

IFT436 – Algorithmes et structures de données  
Université de Sherbrooke

## Examen périodique

Enseignant: Michael Blondin  
Date: samedi 17 octobre 2020  
Durée: 110 min.

### Directives:

- Vous devez répondre aux questions dans le **cahier de réponses**, pas sur ce questionnaire;
- **Une seule feuille** de notes manuscrites au format 8½" × 11" est permise;
- **Aucun matériel additionnel** (notes de cours, fiches récapitulatives, etc.) n'est permis;
- **Aucun appareil électronique** (calculatrice, téléphone, montre intelligente, etc.) n'est permis;
- Vous devez donner **une seule réponse** par sous-question;
- L'examen comporte **4 questions** sur **3 pages** valant un total de **50 points**;
- La correction se base notamment sur la **clarté**, l'**exactitude** et la **concision** de vos réponses, ainsi que sur la **justification** pour les questions qui en requièrent une;
- Si la **base d'un logarithme** n'est pas spécifiée, il s'agit de la base 2: « log » dénote « log<sub>2</sub> »;
- Les indices d'une séquence **débutent à 1**; autrement dit,  $s = [s[1], s[2], \dots, s[n]]$  si  $n = |s|$ .

### Question 1: analyse des algorithmes

- (a) Ordonnez ces fonctions  $f_1, f_2, f_3, f_4$  et  $f_5$  selon la notation  $\mathcal{O}$  (de la plus faible vers la plus haute complexité): 7 pts

$$\underbrace{5 \cdot \sum_{i=0}^n \frac{n}{2^i} + 3}_{f_1}, \quad \underbrace{\frac{1}{100} \cdot \sum_{i=1}^n (i^3 - 5i + 1)}_{f_2}, \quad \underbrace{\log(n^8) \cdot \log(32n)}_{f_3}, \quad \underbrace{\frac{1}{4} \cdot \log(4^n) + \left(\frac{3}{2}\right)^n}_{f_4}, \quad \underbrace{\sum_{i=1}^n \sum_{j=1}^n 42j}_{f_5}$$

- (b) Considérons un algorithme  $\mathcal{A}$  qui reçoit en entrée un graphe dirigé de  $n$  sommets et  $m$  arêtes. Après l'avoir analysé, nous avons obtenu  $t_{\max} \in \mathcal{O}(3n \log n + 2n \log m + 5n)$ , où  $t_{\max}$  dénote le temps d'exécution dans le pire cas de  $\mathcal{A}$ . Pouvons-nous simplifier et affirmer que  $t_{\max} \in \mathcal{O}(n \log n)$ ? Justifiez. 2 pts

- (c) Si  $f \in \Theta(g)$  et  $h \in \Omega(g)$ , pouvons-nous déduire  $f \in \Omega(h)$ ? Justifiez. 2 pts

- (d) Dites auxquels de ces ensembles appartient la fonction  $n \log n$ : 2 pts

$$\Omega(n), \quad \mathcal{O}(\sqrt{n}), \quad \Theta(2^n), \quad \Omega(2n \log n), \quad \mathcal{O}(n^2), \quad \mathcal{O}(n!), \quad \Theta(\sqrt{n} \cdot n).$$

**Question 2: tri**

Considérons l'algorithme de tri suivant:

**Entrée:** séquence  $s$  de  $n \in \mathbb{N}$  éléments comparables

**Sortie:**  $s$  triée

```

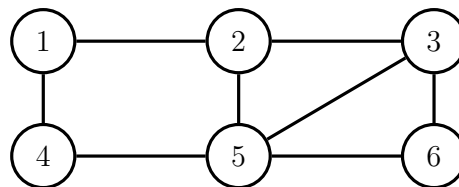
1  $j \leftarrow n - 1$ 
2 faire
3    $fini \leftarrow vrai$ 
4   pour  $i \in [1..j]$ 
5     si  $s[i] > s[i + 1]$  alors
6        $s[i] \leftrightarrow s[i + 1]$  // inverser le contenu de  $s[i]$  et  $s[i + 1]$ 
7        $fini \leftarrow faux$ 
8    $j \leftarrow j - 1$ 
9 tant que  $\neg fini$ 
10 retourner  $s$ 

```

- (a) Exécutez l'algorithme sur l'entrée  $s = [10, 50, 20, 40, 30]$ . Plus précisément, donnez le contenu de la séquence  $s$  et du compteur  $j$  à la fin de chaque tour de la boucle **faire...tant que**. 3 pts
- (b) Les affirmations suivantes sont toutes vraies. Choisissez en *deux* (pas plus!) et montrez leur véracité. 7 pts
- L'algorithme fonctionne en temps  $\Omega(n^2)$  dans le pire cas.
  - L'algorithme fonctionne en temps  $\mathcal{O}(n^2)$  dans le pire cas.
  - L'algorithme fonctionne en temps  $\Omega(n)$  dans le meilleur cas.
  - L'algorithme fonctionne en temps  $\mathcal{O}(n)$  dans le meilleur cas.
  - L'algorithme est correct.
- (c) Dites si l'algorithme est stable et/ou s'il est sur place. Justifiez. 4 pts

