# IFT436 – Algorithmes et structures de données

Université de Sherbrooke

# Devoir 1

| | |
|---|---|
| Enseignant: | Michael Blondin |
| Date de remise: | jeudi 12 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 |

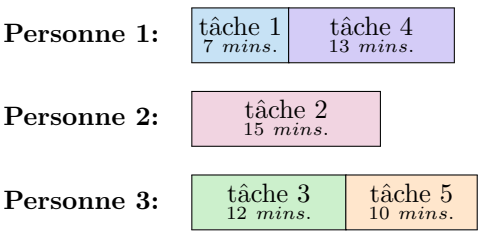
**Question 1.**

Vous organisez une fête et vous avez $m \in \mathbb{N}_{>0}$ tâches à réaliser afin d'être prêt·e à recevoir vos invité·e·s. Ces tâches requièrent précisément $t[1], t[2], \ldots, t[m] \in \mathbb{N}_{>0}$ minutes, respectivement. Ainsi, en effectuant les tâches par vous-même, vous mettriez $t[1] + t[2] + \cdots + t[m]$ minutes à organiser la fête. Heureusement, des ami·e·s se portent volontaires pour vous aider et vous êtes donc $n \in \mathbb{N}_{\geq 2}$ personnes au total. Ainsi, vous pouvez accomplir jusqu'à $n$ tâches simultanément (en parallèle).

Afin de compléter l'organisation le plus rapidement possible, vous vous entendez sur l'approche suivante. On suppose que les personnes sont numérotées de 1 à $n$, et les tâches sont distribuées à tour de rôle à la personne qui détient la plus petite charge de travail. Si plusieurs personnes possèdent une plus petite charge de travail, alors la personne avec le plus petit numéro d'identification reçoit la tâche. Par exemple, supposons qu'il y ait $n = 3$ personnes et ces $m = 5$ tâches:

| | Tâche | Temps requis |
|---|---|---|
| 1 | *Choisir une liste de lecture* | 7 mins. |
| 2 | *Faire l'épicerie* | 15 mins. |
| 3 | *Passer l'aspirateur* | 12 mins. |
| 4 | *Faire la vaisselle* | 13 mins. |
| 5 | *Décorer* | 10 mins. |

Initialement, chaque personne possède une charge de travail de 0 minute. Donc la première personne reçoit la tâche 1. Les trois personnes possèdent maintenant une charge de travail de 7, 0 et 0 minutes respectivement. Ainsi, la deuxième personne reçoit la tâche 2. Ensuite, la troisième personne reçoit la tâche 3. À ce stade, les personnes possèdent une charge de 7, 15 et 12 minutes respectivement. La première personne possède donc la plus petite charge de travail et reçoit ainsi la tâche 4. Les charges sont donc maintenant de 20, 15 et 12 minutes. La tâche 5 est donc assignée à la troisième personne. Nous obtenons donc la distribution suivante:




Ainsi, l'organisation de la fête sera complétée en 22 minutes.

(a) Décrivez la procédure de distribution des tâches 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 structure de données. Vous ne pouvez pas invoquer des algorithmes existants. Sur l'entrée de la page précédente, l'algorithme devrait retourner $[[1, 4], [2], [3, 5]]$. 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 22 minutes est la durée minimale dans l'exemple de la page précédente; autrement dit, prouvez qu'il n'existe *aucune* distribution des cinq tâches qui donne une durée *inférieure* à 22 minutes. Pensez à un argument plus court qu'une énumération de toutes les distributions possibles. 4 pts

(d) Montrez que la procédure ne retourne pas toujours une distribution de durée minimale en donnant un contre-exemple. 4 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. 6 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:

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

Une implémentation de cette énumération vous est fournie sur GitHub.

(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. ★ 2.5 pts

**Question 2.** 10 pts

Vous tentez de choisir une date pour l'organisation de votre fête. Vous avez une séquence $s$ de dates auxquelles vous êtes disponible, et les personnes invitées vous ont fourni collectivement une séquence $t$ des dates auxquelles elles sont disponibles. Vous cherchez à déterminer si une date convient à la fois à vous et vos invité·e·s. Afin de vous simplifier la vie, les deux séquences sont déjà *triées en ordre croissant* et possèdent chacune $n \in \mathbb{N}_{>0}$ éléments. Donnez un algorithme qui détermine s'il existe une date commune, et qui retourne une telle date le cas échéant.

