Examen d'algorithmique (session 2)

Lundi 18 juin 2012 8h30/11h30 – Aucun document autorisé

Mode d'emploi : Le barème est donné à titre indicatif.
La qualité de la rédaction des algorithmes et des preuves sera très fortement prise en compte pour la note.
On peut toujours supposer une question résolue et passer à la suite.

Problème : k-ème plus petit (11 points)

Cet exercice porte sur la recherche du k-ème plus petit élément d'un tableau de n entiers. Dans la suite on pourra supposer que les entiers considérés sont distincts.

Algorithme classique. D'abord, on souhaite modifier l'algorithme du Quicksort pour résoudre ce problème. En utilisant l'algorithme de pivotage `indicepivot(T,bg,bd)` vu en cours (inutile de le redonner !), écrire un algorithme `Select(T,k,bg,bd)` qui retourne le k-ème plus petit élément du tableau `T[bg,...,bd]`. On suppose que $\texttt{bg} \leqslant \texttt{bd}$ et $\texttt{k} \in \{1, \ldots, \texttt{bd} - \texttt{bg} + 1\}$.

Rappel : `indicepivot(T,bg,bd)` choisit un pivot p, place les valeurs inférieures à p au début et celle supérieures à la fin du tableau et retourne l'indice de p après cette réorganisation.

Quelle est la complexité de `Select(T,k,bg,bd)` ? (NB : la complexité du pivotage est linéaire).

Arbres min. Maintenant, nous allons utiliser des *arbres Min*. Ce sont des arbres binaires composés de deux sortes de noeuds : les feuilles auxquelles sont associées des valeurs (ici des entiers), et les noeuds internes (appelés les noeuds "min") qui ont deux fils. On suppose de plus que ces arbres sont *parfaits*, c'est-à-dire que tous les niveaux de l'arbre sont remplis sauf éventuellement le dernier où toutes les feuilles sont rangées à gauche (comme dans les tas). Les feuilles se situent donc sur un ou deux niveaux. On peut voir que le nombre de feuilles fixe la structure de l'arbre.

Un arbre Min peut se voir comme un arbre de calcul : les feuilles fournissent des données en entrée, et la valeur associée à un noeud "min" est le minimum des valeurs associées à ses deux fils. La racine contient donc le minimum des valeurs stockées dans les feuilles.

La figure 1 donne un exemple d'arbre Min. On représente les feuilles par des carrés et les noeuds "min" par des ronds. Dans chaque noeud, on note la **valeur** associée au noeud et à coté du noeud, on note son **numéro**.

Dans la suite, étant donné un arbre Min, on note n_f le nombre de feuilles et n_{min} le nombre de noeuds "min".

1. Montrer que l'on a $n_{min} = n_f - 1$. Quelle est la condition sur n_f pour que les feuilles se situent sur un seul niveau de l'arbre ?

   Montrer que la hauteur d'un arbre Min est inférieure ou égale à $\lceil \log(n_f) \rceil$ (on suppose que $n_f \geqslant 1$).

