Examen d'algorithmique

Vendredi 6 janvier 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 (10 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.

Pour cela, 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**.

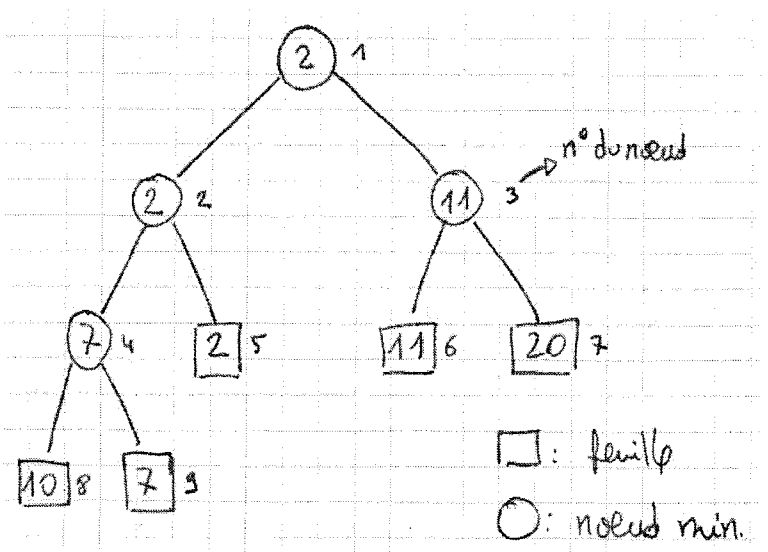

FIGURE 1 – Exemple d'arbre Min

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 $n_f \geqslant 1$).

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 $T[1 \ldots 2n_f - 1]$ où seules les feuilles (c.-à-d. les noeuds $n_f, n_f + 1, \ldots, 2n_f - 1$) 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.

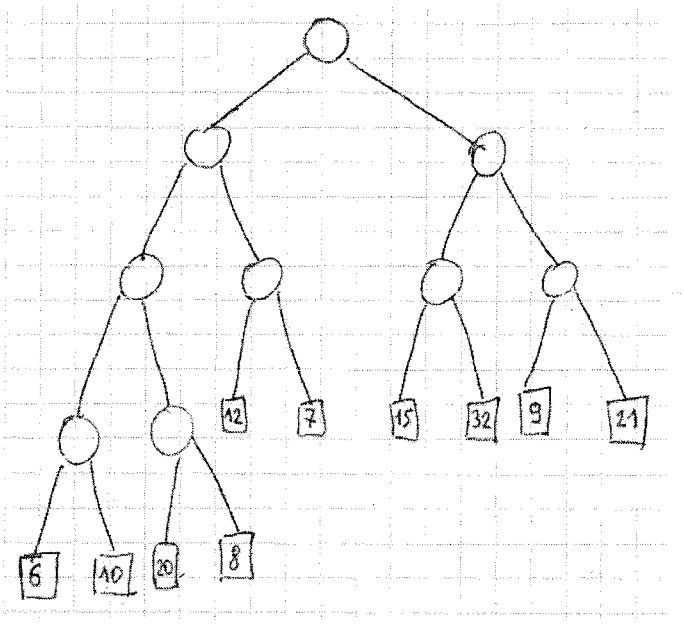

FIGURE 2 – Exemple pour les question 2 et 4

3. Donner un algorithme (efficace !) `MaJVal` qui, étant donnés un arbre Min $T[1 \ldots 2n_f - 1]$, 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 18 avec la valeur 5.
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 $T[i] = i$.

Appliquer votre algorithme à l'exemple de la figure 2.

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...n], k) :
1 : Appliquer CalculArbre sur un arbre avec les n−k+2 feuilles V[1], ..., 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 : Remplir un camion (5 points)

On dispose de n ($n \geqslant 1$) objets ayant des poids $p_1, p_2, \ldots, p_n$ et des valeur $v_1, v_2, \ldots, v_n$. On suppose que les p_i et v_i sont des entiers.

Dans les questions ci-dessous, on cherche des algorithmes (efficaces !) pour charger un camion dont la capacité maximale est donnée par un poids K en cherchant à maximiser la valeur des objets transportés. Pour chaque question, on demande d'expliquer l'algorithme proposé et de donner sa complexité.

1. On suppose que $v_i = 1$ pour tout $i = 1, \ldots, n$.
2. On suppose que $v_i = p_i$ pour tout $i = 1, \ldots, n$.
3. Cas général : sans contrainte sur les v_i.

