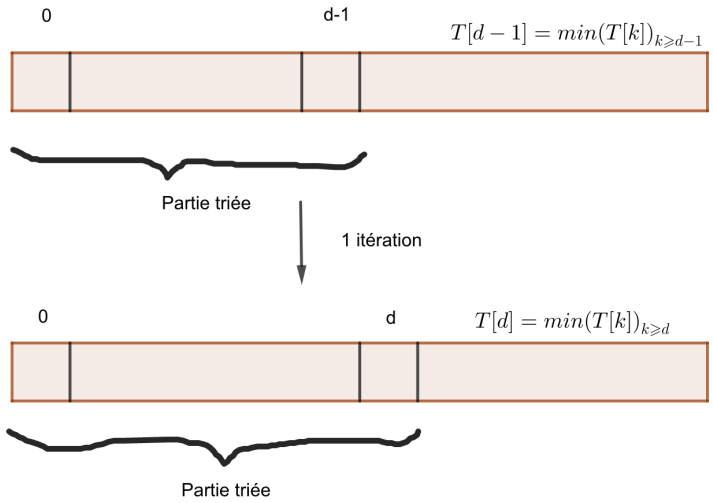
Voir ici une animation `http://www.sorting-algorithms.com/selection-sort`
Comment être sûr que ce programme est "juste" ?

1.2 Complexité du tri par sélection

Pour simplifier l'évaluation de la complexité on va regrouper toutes les affectations dans l'échange (lignes 11 à 13) pour un coût de c_1. Ensuite on majore les comparaisons (ligne 6) par un coût c_2

Combien d'échanges ?
Dans la suite on note d la variable *debut*. La variable d parcourt l'intervalle de 0 à $n-2$ donc il y a $n-1$ échanges
Combien de comparaisons ? Lorsque $d = 0$, i varie de 1 à $n-1$ donc $n-1$ comparaisons, lorsque $d = 1$, i varie de 2 à $n-1$ donc $n-2$ comparaisons puis finalement $d = n-2$ et i varie de $n-1$ à $n-1$ donc une seule comparaison
En tout il y a $(n-1)+(n-2)+...+2+1 = \dfrac{(n-1)n}{2} = \dfrac{n^2}{2} - \dfrac{n}{2}$ comparaisons
Donc $T(n) \leqslant (\dfrac{n^2}{2} - \dfrac{n}{2})c_2 + (n-2)c_1 = \dfrac{n^2}{2}c_2 + n(c_1 - \dfrac{c_2}{2}) - 2c_1$
Donc quand n devient "très grand" le terme en n^2 l'emporte sur celui en n qui l'emporte sur le terme constant, on va donc dire que la complexité du tri par sélection est **dans tous les cas** en $O(n^2)$ ou **quadratique**

1.3 Preuve du programme



Dans un premier temps on peut faire des tests (Voir TP) mais nous allons voir une **preuve** de programme qui consiste à rechercher une propriété qui reste invariante à chaque tour de boucle et qui nous aiderait pour la résolution de notre problème c'est à dire trier le tableau dans l'ordre croissant
Une propriété invariante à chaque tour de boucle s'appelle un invariant de boucle
Soit $P(d)$ "les éléments du tableau de 0 à $d-1$ sont triés dans l'ordre croissant et $T[d-1]$ est plus petit que tous les éléments après lui dans le tableau"
Montrons que $P(d)$ est un invariant de boucle pour $d \geqslant 1$:
Supposons que $P(d)$ est vraie pour $d \geqslant 1$ autrement dit tous les éléments $T[j]$ sont dans l'ordre croissant pour $0 \leqslant j \leqslant d-1$ avec $d \geqslant 1$, et $T[d-1] \leqslant T[k]$ pour tout $k \geqslant d$ au tour suivant de boucle $d \leftarrow d+1$ et alors soit $d < n-1$ soit $d = n-1$
Si $d = n-1$ alors le tableau est trié dans l'ordre croissant sinon si $d < n-1$ après avoir séléctionné dans le reste du tableau le plus petit élément et après échange $T[d]$ sera le plus petit élément de la partie du tableau entre d et $n-1$ d'où l'invariance

Enfin le programme s'arrête au bout d'un moment car l'itération se fait avec des boucles pour

1.4 Tri par insertion

L'idée et le nom du tri par insertion vient des jeux de cartes. Oublions les couleurs et considérons que les hauteurs sont classées ainsi dans l'ordre croissant $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, V, D, R]$
Imaginons que l'on vous distribue 5 cartes et que la première est V donc votre main est pour l'instant [V] puis la deuxième est 4 alors puisque 4 < V on insère 4 à sa place et votre main devient [4,V], puis la troisième carte distribuée est 7 donc insère 7 à sa place donc votre main devient [4,7,V] etc...
D'où l'algorithme

Algorithme 2 : Tri par insertion

Données : Un tableau T ayant $n \geqslant 2$ éléments comparables
Résultat : T est trié dans l'ordre croissant

```
1  début
2      pour i ∈ ⟦0; n⟦ faire
3          pour j ∈⟧0; i⟧ faire
4              si T[j] <T[j-1] alors
5                  //On échange les valeurs en position j et j-1
6                  temp ← T[j]
7                  T[j] ← T[j − 1]
8                  T[j − 1] ← temp
9              sinon
10                 sortir de la boucle
11             fin
12         fin
13     fin
14 fin
```

Voici un exemple d'exécution sur le tableau T = [L,I,G,N,E]

i	j	0	1	2	3	4
		L	I	G	N	E
1	0	I	L	G	N	E
2	0	G	I	L	N	E
3	3	G	I	L	N	E
4	0	E	G	I	L	N

1.5 Exercices -Tris par sélection et insertion

Ex 1

Soit $S(n) = 1 + 2 + 3 + ... + n - 1 + n = n + n - 1 + ... + 3 + 2 + 1$

1. Montrer que $2S(n) = n(n + 1)$
2. Que vaut $S(n)$?
3. Que vaut $1 + 2 + 3 + 4 + ... + 1000$?

Ex 2

Faire tourner l'algorithme du tri par sélection sur le tableau T = [3,1,4,1,5] avec un tableau comme dans le cours

Ex 3

Ecrire en Python et en Javascript l'algorithme du tri par sélection

Ex 4

Faire tourner l'algorithme du tri par insertion sur le tableau T = [3,1,4,1,5] avec un tableau comme dans le cours

Ex 5

Quel algorithme (tri par insertion ou sélection) sera le plus rapide sur un tableau tel que :

1. Toutes les valeurs sont identiques
2. Le tableau est déjà trié
3. Le tableau est trié dans l'ordre descendant

Ex 6

Faire une preuve du programme du tri par insertion

Ex 7

Montrer que la complexité du tri par insertion est quadratique dans le pire des cas (tableau trié dans l'ordre descendant) et linéaire dans le meilleur des cas (tableau déjà trié)

Ex 8

Faire une preuve de l'exercice 9 (Parcours séquentiel d'un tableau)

Ex 9

Ecrire en Python et en Javascript l'algorithme du tri par insertion