COMPLEX - Partie 2
2022-2023

Énoncé

Nom ou numéro d'anonymat :

Durée : 2 heures

Notes manuscrites et documents de cours autorisés
L'utilisation de tout matériel électronique (en dehors d'une montre non connectée) est interdite

Les exercices sont indépendants.
Une rédaction claire et concise sera appréciée. Toute affirmation devra être justifiée.
Une question non résolue n'empêche pas de faire les suivantes
(dans ce cas indiquez clairement que vous admettez le(s) résultat(s) de la question non faite).

Exercice 1 : Listes chainées circulaires triées

Nous considérons un ensemble de n valeurs entières différentes qui sont stockées sous la forme d'un tableau trié comme dans l'exemple suivant (avec $n = 9$) :

i	1	2	3	4	5	6	7	8	9
valeur(i)	83	54	16	31	45	99	78	62	27
suivant(i)	6	8	9	5	2	3	1	7	4

Étant donné un entier i, il est possible d'accéder en temps constant à la valeur **valeur**(i) contenue dans cette case ainsi qu'à l'indice **suivant**(i) associé à la valeur suivante dans la liste. Cet indice correspond soit à la case contenant la prochaine valeur qui est supérieure à **valeur**(i), soit au minimum si **valeur**(i) est le maximum des valeurs stockées (ce qui est le cas pour $i = 6$ dans l'exemple). Cette structure de données sera appelée *liste chainée circulaire triée* et ne permet pas d'accéder directement au minimum ou au maximum des valeurs stockées.

1.a] Remplir la ligne **suivant** du tableau ci-dessous pour en faire une liste chainée circulaire triée :

i	1	2	3	4	5	6	7	8	9
valeur(i)	17	57	23	6	67	43	89	32	42
suivant(i)									

1.b] Proposer un algorithme déterministe qui étant donné une liste chainée circulaire triée $\mathcal{L}$ et un entier m, détermine si m appartient à $\mathcal{L}$. Donner sa complexité.

1.c] Proposer un algorithme probabiliste de type Las Vegas qui, étant donné une liste chainée circulaire triée $\mathcal{L}$ et un entier m, détermine si m appartient à $\mathcal{L}$ en $O(\sqrt{n})$ appels aux fonctions **valeur**(i) et **suivant**(i) en moyenne.

1.d] Décrire brièvement une variante probabiliste du tri par insertion utilisant cette structure de données qui effectue $O(n\sqrt{n})$ comparaisons et $O(n\sqrt{n})$ affectations en moyenne (les détails ne sont pas demandés).

Exercice 2 : Coupes minimales

Une *coupe* d'un graphe est une partition de l'ensemble des sommets en deux sous-ensembles non-vides (on appelle parfois aussi coupe l'ensemble des arêtes ayant une extrêmité dans chaque sous-ensemble de la partition). Étant donné un graphe non orienté $G = (V, E)$, un sous-ensemble S de V, différent de V et $\emptyset$, définit la coupe $\{S, V \setminus S\}$ et le *poids* de cette coupe, noté $c(S)$ est le nombre d'arêtes de E ayant une extrêmité à l'intérieur de cet ensemble et l'autre à l'extérieur. Le problème de la coupe minimal est le suivant : étant donné un graphe non orienté $G = (V, E)$, trouver un coupe (ou un sous-ensemble S) de poids minimum (parmi toutes les coupes de G).

2.a] En utilisant l'analyse de la probabilité de succès de l'algorithme de Karger vue en cours, montrer que le nombre de coupes minimales d'un graphe G à $n \geq 2$ sommets est inférieur ou égal à $\binom{n}{2} = n(n-1)/2$.

Indication : On pourra utiliser la *méthode probabiliste* également vue en cours.

2.b] Donner une famille infinie de graphes à n sommets (pour tout entier $n \geq 2$) qui possède $\binom{n}{2}$ coupes minimales différentes.

Exercice 3 : Algorithme probabiliste pour 3-SAT

Nous considérons l'algorithme suivant :

Algorithme 1 : Algorithme probabiliste pour 3-SAT

Entrée : Formule Φ de m clauses $C_1, \ldots, C_m$ en n variables $x_1, \ldots, x_n$; $T \in \mathbb{N}$
▷ $C_1, \ldots, C_k$ pour $k \leq m$ sont les 3-clauses de Φ
▷ $C_{k+1}, \ldots, C_m$ sont les 1-clauses et les 2-clauses de Φ

Sortie : $b \in \{\text{SATISFIABLE}, \text{NON-SATISFIABLE}\}$

pour j *de* 1 *à* T **faire**
    $\Phi' \leftarrow \{C_{k+1}, \ldots, , C_m\}$ ▷ Φ' est constituée des 1-clauses et de 2-clauses de Φ
    pour i *de* 1 *à* k **faire**
        $\ell_1^i \vee \ell_2^i \vee \ell_3^i \leftarrow C_i$
        $t \longleftarrow \{1, 2, 3\}$ ▷ supprime un littéral aléatoire à la 3-clause C_i
        si $t = 1$ **alors**
            $C_i' \leftarrow \ell_2^i \vee \ell_3^i$
        si $t = 2$ **alors**
            $C_i' \leftarrow \ell_1^i \vee \ell_3^i$
        si $t = 3$ **alors**
            $C_i' \leftarrow \ell_1^i \vee \ell_2^i$
        $\Phi' \leftarrow \Phi' \cup \{C_i'\}$ ▷ ajoute la 2-clause C_i' obtenue à Φ'
    si Φ' *est satisfiable* **alors**
        retourner SATISFIABLE ▷ en temps $O(n+m)$ car Φ' est une formule 2-SAT
retourner NON-SATISFIABLE

Rappel : Un littéral ℓ_i est une variable propositionnelle x_j (littéral positif) ou la négation d'une variable propositionnelle $\neg x_j$ (littéral négatif). Une k-clause (pour $k \in \mathbb{N} \setminus \{0\}$) est une disjonction de la forme $\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k$ où ℓ_i est un littéral pour $i \in \{1, \ldots, k\}$. Dans la suite, nous supposerons toujours que les variables apparaissant dans k littéreaux d'une k-clauses sont toutes distinctes.

