UQAR
Université du Québec à Rimouski
300, allée des Ursulines, C. P. 3300, succ. A
Rimouski (Québec) G5L 3A1, CANADA
www.uqar.ca

Solutions

DÉPARTEMENT DE MATHÉMATIQUES, D'INFORMATIQUE ET DE GÉNIE

Structures de données et algorithmes
Examen Intra — Hiver 2023

SIGLE : INF21307
TITRE : Structures de données et algorithmes
GROUPE : MS
PROFESSEUR : Steven Pigeon
K-212
steven_pigeon@uqar.ca
DURÉE : **3h00**

➜ Nom : ______________________
Code permanent : ______________

Question	Points	Bonus	Obtenus
1	10	0	
2	10	0	
3	10	5	
Total:	30	5	

Modalités : Toutes les notes (données en classe, manuscrites) sont permises. Aucun appareil électronique n'est permis. La durée de l'examen est 3h00. Toute contravention aux directives expose l'étudiant aux mesures disciplinaires prévues au règlement 5, article 15. Répondez à l'encre sur le formulaire (dûment identifié à votre nom).

1. Questions vrai ou faux, à choix multiple et « buffet. » Les questions marquées à choix multiple demandent de ne faire qu'un seul choix, tandis que les questions « buffet » demandent de cocher tout ce qui s'applique. Pour les questions vrai ou faux et à choix multiple, une bonne réponse donne les points, mais une mauvaise donne zéro. Pour les questions « buffet » chaque bon choix donne un point (+1), une erreur enlève un point (−1) mais on arrête à zéro.

   (a) (1 point) (Vrai ou faux) Pour stocker un booléen, il faut au moins un octet.

   ☐ vrai ☑ faux   un bit suffit!

   (b) (1 point) (Vrai ou faux) Dans un nombre en virgule flottante, le nombre de chiffres après le point est variable.

   ☐ vrai ☑ faux   il y a un nombre de bits fixe, donc un nombre de chiffres fixe.

   (c) (1 point) (Vrai ou faux) Les nombres entiers sont habituellement représentés en signe/magnitude.

   ☐ vrai ☑ faux   compliment à 2.

   (d) (1 point) (Vrai ou faux) $3 - n$ est une fonction de complexité valide pour un algorithme.

   ☐ vrai ☑ faux   c'est négatif.

   (e) (1 point) (Vrai ou faux) Si $f(n) = n\sqrt{n}$ et $g(n) = n^2$, alors $g(n) \in O(f(n))$.

   ☐ vrai ☑ faux $n^2 \gg n\sqrt{n} = n^{3/2}$!

   (f) (1 point) (Vrai ou faux) Pour deux fonctions $f(n)$ et $g(n)$, si $f(n) \in O(g(n))$ alors nous avons forcément $g(n) \in O(f(n))$.

   ☐ vrai ☑ faux

   (g) (1 point) (Choix multiple) Le seuil n_0, c'est...

   A. la constante qu'on obtient avec le test de la limite,
   B. un n à partir duquel les deux fonctions se séparent définitivement,
   C. n'importe quel n où les deux fonctions sont déjà séparées,
   D. une valeur où l'une des deux fonctions est minimisée.

(h) (1 point) (Vrai ou faux) La meilleure valeur pour initialiser un booléen, c'est...

■ vrai ■ faux   un ou l'autre ; préférence à faux?

(i) (1 point) (Choix multiple). Triez en « ordre de » les fonctions suivantes $n^{\frac{3}{2}}$, $\frac{n}{\lg n}$, $n \lg n$ et nc^n (avec $c > 1$).

A. $nc^n < n^{\frac{3}{2}} < \frac{n}{\lg n} < n \lg n$,

B. $\frac{n}{\lg n} < n^{\frac{3}{2}} < n \lg n < nc^n$,

C. $\frac{n}{\lg n} < n \lg n < n^{\frac{3}{2}} < nc^n$,

D. $n^{\frac{3}{2}} < \frac{n}{\lg n} < nc^n < n \lg n$,

E. Aucune de ces réponses.

(j) (1 point) (Vrai ou faux) Si on visite un tableau dans l'ordre habituel (rangée par rangée, colonne par colonne), on peut réduire grandement le calcul d'adresse.

☑ vrai ☐ faux   la prochaine case du tableau est toujours la case suivante en mémoire!

2. Questions à développement court. Les questions qui suivent demandent une réponse courte (quelques phrases) mais claire.

   (a) (2 points) Pourquoi est-ce utile d'avoir un algorithme qui répond « non » avec certitude mais aussi parfois « peut-être » ?

   Si cet algorithme est (très) rapide et répond « non » beaucoup plus souvent que « peut-être », on évitera alors souvent d'avoir recours à l'algorithme exact mais plus dispendieux : ça sera plus rapide en moyenne !

   (b) (2 points) D'après vous, si un *bitmap* est très très peu occupé (beaucoup plus de zéros que de uns), à quel moment ça devient plus économique d'utiliser un tableau pour dire quels bits sont à 1 ?

   Si le tableau stocke les positions des bits à 1, ça sera dès que :

   (nombre de bits à 1) × (taille des entiers) ≤ taille du bitmap.

   les entiers qui représentent les positions !

   (c) (2 points) Pensez-vous que le calcul des adresses dans un tableau dont les dimensions sont des puissances de 2 puisse être réalisé plus rapidement que pour un tableau dont les dimensions sont quelconques ?

