Thèmes d'étude

Thème : Tris.
Commentaires :
- algorithmique quadratique : tri par insertion, par sélection ;
- tri par partition-fusion ;
- tri par comptage.

On fait observer différentes caractéristiques (par exemple stable ou non, en place ou non, comparatif ou non ...).

1 Présentation

Le tri de données ou de valeurs est omniprésent en informatique. Pour cela, beaucoup d'algorithmes ont été développés afin de réaliser des tris rapidement, notamment lorsque le nombre de données est important.

Définition Stabilité

Définition Tri en place
Un tri est effectué en place lorsque la liste à trier est modifiée jusqu'à devenir triée. Dans le cas contraire, la fonction de tri pourra renvoyer une novelle liste contenent les mêmes éléments, mais triés.

Définition Tri comparatif
Un tri est dit comparatif lorsqu'il s'appuie uniqument sur la comparaison deux à deux des éléménts de la liste et pas sur la valeur ce ces éléments.

2 Tris

Définition Tri par insertion

À partir d'une sous-liste triée, le tri par insertion consiste à parcourir les éléments non triés et de les insérer successivement dans la sous-liste déjà triée.

```
def insere(t, j):
    k, a = j, t[j]
    while k > 0 and a < t[k-1]:
        t[k] = t[k-1]
        k = k-1
    t[k] = a

def insertionSort(t):
    for j in range(1, len(t)):
        insere(t, j)
```

Définition Tri rapide

Soit une liste L non triée. Soit p un terme appelé pivot. Le tri rapide consiste à répartir les éléments strictement inférieur au pivot avant ce dernier et les termes plus grand après le pivot (segemntation). Le pivot est à ce stade trié correctement par rapport aux autres valeurs de la liste. Ce principe est alors appliqué récursivement aux deus-sous listes séparées par le pivot.

```
def segmente(t, i, j):
    p = t[j-1] # On prend comme pivot le dernier élément de la sous liste.
    a = i
    for b in range(i, j-1):
        if t[b] < p:
            t[a], t[b] = t[b], t[a]
            a += 1
    t[a], t[j-1] = t[j-1], t[a] # On positionne le pivot "à sa place".
    return a # On retourne l'index du pivot. Le tableau a été modifié en place.
```

```
def quickSort(t, i, j):
    if i + 1 < j:
        a = segmente(t, i, j)
        quickSort(t, i, a)
        quickSort(t, a + 1, j)
```

```
# Instruction pour trier une liste
quickSort(t, 0, len(t))
```

Définition Tri fusion
Il s'agit d'un tri s'appuyant sur la stratégie diviser pour régner.
L'algorithme est le suivant :
- on divise la liste en deux listes de tailles quasi-identiques ;
- on trie récursivement ces deux listes ;
- on fusionne les deux listes triées.

```
def placer(L :list, p :int, x) :
    """Place un élément x àsa place dans une liste L triée àpartir de l'indice p
    Entrée :
        L : une liste, p : un entier, x : un élément
    Sorties :
        La liste est modifiée mais n'est pas renvoyée. k la valeur de l'indice de la liste où l'
            élément a été placé"""
    k = p
    while ( k < len(L) and x > L[k]) :
        k = k+1
    L.insert(k, x)
    return k

def fusion(a:list, b:list) :
    """Fusionne les deux listes
    Entrée : deux listes a et b triées.
    Sortie : La liste b modifiée"""
    p = 0
    for x in a :
        p = placer(b, p, x)+1
    return b

def tri_fusion(t : list) :
    """Trie la liste t
    Entrée : une liste.
    Sortie : la liste est modifiée."""
    if len(t) < 2 :
        return (t)
    else :
        m = len(t) // 2
        return (fusion (tri_fusion(t[:m]) , tri_fusion(t[m:]) ))
```

Proposition **Complexité des algorithmes** On note $T(n)$ le nombre de comaraisons nécessaire pour trier une liste de longueur n. On montre que dans le pire des cas, les complexités sont les suivantes :
- tri par insertion : $T_{\text{Max}}(n) = \mathcal{O}\left(n^2\right)$;
- tru rapide : $T_{\text{Max}}(n) = \mathcal{O}\left(n^2\right)$;
- tru partition-fusion : $T_{\text{Max}}(n) = \mathcal{O}\left(n \log n\right)$.

