IFT2015 Hiver 2011 — Devoir 2

Miklós Csűrös

3 février 2011

À remettre avant 20 :15 le 8 février. Remettez un rapport écrit par courriel (à csuros@iro. . .) en format PDF. Le devoir vaut 10 points ; vous pouvez avoir $\sqrt{5} = 2.236\cdots$ points de boni additionnels pour une solution à l'Exercice 2.3.

2.1 File de files (6 points)

Dans cet exercice, on veut implanter une file à «très peut de priorités». Notamment, on veut supporter les opérations usuelles de min-tas : deleteMin et insert, et cela dans un temps de $O(\log k)$ [au pire], où k est le nombre de priorités *distinctes* dans la file. Quand beaucoup d'éléments ont la même priorité (p.e., la priorité est toujours entre 0 et 255), une telle implantation est plus efficace que l'implantation traditionnelle. En particulier, si $k = 2^{o(\log n)}$ (donc k est sous-polynomial en n), alors $O(\log k) = o(\log n)$. Par exemple, $k = O(\log n)$ ou $k = O(1)$ mènent à $O(\log \log n)$ ou $O(1)$ par opération. L'idée de base est de construire un tas binaire dont les éléments sont des queues. ▶ Montrez l'implantation des opérations standardes avec un min-tas binaire stocké dans un tableau $H[1..]$. Chaque $H[i]$ est une file FIFO sur des éléments de même priorité. N'accédez pas aux files FIFO qu'à travers de l'interface standarde : par $H[i]$.enqueue, $H[i]$.dequeue et $H[i]$.isEmpty. (Donc ne montrez pas l'implantation des files FIFO.) L'ensemble de priorités n'est pas connu en avance : il faut créer et abandonner les queues comme nécéssaire.

Vous avez le droit d'utiliser aussi une structure de données D qui implante un dictionnaire pour associer les priorités et leurs queues. D contient des associations entre priorité et file FIFO. L'opération D.search(x) retourne null *si x n'est pas une priorité dans D, ou l'unique file FIFO associée. L'opération D.put(x, Q) associe la queue Q avec la priorité x. Finalement, l'opération D.delete(x) supprime l'association avec x. L'implantation de D assure que toutes les opérations s'exécutent en $O(\log k)$ quand il y a k paires dans D.*

⇐ *instructions ajoutées le 3 février*

▶ Expliquez comment votre implantation assure un temps de calcul $O(\log k)$ (enfiler/défiler sur une file FIFO prend $O(1)$).

2.2 Arbre vs. arbre (4 points)

►Donnez un algorithme MEMEARBRE(x, y) qui évalue si deux arbres binaires sont identiques. Deux nœuds sont identiques entre les deux arbres seulement s'ils sont externes ou internes en même temps. Dans le cas de deux nœuds internes, ils doivent aussi porter le même attribut (ou clé) .info, et avoir des sous-arbres identiques. **Indice :** faire des parcours synchronisés entre les deux arbres.

2.3 Tri local ($\sqrt{5}$ points de boni)

On considère le tri d'un tableau $A[0..n-1]$ d'entiers distincts, en un ordre quelconque. On caractérise le **désordre** du tableau par la différence maximale entre la position initiale et la position triée parmi les éléments. Si $A[i_0] < A[i_1] < \ldots < A[i_{n-1}]$ est l'ordre trié des éléments, alors le niveau de désordre Δ est défini par $\Delta = \max_{j<n} |i_j - j|$. Par exemple, pour le tableau $A[] = (2, 1, 4, 3, 5, 7, 6)$, $\Delta = 1$. Un tableau trié est de désordre $\Delta = 0$. ► Supposons que Δ (ou une borne supérieure) est connu en avance. Montrez comment trier le tableau dans un temps $O(n \log \Delta)$, à l'aide d'un espace de travail de taille $O(\Delta)$ (donc c'est un «tri en place» si $\Delta = o(n)$). Justifier l'efficacité de votre solution.