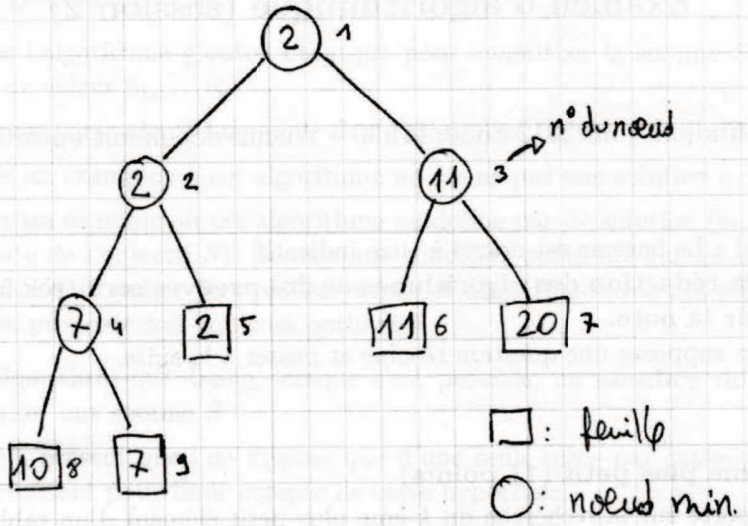

FIGURE 1 – Exemple d'arbre Min

Pour représenter un arbre Min, on utilise un tableau d'entiers $T[-]$ de taille $2n_f - 1$ et commençant à l'indice 1 : la valeur associée au noeud i est stockée en $T[i]$. Cela permet d'obtenir facilement le père $(i/2)$, le fils gauche $(2i)$ et le fils droit $(2i+1)$ du noeud i (NB : on utilise la même idée que pour les tas). L'entier stocké sur un noeud sera soit sa valeur initiale si c'est une feuille, soit le min des valeurs des fils.

2. Donner un algorithme (efficace !) `CalculValArbre` qui, étant donné un arbre Min $T[1 \dots 2n_f - 1]$ où seules les feuilles (c.-à-d. les noeuds $n_f, n_f+1, \dots, 2n_f-1$) ont été initialisées à leur valeur, calcule la valeur de tous les noeuds "min". Donner sa complexité.
   Appliquer votre algorithme à l'exemple de la figure 2.
   Donner le tableau T à la fin du calcul.

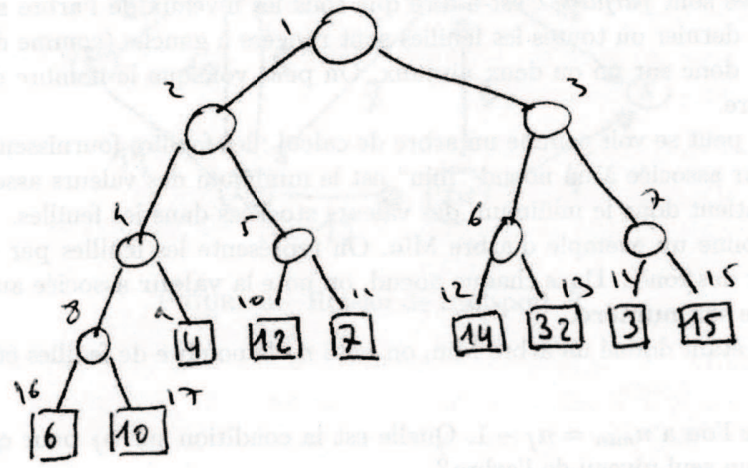

FIGURE 2 – Exemple pour les questions 2 et 4

3. Donner un algorithme (efficace !) `MaJVal` qui, étant donnés un arbre Min $T[1 \dots 2n_f - 1]$,

une valeur v et un numéro de feuille f change la valeur de la feuille f avec v et met à jour les noeuds "min" de T. Quelle est sa complexité ?
Appliquer votre algorithme sur l'arbre obtenu à la question précédente en mettant à jour le noeud 12 avec la valeur 1.

4. On souhaite à présent reprendre les deux algorithmes ci-dessus de manière à stocker non pas les valeurs dans les noeuds "min" mais le **numéro** de la feuille correspondant au résultat de l'opération. Ainsi en $T[1]$ on aura le numéro de la feuille où se trouve le minimum.
Réécrire ainsi les deux algorithmes précédents qu'on appellera `CalculArbre`(T) et `MaJ`(T, v, f). Pour cela, on pourra soit garder le même modèle que précédemment ($T[i]$ contient soit une valeur si i est une feuille, soit l'indice de la feuille contenant le résultat si i est un noeud "min"), soit le modifier en supposant que les valeurs des feuilles sont données par un tableau `F[-]` et que pour les feuilles i on a $T[i] = i$ (on suppose alors que `F` est un tableau d'indices $n_f, \ldots, 2n_f - 1$).
Appliquer votre algorithme à l'exemple de la figure 2. Donner le tableau T et (si vous l'utilisez) le tableau F.
5. Etant donné un arbre Min obtenu avec `CalculArbre`(T), trouver le deuxième plus petit élément en utilisant `MaJ`.
6. Etendre l'idée précédente pour obtenir un algorithme qui trouve le k-ème plus petit élément (avec $k \in \{1, \ldots, n\}$). Donner sa complexité.
7. On considère à présent l'algorithme suivant où V est un tableau d'entiers de taille n et k est un entier vérifiant : $1 < k \leqslant n$.