Vous n'avez pas à vous préoccuper du format des dates, nous supposons que ce sont simplement des entiers naturels où 0 correspond à aujourd'hui, 1 à demain, 2 à après-demain, et ainsi de suite. Par exemple, votre algorithme doit retourner 8 sur l'entrée:

$$s = [2, 5, 6, 8, 30], \quad t = [1, 3, 8, 20, 25],$$

et doit retourner « *conflit d'horaire* » sur l'entrée:

$$s = [2, 5, 6, 7, 30], \quad t = [1, 3, 8, 20, 25].$$

Vous pouvez obtenir jusqu'à:

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

Vous devez respecter les contraintes suivantes:

- L'algorithme doit être décrit sous forme de *pseudocode*.
- Seules les séquences (tableaux et listes) sont permises comme structure de données. Vous ne pouvez pas invoquer des algorithmes existants.
- Vous n'avez *pas* à démontrer le temps d'exécution de votre algorithme.

★ Considérez le problème plus général où $m \in \mathbb{N}_{\geq 2}$ personnes fournissent leur propre séquence de $n \in \mathbb{N}_{>0}$ dates triées. Donnez un algorithme qui fonctionne en temps $\mathcal{O}(m \cdot n)$ et qui détermine si une date convient à *toutes* les personnes. Par exemple, s'il y a $m = 3$ personnes qui fournissent $n = 4$ dates chacune, alors votre algorithme doit retourner 8 sur l'entrée: ★ 2.5 pts

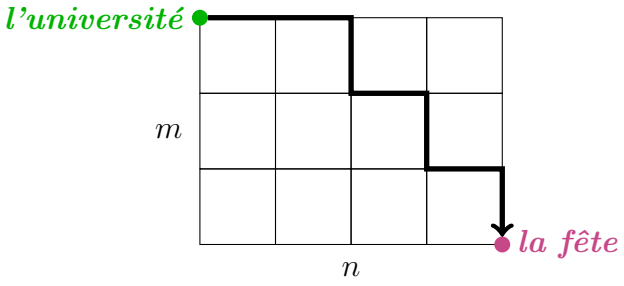
$$[[2, 5, 8, 30],\ [5, 8, 20, 50],\ [1, 8, 20, 30]],$$

et doit retourner « *conflit d'horaire* » sur l'entrée:

$$[[2, 5, 8, 30],\ [4, 5, 20, 50],\ [1, 2, 8, 30]].$$

**Question 3.**

Vos invité·es comptent se rendre à votre fête en groupe à partir de l'université. Le quartier est organisé sous forme de grille de $m$ cases par $n$ cases, où chaque segment correspond à une rue. L'université se situe au coin supérieur gauche de la grille, et la fête a lieu au coin inférieur droit. Vous vous demandez *combien* il existe de chemins *différents* afin de se rendre à la fête en se déplaçant *uniquement* vers l'est (la droite) et vers le sud (le bas), et ce sans revenir sur vos pas. Par exemple, pour $m = 3$ et $n = 4$, un chemin possible est:




Soit $f\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ la fonction telle que $f(m, n)$ indique le nombre de chemins différents permettant de se rendre à la fête dans un quartier de $m \times n$ cases. Une invitée qui a déjà suivi IFT436 prétend que $f(m, n) = \binom{m+n}{n}$. Démontrez qu'elle a raison en répondant aux deux sous-questions suivantes:

(a) Prouvez que $\displaystyle\sum_{i=0}^{n} \binom{m+i-1}{i} = \binom{m+n}{n}$ pour tous $m \in \mathbb{N}_{\geq 1}$ et $n \in \mathbb{N}$. 6 pts

(b) Prouvez que $f(m, n) = \binom{m+n}{n}$ pour tous $m, n \in \mathbb{N}$. 6 pts

*Indices et rappels:*

- En (a) et (b), considérez une preuve par induction sur $m$ *ou* sur $n$.
- En (b), considérez réfléchir récursivement et exploiter (a).
- $\binom{b}{a}$ dénote le *coefficient binomial* qui indique le nombre de façons de choisir $a$ objets parmi $b$ objets distincts, sans tenir compte de l'ordre (voir notes de cours au besoin).
- Si cela vous aide, vous pouvez utiliser la formule de Pascal:

$$\binom{b}{a} = \binom{b-1}{a} + \binom{b-1}{a-1} \qquad \text{pour tous } a, b \in \mathbb{N} \text{ tels que } b > a.$$