

IPT Devoir 5- Proposition de solutions

Solution 1 Tri par comptage

1. La fonction stat :

Dénombrement par élément

```
def stat(L,N):  
    C=(N+1)*[0]  
    for e in L: C[e]=C[e]+1  
    return C
```

2. L'appel stat([[2,1,1,0,1,4],4) retourne [1,3,1,0,1].

3. Tri par comptage :

Tri par comptage

```
def tri_comptage(L,N):  
    C=stat(L,N)  
    T=[]  
    for i,e in enumerate(C): T=T+e*[i]  
    return T
```

4. Un invariant de la i ème étape de tri_comptage est : "tous les éléments inférieure strictement à i , de L, sont ordonnés et constitue la liste T".

5. La fonction stat crée une liste de longueur $N+1$ et parcourt la liste L une fois. Le nombre d'affectation est de $N+1+n$ avec n la longueur de L.

La boucle for reconstitue élément par élément la liste triée et donc il y a n affectation et la longueur de la boucle est N .

Au final, la complexité est en $\mathcal{O}(N+n)$

Solution 2 1. Suivi de la fonction g :

Entrées	Sorties
$e=2, [1,1,3,5,4,3,1], pos=4$	$[1,1,2,3,5,4,3,1]$
$e=2, [1,1,3,5,4,3,1], pos=3$	$[1,1,2,3,5,4,3,1]$
$e=2, [1,1,3,5,4,3,1], pos=2$	$[1,1,2,3,5,4,3,1]$
$e=2, [1,1,3,5,4,3,1], pos=1$	$[1,1,2,3,5,4,3,1]$

2.

Méthode

→ Le tri par **insertion** consiste à :

- parcourir la liste de gauche à droite ;
- à chaque étape, l'élément considéré, est classé parmi les éléments qui le précèdent (et donc qui sont déjà ordonnés).

3. Une fonction du tri par insertion utilisant g :

```
def tri(L):  
    for i in range(1,len(L)):  
        e=L.pop(i)  
        L=g(e,L,i-1)  
    return L
```

Solution 3 Les différents tris :

Tri par sélection

Méthode

→ Le tri par sélection consiste à :

- pour i variant sur les indices de la liste (de longueur n), en partant de la gauche :
- on recherche un minimum parmi les $n-i$ derniers éléments de la liste
- on le place en position i par un échange d'éléments
- Dans tous les cas, la complexité est $\mathcal{O}(n^2)$.

- Ce tri est au programme.

Tri par comptage

Méthode

→ Le tri par comptage consiste à :

- sachant que les éléments d'une liste sont dans $[0,N]$, on dénombre les éventuels 0, les 1, ... les N .
- on construit une liste comportant autant de 0 suivis d'autant de 1, ... et d'autant de N que la liste de départ.
- Dans tous les cas, la complexité est $\mathcal{O}(n+N)$.
- Ce tri est au programme.

Tri par comparaison

Méthode

→ Le tri par comparaison consiste à :

- pour chaque élément, on parcourt la liste et détermine sa position dans la liste triée : la position de $L[i]$ est le nombre d'éléments $j \neq i$ tels que :
 - $L[j] \leq L[i]$ lorsque $j < i$,
 - $L[j] < L[i]$ lorsque $j > i$,
- enfin, on place chaque élément à sa place dans la liste triée.
- Dans tous les cas, la complexité est $\mathcal{O}(n^2)$.
- Ce tri n'est pas au programme.

Tri par insertion

Méthode

→ Le tri par **insertion** consiste à :

- parcourir la liste de gauche à droite ;
- à chaque étape, l'élément considéré, est classé parmi les éléments qui le précèdent (et donc qui sont déjà ordonnés).
- Dans le cas le meilleur, la complexité est $\mathcal{O}(n)$. C'est le cas d'une liste déjà triée.
- Dans le cas le pire, la complexité est $\mathcal{O}(n^2)$. C'est le cas d'une triée dans l'ordre décroissant.
- Ce tri n'est pas au programme.

Tri rapide

Méthode

→ Le tri rapide consiste à :

- si la liste contient au plus un élément, alors on retourne la liste ; sinon, on considère le premier élément ;
- on crée deux sous-listes, une contenant les éléments qui lui sont inférieurs et l'autre pour les autres éléments (ceux qui lui sont strictement supérieurs) ;
- on applique le tri aux deux sous-listes obtenues et on concatène le résultat.
- Dans le cas le meilleur, la complexité est $\mathcal{O}(n \ln(n))$. C'est le cas d'une liste où chaque découpe se fait en deux sous listes de tailles similaires.
- Dans le cas le pire, la complexité est $\mathcal{O}(n^2)$. C'est le cas d'une liste triée.
- Ce tri est au programme.

► Tri fusion

Méthode

→ Le tri fusion consiste à :

- si la liste contient au plus un élément, alors on retourne la liste ; sinon, on la découpe en deux sous-listes de tailles comparables ;
- on applique le tri sur chacune d'elle et on fusion les résultats
- la fusion est une sous fonction qui permet de constituer une liste triée à partir de deux listes déjà triées
- Dans tous les cas, la complexité est $\mathcal{O}(n \ln(n))$.
- Ce tri est au programme.

► Tri à bulles

Méthode

→ Le tri à bulles consiste à :

- on parcourt la liste de gauche à droite ;
- dès que deux éléments consécutifs sont mal ordonnés, on les échange ;
- on recommence le parcours jusqu'à ce que la liste soit triée.
- Dans le cas le meilleur, la complexité est $\mathcal{O}(n)$. C'est le cas d'une liste déjà triée.
- Dans le cas le pire, la complexité est $\mathcal{O}(n^2)$. C'est le cas d'une liste triée dans l'ordre inverse.
- Ce tri n'est pas au programme.

Solution 4 1. La problématique est d'identifier une racine de la fonction $t \mapsto e^t - 2$ sur $[0, 2]$: la fonction retourne une approximation à la précision p d'une telle racine.

Dans le cas présent, il s'agit de $\ln(2)$.

2. Script récursif de la dichotomie :

```
def dichotomie(f,a,b,p):
    if abs(b-a)<2*p:
        return (a+b)/2
    else:
        c=(a+b)/2
        if f(c)*f(a)>0:
            return dichotomie(f,c,b,p)
        else:
            return dichotomie(f,a,c,p)
```