Algorithmique Avancée — Examen Réparti 1

AVEC CORRECTION

Les seuls documents autorisés sont les polys de cours, ainsi que la copie personnelle. Le barème donné est indicatif.

Exercice 1 *: Exercices de cours [4 points]*

Question 1 Quel est le nombre de comparaisons nécessaires pour faire la fusion d'une file binomiale de 2966 éléments avec une file binomiale de 371 éléments ?

Question 2

a. Étant donné l'ensemble des arbres 2-3-4 contenant n clés et dont la racine est un nœud de type 4. Montrer qu'il existe des arbres dont le sous arbre dela racine le plus à droite peut contenir $\Theta(\sqrt{n})$ clés.

b. Qu'en est-il si la racine est un nœud de type 2 ?

Solution

1.

$$2966 = 2^1 + 2^2 + 2^4 + 2^7 + 2^8 + 2^9 + 2^{11}.$$

$$371 = 2^0 + 2^1 + 2^4 + 2^5 + 2^6 + 2^8.$$

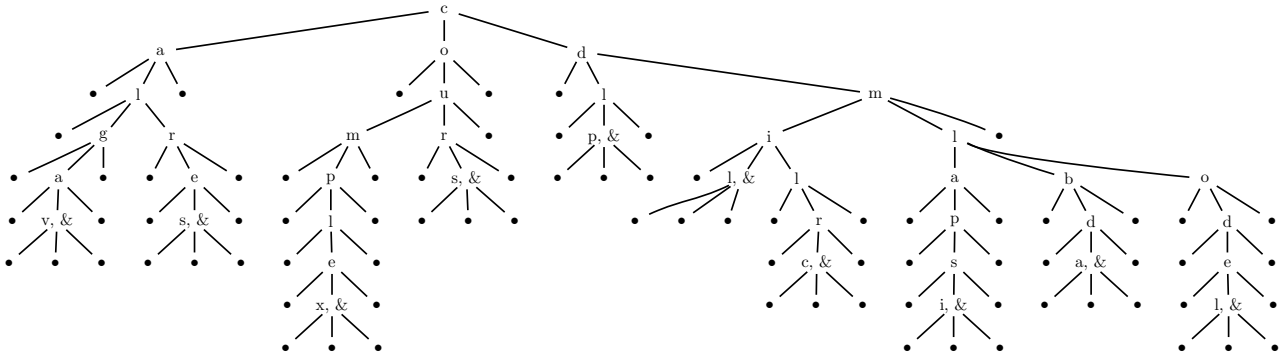
8 comparaisons : 1-1 -> 2-2 -> 4-4 -> 5-5 -> 6-6 -> 7-7 -> 8-8 -> 9-9

2. a. prendre des 4-nœuds partout sauf sur le sous-arbre de droite où l'on prend uniquement des 2-nœuds nombre de clés : $n = 3(racine) + 2^h - 1(sousarbredroite) + 3(4^h - 1)(les3autres) => h = \log_4 n$ donc dans le sous arbre droit le nombre de clés est de l'ordre de $2^{\log_4 n} = n^{\log 2/\log 4} = \sqrt{(n)}$

b. même résultat

Exercice 2 *: Trie hybride [12 points]*

Dans cet exercice, on considère la structure de recherche de type trie hybride. Dans la représentation suivante, on indique les fins de mot avec le caractère spécial & et l'arbre vide avec le caractère •.



Lorsque vous utilisez des primitives sur les arbres, indiquer leur signature et leur rôle.

Question 1 Donner une liste des mots stockés dans l'arbre précédent, en les ordonnant de telle sorte que si l'on construit l'arbre hybride en faisant des insertions successives avec cette liste, on retrouve l'arbre précédent.
Remarque : Plusieurs ordres des mots sont valides, mais une seule liste est demandée.

Question 2 Donner le pseudo-code d'un algorithme renvoyant la longueur du plus long mot stocké dans un trie hybride (on s'interdit de construire la liste des mots puis de chercher le plus long, on travaille directement sur l'arbre).

Question 3 Donner le pseudo-code de l'algorithme générant la liste des mots stockés dans un arbre hybride, dans l'ordre alphabétique.

Remarque : On a à notre disposition une fonction `concat` *qui prend en argument deux chaînes de caractères et qui renvoie leur concaténation.*

Question 4 Étant donnés deux arbres hybrides T_1 et T_2, modifier le second arbre pour y insérer les mots stockés dans T_1. On utilisera les deux questions précédentes.

Question 5 Que se passe-t-il pour le résultat de l'algorithme précédent si l'arbre T_2 est vide ?

Question 6 Étant donnés deux arbres hybrides T_1 et T_2, on souhaite construire un nouvel arbre hybride T_3 qui contient l'ensemble des mots de T_1 et T_2, en utilisant un algorithme plus efficace que l'insertion de la liste des mots du premier arbre dans le second arbre. Proposer le pseudo-code d'un tel algorithme.

Question 7 Dessiner l'arbre de l'exemple dans lequel on a effacé le mot $'complex'$.

