Algorithme de recherche d'éléments

TD-P#4B – Remise à niveau pour l'informatique

1 Introduction

Il s'agit d'étudier ici les algorithmes de recherche d'un élement dans un tableau. Algorithmes de recherche linéaire, dichotomique, trichotomiques. Présenter les codes, résultats, discussions et conclusions dans un rapport LaTeX, Python note book ou autre.

2 Recherche séquentielle

La recherche séquentielle ou recherche linéaire est un algorithme pour trouver une valeur dans une liste. Elle consiste simplement à considérer les éléments de la liste les uns après les autres, jusqu'à ce que l'élément soit trouvé, ou que toutes les cases aient été lues. Elle est aussi appelée recherche par balayage.[1]

3 Recherche par dichotomie

[2]*La recherche dichotomique, ou recherche par dichotomie (binary search), est un algorithme de recherche pour trouver la position d'un élément dans un tableau trié. Le principe est le suivant : comparer l'élément avec la valeur de la case au milieu du tableau ; si les valeurs sont égales, la tâche est accomplie, sinon on recommence dans la moitié du tableau pertinente. L'algorithme est le suivant :*

- *Trouver la position la plus centrale du tableau (si le tableau est vide, sortir).*
- *Comparer la valeur de cette case à l'élément recherché.*
- *Si la valeur est égale à l'élément, alors retourner la position, sinon reprendre la procédure dans la moitié de tableau pertinente.*

[1] Wikipedia
[2] Wikipedia

```
ALGORITHME Dichotomie(T,x):
DONNEES
  T: tableau de n entiers trie
  x: un entier
VARIABLES
  g,d,m: entiers
DEBUT
  g,d←1,n
  TQ g ≤ d FAIRE
    m←⌊(g+d)/2⌋
    SI T[m] < x ALORS
      g←m+1
    SINON SI T[m] > x ALORS
      d←m-1
    SINON
      RENVOYER m
    FSI
  FTQ
  RENVOYER 0
FIN
```

- Arrêt: la suite des valeurs prises par $d - g$ est strictement décroissante
- Validité: $(T[g] \leq x \leq T[d])$ est un invariant

Questions 1.

1. Écrire une fonction `LinearSearch(T,e)` qui retourne l'indice de l'élement e dans le tableau T s'il existe et -1 sinon.
2. Écrire une fonction qui génére un tableau de n valeurs aléatoires entières.
3. Décrire l'instance du pire et du meilleur des cas. En déduire la complexité en nombre de comparaisons avec les éléments du tableau.

4. Écrire une fonction `DichoSearch(T,e)` qui retourne l'indice de l'élement e dans le tableau T s'il existe et -1 sinon par la méthode dichotomique.
5. Décrire l'instance du pire et du meilleur des cas. En déduire la complexité en nombre de comparaisons avec les éléments du tableau.
6. Écrire une version récursive de la recherche dichotomique.

4 Recherche par saut

On considère un tableau trié dans l'ordre croissant de taille n. La stratégie pour trouver un élément e dans le tableau, à k fixé :

- on teste les éléments d'indice $1, 1+k, 1+2k, 1+3k, \ldots$ pour déterminer le sous-tableau de k éléments qui contient l'élément cherché
- on fait une recherche séquentielle dans le tableau trouvé précédemment.

Par exemple si on cherche 6 avec $k = 3$ dans le tableau `T=[1,2,3,4,5,6,7,8,9]`, on teste les valeurs $T[1] < 6, T[1+3] < 6, T[1+6] > 6$, on fait une recherche séquentielle dans le sous-tableau d'indices 5 à 6 ($T[4]$ a déjà été testé), on teste $T[5] < 6$ puis $T[6] = 6$.

Questions 2.

1. Écrire un algorithme qui fait la recherche par saut d'un élément e dans un tableau T de taille n.
2. Décrire le meilleur des cas et estimer sa complexité en nombre de comparaisons aux éléments du tableau.
3. Décrire le pire des cas et estimer sa compléxité en nombre de comparaions.
4. Déterminer la valeur optimale de k dans la cas général.
5. Implanter les fonctions correspondantes à chacune des questions précédentes et faire des tests sur des tableaux de n valeurs aléatoires.
6. Vérifier que le nombre de comparaisons est en $O(\sqrt{n})$

5 Rapport

Présenter les codes, résultats, discussions et conclusions dans un rapport LaTeX, Python note book ou autre.