Examen du 10 décembre 2024

Consignes.
L'examen dure 2 heures. Aucun document n'est autorisé.
Le sujet est constitué de quatre exercices indépendants. Le barème total est sur 21 points. Toute question non résolue peut être admise dans la suite.
Les réponses doivent être justifiées et correctement rédigées.

Exercice 1 (sur 3 pts). *Plus grands éléments*
Soit T un tableau de n entiers. On souhaite trouver ses k plus grands éléments.
Exemple : si $T = [7, 1, 5, 3, 4]$ *et* $k = 2$*, on souhaite avoir* 7 *et* 5.
Les complexités dans cet exercice doivent être exprimées en fonction de n et k. Les algorithmes demandés doivent être de la meilleure complexité possible.

1. On trie T puis on renvoie ses k derniers éléments. Quelle est la complexité obtenue ?
2. On admet qu'on sait transformer T en un tas en temps $O(n)$. Décrire un algorithme pour obtenir les k plus grands éléments de T en commençant par le transformer en tas, et donner sa complexité.
3. On rappelle que l'algorithme QUICKSELECT renvoie le $i^{\text{ème}}$ plus petit élément de T en temps $O(n)$. Décrire un algorithme basé sur QUICKSELECT pour calculer les k plus grands éléments de T et donner sa complexité.

Exercice 2 (sur 3 pts). *Rendu de monnaie*
On considère un ensemble de pièces et billets. Chaque pièce ou billet a une valeur : on suppose qu'il y a n valeurs différentes $v_0 = 1 < v_1 < \cdots < v_{n-1}$. *Par exemple, les euros vont de* $v_0 = 1$ *centime, jusqu'à* v_{14} *qui vaut 500€, soit 50 000 centimes.*
Le problème est le suivant : étant donné une somme S à rendre, on veut trouver le plus petit nombre $p(S)$ de pièces et billets dont la somme soit S. Point important : il est autorisé de rendre plusieurs fois la même pièce ou billet. *Exemple : pour rendre 2€42, le mieux est une pièce de 2€, deux de 20 centimes et une de 2 centimes, donc* $p(242) = 4$ *en euros.*

1. Montrer que $p(S) = 1 + \min\{p(S - v_i) : 0 \leq i < n \wedge v_i \leq S\}$.
2. En déduire un algorithme qui étant donné S et $v_0, \ldots, v_{n-1}$, calcule $p(S)$.
3. Calculer sa complexité en temps et en espace, en fonction de n et S.

Exercice 3 (sur 8 pts). MAX-3-SAT

Rappel. Une formule 3-CNF est une conjonction de clauses, chaque clause étant une disjonction de *trois* littéraux, c'est-à-dire de trois variables ou leur négation. Exemple :

$$\varphi = (x_0 \vee \neg x_1 \vee x_2) \wedge (\neg x_0 \vee \neg x_1 \vee x_3) \wedge (x_0 \vee x_1 \vee \neg x_4) \wedge (\neg x_2 \vee x_3 \vee x_4).$$

Dans cet exercice on s'intéresse au problème MAX-3-SAT : en entrée, on dispose d'une formule 3-CNF φ ; en sortie, on cherche une affectation A des variables qui satisfasse *autant de clauses* que possible. Si la formule est satisfiable, la sortie est une affectation qui satisfait la formule ; sinon, l'affectation ne satisfait que *certaines* clauses. On va décrire un algorithme d'approximation, probabiliste, pour ce problème.

Supposons que φ soit une formule 3-CNF avec n variables $x_0, \ldots, x_{n-1}$ et m clauses $C_0, \ldots, C_{m-1}$. *Dans l'exemple, $n = 5$ et $m = 4$.* On note m_{OPT} le nombre maximal de clauses qui peuvent être satisfaites dans φ.

1. **i.** Combien vaut m_{OPT} si φ est satisfiable ?
ii. Donner une borne supérieure, dans le cas général, sur m_{OPT}.

L'algorithme proposé est le suivant : pour chaque variable x_i, on affecte $x_i = \text{VRAI}$ avec probabilité $\frac{1}{2}$, et $x_i = \text{FAUX}$ avec probabilité $\frac{1}{2}$ (les tirages aléatoires sont indépendants). Soit A l'affectation obtenue.

2. Soit C_i une clause de φ. On note Z_i la variable aléatoire qui vaut 1 si C_i n'est pas satisfaite par l'affectation et 0 sinon. Montrer que $\Pr[Z_i = 1] = \frac{1}{8}$.

