COMPLEX - Partie 2
2024-2025

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 : Tableaux presque-triés

Soit T un tableau d'entiers de longueur n et $\epsilon \in [0,1]$; T est dit *ε-presque trié* si il existe une suite strictement croissante $1 \leq i_1 < i_2 < \cdots < i_\ell \leq n$ de longueur ℓ avec $\ell \geq \lfloor (1-\varepsilon)n \rfloor$ telle que le sous-tableau $[T[i_1], \ldots, T[i_\ell]]$ est trié (pour l'ordre croissant).

En particulier, un tableau 0-presque trié est un tableau trié et tout tableau d'entiers est 1-presque trié. Le tableau $[1,2,3,7,4,5,6]$ est $1/7$-presque trié mais pas 0-presque trié (le tableau complet n'est pas trié mais le sous-tableau $[1,2,3,4,5,6]$ de longueur $6 = (1-1/7)\cdot 7$ est trié par ordre croissant).

L'objectif de cet exercice est d'étudier des algorithmes probabilistes efficaces qui étant donné un tableau d'entiers T et un nombre réel ε déterminent si T est ε-presque trié ou non.

1.a] Décrire brièvement un algorithme déterministe de complexité $O(n^2)$ qui étant donné un tableau d'entiers T de longueur n et un nombre réel ε retourne 1 si T est ε-presque trié et 0 sinon.

Indication : On pourra utiliser un algorithme de programmation dynamique.

1.b] Montrer que le tableau $T = [\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \ldots, n, 1, 2, \ldots, \lfloor n/2 \rfloor]$ de longueur $n \geq 4$ n'est pas $1/10$-presque trié.

1.c] Considérons l'algorithme probabiliste qui consiste à tirer uniformément aléatoirement k positions dans le tableau (pour un entier k donné en entrée) et à tester si les trois valeurs consécutives du tableau à cet emplacement sont bien ordonnées ; l'algorithme retourne 1 si c'est le cas pour les k positions testées et 0 sinon. Plus précisément, nous considérons l'algorithme Est_Presque_Trié_1 suivant :

Algorithme 1 Est_Presque_Trié_1(T, k)

Entrée : T un tableau d'entiers de longueur n et $k \in \mathbb{N}$
Sortie : $b \in \{0, 1\}$

```
pour i de 1 à k faire
    Tirer i uniformément aléatoirement dans {2, ..., n − 1}
    si (T[i − 1] > T[i]) ou (T[i] > T[i + 1]) alors
        retourner 0
    fin si
fin pour
retourner 1
```

1. Montrer que cet algorithme retourne 0 avec probabilité $(1 - (1 - 2/(n-2))^k)$ sur le tableau T de la question précédente (avec $n \geq 4$).

2. En déduire qu'il existe une constante $c > 0$ telle que l'algorithme retourne 0 avec probabilité au moins $2/3$ si $k > c \cdot n$.

1.d] L'algorithme de la question précédente demande donc au moins $\Omega(n)$ opérations pour obtenir une réponse correcte avec une probabilité supérieure ou égale à $2/3$ pour le tableau T considéré (pour $\varepsilon = 1/10$).

Nous allons étudier un autre algorithme qui va effectuer une recherche dichotomique dans le tableau T (même si celui n'est pas nécessairement trié).

Algorithme 2 REC_DICHOTOMIQUE(T, x, d, f)

Entrée : T tableau d'entiers de longueur n, $x \in \mathbb{N}$, $d, f \in \{1, \ldots, n\}$ avec $d \leq f$
Sortie : $i \in \{d, \ldots, f\} \subseteq \{1, \ldots, n\}$

```
si d = f alors
  retourner d
sinon
  m ← ⌊(d + f)/2⌋                ▷ m est le milieu de l'intervalle d'entiers {d, ..., f}
  si x ≤ T[m] alors
    retourner REC_DICHOTOMIQUE(T, x, d, m)
  sinon
    retourner REC_DICHOTOMIQUE(T, x, m + 1, f)
  fin si
fin si
```

L'algorithme REC_DICHOTOMIQUE(T, x, d, f) retourne toujours un entier i dans l'intervalle d'entiers $\{d, \ldots, f\}$. Si le tableau T est trié et si x est un entier, l'exécution de l'algorithme REC_DICHOTOMIQUE sur l'entrée $(T, x, 1, n)$ retourne la position où insérer cet élément x dans le tableau pour qu'il reste trié. Soit T un tableau arbitraire dont tous les éléments sont supposés distincts. Soient x et y deux entiers et notons i et j les entiers retournés par REC_DICHOTOMIQUE$(T, x, 1, n)$ et REC_DICHOTOMIQUE$(T, y, 1, n)$ (respectivement). Montrer que si $i < j$, alors $x < y$.

1.e] Nous considérons maintenant l'algorithme probabiliste suivant :

Algorithme 3 EST_PRESQUE_TRIÉ_2(T, k)

Entrée : T tableau d'entiers de longueur n, $k \in \mathbb{N}$
Sortie : $b \in \{0, 1\}$

```
pour i de 1 à k faire
  Tirer i uniformément aléatoirement dans {1, ..., n}
  j ← REC_DICHOTOMIQUE(T, T[i], 1, n)
  si i ≠ j alors
    retourner 0
  fin si
fin pour
retourner 1
```

Montrer que si le tableau T (dont tous les éléments sont supposés distincts) est trié, alors l'algorithme EST_PRESQUE_TRIÉ_2 retourne 1 sur toute entrée (T, k) pour tout entier $k \in \mathbb{N}$.