Le problème 3-SAT consiste, étant donné une formule logique propositionnelle Φ donnée sous forme d'une conjonction de k-clauses avec $k \leq 3$, à décider si il existe une assignation des variables satisfaisant toutes les clauses.

3.a] Montrer que si cet algorithme retourne SATISFIABLE sur une formule booléenne Φ alors la formule Φ est effectivement satisfiable.

Nous allons désormais analyser la probabilité que l'algorithme retourne NON-SATISFIABLE sur une formule booléenne Φ satisfiable. Soit $\vec{z} \in \{0,1\}^n$ une assignation des variables qui rend la formule Φ vraie.

3.b] Montrer que pour tout $i \in \{1, \ldots, k\}$ (dans la boucle des lignes 3 à 12 de l'algorithme), l'assignation $\vec{z}$ rend C'_i vraie avec probabilité supérieure ou égale à $2/3$.

3.c] En déduire que l'assignation $\vec{z}$ rend la formule Φ' (construite à la fin de la boucle des lignes 3 à 12) vraie avec probabilité supérieure ou égale à $(2/3)^k$.

3.d] Donner un choix du paramètre T (en fonction de n, m, k) qui rend la probabilité d'erreur de cet algorithme inférieure à $1/e$. Une analyse précise de cette probabilité d'erreur et du temps d'exécution de l'algorithme obtenu est demandée.

Exercice 4 : Classe de complexité probabiliste $\mathcal{PP}$

Soit Σ un alphabet arbitraire fini (avec $\#\Sigma > 1$). Nous considérons la classe de complexité $\mathcal{PP}$ définie comme étant l'ensemble des langages $L \subseteq \Sigma^*$ pour lesquels il existe une machine de Turing probabiliste $\mathcal{M}$ telle que :

(1) $\mathcal{M}$ s'arrête sur toute entrée $x \in \Sigma^*$ et s'exécute en temps polynomial $p(|x|)$ où $|x|$ désigne la longueur de x ;

(2) pour tout $x \in L$, $\Pr[\mathcal{M}(x) = 1] > 1/2$;

(3) pour tout $x \notin L$, $\Pr[\mathcal{M}(x) = 0] > 1/2$.

4.a] Montrer que si L est un langage défini sur Σ dans $\mathcal{PP}$ et si $\mathcal{M}$ est une machine de Turing probabiliste qui vérifie les propriétés (1) et (2) précédentes, alors nous avons

(2') pour tout $x \in L$, $\Pr[\mathcal{M}(x) = 1] \geq \dfrac{1}{2} + \dfrac{1}{2^{p(|x|)}}$ où $|x|$ est la longueur de x.

4.b] Nous considérons la classe de complexité $\mathcal{PP}'$ définie comme étant l'ensemble des langages L définis sur Σ pour lesquels il existe une machine de Turing probabiliste $\mathcal{M}$ qui vérifie les propriétés (1), (2) et la propriété (3') suivante :

(3') pour tout $x \notin L$, $\Pr[\mathcal{M}(x) = 0] \geq 1/2$;

4.c] Montrer que $\mathcal{PP} \subseteq \mathcal{PP}'$

4.d] Soient L un langage de Σ^* et une machine de Turing probabiliste $\mathcal{M}$ vérifiant les propriétés (1), (2') et (3'). Considérons la machine de Turing probabiliste $\mathcal{M}'$ qui exécute $\mathcal{M}$ sur son entrée x et :

— si $\mathcal{M}$ rejette x, $\mathcal{M}'$ rejette x ;

— si $\mathcal{M}$ accepte x, $\mathcal{M}'$ rejette x avec probabilité $2^{-(p(|x|)+1)}$ et accepte x avec probabilité $1 - 2^{-(p(|x|)+1)}$.

Montrer que $\mathcal{M}'$ vérifie les propriétés (1), (2) et (3) pour le langage L.

4.e] Conclure.

4.f] En déduire que pour tout langage L de $\mathcal{PP}$, le langage $\overline{L} = \Sigma^* \setminus L$ appartient à $\mathcal{PP}$.

Les deux dernières questions sont indépendantes des précédentes. Nous considérons désormais la classe de complexité $\mathcal{PP}^+$ définie comme étant l'ensemble des langages $L \subseteq \Sigma^*$ pour lesquels il existe une machine de Turing probabiliste $\mathcal{M}$ qui vérifie les propriétés (2), (3) et la propriété (1') suivante :

(1') $\mathcal{M}$ s'arrête sur toute entrée $x \in \Sigma^*$ et s'exécute en temps polynomial **espéré** $p(|x|)$ où $|x|$ désigne la longueur de x ;

4.g] Montrer que tout langage L de $\mathcal{PP}^+$ est décidable (c'est-à-dire qu'il existe une machine de Turing déterministe qui s'arrête sur toute entrée de Σ^*, en temps fini arbitraire, qui accepte tout mot $x \in L$ et rejette tout mot $x \notin L$).

4.h] Montrer que tout langage décidable appartient à $\mathcal{PP}^+$.