IFT436 – Algorithmes et structures de données

Université de Sherbrooke

Devoir 1

Enseignant:	Michael Blondin
Date de remise:	mardi 20 septembre 2022 à 23h59
À réaliser:	en équipe de deux ou individuellement
Modalités:	remettre en ligne sur Turnin
Bonus:	les questions bonus sont indiquées par ★
Pointage:	max. 50 points + 4 points bonus

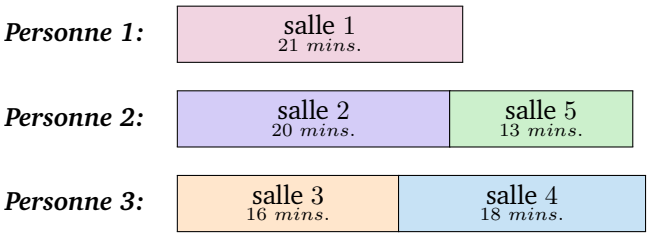
Question 1: rédaction et analyse d'algorithmes

L'escouade étudiante de l'université nettoie les salles de classe chaque matin. En général, il y a $m \in \mathbb{N}_{>0}$ salles à nettoyer par $n \in \mathbb{N}_{>0}$ personnes. Le temps requis pour laver la $i^{\text{ème}}$ salle est de $t[i] \in \mathbb{N}_{>0}$ minutes. Ainsi, une personne qui nettoierait l'entiereté du campus mettrait $t[1] + t[2] + \cdots + t[m]$ minutes pour y arriver. Comme l'escouade est normalement constituée de plusieurs personnes, jusqu'à n salles peuvent être traitées simultanément. Par soucis de distanciation, il n'est pas permis d'assigner plus d'une personne à une même salle.

Dans le but de compléter le nettoyage aussi rapidement que possible, l'université considère l'approche suivante. Les personnes sont numérotées de 1 à n, et les salles sont distribuées à tour de rôle à la personne qui détient la plus petite charge de travail. Si plusieurs personnes possèdent une charge minimale, alors la personne avec le plus petit numéro d'identification reçoit la salle. Par exemple, considérons $n = 3$ personnes et ces $m = 5$ salles:

	Salle	***Temps requis***
1	*D2-1060*	21 mins.
2	*D7-2023*	20 mins.
3	*D4-2011*	16 mins.
4	*D7-2021*	18 mins.
5	*D3-2029*	13 mins.

Initialement, chaque personne possède une charge de travail de 0 minute. La première personne se fait donc assigner la salle 1. Les trois personnes possèdent maintenant une charge de travail de 21, 0 et 0 minutes, respectivement. Ainsi, la deuxième personne reçoit la salle 2, puis, la troisième personne reçoit la salle 3. À ce stade, elles possèdent une charge de 21, 20 et 16 minutes, respectivement. La troisième personne possède donc la plus petite charge de travail et se fait ainsi assigner la salle 4. Les charges sont donc maintenant de 21, 20 et 34 minutes. La salle 5 est ainsi assignée à la deuxième personne. Nous obtenons cette distribution:



Ainsi, le nettoyage sera complété en 34 minutes.

(a) Décrivez la procédure de distribution des salles sous forme de *pseudocode*. Spécifiez le format de l'entrée et de la sortie de l'algorithme. Seules les séquences (tableaux et listes) sont permises comme structures de données. Vous ne pouvez pas invoquer des algorithmes existants en « boîtes noires ». Sur l'entrée de la page précédente, l'algorithme devrait retourner $[[1], [2, 5], [3, 4]]$. 6 pts

(b) Analysez le temps d'exécution de la procédure décrite en (a) dans le *pire cas* et le *meilleur cas* à la manière du diaporama d'introduction. Vous devez compter le nombre d'opérations élémentaires en fonction de m et n (et non pas utiliser la notation $\mathcal{O}$). Indiquez quelles opérations sont considérées élémentaires. Laissez une trace de vos calculs. 5 pts