1.f] Soit T un tableau d'entiers de longueur n dont tous les éléments sont supposés distincts et soit $i \in \{1, \ldots, n\}$. L'élement $T[i]$ du tableau est dit « *bon* » si l'exécution de REC_DICHOTOMIQUE$(T, T[i], 1, n)$ retourne l'entier i et « *mauvais* » dans le cas contraire.
Quels sont les éléments « *mauvais* » dans le tableau $[5, 6, 7, 8, 1, 2, 3, 4]$?

1.g] En utilisant la question **1.d**, montrer que les éléments « *bons* » d'un tableau T forment une sous-suite croissante du tableau T.

1.h] En déduire que si un tableau T n'est pas ε-presque trié, alors au moins $\lfloor \varepsilon \cdot n \rfloor$ éléments de T sont « *mauvais* ».

1.i] En déduire une minoration de la probabilité que Est_Presque_Trié_2(T, k) retourne 0 (en fonction de k et ε) si le tableau T n'est pas ε-presque trié.

Exercice 2 : Algorithme d'approximation probabiliste pour la 3-coloration maximale

En théorie des graphes, le *problème de la* 3-*coloration* est, étant donné un graphe non orienté $G = (V, E)$, de définir une fonction $c : V \to \{1, 2, 3\}$ telle que pour tout arête $\{u, v\} \in E$, $c(u) \neq c(v)$ (*i.e.* d'associer à chaque sommet une « couleur » dans l'ensemble $\{1, 2, 3\}$ de sorte que les sommets reliés par une arête soient de couleur différente).

Le *problème de la* 3-*coloration maximale* pour un graphe non-orienté $G = (V, E)$, consiste à rechercher une fonction $c : V \to \{1, 2, 3\}$ telle que l'ensemble $\{\{u, v\} \in E \,|\, c(u) \neq c(v)\}$ est de cardinal maximal parmi toutes les colorations possibles.

2.a] Considérer l'algorithme probabiliste qui consiste assigner à chaque sommet de G une couleur tirée uniformément aléatoirement dans $\{1, 2, 3\}$. Calculer l'espérance de la variable aléatoire du cardinal de l'ensemble $\{\{u, v\} \in E \,|\, c(u) \neq c(v)\}$ pour une coloration c produite par cet algorithme.

2.b] Considérons l'algorithme probabiliste qui répète l'algorithme simple de la question précédente jusqu'à ce qu'il retourne une coloration c pour laquelle $\#\{\{u, v\} \in E \,|\, c(u) \neq c(v)\} \geq (2 \cdot \#E)/3$ (et retourne alors cette coloration c).
Montrer que cet algorithme retourne une 3/2-approximation pour le problème de la 3-coloration maximale.

2.c] Calculer le temps d'exécution moyen de l'algorithme de la question **2.b**.

Exercice 3 : Classe de complexité probabiliste $\mathcal{BPP}(a, b)$

Soit Σ un alphabet arbitraire fini (avec $\#\Sigma > 1$). Soient $a, b \in [0, 1]$. Nous considérons la classe de complexité $\mathcal{BPP}(a, b)$ 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 $x \in L$, $\Pr[\mathcal{M}(x) = 1] \geq b$;

(3) pour tout $x \notin L$, $\Pr[\mathcal{M}(x) = 1] \leq a$.

La classe $\mathcal{BPP}$ vue en cours correspond donc à la classe $\mathcal{BPP}(1/3, 2/3)$.

3.a] Montrer que $\mathcal{BPP}(0,1) = \mathcal{P}$ et que $\mathcal{BPP}(0,2/3) = \mathcal{RP}$

3.b] Montrer que tout langage défini sur l'alphabet Σ appartient à $\mathcal{BPP}(1,1)$.

3.c] Montrer que tout langage défini sur l'alphabet Σ appartient à $\mathcal{BPP}(3/4,3/4)$.

3.d] Soient $a, b \in]0,1[$ avec $a < b$. Montrer que si un langage $\mathcal{L}$ appartient à $\mathcal{BPP} = \mathcal{BPP}(1/3, 2/3)$ alors il appartient à $\mathcal{BPP}(a, b)$

3.e] Soient $a, b \in]0,1[$ avec $a < b$. Montrer que si un langage $\mathcal{L}$ appartient à $\mathcal{BPP}(a, b)$ alors il appartient à $\mathcal{BPP} = \mathcal{BPP}(1/3, 2/3)$

Indication : On pourra utiliser l'inégalité de Chernoff, par exemple sous la forme suivante :
Soient $X_1, X_2, \ldots, X_n$ des variables aléatoires booléennes (i.e. à valeurs dans $\{0,1\}$) indépendantes, de même espérance p, alors pour tout $\varepsilon > 0$,

$$\Pr\left(\frac{1}{n}\sum_{i=1}^{n} X_i > p + \varepsilon\right) \leq e^{-2\varepsilon^2 n}, \text{ et } \Pr\left(\frac{1}{n}\sum_{i=1}^{n} X_i < p - \varepsilon\right) \leq e^{-2\varepsilon^2 n}.$$

Votre argument devra explicitement utiliser l'hypothèse $a < b$ (comme le montrent les questions **3.b** et **3.c**).