

## Examen d'algorithmique (session 2)

Date : 13 juin 2012 8h30 – 10h30 – Aucun document autorisé

Mode d'emploi : L'examen est donc écrit et oral. Les preuves sera très fortement prise en compte. On peut soulever une question et passer à la suite.

Problème : Le *k*-ième plus petit (11 points)

Cet exercice provient de la recherche de *k*-ième plus petit élément d'un tableau de  $n$  entiers. Dans la suite, on suppose que les entiers considérés sont distincts.

**Algorithme** : D'abord, on rappelle l'algorithme du Quicksort pour résoudre ce problème. On définit l'indice  $\text{pivot}(T, \text{bg}, \text{bd})$  vu en cours (inutile de le redéfinir). On définit l'indice  $\text{select}(T, \text{bg}, \text{bd}, k)$  qui retourne le  $k$ -ième plus petit élément du tableau  $T[\text{bg}.. \text{bd}]$ . On suppose que  $\text{bg} \leq \text{bd}$  et  $k \in \{1, \dots, \text{bd} - \text{bg} + 1\}$ .

**Rappel** :  $\text{select}(T, \text{bg}, \text{bd}, k)$  choisit un pivot  $p$ , place les valeurs inférieures à  $p$  au début du tableau et retourne l'indice de  $p$  après cette réorganisation.

Quelle est la complexité de  $\text{select}(T, \text{bg}, \text{bd}, k)$  ? (NB : la complexité du pivotage est linéaire).

**Arbres min** : On peut aussi résoudre ce problème à l'aide des arbres *Min*. Ce sont des arbres binaires composés de noeuds : les noeuds internes (appelés "min") qui ont deux fils. On suppose de plus que tous les niveaux de l'arbre sont remplis sauf éventuellement le dernier. Les feuilles sont donc rangées à gauche (comme dans les tas). Les feuilles ne sont pas forcément deux à deux. On peut voir que le nombre de feuilles fixe la structure de l'arbre.

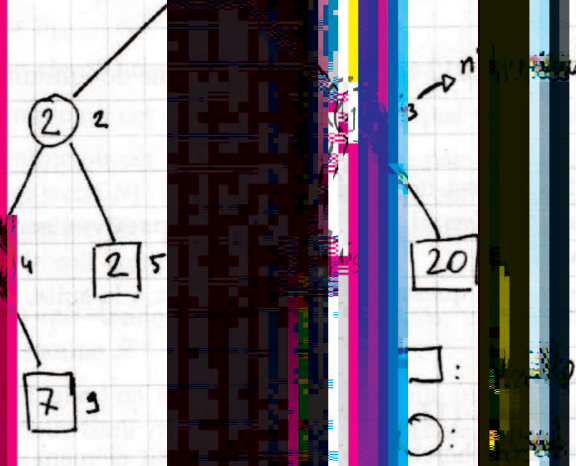
Un arbre *Min* est un arbre binaire tel que : les feuilles fournissent des données en entrée, et la valeur d'un noeud est le minimum des valeurs associées à ses deux fils. La racine est donc le minimum des valeurs stockées dans les feuilles.

La figure ci-dessous représente un arbre *Min*. On représente les feuilles par des carrés et les noeuds "min" par des cercles. Dans chaque carré, on note la valeur associée au noeud et à côté du noeud, on note son indice.

Dans la suite, on suppose que l'arbre *Min* est complet. On note  $n_f$  le nombre de feuilles et  $n_{\min}$  le nombre de noeuds "min".

Montrer que  $n_{\min} = n_f - 1$ . On suppose que les feuilles se situent à une profondeur  $p$  de la racine.

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

[illegible]

Pour repérer un arbre Min, on associe à chaque nœud d'entree  $i$  de  $T$  une valeur  $val[i]$  de taille  $2n_f - 1$  et commençant par 0. On associe à la valeur associée à un nœud  $i$  la valeur associée à son fils gauche  $2i$  et à son fils droit  $2i + 1$ . Cela permet d'obtenir facilement la valeur associée à un nœud  $i$  à partir de son père  $\lfloor i/2 \rfloor$  (NB : on utilise la notation  $\lfloor x \rfloor$  pour désigner l'entier inférieur à  $x$ ). L'entier  $val[0]$  est la valeur associée à la racine. On utilisera souvent la valeur initiale si c'est une feuille, soit 1 si  $i$  est pair et 0 si  $i$  est impair. Les valeurs des fils de  $i$  sont donc  $2i$  et  $2i + 1$ .

2. Donner à  $l$  une valeur (efficace!) et à  $h$  la valeur 1, étape 1. Construire un arbre Min  $T[1 \dots 2n_f - 1]$  où  $l$  est la racine et les feuilles sont les (c.-à-d. les  $2n_f - 1$  feuilles)  $1, \dots, 2n_f - 1$ . Les feuilles ont été initialisées à leur valeur. Donner à  $h$  la valeur de tous les noeuds. Donner à  $l$  la valeur de tous les noeuds. Appliquez l'algorithme à l'étape 2. Donner à  $l$  la valeur de tous les noeuds. Donner à  $h$  la valeur de tous les noeuds. Donner à  $l$  la valeur de tous les noeuds. Donner à  $h$  la valeur de tous les noeuds.