(c) Prouvez que 34 minutes est bel et bien la durée minimale pour l'exemple de la page précédente; autrement dit, prouvez qu'il n'existe *aucune* distribution des cinq salles qui donne une durée *inférieure* à 34 minutes. Donnez un argument plus concis qu'une simple énumération de toutes les distributions possibles. 3 pts

(d) Montrez que la procédure décrite en (a) ne retourne pas toujours une distribution de durée minimale. 3 pts

(e) Donnez un algorithme qui retourne *toujours* une distribution de durée minimale, en complétant le pseudocode ci-dessous. Ne vous souciez pas de son efficacité. 5 pts

```
1 // à compléter
2 pour toute séquence s de m éléments parmi {1, 2, ..., n} faire
3     // à compléter
4 // à compléter
```

Par exemple, pour $m = 3$ et $n = 2$, la boucle principale énumère ces séquences:

$$[1, 1, 1],\ [2, 1, 1],\ [1, 2, 1],\ [2, 2, 1],\ [1, 1, 2],\ [2, 1, 2],\ [1, 2, 2],\ [2, 2, 2].$$

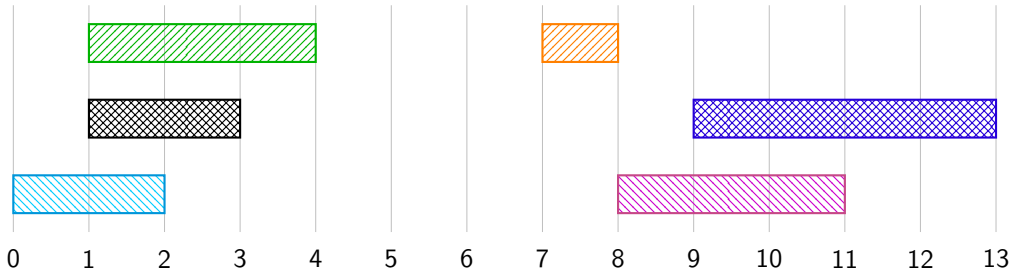
(f) Combien de tours de la boucle principale sont effectués dans le pire cas dans l'algorithme obtenu en (e)? Autrement dit, combien de fois est-ce que la ligne 2 est atteinte dans le pire cas? 3 pts

★ Donnez une implémentation, sous forme de pseudocode, de la procédure décrite en (a), qui fonctionne en temps $\mathcal{O}(n + m \log n)$ dans le pire cas. Vous pouvez cette fois utiliser des structures de données avancées, dont celles introduites en IFT339. ★ 1,5 pts

Question 2: conception d'algorithmes

Dû à la logistique de nettoyage, l'université cherche également à minimiser le nombre de salles utilisées dans une journée donnée. Par exemple, considérons une journée qui compterait trois cours à ces périodes respectives: 8h30 à 10h30, 9h30 à 11h30 et 14h30 à 16h30. Bien que ces cours pourraient avoir lieu dans trois salles distinctes, il est possible d'utiliser deux salles le matin et d'en réutiliser une l'après-midi. De façon générale, nous cherchons ainsi à minimiser le nombre de salles en fonction d'un horaire donné. On ignore ici les tailles physiques et le nombre d'inscriptions.

Formellement, un *cours* est une paire $(i, j) \in \mathbb{N}^2$, telle que $i < j$, où i représente le moment où le cours débute et j représente celui où il se termine. Deux cours (i, j) et (k, ℓ) peuvent avoir lieu dans la même salle si $j \leq k$; autrement dit, s'ils ne se chevauchent pas. Un *horaire* est une séquence finie de cours. Un horaire est réalisable dans m salles si l'on peut assigner à chaque cours une salle parmi m choix, sans créer de conflit. Par exemple, considérons l'horaire $H = [(0, 2), (1, 3), (1, 4), (9, 13), (7, 8), (8, 11)]$ illustré ci-dessous. Celui-ci peut être réalisé avec trois salles, par ex. les salles $[1, 2, 3, 2, 3, 1]$. Il est *impossible* de réaliser H avec moins de salles puisqu'il y a trois cours en simultané au moment 1. Ainsi, la sortie attendue pour cet horaire H est 3.



