IFT436 – Algorithmes et structures de données

Université de Sherbrooke

Devoir 2

Enseignant:	Michael Blondin
Date de remise:	mercredi 7 octobre 2020 à 10h29
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 + 5 points bonus

Question 1: Tri – analyse d'algorithme

Pour cette question, nous considérons (comme dans les notes de cours) que les éléments d'une séquence s de n éléments sont numérotés de 1 à n, c.-à-d. $s = [s[1], s[2], \ldots, s[n]]$. Considérons l'algorithme de tri suivant:

Entrées : séquence T de $n \in \mathbb{N}_{>0}$ éléments comparables
Résultat : séquence T triée

`trier`(T):
    `corriger-inv`(j, k): // sous-routine avec accès à T
        si $T[j] > T[k]$ **alors**
            $T[j] \leftrightarrow T[k]$ // inverser le contenu de $T[j]$ et $T[k]$
            retourner *faux*
        sinon
            retourner *vrai*

    $m \leftarrow n \div 2$
    faire
        terminé $\leftarrow$ *vrai*
        pour $i \in [1..m]$ **faire**
            terminé $\leftarrow$ `corriger-inv`$(2i-1, 2i) \land$ *terminé*
        pour $i \in [1..m]$ **faire**
            si $2i < n$ **alors**
                terminé $\leftarrow$ `corriger-inv`$(2i, 2i+1) \land$ *terminé*
    tant que $\neg$*terminé*
    retourner T

(a) Afin vous familiariser avec l'algorithme, pour chacune des entrées suivantes, exécutez `trier`(T) et donnez le contenu de T à la fin de chaque tour de la boucle **faire ... tant que**: 2 pts

— $T = [66, 99, 100, 88, 77, 200]$;

— $T = [80, 20, 30, 40, 50, 60, 10]$;

— $T = [1002, 1000, 1001, 1006, 1004, 1005]$.

(b) Expliquez brièvement, en mots, le fonctionnement de l'algorithme. 2 pts

(c) Montrez que si une séquence s de n éléments n'est *pas* triée, alors il existe $i \in [1..n]$ tel que $s[i] > s[i+1]$. 3 pts

(d) Expliquez pourquoi l'algorithme termine et est correct. 3 pts

Indice: considérez (c) *et les propriétées des inversions vues en classe.*

(e) Analysez le temps d'exécution dans le pire cas afin de montrer qu'il appartient à $\mathcal{O}(n^3)$. 3 pts

(f) Argumentez que le temps d'exécution dans le pire cas appartient à $\Omega(n^2)$. 4 pts

Indice: pensez aux « pires entrées » possibles.

(g) Le temps d'exécution de l'algorithme dans le meilleur cas appartient-il à $\mathcal{O}(n)$? Justifiez votre réponse. 3 pts

★ Montrez que le temps d'exécution dans le pire cas appartient à $\Theta(n^2)$. ★ 2,5 pts

Question 2: Tri – adaptation d'algorithme

Soit s une séquence d'éléments comparables. Nous écrivons $|s|_x$ afin de dénoter le nombre d'occurrences de x dans s. Un *mode* m de s est une valeur qui apparaît un nombre maximal de fois dans s; autrement dit, tel que $|s|_m = \max\{|s|_x : x \in s\}$. Par exemple, $s = [42, 42, 0, 1, 0, 42, 0]$ possède deux modes: 0 et 42.

(a) Adaptez l'algorithme de tri rapide afin de résoudre ce problème en temps $\mathcal{O}(n \log n)$: 7 pts

ENTRÉE: une séquence s de $n \in \mathbb{N}_{>0}$ éléments comparables
SORTIE: un mode m de s

Vous n'avez *pas* à analyser le temps d'exécution de votre algorithme. On suppose que vous avez accès à une instruction `médiane`(s) qui retourne la médiane d'une séquence s de $n > 0$ éléments en temps $\mathcal{O}(n)$. Cela permet donc d'implémenter une version idéalisée du tri rapide.

Remarque: on pourrait trouver un mode en triant d'abord, puis en effectuant un parcours linéaire; mais ce n'est pas ce qui est demandé, nous cherchons une approche récursive qui résout le problème sans prétraitement.