Question 8 Dessiner l'arbre de l'exemple dans lequel on a effacé le mot $'dlp'$.

Question 9 Dessiner l'arbre de l'exemple dans lequel on a effacé le mot $'mlbda'$.

Question 10 Étant donnés un arbre hybride T et un mot m de T, on souhaite construire un nouvel arbre hybride R qui est l'arbre T privé du mot m. Les structures de R et T sont identiques en dehors du sous-arbre de T contenant le mot m. Proposer le pseudo-code d'un tel algorithme.

Solution

1. Une liste possible : $['cours','algav','dlp','mlbda','mapsi','ares','complex','il','lrc','model']$
sinon pour les listes possibles,
(1) 'cours' suivi de
(2) entrelacement (shuffle) des listes $['algav','ares']$ avec $['complex']$ et $['dlp','mlbda', shuffle(['il','lrc'], shuffle(['mapsi','model']))]$

2.

3.

4.

5. L'arbre penche vers la droite (aucun fils gauche non vide)

6. 2 versions correctes : fusion et fusion2, mais seule le 2e est très bien.

7.

8.

9.

10. pas encore codé

Exercice 3 *: File de priorité [8 points]*

Dans cet exercice, on considère les files de priorité représentées par des tas (on rappelle qu'un tas est un arbre binaire croissant dont toutes les feuilles sont situées au plus sur deux niveaux, les feuilles du niveau le plus bas étant positionnées le plus à gauche possible).

Question 1 Construire un tas (minimum) T d'étiquettes $1, 3, 5, 7, 9, 11, 15, 17, 19$. Insérer 2 dans le tas précédent en indiquant les étapes intermédiaires. Puis supprimer le minimum en indiquant les étapes intermédiaires.

Question 2 Donner le pseudo-code de l'algorithme de suppression du minimum.

Question 3 Donner un encadrement de la taille d'un tas en fonction de sa hauteur, notée h (la racine du tas est à profondeur 0, et la hauteur est la profondeur maximale d'une feuille).

Question 4 Avec cette structure de données, on peut ajouter un élément ou extraire le minimum en $O(\log n)$ comparaisons dans le cas le pire (où n est la taille du tas). Expliquer brièvement cette assertion.

Question 5 Montrer qu'il est impossible de construire une structure de données S représentant une file de priorité, telle que, pour S, on ait à la fois le coût de la construction d'une file de n éléments en $O(n)$ et le coût de la suppression du minimum en $O(1)$.

Question 6 Il s'agit maintenant de montrer que l'on peut extraire le minimum en coût amorti $O(1)$ comparaisons ou échange de valeurs, tout en gardant l'ajout en coût amorti $O(\log n)$ comparaisons ou échange de valeurs. On considère la fonction Φ, qui à un tas T associe

$$\Phi(T) = 3 \cdot \sum_{x \in T} p(x),$$

où $p(x)$ est la profondeur du nœud x dans l'arbre T.

a. Montrer que Φ est une fonction de potentiel.

b. Montrer que le coût amorti de l'ajout d'un élément est en $O(\log n)$.

c. Montrer que le coût amorti de la suppression du minimum est en $O(1)$.

Solution

1.

2. Pour insérer un nouvel élément dans un tas correct, on insère au seul endroit possible (on complète le dernier niveau, sauf s'il est déjà plein, dans ce cas on crée un nouveau niveau et on met la feuille tout à gauche). Puis, on fait remonter, si nécessaire l'étiquette vers la racine, et faisant des échanges successifs avec son père, afin d'obtenir la contrainte d'être croissant.

3. Un arbre de hauteur h au au moins si $h-1$ premier niveau plein, puis au minimum une feuille tout à gauche, ou 2^h dans le niveau lorsqu'il est plein. Donc sa taille n satisfait :

$$1 + 2 + 2^2 + \cdots + 2^{h-1} + 1 = 2^h \leq n \leq 1 + 2 + 2^2 + \cdots + 2^{h-1} + 2^h = 2^{h+1} - 1$$

4. L'ajout c'est parce qu'on ne compare le nouvel éléments au pire qu'à tous les noeuds qui sont sur son chemin vers la racine.
La suppression du min se fait ainsi : On efface la racine (qui contient le min), on cherche la dernière feuille et l'efface et on met son étiquette à la racine. Puis cette étiquette on la fait coulisser vers le bas, en l'échangeant toujours avec son fils dont l'étiquette est la plus petite. Donc on descend un chemin de longueur au plus $logn$.

5. Si on pouvait construire le tas en $O(n)$ puis on supprimerait n fois le minimum (chaque fois en $O(1)$), on aurait un algo de tri par comparaison en $O(n)$ ce qui est impossible. Je l'ai dit plusieurs fois en cours, donc les étudiants n'ont pas besoin d'en dire plus.

6. a.

b.

c. Le 3 est dû au fait que l'orsqu'on descend l'étiquette de la racine vers les feuilles, il faut comparer ses deux fils, pour trouver le min, puis comparer ce min à l'étiquette en question, et enfin procéder éventuellement à l'échange de valeurs.