Informatique

1
Recherche séquentielle - Dictionnaire

TD 1-2
Recherche maximum et suivants

Exercice 1: Détermination du maximum d'une liste de nombres

On cherche à déterminer le maximum d'une liste L de nombres. Pour cela on propose deux algorithmes dont nous allons comparer les temps d'exécution.

Premier algorithme

Le principe est le suivant : on compare chaque élément de L avec tous les éléments de L. Si l'élément en cours est plus grand que tous les autres c'est le maximum.
Afin de programmer cet algorithme, on a écrit la fonction suivante, qui comporte un certain nombre d'erreurs à identifier et corriger :

```
def Maximum_naturel(L):
    """Etant donnée une liste de nombres, renvoie son maximum en
utilisant la    méthode naturelle : comparaison de chaque
élément avec tous les autres"""
    N = len(L)
    if N = 0:
        print(la liste est vide)
        return None
    for i in range(N):
        v1 = L(i)
        est_plus_grand = True
        for j in range(N+1):
            v2 = L[j]
            if i < j:
                est_plus_grand = False
        if est_plus_grand:
            maximum = i
    return maximum
```

Question 1: Tenter dans un premier temps de corriger cette fonction sur papier
Ouvrez le fichier suivant (LIEN) et copiez-collez le code dans votre fichier
Question 2: Appliquer les correction et tester la fonction avec une liste vide ainsi qu'avec la liste [4, 8, 5]
Si la fonction ne renvoie pas la valeur attendue pour la seconde liste, corriger celle-ci, si besoin, en déroulant à la main l'évolution des différentes variables lors de l'exécution du code.

Deuxième algorithme

Le principe est le suivant : on utilise une variable afin de stocker le maximum provisoire. En voici les étapes :

- Si la liste est vide, on renvoie None
- On initialise max_prov à la valeur du 1er élément de la liste
- On parcourt les éléments de la liste à l'aide d'une boucle, et…
    - Si l'élément en cours est plus grand que max_prov on met max_prov à jour
    - Sinon, on ne fait rien
- Une fois arrivé au bout de la liste, on renvoie max_prov

Question 3: Ecrire une fonction Maximum_memoire(L) qui renvoie le maximum de L en utilisant ce deuxième algorithme et la tester sur un exemple simple

Applications

On donne les instructions suivantes :

```
import random as rd  # import du module random qui est renommé rd
L = [rd.randint(0,100*N) for k in range(N)]
```

La liste L est alors une liste de N entiers compris entre 0 et 100*N.
Rappel : *randint(a,b)* du module *random* renvoie un entier choisi aléatoirement entre a et b

Question 4: Recopier ces instructions afin de tester vos deux fonctions maximum sur une liste de N = 1000 valeurs et vérifier qu'elles renvoient bien la même chose (on pourra comparer à la fonction max de python)

Nous allons maintenant comparer les temps d'exécution des deux algorithmes.

Question 5: Intuitivement, quel est pour vous l'algorithme qui sera le plus rapide ?

Pour déterminer le temps d'exécution d'une instruction on utilise la fonction *perf_counter()* du module *time*. Cela peut s'utiliser comme suit :

```
# Utilisation de la fonction perf_counter()
import time
t0 = time.perf_counter()
'''instructions''' # Les instructions dont on veut calculer le temps
d'exécution
t1 = time.perf_counter()
temps_execution = t1 - t0
```

Pour vous rendre compte de l'effet de la ligne « `time.perf_counter()` », tentez de l'exécuter deux fois dans la console après avoir importé le module `time`.

Question 6: Tester vos deux fonctions sur une même liste contenant 1000 valeurs aléatoires afin de vérifier votre conjecture
Question 7: Quelle est la complexité de la fonction Maximum_naturel ?
Question 8: Afin de vérifier cette complexité, calculer et afficher à l'aide d'une boucle le temps d'exécution de votre fonction sur des listes contenant successivement 10^3, 10^4, 10^5 et 10^6 éléments

Exercice 2: Variations autour du maximum

Question 1: Ecrire une fonction Indice_max(L) qui, étant donnée une liste de nombres L, renvoie l'indice d'un maximum de L et le maximum, en ne parcourant qu'une seule fois la liste L (complexité linéaire)
Question 2: Dans le cas où votre maximum est présent plusieurs fois dans la liste, quel indice renvoyez-vous ?
Question 3: Ecrire une fonction Positions_max(L) qui, étant donnée une liste de nombres L, renvoie la liste des positions du maximum dans L en utilisant la fonction Indice_max (2 parcours de L)
Question 4: Ecrire une fonction Positions_max_opt(L) qui, étant donnée une liste de nombres L, renvoie la liste des positions du maximum dans L sans utiliser la fonction Indice_max (1 parcours de L)

Exercice 3: Second maximum

On cherche à écrire une fonction qui renvoie le second maximum d'une liste L. On fait l'hypothèse que la liste possède au moins deux éléments différents.
Attention : aucune des fonctions de cette exercice ne doit modifier la liste L.

Idée

Idée : Utiliser une fonction qui renvoie le minimum et le maximum de L

Question 1: **Ecrire une fonction min_et_max(L) de complexité linéaire qui renvoie le minimum et le maximum d'une liste de nombres L**

Question 2: Utiliser la fonction min_et_max pour écrire une fonction Second_max(L) qui renvoie le second maximum de L

Amélioration

Idée : Améliorer l'algorithme précédent de façon à ne parcourir la liste qu'une seule fois.
On suppose que les deux premiers éléments de L sont distincts.

Question 3: **Ecrire une fonction Second_max_2(L) qui renvoie le second maximum de L en ne parcourant L qu'une seule fois**
Remarque : si vous en avez le temps, réfléchissez à une amélioration afin que la fonction marche même si les deux premiers éléments de L ne sont pas distincts

On suppose toujours que les deux premiers éléments de L sont distincts.
Question 4: Ecrire une fonction Indice_second_max(L) qui renvoie une des positions du second maximum en ne parcourant L qu'une seule fois

Exercice 4: Maximum et suivants

Soit une liste de flottants L de N termes.
Dans cet exercice, on pourra utiliser la fonction « `max` » de Python, aucun import nécessaire.

Question 1: Créer une fonction n_max(L,n) renvoyant dans l'ordre, les n premiers maximum de L
Question 2: Préciser la complexité en temps et ce que renvoie l'appel de n_max(L,N)