Algo

- Numero: 2
- Prof: Alexandre Duret-Lutz
- Date: 19 Octobre 2017

Tri par selection

```
def SelectionSort(A, n):
  for i in range(n - 1):
    min = i
    for j in range(i + 1, n):
      if A[j] < A[min]:
        min = j
    swap(A, min, i)
  return A
```

A la i^{eme} iteration les i^{eme} plus petites valeurs seront triees...

Dans un tri on va souvent comparer les elements entre eux et echanger les elements. On veut donc compter le nombre de comparaison et le nombre d'echange.

$C(n)$ = Nombre de comparaison de `SelectionSort` pour une taille `n`.

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n-i-1) = \frac{(n-1+1)(n-1)}{2} = \frac{n(n-1)}{2}$$

On aimerai avoir un $T(n) =$ temps d'execution. Il faut determiner un cout d'execution de chaque ligne et le multiplier au nombre de fois qu'elle est execute. Cela va donner un encadrement de $C(n)$.

Cela donne:

$a'n^2 + b'n + c \leq T(n) \leq an^2 + bn + c$

- La borne inferieur est le cas ou le tableau est deja trie.
- La borne superieur est le cas ou le tableau est trie dans l'ordre inverse.

Conclusion: $T(n) =$ une fonction quadratique.

Tri par insertion

```
def InsertionSort(A,n):
  for i in range(1, n):
    key = A[i]
    j = i - 1
    while (j >= 0 && A[j] > key):
      A[j + 1] = A[j]
      j-=1
    A[j + 1] = key
  return A
```

A la i^{eme} iteration les i^{eme} premieres valeurs de `A` sont triees...

On fait la meme chose, on compte le nombre de fois que chaque ligne est execute. On leur asssocie ensuite un coup.

On note t_i le nombre d'execution du corps de la boucle **`while`** a l'iteration `i`.

- Le meilleur cas est le tableau trie dans l'ordre croissant. $(t_i = 0)$
- Le pire cas est le tableau trie dans l'ordre decroissant. $(t_i = i)$

Cela donne:

$dn + 2 \leq T(n) \leq an^2 + bn + c$

En moyenne ?

On veut la complexite moyenne sur toutes les mermutations qui existent.

La somme des t_i est la somme de toutes les inversions. Donc on veut savoir le nombr ede permutation en moyenne en chaque iteration.

Il faut imaginer l'ensemble des ordres possible pour un tableau de `n` elements. On peut faire une bijection entre les ordres qui ont une inversion $0 \iff n$

En moyenne un ordre possede donc $\frac{1}{2}\begin{pmatrix} n \\ 2 \end{pmatrix}$ inversions. $\sum_{i=1}^{n-1} t_i$ vaut en moyenne $\frac{n(n-1)}{4}$

$T(n)$ est **quadratique** en moyenne.