> `Algo`$(V[1 \ldots n], k)$:
> 1 : Appliquer `CalculArbre` sur un arbre avec les $n-k+2$ feuilles $V[1], \ldots, V[n-k+2]$.
> 2 : Pour $i = 1$ à $k-2$ faire :
> 3 : Appliquer `MaJ`$(T, V[n-k+2+i], T[1])$
> 4 : Renvoyer le deuxième plus petit élément de l'arbre T.

Remarque : On suppose que l'instruction 4 est basée sur l'algorithme de la question 5.

(a) Appliquer `Algo` au tableau $V = [12, 3, 5, 6, 7, 22, 35, 8, 18, 2, 56, 89, 4, 9]$ et $k = 5$.
(b) Que fait cet algorithme ? Expliquer le (montrer qu'il est correct).
(c) Quelle est sa complexité ?

8. Comparer les algorithmes des questions 6 et 7.
9. Comparer l'algorithme `Algo` de la question 7 avec les deux autres vus en cours (le premier était une simple variante de quicksort, et le second utilisait comme pivot le médian des médians des "paquets de 5 éléments"...).

Exercice 1 : Algorithmes glouton et programmation dynamique (6 points)

On dispose de n ($n \geqslant 1$) sortes de pièces de valeurs $v_1, \ldots, v_n$ (avec $v_i \in \mathbb{N}$ et $v_i \geqslant 1$ pour $i = 1, \ldots, n$). Sauf indication contraire, on suppose que l'on dispose d'un nombre illimité de pièces pour chaque catégorie.

Dans les questions ci-dessous, on cherche des algorithmes (efficaces !) pour constituer une somme S avec un nombre minimal de pièces. Pour chaque question, on demande d'expliquer l'algorithme proposé et de donner sa complexité.

1. Algorithme glouton :
   (a) Donner l'algorithme glouton classique pour constituer la somme S en fonction des pièces de valeur $v_1, \ldots, v_n$.
   (b) Appliquer votre algorithme pour $S = 126$ et $v_1 = 10$, $v_2 = 4$, $v_3 = 50$, $v_4 = 2$.
   (c) Donner un exemple où cet algorithme ne donne pas une solution optimale.
   (d) Donner un exemple où cet algorithme ne donne pas de solution (la somme finale est différente de l'objectif S). Donner une condition suffisante sur les v_i pour que toute somme $S \in \mathbb{N}$ puisse être réalisée avec cet algorithme avec ces pièces (sans que la solution proposée soit toujours optimale).
2. Ecrire un algorithme qui donne, lorsque c'est possible, un **nombre minimal** de pièces pour constituer une somme S.
3. On suppose à présent que l'on dispose que d'une seule pièce par catégorie. Modifier l'algorithme précédent pour tenir compte de cette hypothèse.
4. On suppose à présent que l'on dispose de n_i pièces de valeur v_i. Comment utiliser les algorithmes précédents pour résoudre le problème ?

Exercice 2 : Flots (3 points)

On considère le réseau de transport R de la figure 3. Appliquer l'algorithme de Ford-Fulkerson pour trouver un flot maximal. On donnera les flots intermédiaires et les graphes des augmentations obtenus par l'algorithme.

En déduire une coupe de capacité minimale du réseau R.

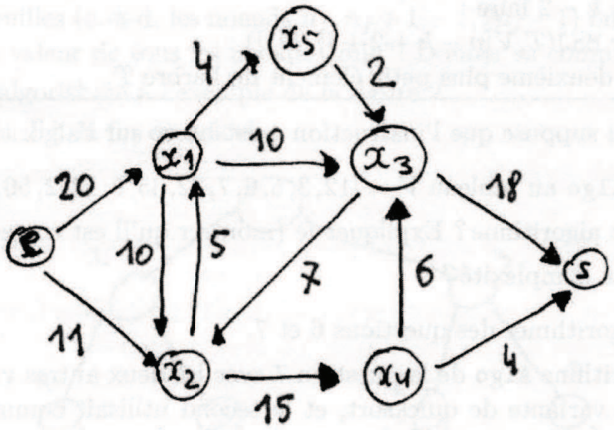

FIGURE 3 – Réseau de transport R