TD/TP 6 : Dichotomie et récurrences

CPES ≪ Sciences des données, arts et cultures ≫ : Introduction à l'algorithmique

Antoine GAUQUIER antoine.gauquier@ens.fr
Pierre SENELLART pierre@senellart.com

12 décembre 2023

Le but de ce TD/TP est de manipuler le théorème maître à travers différents exemples, et d'implémenter la recherche dichotomique.

Exercice 1: Applications du théorème maître

Dans cet exercice, il vous est demandé de noter la notation asymptotique $\Theta()$ associée à chacune des expressions de récurrence présentées ci-dessous. Pour chaque cas, il vous est demandé d'exprimer les paramètres du théorème maître, puis, à partir de ceux-ci, justifier auquel des trois cas chaque expression appartient, et finalement donner la notation asymptotique.

Question 1. $T(n) = T\left(\frac{n}{2}\right) + 1$

Question 2. $T(n) = 2T\left(\frac{n}{2}\right) + n^3$

Question 3. $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$

Question 4. $T(n) = 4T\left(\frac{n}{2}\right) + n \log n$

Question 5. $T(n) = 4T\left(\frac{3n}{4}\right) + 5$

Exercice 2: Recherche dichotomique efficace

Dans cet exercice, vous allez implémenter l'algorithme de recherche dichotomique qui vous a été présenté en cours, à savoir, l'algorithme qui vérifie l'existence d'un élément au sein d'un tableau trié, et qui retourne sa position s'il le trouve (ou retourne une valeur par défaut s'il n'est pas dans le tableau). Vous allez implémenter deux versions de cet algorithme : une version efficace, où à chaque nouvel appel, on va chercher les éléments du tableau qui nous intéressent avec deux indices `gauche` et `droite`, et une version naïve, où à chaque appel, on copie le sous-tableau entre `gauche` et `droite` pour l'appel qui suit. On considère dans cet exercice que les tableaux triés contiennent des nombres à virgule flottante.

Question 1. Implémenter en Python la fonction `recherche_dichotomique_efficace` prenant en paramètres une liste Python triée par ordre croissant `liste_triee`, un nombre à virgule flottante `x` à chercher dans la liste, un entier `gauche` représentant l'indice à partir duquel l'on va traiter la liste `liste_triee` pour l'appel en cours, et un entier `droite` représentant le dernier indice de la liste `liste_triee` que l'on considère pour l'itération en cours.

Question 2. Implémenter en Python la fonction `recherche_dichotomique_copie`, prenant pour paramètres une liste Python `sous_liste_triee` et le nombre à virgule flottante `x` à chercher dans la liste.

Question 3. Vérifier que les deux fonctions implémentées sont correctes pour `x1 = 4.12` et pour `x2 = 2.7` dans la liste initiale `[-2.12, -1.1, -2/3, 0, 0.2, 1.25, 3.14, 4.12, 9.87]`.

Question 4. Mesurer empiriquement les temps mis par chacune des fonctions pour faire des recherches dans des tableaux triés de taille croissante. On souhaite que le temps mesuré reflète une situation au cas moyen. Qu'observez-vous ?

Question 5. On souhaite maintenant confirmer les observations empiriques par le calcul des complexités asymptotiques en $\Theta()$ des deux algorithmes implémentés. On connaît déjà celle de la fonction `recherche_dichotomique_efficace`, puisque sa formule de récurrence correspond à la formule de la question 1 de l'exercice 1. Afin de traiter le cas de l'algorithme implémenté par la fonction `recherche_dichotomique_copie`, déterminer sa formule de récurrence.

Question 6. A l'aide du théorème maître, donner la complexité asymptotique en $\Theta()$ de l'algorithme de recherche dichotomique avec recopie du tableau, en justifiant le résultat obtenu.

Question 7. Comparer les complexités asymptotiques des deux algorithmes. Est-ce en adéquation avec les résultats observés empiriquement ?