Résolution d'examens

Programmation avancée

18/12/2015

Remarque sur le projet

4. Programmation dynamique :
   La recherche de Q^* peut s'effectuer par programmation dynamique en s'appuyant sur la fonction de coût $M(i,q)$ (avec $0 \leq i \leq P-1$ et $q \in \{-R,\ldots,0,\ldots,R\}$), définie de la manière suivante :

$$M(i,q) = \max_{q_0,\ldots,q_{i-1}} \sum_{j=0}^{i-2} S(j,q_j,q_{j+1}) + S(i-1,q_{i-1},q).$$

   $M(i,q)$ est donc le score d'alignement maximum que l'on peut obtenir en décalant la ligne i de q positions et optimisant les décalages $q_0,\ldots,q_{i-1}$ des lignes précédentes.
   (a) Donnez une formulation récursive de la fonction de coût $M(i,q)$ en précisant bien le(s) cas de base.

Structure des examens

- QCM (théorie)
- Tri (théorie) / pseudo-code (application)
- Récurrence (application)
- Structures de données (théorie et/ou application)
- Résolution de problèmes (application)

QCM

Pour chacune des affirmations suivantes, déterminez si elle est vraie ou fausse et justifiez **brièvement** votre choix :

(a) $7^{\log_2 N}$ est $O(N^3 + N^2)$.

(b) Pour un ensemble de valeurs données, il existe un unique tas max.

(b) Pour toutes fonctions $f(n)$, $g(n)$ et $h(n)$, si $f(n) \in O(g(n))$ et $f(n) \in \Omega(h(n))$, alors $g(n) + h(n) \in \Omega(f(n))$.

(d) Il est impossible de construire un arbre binaire de recherche à partir d'un tableau quelconque contenant N clés en $\Theta(N)$ opérations.

(e) Si le facteur de charge d'une table de hachage est plus petit que 1, alors il n'y a pas de collisions.

Question 3 : Analyse de complexité

Soit la fonction suivante :

```
ALGO(n)
1  if n == 1
2        return 0
3  else
4        j = 0
5        i = 1
6        while i < n
7              j = j + 1
8              i = 2i
9        return 2 * (ALGO(n/2) + ALGO(n/2) + ALGO(n/2)) + 2^j
```

En supposant que cette fonction prend toujours une puissance de 2 comme argument :

(a) Donnez une forme analytique équivalente à cette fonction.

(b) Donnez une borne asymptotique sur sa complexité en temps (*Suggestion :* Exprimez cette complexité sous la forme d'une récurrence et résolvez la au moyen du *Master theorem*)

Master theorem

Théorème : Soit la récurrence suivante :

$$\begin{cases} T(n) = c & \text{si } n < d \\ T(n) = aT(\frac{n}{b}) + f(n) & \text{si } n \geq d \end{cases}$$

où $d \geq 1$ est une constantes entière, $a > 0$, $c > 0$ et $b > 1$ sont des constantes réelles, et $f(n)$ est une fonction positive pour $n \geq d$.

1. Si $f(n) \in O(n^{\log_b a - \epsilon})$ pour un $\epsilon > 0$, alors $T(n) \in \Theta(n^{\log_b a})$
2. Si $f(n) \in \Theta(n^{\log_b a})$, alors $T(n) \in \Theta(n^{\log_b a} \log n)$.
3. Si $f(n) \in \Omega(n^{\log_b a + \epsilon})$ pour un $\epsilon > 0$ et si $af(\frac{n}{b}) \leq \delta f(n)$ pour un $\delta < 1$, alors $T(n) \in \Theta(f(n))$.

(Introduction to algorithms, Cormen et al.)

NB : les bornes ne dépendent pas de c et d. $\frac{n}{b}$ peut être interprété soit comme $\lfloor \frac{n}{b} \rfloor$, soit comme $\lceil \frac{n}{b} \rceil$.

