IFT436 – Algorithmes et structures de données

Université de Sherbrooke

Devoir 4

Enseignant:	Michael Blondin
Date de remise:	mardi 29 novembre 2022 à 23h59
À réaliser:	en équipe de deux ou individuellement
Modalités:	remettre en ligne sur Turnin dans un fichier PDF
Bonus:	les questions bonus sont indiquées par ★
Pointage:	max. 50 points + 5 points bonus
Consignes	commentez et décrivez les grandes lignes de vos algorithmes afin de faciliter leur compréhension

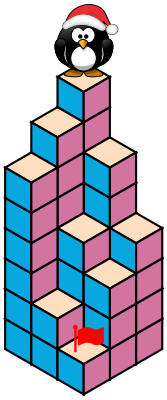
Question 1.

Une *montagne* est une matrice carrée $\mathbf{M} \in \mathbb{N}^{n \times n}$ dont chaque ligne et chaque colonne est ordonnée de façon strictement décroissante:

$$\begin{array}{ccccccc}
\mathbf{M}[1,1] & > & \mathbf{M}[1,2] & > & \cdots & > & \mathbf{M}[1,n] \\
\vee & & \vee & & & & \vee \\
\mathbf{M}[2,1] & > & \mathbf{M}[2,2] & > & \cdots & > & \mathbf{M}[2,n] \\
\vee & & \vee & & & & \vee \\
\vdots & & \vdots & & & & \vdots \\
\mathbf{M}[n,1] & > & \mathbf{M}[n,2] & > & \cdots & > & \mathbf{M}[n,n]
\end{array}$$

L'entrée $\mathbf{M}[i,j]$ indique la hauteur du sol de la montagne à la coordonnée (i,j). Le célèbre manchot *Tux* se situe au sommet d'une telle montagne et désire se rendre tout en bas. Tux peut descendre d'une position vers une position adjacente en glissant sur son ventre le long des parois. Une telle glissade de hauteur h dure $f(h)$ secondes, où $f\colon \mathbb{N}_{>0} \to \mathbb{R}_{>0}$ est une fonction qui tient compte de la glace, du plumage, de l'accélération, etc. Tux désire connaître la durée d'une descente *minimale*. Par exemple, considérons cette montagne:

$$\mathbf{M} = \begin{pmatrix} 8 & 6 & 4 \\ 7 & 4 & 3 \\ 6 & 2 & 1 \end{pmatrix} \quad \equiv$$



Voici la durée de trois des six descentes:

Chemin	Durée (en secondes)	Ex. $f(h) = \sqrt{h}$
→ → ↓ ↓	$f(8-6) + f(6-4) + f(4-3) + f(3-1) = f(2) + f(2) + f(1) + f(2)$	$\approx 5{,}24264$
↓ → → ↓	$f(8-7) + f(7-4) + f(4-3) + f(3-1) = f(1) + f(3) + f(1) + f(2)$	$\approx 5{,}14626$
↓ ↓ → →	$f(8-7) + f(7-6) + f(6-2) + f(2-1) = f(1) + f(1) + f(4) + f(1)$	5

En explorant toutes les descentes, on conclurait que la durée minimale est de 5 secondes lorsque $f(n) = \sqrt{n}$.

(a) Si $f(h) = h$, alors on peut calculer la durée d'une descente minimale en temps $\mathcal{O}(1)$. Pourquoi? Justifiez. 5 pts

(b) Donnez un algorithme de force brute qui identifie la *durée* d'une descente minimale en explorant *toutes* les descentes. Votre algorithme fonctionne-t-il en temps polynomial par rapport à n? Justifiez brièvement. 8 pts

(c) Donnez un algorithme qui exploite la programmation dynamique avec *tableaux* afin d'identifier la *durée* d'une descente minimale. Votre algorithme doit fonctionner en temps $\mathcal{O}(n^2)$. Quel tableau est calculé par l'algorithme sur la montagne ci-dessus avec $f(h) = \sqrt{h}$? Vous pouvez donner les valeurs symboliquement ou numériquement arrondies à cinq décimaux après la virgule. 8 pts

(d) Adaptez votre algorithme obtenu en (c) afin qu'il retourne une séquence de *directions* que Tux peut emprunter afin d'effectuer une descente de durée minimale. Par exemple, votre algorithme devrait retourner la séquence [🡻, 🡻, 🡺, 🡺] pour la montagne ci-dessus avec $f(h) = \sqrt{h}$. 4 pts

Clarifications:

— Le sommet et le bas de la montagne se situent respectivement aux positions $(1, 1)$ et (n, n);

— À partir de la position (i, j), Tux peut glisser aux positions $(i{+}1, j)$ et $(i, j{+}1)$, pourvu qu'elles n'excèdent pas la dimension de la montagne;

— La hauteur de la glissade d'une position (i, j) vers une position adjacente (i', j') est $\mathbf{M}[i, j] - \mathbf{M}[i', j']$;

— On suppose que l'évaluation de $f(h)$ est une opération élémentaire. Imaginez f comme une fonction (pure) offerte par le moteur physique d'un jeu vidéo dont vous ne connaissez pas l'implémentation.

Question 2.