Exercice 2 : Flots (5 points)

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

   En déduire une coupe de capacité minimale du réseau R.
2. Appliquer l'algorithme de Ford-Fulkerson sur le même réseau R que la question précédente mais en choisissant d'autres chemins dans les graphes des augmentations de manières à obtenir le résultat avec un nombre d'itérations différent.

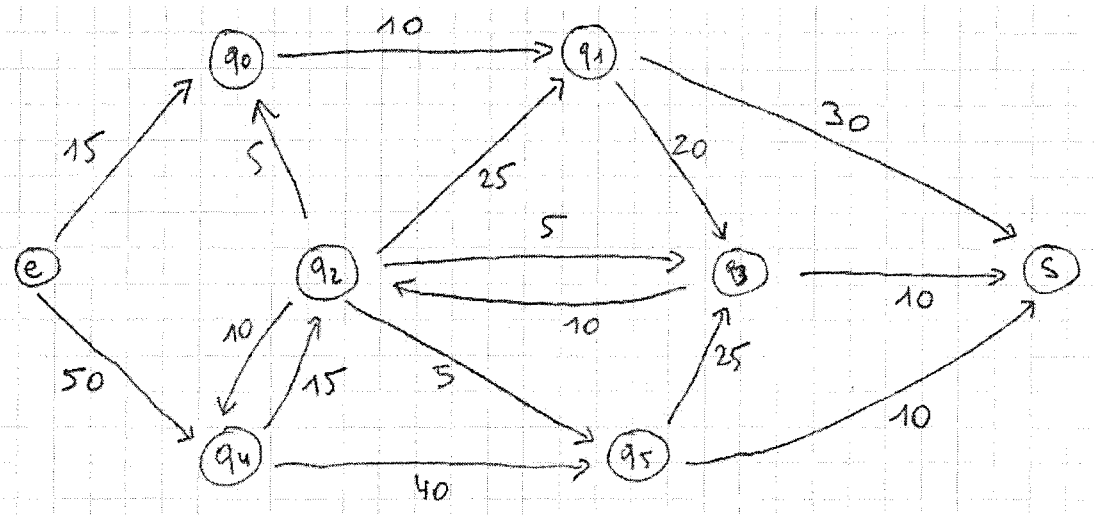

FIGURE 3 – Réseau de transport R des questions 1 et 2 (exercice 2)

3. Un pays dispose d'un ensemble de centrales électriques $\mathcal{C} = \{C_1, \ldots C_m\}$ de différentes tailles : la centrale C_i produit une quantité d'électricité q_i pour $i = 1, \ldots, m$.
   Par ailleurs, il y a un ensemble de villes $\mathcal{V} = \{V_1, \ldots, V_n\}$ qui ont chacune des besoins en électricité : la ville V_j a besoin de la quantité b_j pour $j = 1, \ldots, n$. On suppose que les b_j et les q_i sont des entiers.
   Toutes les villes ne sont pas connectées à toutes les centrales : la ville V_j est connectée à un sous-ensemble $R_j \subseteq \{C_1, \ldots, C_m\}$.
   On s'intéresse maintenant au problème suivant : **trouver une distribution d'électricité maximale (globalement) pour l'ensemble des villes du pays.**

   (a) Réduire ce problème à un problème de flot maximal sur un réseau de transport.

   (b) Appliquer le problème précédent sur l'exemple suivant :
   - $\mathcal{C}$ contient 6 centrales (on note les capacités de production entre parenthèses) : $C_1(120)$, $C_2(320)$, $C_3(170)$, $C_4(200)$, $C_5(230)$, et $C_6(140)$.
   - $\mathcal{V}$ contient les villes suivantes (on note les besoins b_j entre parenthèses) :Paris (300), Lyon (200), Rennes (120), Nantes (130), Marseille(220), Dijon (70), Vitry (130), Montpellier (100), Toulouse (130), et Lille (150).
   - Paris peut se fournir auprès de toutes les centrales exceptées C_2 et C_5. Lyon peut se fournir auprès de C_3, C_4, C_5, Rennes auprès de C_2 et C_6, Nantes auprès de C_2 et C_5, Marseille auprès de C_3 et C_4, Dijon auprès de C_1, C_3 et C_4, Vitry auprès de C_2 et C_4, Montpellier auprès de C_4 et C_5, Toulouse auprès de C_5 et Lille auprès de C_6 et C_1.

   (c) On associe maintenant à chaque ville un couple (m_j, b_j) avec $m_j \leqslant b_j$ et on cherche une solution (c-à-d. une distribution maximale globalement pour l'ensemble des villes) où l'électricité affectée à la ville j est comprise entre m_j et b_j (c.-à-d. m_j est le minimum d'électricité à fournir à V_j). Comment peut-on intégrer cette contrainte dans la méthode précédente ?