FIGURE 2 – *How do you feel about the question?*

3. Donn $\bar{a}$  il  $k$ -esimo (efficace)  $\leq 2n_f$  e  $k$  non  $\leq$  il numero Min  $T[1 \dots 2n_f - 1]$ ,



une valeur  $v$  à un numéro de feuille  $f$  change la valeur de la feuille  $f$  avec  $v$  et met à jour "min" de  $T$ . Quelle est sa complexité ?

Appliquez l'algorithme sur l'arbre donné à la question précédente en mettant à jour avec la valeur 1.

On se propose de reprendre les deux algorithmes ci-dessus de manière à stocker non pas les noeuds "min" mais le numéro de la feuille correspondant au résultat. Ainsi en  $T[1]$  on aura le numéro de la feuille où se trouve le minimum.

Écrivez les deux algorithmes précédents qu'on appellera `CalculArbre( $T$ )` et `MaJ( $T, v, f$ )`.

Pour ce faire, pourra soit garder le même indice que précédemment ( $T[i]$  contient soit une valeur : une feuille, soit l'indice de la feuille contenant le résultat si  $i$  est un noeud "min") soit modifier en supposant que les valeurs des feuilles sont données par un tableau  $F$  que pour les feuilles  $i$  on a  $F[i] = i$  (on suppose alors que  $F$  est un tableau d'indices de  $1$  à  $2n_f - 1$ ).

Appliquez l'algorithme à l'exemple de la figure 2. Donner le tableau  $T$  et (si vous utilisez un tableau  $F$ ) le tableau  $F$ .

Étant donné un arbre `Min` obtenu avec `CalculArbre( $T$ )`, trouver le deuxième plus petit élément en utilisant `MaJ`.

Prendre l'algorithme précédente pour obtenir un algorithme qui trouve le  $k$ -ème plus petit élément (avec  $k \in \{1, \dots, n\}$ ). Donner sa complexité.

On considère l'algorithme suivant où  $V$  est un tableau d'entiers de taille  $n$  et  $k$  est un entier satisfaisant :  $1 < k \leq n$ .

**Algo( $V, k$ ) :**

- 1 : Appliquer `CalculArbre` sur un arbre à  $n - k + 2$  feuilles  $V[1], \dots, V[n - k + 2]$ .
- 2 : Pour  $i$  de  $k - 2$  à  $0$  faire :
- 3 :     Appeler `MaJ( $T, V[n - k + 2 + i], T[1]$ )`.
- 4 : Retourner 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 l'Algo au tableau  $V = [12, 3, 5, 6, 7, 22, 35, 8, 18, 2, 56, 89, 4, 9]$  et  $k = 5$ .
- (b) Quel est cet algorithme ? Expliquer et prouver qu'il est correct.
- (c) Quelle est sa complexité ?

**Comparaison d'algorithmes des questions 5 et 7 :**

On compare l'algorithme Algo de la question 7 avec les deux autres vus en cours (le premier était une variante de quicksort, et le second utilisait comme pivot le médian des "paquets de 5 éléments"...).

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

On dispose de  $(n \geq 1)$  sortes de pièces de valeurs  $v_1, \dots, v_n$  (avec  $v_i \in \mathbb{N}$  et  $v_i \geq 1$  pour  $i = 1, \dots, n$ ). En indication contrainte, 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 et de donner sa complexité.

Algorithme 1 :

Déterminer l'algorithme glouton qui construit la somme  $S$  en fonction des pièces de valeur  $v_1, \dots, v_n$ .

Appliquer l'algorithme précédent avec  $v_1 = 10, v_2 = 4, v_3 = 50, v_4 = 2$ .

Déterminer un exemple où cet algorithme ne donne pas une solution optimale.

Déterminer un exemple où cet algorithme ne donne pas de solution (la somme finale est différente de l'objectif  $S$ ). Indiquer une condition suffisante sur les  $v_i$  pour que toute somme puisse être réalisée avec ces pièces (sans que la somme proposée soit toujours réalisable).

Écrire un algorithme qui donne, si possible, un nombre minimal de pièces pour constituer une somme  $S$ .

Supposons maintenant que l'on dispose d'une seule pièce par catégorie. Modifier l'algorithme précédent pour tenir compte de cette hypothèse.

Supposons maintenant que l'on dispose de pièces de valeur  $v_i$ . Comment utiliser les algorithmes précédents pour résoudre ce problème ?

## Exercice 2 (10 points)

Considérons le problème de transport suivant. Appliquer l'algorithme de Ford-Fulkerson pour trouver une solution optimale. On donne les graphes des augmentations successives de l'algorithme.

Le graphe de capacité initiale est le graphe  $R$ .

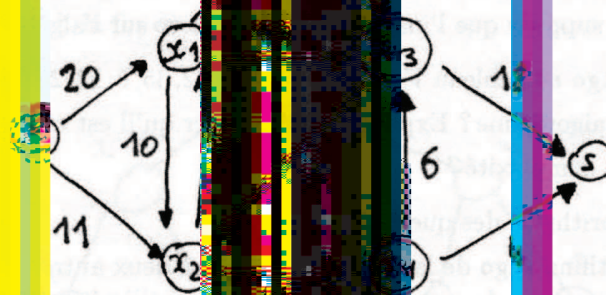


FIGURE 1. Le graphe de transport  $R$ .