3. On cherche maintenant l'espérance du nombre de clauses non satisfaites. Soit Z la variable aléatoire qui est égale au nombre de clauses non satisfaites.

i. Montrer que $\mathbb{E}[Z] = \frac{1}{8}m$. *Indication. Exprimer Z en fonction des Z_i et utiliser la linéarité de l'espérance.*
ii. Que peut-on dire du facteur d'approximation de cet algorithme ?
iii. Quelle est la probabilité que l'algorithme renvoie une affectation qui satisfasse moins de $\frac{3}{4}m$ clauses ? *Indication. Exprimer la question en fonction de Z et utiliser l'inégalité de Markov.*

On dit que l'affectation renvoyée par l'algorithme est *correcte* si $\geq \frac{7}{8}m$ clauses sont satisfaites. On *admet* que l'algorithme renvoie une affectation correcte avec probabilité $\varepsilon \geq 1/(m+1)$.

4. **i.** On répète N fois l'algorithme, et on garde la meilleure affectation (celle qui satisfait le plus de clauses). Quelle est la probabilité que cette affectation soit correcte ? *Exprimer le résultat en fonction de ε.*
ii. Proposer une version de type Las Vegas de l'algorithme, qui renvoie toujours une affectation correcte.
iii. Analyser l'espérance de la complexité de votre algorithme.

Exercice 4 (sur 7 pts). *Test de puces*

On dispose de n puces électroniques $p_0, \ldots, p_{n-1}$, toutes identiques. Certaines fonctionnent correctement, les autres sont défectueuses. On dispose d'un banc de test qui prend deux puces p_i et p_j et renvoie une valeur booléenne $\text{TEST}(p_i, p_j)$, qui donne l'information suivante :

— si $\text{TEST}(p_i, p_j) = \text{VRAI}$, soit p_i et p_j sont toutes les deux correctes, soit elles sont toutes les deux défectueuses ;
— si $\text{TEST}(p_i, p_j) = \text{FAUX}$, *au moins une* des deux puces est défectueuse.

On suppose que si on reteste deux puces déjà testées, on retrouve le même résultat à chaque fois.
L'objectif est de déterminer les puces qui fonctionnent correctement.

1. **i.** Soit p_i et p_j deux puces correctes. Que peut renvoyer $\text{TEST}(p_i, p_j)$?
ii. Soit p_i une puce correcte et p_j une puce incorrecte. Que peut renvoyer $\text{TEST}(p_i, p_j)$?
iii. Soit p_i et p_j deux puces incorrectes. Que peut renvoyer $\text{TEST}(p_i, p_j)$?

2. Supposons qu'on connaisse une puce p_i correcte. Écrire un algorithme pour déterminer le statut (correcte ou défectueuse) de chacune des autres puces, et déterminer le nombre de tests effectués.

Dans la suite, on se concentre donc sur le problème d'identifier *une puce correcte* parmi toutes les puces.

3. On adopte la stratégie suivante pour réduire le nombre de puces. On effectue tous les tests $\text{TEST}(p_{2j}, p_{2j+1})$ pour $j = 0$ à $\lfloor n/2 \rfloor$. Si $\text{TEST}(p_{2j}, p_{2j+1}) = \text{FAUX}$, on jette les puces p_{2j} et p_{2j+1}. Si $\text{TEST}(p_{2j}, p_{2j+1}) = \text{VRAI}$, on jette la puce p_{2j+1} et on garde la puce p_{2j}. On garde enfin la puce p_{n-1} si n est impair.

i. Montrer qu'après ces tests, on garde au plus $\lceil n/2 \rceil$ puces.
ii. On suppose qu'initialement, $> n/2$ puces sont correctes, et qu'on en garde $m \leq \lceil n/2 \rceil$. Montrer que parmi les m puces gardées, il y en a au moins $> m/2$ de correctes. *Indication : combien de puces correctes peuvent être jetées ?*
iii. En déduire un algorithme de type « diviser pour régner » pour trouver une puce correcte, en supposant qu'initialement il en existe $> n/2$. *Attention à bien réfléchir au cas de base.*
iv. Déterminer le nombre de TESTs effectués. *On attend une réponse asymptotique en $O(\ldots)$.*

4. Justifier que si le nombre de puces correctes est $\leq n/2$, il n'est pas possible d'en trouver une correcte à coup sûr. *Indication. On peut supposer que lorsqu'on* TEST*e deux puces défectueuses, le résultat est toujours* VRAI.