M1 informatique 2021-2022

Partiel d'algorithmique - sujet a

Mardi 9 novembre 2021 14h–16h / Aucun document autorisé

Nom :

Prénom :

Numéro étudiant :

Groupe :

Exercice 1. Application du Master theorem - 7 points

On considère l'algorithme Algo ci-dessous qui prend en argument un tableau T et deux indices g et d définissant une zone de T. L'objectif ici est de trouver la complexité de Algo en utilisant le Master Theorem (rappelé en annexe). On exprimera cette complexité en fonction de la taille n de la zone délimitée par g et d (inclus), c'est-à-dire de $n = d - g + 1$.

L'algorithme `Algo` dépend de la fonction $\texttt{F}(T, g, d)$ non décrite ici. La complexité de la fonction `F` est aussi évaluée en fonction de la taille $n = d - g + 1$.

Donner la complexité de l'algorithme `Algo` dans les deux cas ci-dessous :

1. Si $\texttt{F}(T, g, d)$ est de complexité $2 \cdot n$.
2. Si $\texttt{F}(T, g, d)$ est de complexité $n^{\log_2 3} + 1$.

NB : Dans les deux cas, on expliquera comment on applique le master Theorem : donner la valeur de a, celle de b, celle de c si besoin, indiquer le cas (parmi les trois possibles) *etc.* Bref, justifiez vos choix !

Procédure `Algo` (T, g, d)
`//`T `: un tableau de taille` k `avec` $0 \leq g, d \leq k - 1$.
begin
  si $g == d + 1$ **alors**
    $T[g] = T[g] + T[d]$
  si $d > g + 1$ **alors**
    `Algo` $(T, g, g + \frac{d-g+1}{2} - 1)$
    `Algo` $(T, g + \frac{d-g+1}{2}, d)$
    `Algo` $(T, g + \frac{d-g+1}{4} + 1, g + 3 \cdot \frac{d-g+1}{4})$
    `F` (T, g, d)

Exercice 2. Programmation dynamique - 9 points

On considère un graphe orienté valué $G = (S, A, w)$ où l'ensemble des sommets est l'ensemble $\{1, \ldots, n\}$ avec n un entier fixé, les arcs $(i, j) \in A$ vérifient $i < j$ (NB : le graphe est acyclique) et la fonction $w : A \to \mathbf{R}$ associe un poids à chaque arc du graphe.

Le poids d'un chemin $i_1 \cdot i_2 \cdots i_k$ (où $(i_j, i_{j+1}) \in A$ pour $j = 1 \ldots, k-1$) est défini par $\displaystyle\sum_{j=1\ldots k-1} w(i_j, i_{j+1})$.

L'objectif est de calculer le poids maximal d'un chemin arrivant au sommet n (peu importe son point de départ).

1. Considérons le graphe de la figure ci-dessous. Donner un chemin (et son poids !) de poids maximal arrivant en 8.

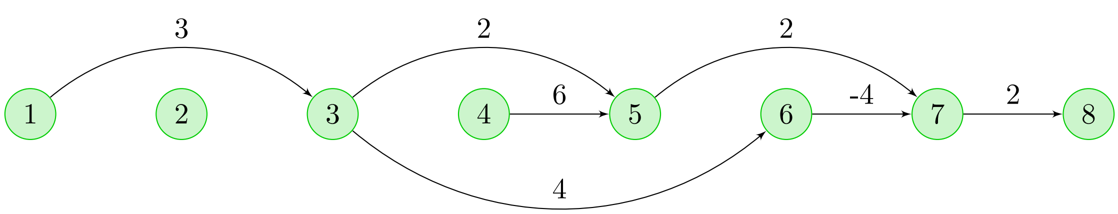

Figure 1 – Graphe G_1.

2 On souhaite calculer une table $D[i]$ avec $1 \leq i \leq n$ telle que $D[i]$ corresponde au poids maximal d'un chemin arrivant en i. Définir $D[1]$ et donner la récurrence pour $D[i+1]$ en fonction de $D[j]$ avec $j < i+1$.

3 Déduire de la question précédente, un algorithme qui étant donné $G = (S, A, w)$ défini comme précédemment, calcule le poids maximal d'un chemin arrivant au sommet n. Quelle est la complexité de votre algorithme ? Comment doit-on représenter G pour avoir une bonne complexité.

Exercice 3. Divers - 4 points

A partir des algorithmes vus en cours (qu'il est inutile de présenter), proposer deux algorithmes efficaces qui, étant donnés un tableau T de n valeurs distinctes et un entier k avec $1 \leq k \leq n$, renvoie **l'ensemble** des k plus petites valeurs de T. Donner leurs complexités.

Annexe

On rappelle que le Master theorem permet d'évaluer les fonctions de la forme $t(n) = a \cdot t(\frac{n}{b}) + f(n)$ qui apparaissent dans les algorithmes diviser-pour-régner, et où a et b sont des rationnels tels que $a \geq 1$ et $b > 1$, et où $f(n)$ est une fonction positive. On distingue trois cas :

1. Si $f(n) = O(n^{\log_b(a)-\varepsilon})$ pour $\varepsilon > 0$, on a : $t(n) = \Theta(n^{\log_b a})$.
2. Si $f(n) = \Theta(n^{\log_b a})$, on a : $t(n) = \Theta(n^{\log_b a} \cdot \log n)$.
3. Si $f(n) = \Omega(n^{\log_b(a)+\varepsilon})$ pour $\varepsilon > 0$ et si il existe $c < 1$ tel que $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$ pour n assez grand, alors $t(n) = \Theta(f(n))$.

NB : $\frac{n}{b}$ peut aussi désigner $\lceil \frac{n}{b} \rceil$ ou $\lfloor \frac{n}{b} \rfloor$.