**Question 3: graphes**

(a) Considérons ce graphe non dirigé  $\mathcal{G}$ :



- Donnez la *liste d'adjacence* de  $\mathcal{G}$  et le degré de chacun de ses sommets. 2 pts
  - Parcourez  $\mathcal{G}$  en profondeur à partir du sommet 1, en laissant une trace de l'ordre dans lequel les sommets sont marqués et de l'arbre couvrant formé des arêtes qui contribuent à la découverte d'un sommet. Lors de l'exploration d'un sommet, considérez ses voisins en ordre croissant de leur identifiant numérique. 2 pts
  - Quels sont les *cycles simples* de  $\mathcal{G}$ ? 2 pts
- (b) Soit  $\mathcal{G}$  une forêt. Supposons que l'on ajoute une direction à chaque arête de  $\mathcal{G}$  de façon arbitraire. Expliquez pourquoi le graphe *dirigé* résultant possède forcément un ordre topologique. 2 pts

(c) Donnez un algorithme, sous forme de pseudocode, qui résout le problème suivant:

5 pts

ENTRÉE: un graphe dirigé  $\mathcal{G} = (V, E)$  sous forme de liste d'adjacence  
 SORTIE:  $\mathcal{G}$  est-il fortement connexe?

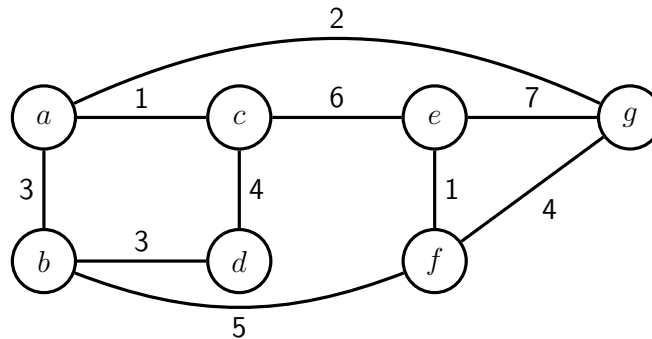
Analysez son temps d'exécution dans le pire cas. Pour obtenir tous les points, il doit appartenir à  $\mathcal{O}(|V|+|E|)$ .

Précision: au besoin, vous pouvez invoquer des algorithmes couverts en classe sans les réimplémenter.

#### Question 4: algorithmes gloutons

(a) Identifiez un arbre couvrant minimal de ce graphe pondéré avec l'algorithme de Prim–Jarník ou de Kruskal. Laissez une trace de l'ordre dans lequel les arêtes de l'arbre sont sélectionnées.

3 pts



(b) Soit  $\mathcal{G}$  un graphe non dirigé pondéré par  $p$ . Supposons que  $\mathcal{G}$  soit connexe et que ses poids soient *distincts*, c.-à-d.  $p[e] \neq p[f]$  pour toutes arêtes  $e \neq f$ . Montrez que  $\mathcal{G}$  possède *exactement un* arbre couvrant minimal.

2 pts

(c) Considérons le problème qui consiste à maximiser le nombre de cours qui peuvent avoir lieu, sans conflit, dans *une seule* salle. Comme au devoir 1, un cours est une paire  $(i, j) \in \mathbb{N}^2$ , où  $i$  indique son début et  $j$  sa fin, et où deux cours  $(i, j)$  et  $(k, \ell)$  ne sont pas en conflit ssi  $j \leq k$ .

5 pts

Une approche naïve pour résoudre ce problème consiste à énumérer toutes les façons de choisir un ensemble de cours parmi  $n$  cours; vérifier s'ils créent un conflit; et si ce n'est pas le cas, mettre à jour la meilleure solution découverte. Afin de réduire la complexité, on pourrait plutôt utiliser cette approche gloutonne:

---

**Entrée:** une séquence de cours  $H = [(i_1, j_1), \dots, (i_n, j_n)]$

**Sortie:** le nombre maximal de cours de  $H$  qu'on peut assigner dans une même salle

- 1  $S \leftarrow \emptyset$
  - 2 **tant que**  $H$  est non vide
  - 3   **sélectionner** et **retirer** un cours  $c$  de  $H$
  - 4   **si**  $c$  ne crée pas de conflit dans  $S$  **alors ajouter**  $c$  à  $S$
  - 5 **retourner**  $|S|$
- 

Considérons ces trois implémentations de l'opération « **sélectionner** »:

- (i) choisir un cours  $c = (i, j)$  de plus petite durée  $j - i$ ;
- (ii) choisir un cours  $c = (i, j)$  qui termine le plus tôt;
- (iii) choisir un cours  $c = (i, j)$  qui débute le plus tôt.

L'approche gloutonne ci-dessus est correcte avec *une seule* de ces trois implémentations. Identifiez-la. Dites aussi combien d'itérations l'approche naïve doit effectuer dans le pire cas. Justifiez.