Considérons quatre types de *potions*, représentés par les symboles a, b, c et x, que nous pouvons combiner. Le résultat des combinaisons des potions est décrit par une « table de multiplication » comme celle-ci:

⊗	a	b	c	x
a	a	b	x	a
b	b	b	a	a
c	x	a	b	c
x	a	a	c	b

On peut combiner une séquence de potions en appliquant l'opération $\otimes$. Puisque $\otimes$ n'est pas nécessairement associative, l'ordre d'évaluation a une incidence sur le résultat. Par exemple, en utilisant l'opération ci-dessus, la séquence [x, b, c] peut être combinée de deux façons:

$$(x \otimes b) \otimes c = a \otimes c = x$$

$$x \otimes (b \otimes c) = x \otimes a = a$$

Une druidesse désire combiner consécutivement une séquence de potions afin d'obtenir une potion de type x. Autrement dit, elle cherche à résoudre ce problème:

ENTRÉE: une séquence p de $n \in \mathbb{N}_{\geq 1}$ symboles parmi $\{a, b, c, x\}$, et la table T d'une opération binaire $\otimes$ sur $\{a, b, c, x\}$

SORTIE: est-ce possible de parenthéser l'expression $p[1] \otimes p[2] \otimes \cdots \otimes p[n]$ de telle sorte qu'elle évalue à x?

(a) Dites si les séquences $p = [x, b, c, a]$ et $q = [c, b, c, a]$ peuvent évaluer à x selon la table ci-dessus. Justifiez. 5 pts

(b) Aidez la druidesse en donnant un algorithme qui résout son problème. Vous pouvez accéder à la table T de l'opération $\otimes$ en temps constant, par exemple: pour la table ci-dessus, nous avons $T[a, b] = b$ et $T[c, a] = x$. Votre algorithme doit fonctionner en temps polynomial par rapport à n. Expliquez pourquoi c'est le cas. 10 pts

Indices:

— On peut procéder récursivement (approche descendante) ou en remplissant un tableau 2D (approche ascendante); le temps d'exécution d'une approche ascendante est typiquement plus facile à analyser;

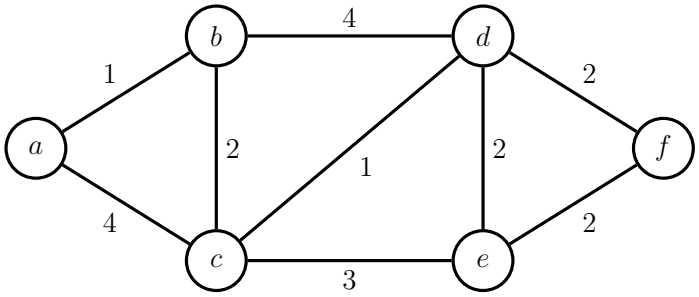
— Pensez aux deux paramètres comme étant la position de début et de fin d'une sous-séquence de potions;

— Pensez aux façons de placer les deux premières paires de parenthèses: $(\cdots)(\cdots)$;

— Une case d'un tableau de programmation dynamique peut dépendre d'un nombre *linéaire* de cases, pas nécessairement une, deux ou trois cases. De plus, ces cases ne sont pas nécessairement tous sur une même ligne.

Question 3.

(a) Exécutez l'algorithme de Dijkstra sur le graphe pondéré suivant à partir du sommet a. Laissez une trace de chaque itération de l'algorithme en indiquant les sommets marqués et les distances partielles identifiées. 5 pts

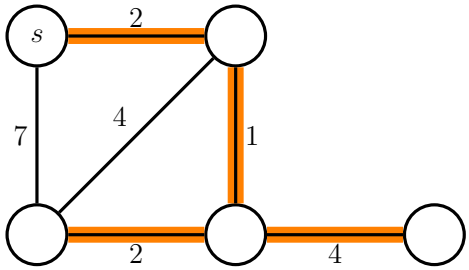


(b) Soit $\mathcal{G} = (V, E)$ un graphe pondéré, non dirigé et connexe. Soit $s \in V$ un sommet de $\mathcal{G}$. Un *arbre de plus courts chemins (par rapport à s)* est un sous-graphe $\mathcal{T}$ de $\mathcal{G}$ qui satisfait ces propriétés:

— $\mathcal{T}$ est un arbre couvrant de $\mathcal{G}$,

— pour tout $v \in V$, l'unique chemin simple de s vers v dans $\mathcal{T}$ est un plus court chemin de s vers v.

Par exemple, voici un arbre de plus courts chemins par rapport au sommet s de ce graphe:



(i) Cet arbre est un arbre couvrant *minimal*. On pourrait donc être tenté de croire que les deux notions coïncident. Montrez que ce n'est *pas* le cas. Autrement dit, identifiez un graphe pondéré $\mathcal{G}$ et un sommet s tel qu'un arbre de plus courts chemins par rapport à s ne correspond *pas* à un arbre couvrant minimal de $\mathcal{G}$. Justifiez. 3 pts

(ii) Identifiez un arbre de plus courts chemins par rapport au sommet a du graphe de la sous-question (a). 2 pts

Remarque: Si vous l'avez déjà fait, indiquez simplement « voir en (a) ».

★ Si l'on connaît des plus courts chemins (non dirigés) à partir de s et à partir de t, et qu'on retire une arête e du graphe, alors il est possible de recalculer la distance entre s et t en temps linéaire. Montrez que c'est le cas en donnant un algorithme qui résout le problème suivant en temps $\mathcal{O}(|V| + |E|)$: ★ 5 pts

ENTRÉE: un graphe pondéré, non dirigé et connexe $\mathcal{G} = (V, E)$, une arête $e \in E$, deux sommets $s, t \in V$, et deux arbres de plus courts chemins $\mathcal{T}_s$ et $\mathcal{T}_t$ par rapport à s et t respectivement

SORTIE: la distance entre s et t dans le graphe $\mathcal{G}$ auquel on retire e