TP Tris

# Travaux préliminaires

### Explication de code

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | L=[1,5,8,12,14,16,53] | | |
| 2 | res=True | | |
| 3 | for i in range(1,len(L)): | | |
| 4 |  | if L[i-1]>L[i]: | |
| 5 |  |  | res=False |
| 6 | print(res) | | |

1. Que fait l’algorithme ci-dessus ?
2. Que se passerait-il si on avait L=[4,6,9,11] ?

## Minimum et sous liste d'une liste

1. Ecrire une fonction mini ayant comme argument une liste qui détermine le plus petit élément de cette liste. Tester la fonction lorsque 5.
2. Ecrire une fonction creerPairImp ayant comme argument une liste d'entiers qui renvoie deux listes :  
    - La liste des pairs notée Pa  
    - La liste des impairs notée Imp

# Tri par insertion et complexité

## Tri par insertion

On donne ci-dessous l'algorithme de tri par insertion pour une liste de éléments :

|  |  |  |
| --- | --- | --- |
| 1 | Pour allant de 2 à n | |
| 2 |  | Insérer l'élément L[i] à sa place dans L[0,i-1] |
| 3 | Fin pour | |

1. Ecrire une fonction triInsertion prenant en argument la liste L1et qui renvoie la liste L2contenant les éléments de L1triés par ordre croissant.
2. Modifier la fonction triInsertion pour qu'elle renvoie également le nombre de comparaisons d'entiers réalisées.

## Complexité du tri par insertion

Soit n la longueur de la liste L1. On souhaite tracer l’évolution de la complexité temporelle (évaluée ici par le nombre de comparaisons à effectuer pour réaliser le tri de la liste L1) en fonction de la longueur n de la liste L1.

Afin que la complexité obtenue pour un n donné soit fiable il est nécessaire de répéter plusieurs fois la même mesure et d’utiliser la valeur moyenne des résultats obtenus. On fera donc varier n de à par pas de et on répètera chaque mesure fois.

On peut montrer que le nombre de comparaisons en moyenne est équivalent à .

1. Ecrire un script qui permet en traçant un graphe de vérifier cette propriété. On pourra construire des listes aléatoires avec la fonction randint du module random. Il est possible que vous soyez obligés de modifier la valeur maximale de n si le temps de calcul est trop important.

# Tri fusion

## Tri fusion

Soient deux listes L1 et L2 préalablement triées par ordre croissant que l'on souhaite fusionner dans la liste L.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Tant qu'une des deux listes n'est pas vide | | |
| 2 |  | Si le premier élément de L1 est plus petit que le premier élément de L2. | |
| 3 |  |  | Retirer le premier élément de L1 et l'ajouter à L |
| 4 |  |  | Sinon, retirer le premier élément de L2 et l'ajouter à L |
| 5 | Si L1 n'est pas vide | | |
| 6 |  | Elle est ajoutée à L | |
| 7 |  | Sinon L2 est ajoutée à L | |
| 8 | Renvoyer L | | |

1. Ecrire une fonction fusion qui prend en argument les deux listes L1 et L2 et qui renvoie une liste L3, fusion des deux listes L1 et L2 et dont les éléments sont rangés par ordre croissant.
2. Ecrire une fonction triFusion qui prend en argument une liste d'entiers L non ordonnée et qui renvoie cette liste triée par la méthode du tri fusion.

## Complexité du tri fusion

1. Modifier la fonction triFusion pour qu'elle renvoie également le nombre de comparaisons nécessaires pour effectuer le tri.
2. Modifier le script de la question 7 pour que celui trace sur le même graphe l'évolution des complexités des deux méthodes de tri. Commenter la complexité du tri fusion et la comparer à celle du tri par insertion.

# Tri rapide (QuickSort)

## Tri rapide (en place)

|  |  |
| --- | --- |
| Idée | Le tri rapide a été inventé en 1960 par Tony Hoare.  C'est une méthode récursive qui s'appuie sur le principe "diviser pour régner" :   * on choisit un pivot (aléatoirement ou pas) parmi tous les éléments à trier, * on réorganise tous les éléments de sorte que tous les éléments inférieurs soient à gauche du pivot et les éléments supérieurs à droite * trier chaque sous-ensemble (droite et gauche) en y choisissant un nouveau pivot * lorsque chaque sous-ensemble ne contient qu'un élément, la liste est triée |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | On choisit le premier élément de L comme pivot | | |
| 2 | On initialise le curseur à c=1 (premier élément après le pivot) | | |
| 3 | Pour tous les éléments qui suivent le pivot | | |
| 4 |  | Si l'élément d'indice i est inférieur au pivot | |
| 5 |  |  | On l'échange avec l'élément d'indice c et on fait c=c+1 |
| 6 |  |  | Sinon on ne fait rien |
| 7 | On échange le pivot avec le terme d'indice c-1 | | |