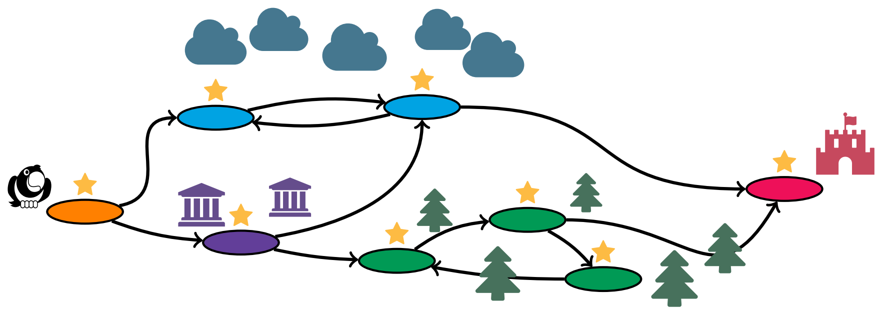
Précision: Dans ce contexte, la médiane réfère à la valeur qui se retrouve à la position $\lceil n/2 \rceil$ *lorsque* s *est triée; par ex. la médiane de* $[20, 40, 10, 50, 60, 30]$ *est* 30 *et la médiane de* $[70, 20, 40, 10, 50, 60, 30]$ *est* 40. *Ainsi, pour les séquences de taille paire, on ne prend* pas *la moyenne des deux valeurs milieux comme en statistiques.*

(b) Afin d'illustrer le fonctionnement de votre algorithme, tracez son arbre de récursion sur entrée: 3 pts

$$s = [10, 20, 20, 20, 10, 30, 70, 20, 80, 30, 60, 40, 20, 10, 50, 10, 70, 80].$$

Question 3: Algorithmes de graphes

Considérons un jeu vidéo non linéaire où la complétion d'un niveau permet de passer à un niveau subséquent parmi certains choix. La première fois qu'on complète un niveau, on obtient une étoile. Il est possible de rejouer un niveau si on y revient, mais cela n'augmente pas le nombre d'étoiles obtenues. Par exemple, supposons que le jeu soit constitué de huit niveaux organisés de cette manière:



Le personnage, qui débute au premier niveau (à gauche), peut obtenir quatre étoiles en complétant les deux niveaux du ciel, puis en se déplaçant au château (le dernier niveau à droite). Toutefois, la solution qui maximise le nombre d'étoiles visite la cité, suivie d'un tour des trois niveaux de la forêt, complété d'une visite du château.

Cherchons à automatiser la recherche d'un tel score maximal. Nous supposons qu'un jeu est représenté par un graphe dirigé de n sommets et m arêtes, sous forme de liste d'adjacence. Pour chacune des questions suivantes, vous pourrez obtenir tous les points si votre algorithme fonctionne en temps $\mathcal{O}(n + m)$ dans le pire cas.

(a) Nous disons qu'un jeu est *bien conçu* si: 3 pts

— il possède un (seul) niveau accessible à partir d'aucun autre (le *premier niveau*);
— il possède un (seul) niveau qui ne peut pas en atteindre d'autres (le *dernier niveau*);
— tous les niveaux sont accessibles à partir du premier niveau;
— tous les niveaux peuvent atteindre le dernier niveau.

Expliquez comment déterminer algorithmiquement si un jeu est bien conçu. Vous pouvez le faire en mots, sous forme de pseudocode, en invoquant des algorithmes vus en classes, etc.

(b) Nous disons que deux niveaux x et y appartiennent au même *monde* si x peut atteindre y et vice-versa, avec un nombre arbitraire de déplacements. Par exemple, il y a cinq mondes dans le jeu ci-dessus. En particulier, les mondes du ciel, de la cité et de la forêt contiennent respectivement deux, un et trois niveaux.

(i) Supposons qu'on ait accès à une routine qui répond, en temps constant, aux questions de cette forme: 6 pts

le niveau x peut-il atteindre le niveau y avec un nombre arbitraire de déplacements?

Donnez un algorithme qui identifie les mondes d'un jeu bien conçu. Il doit numéroter chaque niveau avec un nombre de $[1..k]$, où k est le nombre de mondes. L'ordre dans lequel ils sont attribués n'importe pas, pourvu que deux niveaux aient le même nombre ssi ils font partie du même monde.

(ii) Est-ce possible de passer d'un monde i vers un monde $j \neq i$ et d'éventuellement revenir au monde i? 3 pts
Autrement dit, le « graphe des mondes » est-il acyclique? Justifiez brièvement.

(c) **Donnez un algorithme qui détermine le nombre *maximal* d'étoiles qu'on peut obtenir dans un jeu bien conçu, où chaque monde ne contient qu'*un seul* niveau.** 8 pts

Indice: pensez à un ordre dans lequel considérer les niveaux.

★ Donnez un algorithme qui effectue la même tâche, mais cette fois sans restriction sur la taille des mondes. ★ 2,5 pts