(a) Quel est le plus petit nombre et le plus grand nombre de salles pouvant être requis pour un horaire de n cours? Dans les deux cas, donnez un exemple générique où cela se produit. 1 pt

(b) Combien y a-t-il de façons d'assigner m salles à n cours (en incluant celles qui peuvent créer des conflits)? Un algorithme qui génèrerait toutes ces assignations fonctionnerait-il en temps polynomial? 1 pt

(c) Quel est le nombre minimal de salles permettant de réaliser l'horaire ci-dessous? 1 pt

$$[(0, 5), (1, 12), (8, 11), (3, 8), (12, 14), (13, 15), (18, 20), (1, 16)]$$

(d) Donnez un algorithme, sous forme de pseudocode, qui résout ce problème: 7 pts

ENTRÉE: un horaire $H = [(i_1, j_1), \ldots, (i_n, j_n)]$
SORTIE: le nombre minimal de salles qui permet de réaliser H

Vous pouvez utiliser une instruction **trier** qui trie une séquence de n éléments, selon l'ordre de votre choix, en temps $\mathcal{O}(n \log n)$ dans le pire cas. Analysez le temps d'exécution de votre algorithme asymptotiquement dans le pire cas. Vous obtiendrez jusqu'à:

— 7 points si votre algorithme fonctionne en temps $\mathcal{O}(n \log n)$,

— 3,5 points si votre algorithme fonctionne en temps $\mathcal{O}(n^2)$.

Question 3: notation et analyse asymptotique

(a) Considérons un algorithme $\mathcal{A}$ lancé sur un ordinateur qui exécute 225 000 millions d'opérations élémentaires par seconde. Soit $t(n)$ le temps d'exécution de $\mathcal{A}$ dans le pire cas. Pour chacune des fonctions suivantes, donnez la plus grande valeur (entière) de n où $\mathcal{A}$ termine (à coup sûr) en une minute ou moins. 3 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 de $\mathcal{A}$ dans le *pire cas*, et soit $g(n)$ le temps d'exécution de $\mathcal{B}$ dans le *meilleur cas*. Dans les deux scénarios suivants, dites à partir de quelle valeur (entière) de n l'algorithme $\mathcal{A}$ est *toujours plus rapide* que $\mathcal{B}$: 1 pt

(i) $f(n) = 100000 \cdot n^3$ et $g(n) = 3^n$,

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

Il n'est pas nécessaire de justifier vos réponses.

(c) Montrez que $2(n-4)(n+2)(n+3) + 8 \log n \in \Theta(n^3)$ sans utiliser les propositions vues en classes (donc en identifiant explicitement les constantes multiplicatives et seuils). 3 pts

(d) Ordonnez les fonctions suivantes selon la notation $\mathcal{O}$ (de la plus faible vers la plus haute complexité): 4 pts

$$\log_3(9n^2) + 128!, \qquad 2^{4 \cdot \log_2(n)} + 100\sqrt{n}, \qquad \sum_{i=1}^{n}(7i+2), \qquad \sum_{i=1}^{n}(2^i - 5i), \qquad \frac{n}{3} + 1000000$$

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 indice i de votre choix.

(e) Soit un algorithme $\mathcal{A}$ qui fonctionne en temps $f(n) \coloneqq \sum_{i=0}^{n}(i/2^i)$ dans le pire cas. Montrez, par induction, que $f(n) = (2^{n+1} - n - 2)/2^n$ pour tout $n \in \mathbb{N}$. Expliquez pourquoi $\mathcal{A}$ fonctionne en temps constant. 4 pts

★ Montrez qu'il existe des fonctions $f, g \in \mathcal{F}$ telles que $f \notin \mathcal{O}(g)$ et $g \notin \mathcal{O}(f)$; donc, incomparables selon $\mathcal{O}$. ★ 2,5 pts