1. Ecrire une fonction partition prenant en argument une liste L et deux indices g et d () qui définissent une sous liste à partitionner. Cette fonction renverra la position du pivot dans la liste L après partition.
2. Etablir le code Python d'une fonction triRapideRec qui prend en argument une liste d'entiers L et les deux indices g et d qui définissent la sous-liste à trier. Cette fonction doit effectuer le tri rapide de L[g:d].
3. Ecrire le code Python de la fonction triRapide qui prend en argument une liste d'entiers non triés L et qui la renvoie triée.

## Influence du choix du pivot

Dans cette partie on souhaite évaluer l’influence du choix du pivot sur la rapidité du tri rapide. Pour cela on envisage trois stratégies :

* On prend, comme pivot, le premier élément de la liste à partitionner,
* On choisit l’élément le plus proche de la valeur moyenne des éléments de la liste à partitionner,
* On choisit le pivot au hasard parmi les éléments de la liste à partitionner.

En modifiant le code de la fonction partition, écrire trois fonctions de partition appliquant ces stratégies.

1. Modifier la fonction triRapide pour qu'elle renvoie le temps t1 nécessaire à la réalisation du tri et le nombre nc de comparaisons. Le temps t1 englobe tout (y compris le temps nécessaire au choix du pivot), en revanche, le nombre de comparaisons nc ne comprendra pas les éventuelles comparaisons nécessaires au choix du pivot.
2. Ecrire un script qui fait varier la longueur n de la liste à trier de 10000 à 100000 éléments par pas de 10000 éléments et qui trace sur un graphe l’évolution du temps t1 en fonction de n, et sur un second graphe l’évolution de nc en fonction de n pour les trois stratégies de choix du pivot. Afin que les valeurs de t1 et nc soient fiables on fera une moyenne sur 10 essais pour chaque valeur de n.

## Complexité du tri rapide

1. Modifier les fonctions triInsertion et triFusion pour qu’elles renvoient également le temps nécessaire pour effectuer le tri.
2. Modifier le script de la question 16 pour que celui-ci trace sur le même graphe l’évolution du temps d’exécution des 3 méthodes de tri en fonction de la longueur n de la liste à trier. Que peut-on dire des complexités des trois méthodes.

# Tri à bulles (hors programme)

## Définition

|  |  |
| --- | --- |
| Déf. | Soit une liste L de n éléments. Le tri à bulles consiste à comparer les deux premiers termes de la liste. S’ils sont désordonnés, on les échange. On recommence ensuite avec le deuxième et le troisième terme du tableau et ainsi de suite.  Ainsi, au bout d’un premier passage, l’élément le plus grand est bien placé : il est remonté comme une bulle. On recommence l’opération jusqu’à ce que le tableau soit trié. |

Pour une liste L de n éléments, on trouve l'algorithme suivant :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | permut=vrai | | | |
| 2 | Tant que permut=vrai faire : | | | |
| 3 |  | permut=faux | | |
| 4 |  | Pour j allant de 1 à n-1 | | |
| 5 |  |  | si L[j]>L[j+1] alors : | |
| 6 |  |  |  | échanger L[j] et L[j+1] |
| 7 |  |  |  | permut=vrai |
| 8 |  |  | fin si | |
| 9 |  | fin pour | | |
| 10 | fin tant que | | | |

## Tri à bulles

1. Ecrire une fonction triAbulles qui prend en argument une liste d'entiers L et qui renvoie cette même liste triée par la méthode du tri à bulles.

## Complexité du tri à bulles

1. Modifier la fonction triAbulles pour qu’elle renvoie également le temps nécessaire et le nombre de comparaisons pour effectuer le tri.
2. Modifier le script de la question 18 pour que celui-ci trace sur le même graphe l’évolution du temps d’exécution des 4 méthodes de tri en fonction de la longueur n de la liste à trier. Que peut-on dire des complexités des quatre méthodes.