Université de Paris

L2 informatique - Année 2021–2022

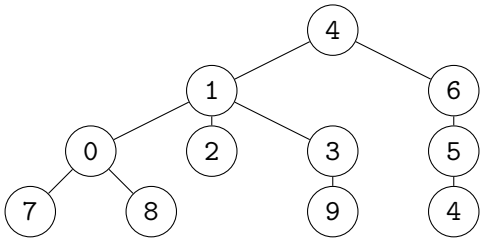
TD d'Éléments d'Algorithmique n° 11

*** L'exercice marqué d'une étoile est à faire à la maison.**

Exercice 1. *Parcours en Largeur.*

1. En utilisant un parcours préfixe ou sufixe, implémenter un algorithme qui prend en argument un arbre et une valeur, et renvoie la profondeur minimale à laquelle cette valeur se trouve (on renverra 0 si la valeur n'est pas dans l'arbre).
2. Quelle est la complexité de cet algorithme si la valeur se trouve à hauteur h de l'arbre ?

Le principe du parcours en largeur est d'explorer tous les noeuds, profondeur par profondeur. Cela revient à regarder les noeuds "ligne par ligne". Par exemple, sur l'arbre suivant, on va explorer les noeuds dans l'ordre : 41602357894.
On considère l'arbre suivant :



Ce parcours ne peut du coup pas être implémenter récursivement, vu qu'on va explorer les enfants de 1, puis celui de 6, avant de revenir aux petits-enfants de 1.

1. Quelle structure de donnée proposez-vous d'utiliser pour se souvenir de quels noeuds on doit traîter après celui en cours ?
2. Implémenter un parcours en largeur qui résout le problème de la Question 1.
3. Si on donne en entrée un arbre binaire complet, quelle est la complexité dans le pire des cas de cet algorithme si la valeur se trouve à hauteur h de l'arbre ?

Dans la suite du TD, à chaque fois qu'on parle d'un *tas*, il s'agit d'un *tas-max*, c'est-à-dire d'un arbre binaire parfait qui satisfait à la *condition du tas* : chaque élément est plus petit que son père. Un tas est équipé des deux opérations suivantes :

- `insert(e)` qui insère l'élément `e` dans l'arbre à la seule position possible, puis qui l'échange avec son père tant qu'il est supérieur à ce dernier ;
- `remove` qui enlève et renvoie la racine de l'arbre, la remplace par la dernière feuille, qu'elle échange itérativement avec son fils le plus grand tant qu'elle est inférieure à ce dernier.

Exercice 2. *Tas.*

1. Dans un tas, où se trouve l'élément maximal ?
2. Dans le cas où tous les éléments sont distincts, où peut se trouver l'élément minimal ?
3. Dessinez le tas qui est produit quand on effectue les opérations suivantes :

`insert`(1), `insert`(5), `insert`(2), `insert`(6), `remove`, `insert`(8), `remove`, `insert`(7), `insert`(3).

Exercice 3. *Implémentation efficace.*
On implémente maintenant un tas par un tableau contenant les éléments du tas énumérés en largeur. On commencera à placer les éléments à partir de la case 1, la case 0 contenant le nombre d'éléments dans le tas (ainsi, on peut supposer que le tableau possède des cases vides qui serviront si on veut ajouter des éléments dans le tas).

1. Quel est le contenu du tableau qui représente le tas obtenu à l'exercice 2 ?
2. Écrivez un algorithme qui prend en argument un tas sous forme d'arbre binaire et sa taille, et renvoie le tableau correspondant.
3. Quel est le numéro de la case contenant le père de la case i ? Les deux fils de la case i ?
4. Écrivez un algorithme qui retourne vrai si et seulement si un tableau d'entiers correspond à une implémentation d'un tas.
5. Écrivez un algorithme qui retourne le plus petit élément d'un tas.
6. Écrivez les algorithmes `insert` et `remove` sur les tableaux.

Exercice 4. *Files de priorité.*
Une *file de priorité* est une structure de données abstraite qui a deux opérations :

- `insert(e, p)` qui insère l'élément e avec la priorité p ;
- `remove` qui retire et retourne un élément de priorité maximale (il peut en général y en avoir plusieurs).

1. On se propose d'implémenter une file de priorité à l'aide d'une liste chaînée simple non-triée. L'opération `insert` insère la paire (e, p) en tête, l'opération `remove` parcourt la liste et retire un élément de priorité maximale. Quelle est la complexité de chacune de ces deux opérations ?
2. On implémente maintenant une file de priorité à l'aide d'une liste triée en ordre décroissant de priorités. L'opération `insert` insère la paire (e, p) de façon à ce que la liste reste triée, l'opération `remove` retire l'élément de tête. Quelle est la complexité de chacune de ces deux opérations ?

3* Implémenter une file de priorité à l'aide d'un tas. Vous pouvez utiliser une version avec des tableaux ou une version avec des arbres binaires. Votre implémentation devra contenir les fonctions `insert` et `remove`. Précisez en commentaire quelle est la complexité de votre implémentation.