Université de Mons–Hainaut
FS/1/5684 – Algorithmique
Examen de première session – Partie théorique

Le 23 janvier 2009

Consignes

- Pour cette partie, vous n'avez pas le droit d'utiliser de notes.
- Cette partie de l'examen dure 1 heure 15 minutes.
- Veillez à bien justifier vos réponses. Une réponse mal justifiée, même correcte, ne permet pas d'obtenir le maximum des points.
- Quand vous indiquez une complexité, veillez à bien expliquer ce que sont les paramètres qui apparaissent dans le $\mathcal{O}$. Par exemple, $\mathcal{O}(n^2)$ n'a aucun sens si n n'apparaît pas dans l'algorithme ou dans la définition de la structure qui est traitée…

Question 1 – 6 points

(3 points) Donnez l'algorithme qui permet de rechercher la présence d'une information i dans un vecteur trié d'entiers de taille n, et ce, en $\mathcal{O}(\log(n))$. Justifiez la complexité (il n'est pas nécessaire de donner les détails de la démonstration).

(3 points) Comment peut-on implémenter cet algorithme dans une *liste* triée ? Obtient-on alors la même complexité ? Expliquez.

Correction

L'algorithme demandé est celui de la recherche dichotomique (voir cours, Section 8.2.3). Pour justifier la complexité, il suffisait de rappeler que la recherche dichotomique procède en divisant l'intervalle de recherche par deux à chaque étape. Ainsi, s'il faut initialement effectuer la recherche dans un intervalle de taille n, cet intervalle de recherche ne contient plus que $n/2$ cases à la deuxième étape, $n/4$ cases à la troisième étape,…, $n/(2^k)$ cases à la k^{e}étape. Le pire cas est obtenu quand la valeur recherchée n'est pas dans le vecteur. Pour s'en rendre compte, il faut que la recherche atteigne une étape à laquelle l'intervalle de recherche ne contient plus aucune case, c'est-à-dire quand $n \approx 2^k$. Il faut donc de l'ordre de $k = \log(n)$ étapes de calcul (toutes en temps constant).

Cet algorithme peut naturellement être implémenté dans une liste, puisque la seule chose nécessaire est de pouvoir accéder à un élément arbitraire de la structure, ce qui se fait, dans le cas des listes à l'aide de la fonction **KIemeElement** par exemple (voir cours, Algorithme 13).

Malheureusement, cette opération **KIemeElement** est en $\mathcal{O}(n)$, alors qu'il est possible d'accéder en temps constant à n'importe quelle case d'un vecteur. Comme l'opération d'accès à un élément doit être effectuée au moins une fois par étape de calcul, la complexité totale est maintenant en $\mathcal{O}(n\log(n))$, ce qui est bien moins efficace que $\mathcal{O}(\log(n))$ (remarquons que c'est même moins efficace que la recherche linéaire simple, qui est en $\mathcal{O}(n)$). L'efficacité de la recherche dichotomique dépend donc de manière cruciale de la possibilité d'accéder en $\mathcal{O}(1)$ à n'importe quel élément d'un vecteur.

Question 2 – 4 points

(2 points) Expliquez comment on peut implémenter une file dans un vecteur.
(2 points) Dans ce contexte, donnez les algorithmes qui permettent de réaliser l'insertion et la consultation du début. Commentez brièvement.

Correction

Voir cours, section 5.1.2 (en particulier l'Algorithme 24). Il importait de bien expliquer comment les informations sont stockées de façon circulaire dans le vecteur, en utilisant l'opération *modulo*.