Question 4 : Arbre binaire de recherche

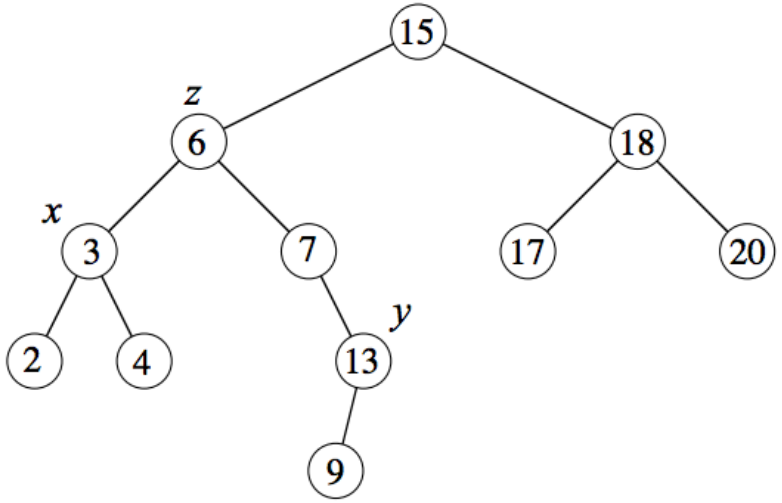
(a) Qu'est-ce qu'un arbre binaire de recherche ?

(b) Le parcours préfixe d'un arbre binaire de recherche donne la séquence suivante :

$$5, 1, 2, 11, 8, 7, 13, 12.$$

Dessinez un arbre correspondant à ce parcours. Cet arbre est-il unique ?

(c) Ecrivez une fonction GETLCA(T, x, y) renvoyant l'ancêtre commun le plus proche des deux nœuds x et y dans un arbre binaire de recherche T. L'ancêtre commun le plus proche de deux nœuds x et y est l'ancêtre z de x et y qui est le plus profond dans l'arbre. Pour cette définition, on considérera un nœud comme un ancêtre de lui-même. Par exemple, pour l'arbre ci-dessous, les deux appels GETLCA(T, x, y) et GETLCA(T, z, y) doivent renvoyer le nœud z.
(suggestion : tirez profit de la propriété d'arbre binaire de recherche)



(d) Analysez la complexité de votre fonction dans le pire et dans le meilleur cas en supposant que l'arbre contient N clés.

Résolution de problèmes

Question 5

Soit un tableau $A[1\,..\,n]$ de n entiers, chacun pris dans l'intervalle $[1, k]$. On cherche à déterminer s'il existe un sous-ensemble des entiers qui somme exactement à $S/2$, où S est la somme de tous les entiers de la liste.

(a) Ecrivez un algorithme efficace pour résoudre ce problème en utilisant la programmation dynamique.
Suggestion : définissez une fonction $f(i, y)$ valant 1 s'il existe un sous-ensemble des i premiers entiers qui somme exactement à y, 0 sinon. Il s'agit alors de calculer efficacement $f(n, S/2)$.

(b) Analysez la complexité de cet algorithme en fonction de n et de k.

Résolution de problèmes

Question 5

On cherche à écrire une fonction LONGEST-CST-SUBARRAY qui renvoie les bornes d'une plus longue suite de valeurs constantes dans un tableau $A[1\,..\,n]$. Si $(i, j) =$ LONGEST-CST-SUBARRAY(A), $A[i\,..\,j]$ est une plus longue suite de valeurs constantes de A $(A[i] = A[i+1] = \ldots = A[j])$.

(a) Expliquez le principe d'une approche force-brute pour résoudre ce problème et donnez sa complexité.

(b) Ecrivez une solution de type diviser-pour-régner et calculez sa complexité.

(c) Ecrivez une solution par programmation dynamique et calculez sa complexité.

Conclusion

- Boite à outils
  - Sommation & récurrence
  - Complexité asymptotique
- Structures simples
- Structures complexes
  - Arbre (binaire de recherche)
  - Tas/File à priorité
  - Table de hachage
- Résolution de problèmes