IFT436 – Algorithmes et structures de données

Université de Sherbrooke

Devoir 2

Enseignant:	Michael Blondin
Date de remise:	jeudi 26 septembre 2019 à 10h30
À réaliser:	en équipe de deux ou individuellement
Modalités:	à remettre en classe, en copie imprimée ou copie manuscrite lisible
Bonus:	les questions bonus sont indiquées par ★
Pointage:	max. 50 points + 5 points bonus

Question 1.

(a) Considérons un algorithme $\mathcal{A}$ lancé sur une architecture qui exécute 300 000 opérations élémentaires par seconde. Soit $t(n)$ le temps d'exécution dans le pire cas de $\mathcal{A}$. Pour chacune des fonctions suivantes, donnez la plus grande valeur (entière) de n où $\mathcal{A}$ termine (à coup sûr) en une heure ou moins. 4 pts

(i) $t(n) = n$

(ii) $t(n) = n \log_2 n$

(iii) $t(n) = n^2$

(iv) $t(n) = n^3$

(v) $t(n) = 2^n$

(vi) $t(n) = n!$

Il n'est pas nécessaire de justifier vos réponses.

(b) Considérons deux algorithmes $\mathcal{A}$ et $\mathcal{B}$. Soit $f(n)$ le temps d'exécution dans le *pire cas* de $\mathcal{A}$, et soit $g(n)$ le temps d'exécution dans le *meilleur cas* de $\mathcal{B}$. Pour les fonctions suivantes, dites à partir de quelle valeur (entière) de n l'algorithme $\mathcal{A}$ est *toujours plus rapide* que $\mathcal{B}$: 3 pts

— $f(n) = 65536n^2$ et $g(n) = \frac{1}{65536} \cdot 2^n$,

— $f(n) = 1000n \log_2 n$ et $g(n) = n^2$.

Il n'est pas nécessaire de justifier vos réponses.

(c) Montrez que $(n-1)(n+2)(n+3) \in \Theta(n^3)$ sans utiliser les propositions vues en classes (donc en identifiant explicitement les constantes multiplicatives et seuils). 5 pts

(d) Ordonnez les fonctions suivantes selon la notation $\mathcal{O}$ (de la plus faible vers la plus haute complexité): 5 pts

$$2^{3 \cdot \log_2(n)} - 100, \qquad 5n \log_3(8n^2) + 900n, \qquad \sum_{i=1}^{2n} (7i - 3), \qquad \sum_{i=1}^{n} (2^i - i), \qquad \frac{n}{128!} + 10000\sqrt{n}$$

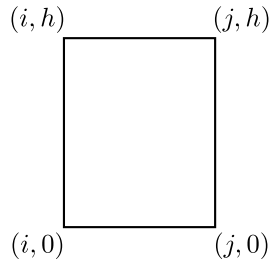
Autrement dit, votre réponse devrait être de la forme « $\mathcal{O}(f_1) \subset \mathcal{O}(f_2) \subset \mathcal{O}(f_3) \subset \mathcal{O}(f_4) \subset \mathcal{O}(f_5)$ ». Justifiez *une seule* inclusion en montrant que $f_i \in \mathcal{O}(f_{i+1})$ et $f_{i+1} \notin \mathcal{O}(f_i)$ pour un i de votre choix.

(e) Dites si 2^n appartient à $\mathcal{O}(2^{2n})$, $\Omega(2^{2n})$ et/ou $\Theta(2^{2n})$. Justifiez votre réponse. 3 pts

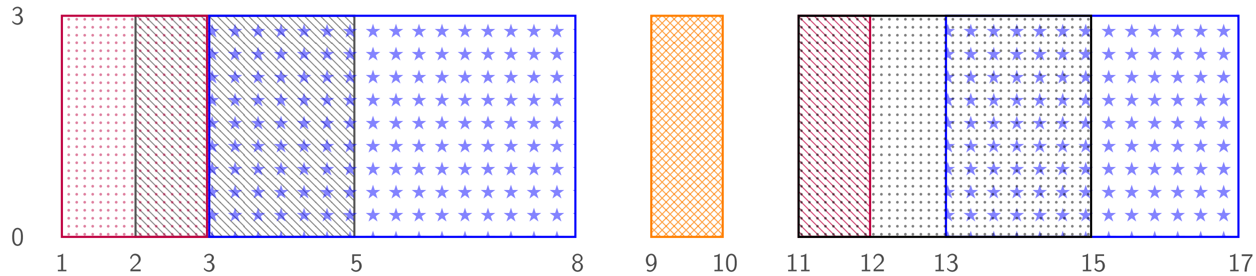
★ Montrez qu'il existe des fonctions $f, g \in \mathcal{F}$ telles que $f \notin O(g)$ et $g \notin O(f)$. ★ 2.5 pts

Question 2.