   - Il est possible que le jeu d'instruction puisse aider avec certaines puissances de 2 (1, 2, 4, 8, 16...)
   - Multiplier par une puissance de deux, c'est un décalage à gauche «, beaucoup moins dispendieux qu'une « vraie » multiplication.

   ... de toute façon : les adresses sont précalculées autant que possible au moment de la compilation

(d) (2 points) Supposons que nous ayons un ordinateur avec un processeur 4 bits. Est-il possible de calculer correctement $5 - 9$ en complément à deux ? Si oui, comment ? Si non, pourquoi ? Est-ce accidentel ?

0	−16	"négatifs" quand même.
1	−15	
2	−14	
3	−13	
4	−12	
5	−11	
6	−10	
7	−9	
8	−8	"négatifs" habituels
9	−7	
10	−6	
11	−5	
12	−4	
13	−3	
14	−2	
15	−1	

Oui, non ; ça dépend de l'interprétation du complément à 2 :

-9 serait $\equiv 7 \pmod{16}$,

et donc $5-9 \equiv 5+7 = 12 \equiv -4 \pmod{16}$.

(e) (2 points) Expliquez pourquoi l'arithmétique en point fixe, malgré son manque de souplesse, peut tout de même être un bon choix ?

$\Rightarrow$ l'arithmétique se fait entièrement en nombres entiers

($+, -, \times, \div$ et «, ») .

3. Questions à développement. Les questions qui suivent demandent un développement un peu plus long des idées ; privilégiez une approche « droit au but. ».

(a) (5 points) Expliquez le compromis entre la vitesse de croissance d'un tableau dynamique et le coût d'insertion. Existe-t-il un facteur de croissance optimal ? Pourquoi ?

Le compromis, c'est balancer temps d'insertion espéré vs quantité de mémoire consommée ; deux choses qui s'optimisent en sens contraire.

Le compromis « optimal » dépend donc des poids qu'on assigne aux deux quantités, et ça dépend des situations!

(b) (5 points) Soient $f(n) = n \log n$ et $g(n) = n\sqrt{n}$. Est-ce que $f(n) \in O(g(n))$? Si oui, donnez c et n_0.

Test limite :

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n \log n}{n\sqrt{n}} = \lim_{n\to\infty} \frac{\log n}{\sqrt{n}} \overset{R.H.}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} \frac{1}{n}\cdot\frac{2\sqrt{n}}{1} = \lim_{n\to\infty} \frac{2}{\sqrt{n}} = 0$$

Puisque $f(n) \ll g(n)$, la constante existe, or, le test dit que la limite est 0, mais zéro est interdit. Donc $c=1$ fera l'affaire. Pour trouver le ~~cons~~ seuil : on trouve n t.q.

$$n \log_b n \leq c\, n\sqrt{n}$$

mais comme $c=1$, c'est

$$n \log_b n \leq n\sqrt{n}$$

ou

$$\log_b n \leq \sqrt{n}$$

et $\sqrt{n}$ est <u>toujours</u> plus grand que $\ln n$, donc $n_0 = 1$

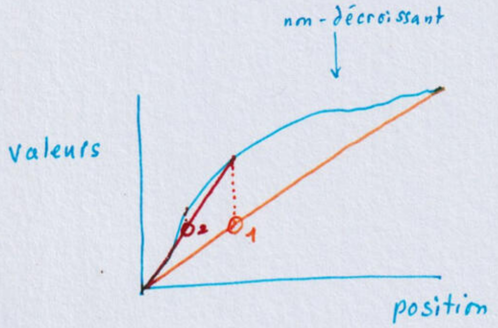
Si $b=2$, $\log_2 n = \sqrt{n}$ pour $n = 4, 16$, et donc $n_0 = 16$.

etc.

(c) (5 points (bonus)) Dans un tableau trié, on va évidemment préférer utiliser la recherche dichotomique à la recherche séquentielle, car on peut déterminer la position (ou l'absence !) d'une valeur en $O(\lg n)$ plutôt qu'en $O(n)$. Pensez-vous qu'on peut faire nettement mieux que $O(\lg n)$ si on sait que les valeurs sont déjà triées ? Comment ?

Oui ! Si on sait — ou on suppose — quelque chose de spécial à propos des valeurs, on peut l'exploiter pour un meilleur estimé de la position de l'élément recherché dans le tableau.

EX (Algorithme de Perl, p. 107). Si on suppose que les valeurs (triées) sont à peu près en progression régulière, on peut utiliser l'interpolation linéaire pour calculer un meilleur estimé de la position de l'item :



① Premier round : on calcule où la valeur serait si les valeurs étaient « linéaires ». Cette position (le ① sur le dessin) sert de premier pivot : ensuite on regarde si la valeur est trouvée, ou si elle devrait être à gauche, ou à droite.

② Deuxième round : on calcule où la valeur serait si les valeurs étaient « linéaires » sur cet intervalle : c'est notre 2ᵉ « guess »...

Sur le dessin on voit qu'après deux itérations déjà on est très près de la valeur... en fait, c'est une procédure de recherche en $O(\log \log n)$!

Indices

$$n^a n^b = n^{a+b}, \qquad \sqrt{n} = n^{\frac{1}{2}}, \qquad (n^a)^b = n^{ab}.$$

$$\frac{\mathrm{d}}{\mathrm{d}n} \log_b n = \frac{1}{n \ln b}, \qquad \frac{\mathrm{d}}{\mathrm{d}n} c^n = c^n \ln c, \qquad \frac{\mathrm{d}}{\mathrm{d}n} \sqrt{n} = +\frac{1}{2\sqrt{n}}.$$

erreur !