Algorithme et Programmation 3 - TP Semaine 6

Diviser pour Régner

20/11/2023

1 Recherche dans une liste

Dans cette première partie, nous nous intéressons au problème de recherche dans une liste.

1. Programmer une fonction `recherche(L, x)` qui étant donné une liste `L` retourne la position de x dans la liste si x appartient à la liste et -1 sinon.
2. Programmer une fonction `isInList(L, x)` qui renvoit `True` si x appartient à la liste et `False` sinon. On peut appeler la fonction précédente.
3. Quelle est la complexité de votre fonction `recherche`?

Nous allons maintenant supposer que la liste qui nous est donné en paramètre est triée dans l'ordre croissant. On peut alors faire une recherche dichotomique dans la liste triée. Supposons que la liste soit de taille n, et que l'on cherche l'élément x dans la liste.

1. Si la liste est de taille $n = 1$, il suffit de vérifier que l'élement dans la liste est x.
2. Sinon, on regarde l'élement au milieu de la liste, en position $\lfloor \frac{n}{2} \rfloor$.
   - Si cet élément est égal à x, alors on peut retourner le position du milieu.
   - Si l'élément du milieu est inférieur à x, alors cela signifie que x ne peut être que dans la partie droite de la liste, c'est à dire dans la deuxième moitié. On peut donc ré-appeler la recherche dichotomique sur cette sous liste de longueur $n/2$.
   - Si l'élément du milieu est supérieur à x, alors cela signifie que x ne peut être que dans la partie gauche de la liste, c'est à dire dans la première moitié. On peut donc ré-appeler la recherche dichotomique sur cette sous liste de longueur $n/2$.
4. Pour comprendre comment ça marche, imaginez les différentes étapes de l'algorithme (à la main sur une feuille) avec `L = [1,4,5,6,8,9,15,16,19,25,31,42,43,55,68,80]` et `x=15`, puis `x=71`.

5. Implémenter la fonction `rechercheDichotomique(L, x)` qui étant donné une liste `L` triée retourne la position de x dans la liste si x appartient à la liste et -1 sinon.

6. Quelle est la complexité de cette fonction?

7. En faisant `import random` et en utilisant la fonction `random.randint(a,b)`, coder une fonction `randomList(n,maxVal)` qui retourne une liste de taille n contenant des entiers entre 0 et `maxVal`.

8. En utilisant la fonction précédente, coder une fonction `randomSortedList(n,maxVal)` qui retourne une liste triée de taille n contenant des entiers entre 0 et `maxVal`.

9. Lire et lancer le code suivant. Que fait-il? Que pouvez vous conclure des résultats?

```
import time

time_naif = 0
time_dicho = 0

for i in range(1000):
    maListe = randomSortedList(20000, 10000)
    monEntier = random.randint(0,10000)
    t1 = time.time()
    recherche(maListe, monEntier)
    t2 = time.time()
    rechercheDichotomique(maListe, monEntier)
    t3 = time.time()

    time_naif += t2-t1
    time_dicho += t3-t2

print("Temps moyen pour algo naïf :",time_naif/1000)
print("Temps moyen pour algo dichotomoque :",time_dicho/1000)
```

Si vous avez fini en avance, reprenez votre TP de la semaine dernière (sans regarder le corrigé) et finissez.