Journées ALÉA 2021

Exercices sur les bornes inférieures de complexité

Ces exercices sont inspirés des cours de Jeff Erikson, que vous trouverez ici :

`http://jeffe.cs.illinois.edu/teaching/algorithms/`

► Exercice 1 ◄ On dispose de n vis et de n écrous, tous de diamètres différents, tels que chaque vis s'apparie avec exactement un écrou, formant ainsi un boulon (boulon = vis + écrou, appariés). Les diamètres sont très similaires, indistinguables à l'oeil nu, donc la seule opération que l'on peut faire c'est de tester une vis avec un écrou, ce qui peut produire trois résultats : l'écrou est trop petit, c'est le bon écrou ou l'écrou est trop grand.

(a) Montrez qu'il faut $\Omega(n \log n)$ tests dans le pire cas pour reconstituer les n boulons.

(b) On n'a besoin que de former $\lceil n/2 \rceil$ boulons. Montrez qu'il faut quand même $\Omega(n \log n)$ tests dans le pire cas.

(c) (*Passez si vous ne connaissez pas la définition de la complexité d'un algo probabiliste*) Proposez un algorithme probabiliste qui reconstitue tous les boulons avec une complexité $\mathcal{O}(n \log n)$.

Pour la petite histoire, le premier algorithme déterministe en $\mathcal{O}(n \log n)$ n'a été proposé qu'en 1995[1].

► Exercice 2 ◄ Le problème `Fusion` consiste, étant donné deux tableaux S et T triés et de même taille n, à créer un tableau de taille $2n$ qui contient les éléments de S et de T, lui-même trié. Pour le modèle de calcul, on considère uniquement des comparaisons entre des éléments de S et de T, donnés par leurs indices : "est-ce que S[i] est plus petit que T[j] ?".

(a) Proposez un algorithme qui résout le problème `Fusion` en faisant au plus $2n-1$ comparaisons.

On veut établir une borne inférieure au problème `Fusion` en utilisant un argument d'adversaire. L'adversaire commence avec

$$S = [0, 2, \ldots, 2n-2] \text{ et } T = [1, 3, \ldots, 2n-1]$$

Il utilise ce remplissage pour répondre aux questions de l'algorithme. Il peut changer ces valeurs s'il le souhaite tant qu'il ne se contredit pas.

(b) Montrez que si, quand il a terminé, l'algorithme n'a pas comparé S[i] et T[i] alors l'adversaire peut échanger les valeurs de ces deux cases sans se contredire, pour $0 \leq i < n$.

(c) Montrez que c'est également vrai pour S[$i+1$] et T[i], pour $0 \leq i < n-1$.

(d) En déduire une borne inférieure sur le nombre de comparaisons nécessaires pour `Fusion`.

► Exercice 3 ◄ On s'intéresse à tester la connexité d'un graphe non-orienté (sans boucle) avec n sommets étiquetés de 1 à n. Dans notre modèle de calcul, les algorithmes peuvent juste poser des questions de la forme **"Est-ce qu'il y a une arête entre i et j ?"**, pour i et j entre 1 et n.

On veut montrer que tout algorithme qui teste la connexité a besoin d'au moins $\binom{n}{2}$ questions dans le pire cas, il doit reconstituer tout le graphe. Pour cela, on utilise un argument d'adversaire : Alice maintient deux graphes A et B avec le même nombre de sommets. Initialement, A est le graphe complet et B le graphe vide. Quand l'algorithme demande s'il y a une arête $i-j$ pour la première fois, on l'enlève de A si cela ne déconnecte pas A et on répond "non"; sinon, on l'ajoute à B et on répond "oui".

(a) Vérifiez qu'à tout moment A et B sont compatibles avec les réponses données par Alice.

(b) Montrez que $B \subseteq A$ et que si A possède un cycle $\mathcal{C}$, alors B ne contient aucune arête de $\mathcal{C}$.

(c) Montrez que si $A \neq B$ alors B n'est pas connexe. Conclure.

► Exercice 4 ◄ On a une séquence S de n bits. On note $S[i]$ le $i+1$-ème bit de S (on commence donc les indices à 0). On cherche à déterminer si S contient le motif 01, c'est-à-dire s'il existe $0 \leq i < n-1$ tel que $S[i] = 0$ et $S[i+1] = 1$. Dans notre modèle de calcul, on a juste le droit de demander quelle est la valeur de $S[i]$, pour i entre 0 et $n-1$.

(a) On suppose que n est impair. On considère un algorithme qui commence par demander les valeurs de tous les bits en positions impaires dans S : $S[1]$, $S[3]$, $\ldots S[n-2]$. En distinguant différent cas, montrez qu'on peut compléter l'algorithme en une solution qui demande la valeur d'au plus $n-1$ bits.

(b) On suppose que n est pair. Montrez par un argument d'adversaire que tout algorithme qui résout le problème a besoin d'au moins n questions dans le pire cas.

---

[1] Phillip G. Bradford, Matching nuts and bolts optimally, Technical Report MPI-I-95-1-025, Max-Planck-Institut für Informatik, September 1995. Aussi : János Komlós, Yuan Ma, and Endre Szemerédi, Sorting nuts and bolts in $O(n \log n)$ time, SIAM J. Discrete Math 11(3):347–372, 1998. Les deux algos et leurs analyses sont compliqués.