PYTHON № 7 : MÉTHODES DE TRI

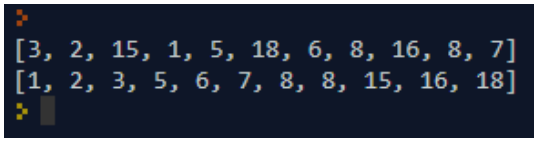
Nous allons présenter plusieurs méthodes de tri :

1. le tri sélection
2. le tri insertion
3. le tri à bulles
4. le tri dénombrement
5. le tri fusion
6. le tri rapide ou dichotomique
7. le tri topologique

1 le TRI SÉLECTION

à l'étape i : on cherche dans les éléments restants le plus petit
après l'étape i : les éléments de 0 à i sont par ordre croissant et sont inférieurs aux restants

```
1  # tri par sélection
2  def tri_selection(L):
3    n=len(L)
4    for i in range(0,n-1):
5      # recherche du min dans L[i:n]
6      m=i
7      for j in range(i+1,n):
8        if L[j]<L[m]:
9          m=j
10     L[m],L[i]=L[i],L[m]   # échange L[i] et L[m]
11
12 L=[3,2,15,1,5,18,6,8,16,8,7]
13 print(L)
14 tri_selection(L)
15 print(L)
```



Bien prendre le temps d'analyser le programme et en particulier de comprendre pourquoi i varie de 0 à n-2 en ligne 4

2 le TRI INSERTION

à l'étape i : on place de l'élément i dans la liste 0 à i-1

après l'étape i : les éléments de 0 à i sont par ordre croissant ; rapidité $\sim \frac{n^2}{2}$

```
1  # tri par insertion
2  def tri_insertion(L):
3    n=len(L)
4    for i in range(0,n):
5      # on insère l'élément à sa place en décalant
6      # vers la droite les plus grands
7      x=L[i] ; j=i
8      while j>0 and L[j-1]>x:
9        L[j]=L[j-1] ; j=j-1
10     L[j]=x
11
12 L=[3,2,15,1,5,18,6,8,16,8,7]
13 print(L)
14 tri_insertion(L)
15 print(L)
```

3 le TRI à BULLES (de champagne)

à l'étape j : on parcourt la liste en inversant les couples consécutifs non triés

conséquence : les gros nombres remontent comme des bulles en fin de liste ; rapidité $\sim \frac{n^2}{2}$

```
1  # tri à bulles
2  def tri_bulle(L):
3    n=len(L)
4    for j in range(0,n-1):
5      for i in range(0,n-j-1):
6        if L[i+1]<L[i]:
7          L[i],L[i+1]=L[i+1],L[i]
```

4 le TRI par DÉNOMBREMENT

objectif : on crée une autre liste qui regroupe les occurrences de 0 et max(L)

conséquence : rapidité $\sim n$ mais demande beaucoup de place ; idéal tri des âges (petit max)

```
1  # tri par dénombrement
2  def tri_denb(L):
3    maxi=max(L)
4    V = [0 for j in range(maxi+1)]; n = len(L)
5    # remplissage du tableau des occurrences
6    for i in range(0,n):
7      V[L[i]] = V[L[i]] + 1
8    # tri du tableau L
9    k = 0
10   for j in range(maxi+1):
11     L[k:k+V[j]]=[j for i in range(V[j])]
12     k=k+V[j]
13   return(L)
```

5 le TRI par FUSION ou DICHOTOMIQUE

objectif : on partage la liste en 2 sous-liste que l'on trie de façon récursive

conséquence : rapidité $\sim n \log_2 n$

```
1  # tri par fusion ou dichotomique
2  def fusion(L1,L2):
3    if L1==[]:
4      return L2
5    elif L2==[]:
6      return(L1)
7    elif L1[0]<L2[0]:
8      return([L1[0]]+fusion(L1[1:],L2))
9    else:
10     return([L2[0]]+fusion(L1,L2[1:]))
11
12 def tri_fusion(L):
13   n=len(L)
14   if n<=1:
15     return L
16   else:
17     m=n//2
18     L[0:m]=tri_fusion(L[0:m])       # tri du tableau de gauche
19     L[m:n]=tri_fusion(L[m:n])       # tri du tableau de droite
20     L[0:n]=fusion(L[0:m],L[m:n])    # fusion des 2 listes triées
21     return(L)
```


6 le TRI RAPIDE

objectif : on choisit un pivot qui partage la liste en 2 : ceux au dessus et ceux en dessous

conséquence : rapidité $\sim 2n \ln n$ (cas aléatoire)

```
# tri rapide
def partition(L,p,r):
  x=L[r]                # le pivot x est l'élément le plus à droite
  i=p-1
  for j in range(p,r):
    if L[j]<=x:
      i=i+1
      L[i],L[j]=L[j],L[i]
  L[i+1],L[r]=L[r],L[i+1]
  return(i+1)           # on renvoie la nouvelle place du pivot

def tri_rapide(L,p,r):
  if p<r:
    q=partition(L,p,r)
    tri_rapide(L,p,q-1)
    tri_rapide(L,q+1,r)

def tri_rapide_fct(L):
  tri_rapide(L,0,len(L)-1)
```

7 le TRI TOPOLOGIQUE

Voir : Tri topologique

8 TRI : résumé

- animations intéressantes ici ou là
- Voici un tableau de synthèse pour la complexité des tris au programme nsi :

tri	complexité	meilleur cas - en moyenne - pire des cas
sélection	O(n^2)	
insertion	O(n^2)	
bulle	O(n^2)	
dénombrement		
fusion	O(n log n)	
rapide (quicksort)	O(n log n)	
topologique		