Algorithmique et Programmation

Devoir n° 4

École normale supérieure – Département d'informatique
algoL3@di.ens.fr

2014-2015

- Devoir à faire en binôme.
- Règles de collaboration : vous ne devez collaborer qu'avec votre binôme. Vous ne devez pas consulter internet ou d'autres sources pour essayer de trouver des idées de solution. Toutes vos questions doivent être adressées à vos enseignants.
- Devoir à rendre en cours le lundi 20 octobre à 15h30

Le but de ce problème est d'étudier plusieurs structures de données algorithmiques pour implanter un *dictionnaire ordonné*. Il s'agit d'un ensemble S d'éléments appartenant à un univers $\mathcal{U}$ et permettant d'effectuer les opérations suivantes :

- inserer(x) qui ajoute l'élément x à S ;
- est_vide() qui retourne si $S = \emptyset$;
- chercher(x) qui retourne si $x \in S$;
- supprimer(x) qui supprime l'élément x de S ;
- max() qui retourne l'élement maximal de S ;
- min() qui retourne l'élement minimal de S ;
- successeur(x) qui retourne le plus petit élément de S supérieur à x ;
- predecesseur(x) qui retourne le plus grand élément de S inférieur à x.

1. Montrer qu'un arbre binaire équilibré permet d'implanter un dictionnaire ordonné où toutes ces opérations ont un coût $O(\log n)$ où $n = \#S$.
2. Montrer qu'une table de hachage permet également d'implanter un dictionnaire ordonné et donner les complexités de ces opérations.
3. Nous supposons désormais que l'univers $\mathcal{U}$ est constitué uniquement d'entiers strictement inférieurs à une borne U connue (*i.e.* $\mathcal{U} \subseteq \{0, 1, \ldots, U-1\}$). En utilisant un tableau binaire de longueur U (où l'accès à chaque case se fait en temps constant), montrer comment implanter un dictionnaire ordonné où les opérations inserer(x), chercher(x) et supprimer(x) se font en temps constant et les autres opérations demandent un temps $\Theta(U)$.
4. Soit B un entier de $\{1, \ldots, U\}$ (que l'on supposera diviseur de U pour simplifier). Nous ajoutons au tableau binaire de la question précédente, un tableau binaire de longueur B tel que le i-ième bit de ce deuxième tableau vaut 0 si le i-ème bloc de longueur U/B du premier tableau est constitué uniquement de 0 et 1 dans le cas contraire. Montrer que pour un bon choix de B, les opérations est_vide(), max(), min(), successeur(x), predecesseur(x) demandent maintenant un temps $\Theta(\sqrt{U})$. Donner le temps de l'opération supprimer(x) dans cette nouvelle structure de données.
5. En appliquant l'idée de la question précédente récursivement, proposer une structure de données où les opérations est_vide() et chercher(x) demandent un temps $O(\log \log U)$ et les autres opérations demandent un temps $O(\log U)$.
6. Nous ajoutons à la structure précédente, à chaque niveau de la récursivité, un stockage du maximum et du minimum de chaque tableau. Montrer que les opérations est_vide(), max() et min() demandent un temps constant, que l'opération chercher(x) demande toujours un temps $O(\log \log U)$ et que les autres opérations demandent un temps $O(\log U)$.
7. En remarquant que l'opération inserer(x) effectue un appel récursif dans le tableau du niveau suivant seulement si celui-ci est vide et que l'insertion dans une structure vide demande un temps constant, montrer que l'opération inserer(x) demande en réalité un temps $O(\log \log U)$. Montrer de même que l'opération supprimer(x) demande un temps $O(\log \log U)$.
8. Proposer enfin deux algorithmes (analogues) pour effectuer les opérations successeur(x) et predecesseur(x) en temps $O(\log \log U)$.