3 Acticité préparatoire

Pour réaliser l'activité associée à ce cours, suivre le lien suivant :
- Sujet : `https://bit.ly/3iAc5do`
- Corrige : `https://bit.ly/3eFuHrt`

4 QCM

Question 1 *On dispose d'une liste de triplets : t = [(1,12,250),(1,12,251), (2,12,250),(2,13,250),(2,11,250),(2,12,249)]. On trie cette liste par ordre croissant des valeurs du second élément des triplets. En cas d'égalité, on trie par ordre croissant du troisième champ. Si les champs 2 et 3 sont égaux, on trie par ordre croissant du premier champ. Après ce tri, quel est le contenu de la liste ?*

1. `[(1,12,249),(1,12,250),(1,12,251),(2,11,250),(2,12,250),(2,13,250)]`.
2. `[(2,11,250),(1,12,249),(1,12,250),(2,12,250),(1,12,251),(2,13,250)]`.
3. `[(2,11,250),(1,12,249),(1,12,250),(1,12,251),(2,12,250),(2,13,250)]`.
4. `[(1,12,249),(2,11,250),(1,12,250),(2,12,250),(2,13,250),(1,12,251)]`.

Question 2 *Quelle valeur retourne la fonction "mystere" suivante ?*

```
def mystere(liste):
    valeur_de_retour = True
    indice = 0
    while indice < len(liste) - 1 :
        if liste[indice] > liste[indice + 1]:
            valeur_de_retour = False
        indice = indice + 1
    return valeur_de_retour
```

1. Une valeur booléenne indiquant si la liste liste passée en paramètre est triée.
2. La valeur du plus grand élément de la liste passée en paramètre.
3. La valeur du plus petit élément de la liste passée en paramètre.
4. Une valeur booléenne indiquant si la liste passée en paramètre contient plusieurs fois le même élément.

Question 3 *Combien d'échanges effectue la fonction Python suivante pour trier un tableau de 10 éléments au pire des cas ?*

```
def tri(tab) :
    for i in range (1, len(tab)) :
        for j in range (len(tab) - i) :
            if tab[j] > tab[j+1] :
                tab[j], tab[j+1] = tab[j+1], tab[j]
```

1. 45.
2. 100.
3. 10.
4. 55.

Question 4 *Que vaut l'expression* `f([7, 3, 1, 8, 19, 9, 3, 5], 0)` *?*

```
def f(t,i) :
    im = i
    m = t[i]
    for k in range(i+1, len(t)) :
        if t[k] < m :
            im, m = k, t[k]
    return im
```

1. 1.
2. 2.
3. 3.
4. 4.

Question 5 *Laquelle de ces listes de chaînes de caractères est triée en ordre croissant ?*

1. `['Chat' , 'Cheval' , 'Chien' , 'Cochon']`.
2. `['Cochon' , 'Chat' , 'Cheval' , 'Chien']`.
3. `['Cheval' , 'Chien‘ , 'Chat' , 'Cochon']`.
4. `['Chat' , 'Cochon' , 'Cheval' , 'Chien']`.

Question 6 *Laquelle de ces listes de chaînes de caractères est triée en ordre croissant ?*

1. ['12','142','21','8'].
2. ['8','12','142','21'].
3. ['8','12','21','142'].
4. ['12','21','8','142'].

Question 7 *Quelle est la valeur de la variable* `table` *après exécution du programme Python suivant ?*

```
table = [12, 43, 6, 22, 37]
for i in range(len(table) - 1):
    if table [i] > table [i+1] :
        table [i] ,table [i+1] = table [i+1] ,table [i]
```

1. [12, 6, 22, 37, 43].
2. [6, 12, 22, 37, 43].
3. [43, 12, 22, 37, 6].
4. [43, 37, 22, 12, 6].

Question 8 *Un algorithme cherche la valeur maximale d'une liste non triée de taille n. Combien de temps mettra cet algorithme sur une liste de taille $2n$?*

1. Le même temps que sur la liste de taille n si le maximum est dans la première moitié de la liste.
2. On a ajouté n valeurs, l'algorithme mettra donc n fois plus de temps que sur la liste de taille n.
3. Le temps sera simplement doublé par rapport au temps mis sur la liste de taille n.
4. On ne peut pas savoir, tout dépend de l'endroit où est le maximum.

Question 9 *Quel est le coût en temps dans le pire des cas du tri par insertion ?*

1. $\mathcal{O}(n)$.
2. $\mathcal{O}\left(n^2\right)$.
3. $\mathcal{O}\left(2^n\right)$.
4. $\mathcal{O}(\log n)$.

Question 10 *On souhaite écrire une fonction* `tri_selection(t)`*, qui trie le tableau* `t` *dans l'ordre croissant : parmi les 4 programmes suivants, lequel est correct ?*

```
def tri_selection(t) :
    for i in range (len(t)-1) :
        min = i
        for j in range(i+1,len(t)):
            if t[j] < t[min]:
                min = j
        tmp = t[i]
        t[i] = t[min]
        t[min] = tmp
def tri_selection(t) :
    for i in range (len(t)-1) :
        min = i
        for j in range(i+1,len(t)-1):
            if t[j] < t[min]:
                min = j
        tmp = t[i]
        t[i] = t[min]
        t[min] = tmp
def tri_selection(t) :
    for i in range (len(t)-1) :
        min = i
        for j in range(i+1,len(t)):
            if t[j] < min:
                min = j
        tmp = t[i]
        t[i] = t[min]
        t[min] = tmp
def tri_selection(t) :
    for i in range (len(t)-1) :
        min = i
        for j in range(i+1,len(t)):
            if t[j] < t[min]:
```

```
                    min = j
            tmp = t[i]
            t[min] = t[i]
            t[i] = tmp
```

