IFT436 – Algorithmes et structures de données

Université de Sherbrooke

Devoir 3

Enseignant:	Michael Blondin
Date de remise:	mardi 15 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 + 3 points bonus
Consignes	commentez et décrivez les grandes lignes de vos algorithmes afin de faciliter leur compréhension

Question 1.

(a) Soit $\mathcal{P}_n$ l'ensemble des pavages, d'une grille $4 \times n$, constitués de tuiles de cette forme (sans rotations):

(i) Donnez une récurrence linéaire t telle que $t(n) = |\mathcal{P}_n|$ pour tout $n \in \mathbb{N}_{\geq 1}$. Expliquez comment vous avez obtenu votre récurrence. 4 pts

(ii) Appliquez la méthode de résolution de récurrences linéaires présentée en classe afin d'obtenir une forme close pour t. Laissez une trace de votre démarche. 5 pts

(iii) Donnez la complexité asymptotique de t aussi précisément que possible avec la notation Θ. Justifiez. 3 pts

(b) Au tout premier cours de la session, nous avons brièvement vu cet algorithme:

Entrées : séquence s de $n \in \mathbb{N}_{\geq 1}$ entiers
Sorties : $\max\{s[i] : 1 \leq i \leq n\}$

```
max-rec(s):
    si n = 1 alors
        retourner s[1]
    sinon
        m  ← max-rec(s[1 : n ÷ 2])
        m' ← max-rec(s[n ÷ 2 + 1 : n])
        si m > m' alors retourner m
        sinon         retourner m'
```

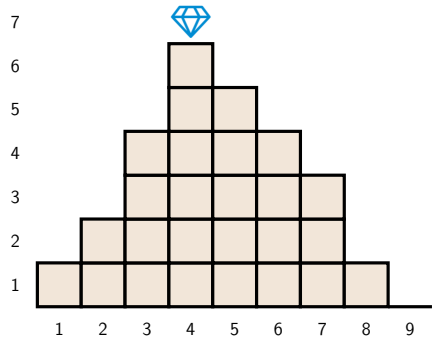
(i) Donnez une récurrence qui décrit le temps d'exécution *exact* de `max-rec` par rapport à $n = |s|$ (donc avec les constantes). Considérez ces opérations comme élémentaires: 4 pts

- comparaison;
- addition;
- division;
- affectation;
- accès au $i^{\text{ème}}$ élément d'une séquence;
- obtention d'une sous-séquence $s[i : j]$ (« *slicing* »).

(ii) L'algorithme `max-rec` est-il plus rapide (asymptotiquement) qu'un algorithme itératif usuel pour le calcul du maximum d'une séquence? Justifiez. Si cela vous aide, vous pouvez supposer que n est une puissance de deux. 4 pts

Question 2.

(a) Imaginons un jeu où un diamant se trouve au sommet d'une colline de blocs. Vous devez atteindre le diamant avec un projectile. Vous avez accès à une decription de la colline sous forme d'une séquence c où $c[i] \in \mathbb{N}$ indique le nombre de blocs de la $i^{\text{ème}}$ colonne de la colline. Par exemple, $c = [1, 2, 4, 6, 5, 4, 3, 1, 0]$ décrit cette colline: 10 pts



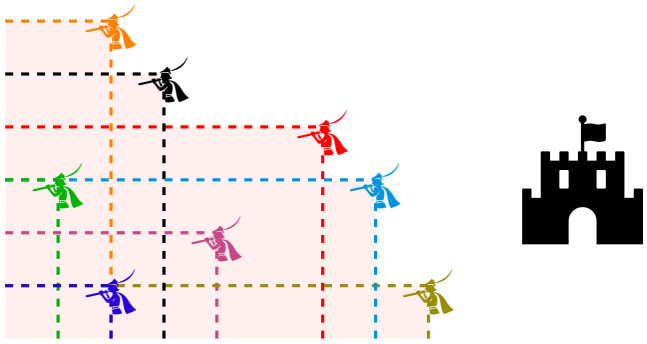
Formellement, une séquence c de $n \in \mathbb{N}_{\geq 3}$ nombres naturels forme une *colline* s'il existe $i \in [1..n]$ tel que

$$c[1] < c[2] < \cdots < c[i-1] < c[i] > c[i+1] > \cdots > c[n-1] > c[n].$$

Cet indice i indique la colonne du sommet de c. Le diamant se trouve sur le bloc au sommet de la colline, par ex. à la position $(4, 7)$ pour la colline ci-dessus. Vous n'avez qu'un seul projectile et très peu de temps pour le lancer. Donnez donc un algorithme qui résout ce problème en temps $\mathcal{O}(\log n)$:

ENTRÉE: séquence c décrivant une colline de $n \in \mathbb{N}_{\geq 3}$ colonnes
SORTIE: position (i, j) du diamant de la colline c

(b) Imaginons un jeu où des gardes munis de sarbacanes défendent une forteresse. Chaque garde se situe à une *position* $(x, y) \in \mathbb{N}^2$ et surveille la région $\{(a, b) \in \mathbb{N}^2 : a \leq x \wedge b \leq y\}$. Par exemple, huit gardes aux positions $[(6, 4), (4, 2), (8, 1), (2, 6), (3, 5), (7, 3), (1, 3), (2, 1)]$ surveillent collectivement cette région globale: 10 pts



Vous désirez améliorer votre forteresse, mais n'avez plus de pièce d'or. Vous désirez donc retirer autant de gardes que possible afin d'obtenir des pièces en échange, et ce *sans changer la région globale* surveillée par les gardes. Par exemple, on peut retirer trois gardes au maximum dans le scénario ci-dessus. Donnez donc un algorithme qui résout ce problème en temps $\mathcal{O}(n \log n)$:

ENTRÉE: séquence p des positions des $n \in \mathbb{N}_{\geq 1}$ gardes
SORTIE: plus grand nombre de gardes qu'on peut retirer de p sans changer la région globale surveillée collectivement

Indice: divisez le royaume pour régner!

Question 3.

Donnez un algorithme qui résout ce problème en temps $\Theta(n^{\log_3 6})$ dans le pire cas: 10 pts

ENTRÉE: $x \in \mathbb{N}$ représenté avec n chiffres en base 10
SORTIE: x^2

Expliquez pourquoi il fonctionne en temps $\mathcal{O}(n^{\log_3 6})$. Vous n'avez pas à justifier l'appartenance à $\Omega(n^{\log_3 6})$.

Précision: nous faisons ces hypothèses (comme en classe):

— *L'addition et la soustraction de deux nombres d'au plus k chiffres s'effectue en temps $\mathcal{O}(k)$;*

— *Le décalage et la troncation de k chiffres d'un nombre s'effectue en temps $\mathcal{O}(k)$.*

Indice: inspirez-vous de l'algorithme de multiplication rapide vu en classe mais en divisant autrement.

★ Donnez un algorithme plus efficace pour le cas spécifique d'un nombre x composé d'un seul chiffre répété, par ex. $x = 3333333333$. Plus précisément, donnez un algorithme qui résout ce problème en temps $\mathcal{O}(n)$: ★ 3 pts

ENTRÉE: $x \in \mathbb{N}$ composé d'un seul chiffre en base 10 répété n fois
SORTIE: x^2

Expliquez brièvement le fonctionnement de votre algorithme et pourquoi il s'exécute en temps $\mathcal{O}(n)$.