Correction TD Complexité

Exercice 1 :

1. a .Donner une liste de 6 éléments pour laquelle le tri par insertion s'effectuera avec une complexité dans le pire des cas.
   b. Écrire toutes les étapes et vérifier que le nombre d'étapes correspond à la complexité vu en cours.

La liste 6 – 5 – 4 – 3 – 2 – 1 réalise le pire cas pour le tri par insertion. En effet, les étapes seront :

6 – 5 – 4 – 3 – 2 – 1 départ
5 – 6 – 4 – 3 – 2 – 1 étape 1
5 – 4 – 6 – 3 – 2 – 1 étape 2
4 – 5 – 6 – 3 – 2 – 1 étape 3
4 – 5 – 3 – 6 – 2 – 1 étape 4
4 – 3 – 5 – 6 – 2 – 1 étape 5
3 – 4 – 5 – 6 – 2 – 1 étape 6
3 – 4 – 5 – 2 – 6 – 1 étape 7
3 – 4 – 2 – 5 – 6 – 1 étape 8
3 – 2 – 4 – 5 – 6 – 1 étape 9
2 – 3 – 4 – 5 – 6 – 1 étape 10
2 – 3 – 4 – 5 – 1 – 6 étape 11
2 – 3 – 4 – 1 – 5 – 6 étape 12
2 – 3 – 1 – 4 – 5 – 6 étape 13
2 – 1 – 3 – 4 – 5 – 6 étape 14
1 – 2 – 3 – 4 – 5 – 6 étape 15

La liste contient 6 éléments donc la complexité dans le pire des cas est $\frac{n\times(n-1)}{2} = \frac{6\times5}{2} = \frac{30}{2} = 15.$

Cela correspond bien

2. a .Donner une liste de 6 éléments pour laquelle le tri par sélection s'effectuera avec une complexité dans le pire des cas.
   b. Écrire toutes les étapes et vérifier que le nombre d'étapes correspond à la complexité vu en cours.

La liste 2 – 3 – 4 – 5 – 6 – 1 réalise le pire cas pour le tri par insertion. En effet, les étapes seront :

2 – 3 – 4 – 5 – 6 – 1 départ
2 – 3 – 4 – 5 – 6 – 1 étape 1 compare
2 – 3 – 4 – 5 – 6 – 1 étape 2 compare
2 – 3 – 4 – 5 – 6 – 1 étape 3 compare
2 – 3 – 4 – 5 – 6 – 1 étape 4 compare
1 – 3 – 4 – 5 – 6 – 2 étape 5 échange
1 – 3 – 4 – 5 – 6 – 2 étape 6 compare
1 – 3 – 4 – 5 – 6 – 2 étape 7 compare
1 – 3 – 4 – 5 – 6 – 2 étape 8 compare
1 – 2 – 4 – 5 – 6 – 3 étape 9 échange
1 – 2 – 4 – 5 – 6 – 3 étape 10 compare
1 – 2 – 4 – 5 – 6 – 3 étape 11 compare
1 – 2 – 3 – 5 – 6 – 4 étape 12 échange
1 – 2 – 3 – 5 – 6 – 4 étape 13 compare
1 – 2 – 3 – 4 – 6 – 5 étape 14 échange
1 – 2 – 3 – 4 – 5 – 6 étape 15 échange

La liste contient 6 éléments donc la complexité dans le pire des cas est $\frac{(n-1)\times n}{2} = \frac{5\times 6}{2} = \frac{30}{2} = 15.$

Cela correspond bien

Exercice 2 :

1. Que font les lignes ci dessous :

```
import random
……..
a = 20
b = 100
T =[a + randint(b – a) for i in range(10)]
```

Ces lignes permettent de générer une liste de 10 entiers choisis aléatoirement entre 20 et 100.

2. Écrire une fonction qui prend en argument trois nombres entiers, n, a et b et qui renvoie une liste de n entiers choisis aléatoirement entre a et b.

```
def tab_aleatoire(n,a,b):
    T=[a+randint(0,b-a) for i in range(n)]
    return T
```

Exercice 3 :

1. En utilisant la fonction ci dessus ainsi que l'algorithme du tri insertion vu en cours, écrire un programme qui mesure le temps de tri (avec le tri insertion) d'une liste de 2^{10} entiers choisis aléatoirement entre -100 000 et 100 000.

2. Modifier le programme ci dessus (avec une boucle) de façon à mesurer le temps de tri pour des listes de de 2^{10} ; 2^{11} ; 2^{12} ; 2^{13}; 2^{14} ; 2^{15} entiers choisis aléatoirement entre -100 000 et 100 000 et stocker ces temps dans une liste nommé `Temps_insertion`.
Faire afficher cette liste.

```
Taille_echantillon=[]
Temps_insertion=[]
for i in range(10,16):
    n=2**i
    Taille_echantillon.append(n)
    T=tab_aleatoire(n,-100000,100000)
    t1=time.time()
    T=tri_insertion(T)
    t2=time.time()
    Temps_insertion.append(t2-t1)
print(Temps_insertion)
```

3. De 2^{10} à 2^{11}, la taille de la liste à triée double et cela est vrai à chaque augmentation de la puissance de 2.
Que pouvez dire des temps d'exécution des tris pour chaque doublement de la taille de la liste à triée ?
Cela est-il en accord avec avec les propriétés vus en classe ?
Combien de temps faudrait-il approximativement de temps (sur votre machine) pour trié une liste d'un milliard d'éléments avec le tri insertion ?

Sur ma machine, on obtient les temps d'exécution suivant :

```
 RESTART: /home/nicolas/Documents/Année_2019-2020/NSI/BLOC_5/Tris_complexite.py
[0.057576894760131836, 0.22125983238220215, 0.8977389335632324, 3.608476400375366, 14.405271530151367, 60.88550662994385]
```

On remarque qu'à chaque fois que la taille de la liste double le temps d'exécution est environ multiplier par 4.

Cela est en accord avec le cours puisque le tri par insertion est d'une complexité quadratique ($O(n^2)$) donc quand la taille de la liste est multipliée par 2 le temps d'exécution est environ multiplier par 2^2 soit 4.

Un milliard est environ égale à 2^{30}. Ainsi pour passer de 2^{15} à 2^{30}, il faut multiplier 15 fois par 2 soit par 2^{15}.
Le temps d'exécution sera donc environ multiplier par 4^{15}.

60 secondes × 4^{15} ≈ 64 000 000 000 secondes ≈ 745654 jours ≈ 2043 années.

Le tri par insertion n'est donc pas un tri très rapide.

4. La fonction ci-dessous permet de faire afficher le graphique représentant le temps d'exécution en fonction du nombre d'éléments à trier.
Adapter la à votre programme de façon à afficher ce graphique.

```
import pylab
…….
def trace(Lx,Ly):
    pylab.clf()
    titre="Temps de tri en fonction de la taille de l'échantillon"
    pylab.title(titre)
    pylab.xlabel("Taille echantillon")
    pylab.ylabel("temps du tri")
    pylab.axis([-5,35000,-5,80])
    pylab.axhline(color='black')
    pylab.axvline(color='black')
    pylab.grid()
    pylab.plot(Lx,Ly,'r')
    pylab.show()
```

Exercice 4 :

Même exercice que le 3 avec le tri sélection