1. Fonction 1.
2. Fonction 2.
3. Fonction 3.
4. Fonction 4.

Question 11 *De quel type de tri s'agit-il ?*

```
def tri(lst):
    for i in range(1,len(lst)):
        valeur = lst[i]
        j = i
        while j>0 and lst[j-1]>valeur:
            lst[j]=lst[j-1]
            j = j-1
        lst[j]=valeur
```

1. Tri par insertion.
2. Tri fusion.
3. Tri par sélection.
4. Tri à bulles.

Question 12 *De quel type de tri s'agit-il ?*

```
def tri(lst):
    nb = len(lst)
    for i in range(0,nb):
        ind_plus_petit = i
        for j in range(i+1,nb) :
            if lst[j] < lst[ind_plus_petit] :
                ind_plus_petit = j
        if ind_plus_petit is not i :
            temp = lst[i]
            lst[i] = lst[ind_plus_petit]
            lst[ind_plus_petit] = temp
```

1. Tri par insertion.
2. Tri fusion.
3. Tri par sélection.
4. Tri à bulles.

Question 13 *Un algorithme est en complexité quadratique. Codé en python, son exécution pour des données de taille 100 prend 12 millisecondes. Si l'on fournit des données de taille 200 au programme, on peut s'attendre à un temps d'exécution d'environ :*

1. 48 millisecondes.
2. 24 millisecondes.
3. 12 millisecondes.
4. 96 millisecondes.

Question 14 *À quel type tri correspond l'invariant de boucle ci-dessous :*

- tous les éléments d'indices 0 à $i-1$ sont déjà triés,
- tous les éléments d'indices i à n sont de valeurs supérieures à ceux de la partie triée.

1. Tri par insertion.
2. Tri fusion.
3. Tri par sélection.
4. Tri à bulles.

Question 15 *Quel est l'invariant de boucle qui correspond précisément à cet algorithme ?*

On considère un algorithme de tri par sélection, dans lequel la fonction `echanger(tab[i], tab[j])` effectue l'échange des ième et jième valeurs du tableau `tab`.

```
nom: tri_sélection

paramètre: tab, tableau de n entiers, n>=2

Traitement:
pour i allant de 1 àn-1:
    pour j allant de i+1 àn:
        si tab[j] < tab[i]:
            echanger(tab[i], tab[j])
renvoyer tab
```

1. Tous les éléments d'indice supérieur ou égal à i sont triés par ordre croissante.
2. Tous les éléments d'indice compris entre 0 et i sont triés et les éléments d'indice supérieurs ou égal à i leurs sont tous supérieurs.
3. Tous les éléments d'indice supérieur ou égal à i sont non triés.
4. Tous les éléments d'indice compris entre 0 et i sont triés, on ne peut rien dire sur les éléments d'indice supérieur ou égal à i.

Question 16 *Quel est le type de tri qui correspond à cet algorithme?*

```
nom: tri_mystere

paramètre: tab, tableau de n entiers, non trié, non vide

Traitement:
pour i allant de 1 àn-1:
    pour j allant de i+1 àn:
        si tab[j] < tab[i]:
            echanger(tab[i], tab[j])
renvoyer tab
```

1. Tri par insertion.
2. Tri fusion.
3. Tri par sélection.
4. Tri rapide.

Question 17 *Quel est l'invariant de boucle qui correspond précisément à cet algorithme?*

```
nom: tri_insertion

paramètre: tab, tableau de n entiers, n >= 2

Traitement:
pour i allant de 2 àn:
    j = i
    tant que j > 1 et tab[j-1] > tab[j]:
        echanger(tab[j-1], tab[j])
        j = j-1
renvoyer tab
```

1. Tous les éléments d'indice compris entre 0 et i sont triés et les éléments d'indice supérieurs ou égal à i leurs sont tous supérieurs.
2. Tous les éléments d'indice supérieur ou égal à i sont triés par ordre croissant.
3. Tous les éléments d'indice compris entre 0 et i sont triés, on ne peut rien dire sur les éléments d'indice supérieur ou égal à i.
4. Tous les éléments d'indice supérieur ou égal à i sont non triés par ordre croissants.

Question 18 *Parmi les propositions suivantes, quelle est celle qui ne correspond pas à une méthode de tri?*

1. Par sélection.
2. Par insertion.
3. Par rotation.
4. Par fusion.

5 TP