Soit $h \in \mathbb{N}_{>0}$. Un *bloc* est une paire $(i, j) \in \mathbb{N}^2$, telle que $i < j$, qui décrit le rectangle suivant dans le plan:



Un *paysage* est un ensemble fini de blocs. Par exemple, le paysage $P = [(1, 3), (3, 8), (11, 15), (9, 10), (2, 5), (11, 12), (13, 17)]$ de hauteur $h = 3$ correspond graphiquement à:



Observons que plusieurs blocs d'un paysage peuvent se chevaucher. La surface d'un paysage ne correspond donc pas nécessairement à la somme de la surface de chacun de ses blocs. Par exemple, la surface du paysage ci-dessus est de $3 \cdot ((8 - 1) + (10 - 9) + (17 - 11)) = 42$.

(a) Quelle est la surface du paysage $P = [(1, 3), (16, 17), (5, 9), (11, 15), (4, 8), (10, 14), (10, 12), (2, 3), (4, 5)]$ de hauteur $h = 5$? 3 pts

(b) Donnez un algorithme, sous forme de pseudocode, qui résoud le problème suivant en temps $\mathcal{O}(n \log n)$ dans le pire cas: 7 pts

ENTRÉE: une hauteur $h \in \mathbb{N}_{>0}$ et un paysage $P = [(i_1, j_1), \ldots, (i_n, j_n)]$
SORTIE: la surface du paysage P de hauteur h

Vous pouvez utiliser une instruction **trier** qui trie une séquence de n éléments, selon un ordre de votre choix, en temps $\mathcal{O}(n \log n)$ dans le pire cas. Vous devez analyser le temps d'exécution de votre algorithme.

Question 3.
Pour cette question, nous considérons (comme dans les notes de cours) que les éléments d'une séquence s de n éléments sont numérotés de 1 à n, c.-à-d. $s = [s[1], s[2], \dots, s[n]]$. Considérons l'algorithme de tri suivant:

Entrées : séquence T de $n \in \mathbb{N}_{>0}$ éléments comparables
Résultat : trie T

```
trier(T):
    corriger-inv(j, k):                          // sous-routine avec accès à T
        si T[j] > T[k] alors
            T[j], T[k] ← T[k], T[j]              // inverser le contenu de T[j] et T[k]
            retourner faux
        sinon
            retourner vrai

    m ← n ÷ 2
    faire
        terminé ← vrai
        pour i ← 1, 2, ..., m faire
            terminé ← corriger-inv(2i - 1, 2i) ∧ terminé
        pour i ← 1, 2, ..., m faire
            si 2i < n alors
                terminé ← corriger-inv(2i, 2i + 1) ∧ terminé
    tant que ¬terminé
```

(a) Pour chacune des entrées suivantes, exécutez trier(T) et donnez le contenu de T à la fin de chaque tour de la boucle **faire ... tant que**: 2 pts

— $T = [66, 99, 100, 88, 77, 200]$;

— $T = [80, 20, 30, 40, 50, 60, 10]$;

— $T = [1002, 1000, 1001, 1006, 1004, 1005]$.

(b) Expliquez brièvement, en mots, le fonctionnement de l'algorithme. 2 pts

(c) Montrez que si une séquence s de n éléments n'est *pas* triée, alors il existe $i \in [n]$ tel que $s[i] > s[i+1]$. 3 pts

(d) Expliquez pourquoi l'algorithme termine et est correct. 3 pts

Indice: considérez (c) et les propriétées des inversions vues en classe.

(e) Analysez le temps d'exécution dans le pire cas afin de montrer qu'il appartient à $\mathcal{O}(n^3)$. 3 pts

(f) Argumentez que le temps d'exécution dans le pire cas appartient à $\Omega(n^2)$. 4 pts

Indice: pensez aux ≪ pires entrées ≫ possibles.

(g) Le temps d'exécution de l'algorithme dans le meilleur cas appartient-il à $\mathcal{O}(n)$? Justifiez votre réponse. 3 pts

★ Montrez que le temps d'exécution dans le pire cas appartient à $\Theta(n^2)$. ★ 2.5 pts