Sujet 1

Exercice 1

On considère le problème suivant `Couverture d'ensemble` :

* **Instance** : trois entiers $m, n, k \in \mathbb{N}^*$ et une collection d'ensembles $C = \{E_1, E_2, \ldots, E_n\}$ où les $E_i \subseteq [\![1, m]\!]$.
* **Question** : existe-t-il une partie de C de cardinal $\leqslant k$ qui forme une couverture de $[\![1, m]\!]$, c'est-à-dire dont l'union est égale à $[\![1, m]\!]$?

Montrer que `Couverture d'ensemble` est est NP-complet. *On pourra admettre la NP-complétude d'un problème vu en cours ou en exercices.*

Exercice 2

On admet l'existence d'un algorithme qui détermine la médiane d'un tableau de taille n en temps $\mathcal{O}(n)$. On cherche à écrire un algorithme probabiliste qui renvoie une valeur comprise entre le premier quartile et le troisième quartile d'un tableau de taille n avec une grande probabilité. On considère l'algorithme suivant :

Entrée : tableau $t = [t_0, \ldots, t_{n-1}]$.
Début algorithme
  Sélectionner au hasard $16 \ln n$ valeurs dans t.
  Renvoyer la médiane exacte de ces valeurs.

1. Décrire un algorithme de complexité logarithmique en n permettant de sélectionner au hasard $10 \log_2 n$ valeurs dans t. On autorisera la modification du tableau t si besoin.
2. En déduire la complexité temporelle de l'algorithme précédent.

On admet l'inégalité de Chernoff : si $X_1, \ldots, X_n$ sont des variables aléatoires indépendantes de Bernoulli de paramètre p et $\varepsilon > 0$, alors :

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} X_i > p + \varepsilon\right) \leqslant \mathrm{e}^{-2\varepsilon^2 n}$$

3. On note $k = 16 \ln n$. Montrer que la probabilité qu'au moins $\frac{k}{2}$ éléments parmi les k choisis au hasard dans t soient inférieurs au premier quartile est inférieure ou égale à $\mathrm{e}^{-\frac{k}{8}}$.
4. En déduire que la probabilité que l'algorithme précédent renvoie une valeur comprise entre le premier quartile et le troisième quartile est supérieure ou égale à $1 - \frac{2}{n^2}$.

Sujet 2

Exercice 1

Un flux de n valeurs défilent successivement avec les défauts suivants :

- n est trop grand pour qu'on puisse garder en mémoire toutes les valeurs déjà vues ;
- n n'est pas connu à l'avance.

Écrire un algorithme qui choisit k éléments parmi les n, de manière équiprobable. La seule génération d'aléatoire dont on dispose est la sélection d'un entier compris entre 0 et $i-1$, pour tout entier i.

Exercice 2

On considère le problème `1LIT-3SAT` :

* **Instance** : une formule φ en 3-FNC.
* **Question** : existe-t-il une valuation μ telle qu'exactement un littéral par clause de φ est évalué à « vrai » ?

Pour une clause $C = x \vee y \vee z$, on définit $\psi_C = (\overline{x} \vee a \vee b) \wedge (\overline{y} \vee c \vee d) \wedge (\overline{z} \vee e \vee f) \wedge (a \vee c \vee e)$.

- Montrer que pour une valuation μ, $\mu(C) = 1$ si et seulement s'il existe une valuation μ_C telle que μ et μ_C coïncident en x, y et z, et μ_C rend vrai exactement un littéral par clause de ψ_C.
- En déduire que `1LIT-3SAT` est NP-complet.

Sujet 3

Exercice 1

On considère le problème `ARBRE COUVRANT` :

* **Instance** : un graphe $G = (S, A, f)$ non orienté pondéré et un seuil $P \in \mathbb{R}$.
* **Question** : existe-t-il un arbre couvrant T de G tel que $f(T) \leqslant P$?

1. Justifier que `ARBRE COUVRANT` $\in$ P.

On considère le problème `ARBRE COUVRANT BORNÉ` :

* **Instance** : un graphe $G = (S, A, f)$ non orienté pondéré, un seuil $P \in \mathbb{R}$ et un entier k.
* **Question** : existe-t-il un arbre couvrant T de G tel que $f(T) \leqslant P$ et tous les sommets de T sont de degré au plus k ?

2. Montrer que `ARBRE COUVRANT BORNÉ` est NP-complet. *On pourra admettre la NP-complétude d'un problème vu en cours ou en exercices.*

Exercice 2

On considère un arbre correspondant à un graphe de jeu à deux joueurs où les coups alternent entre les deux joueurs. On suppose que les états terminaux sont gagnants pour l'un ou l'autre des joueurs (pas de match nul). On cherche à déterminer si la racine de l'arbre est gagnante pour 1 ou pour 2. On se limite au cas d'un arbre binaire où il reste k coups à jouer pour chacun des deux joueurs, en commençant par le joueur 1.

1. Quelle est la complexité temporelle en fonction de k d'un algorithme déterministe pour trouver qui est le joueur gagnant ?
2. Montrer qu'il n'est pas nécessaire de déterminer l'état de chaque nœud de l'arbre pour trouver qui est le joueur gagnant.
3. En déduire un algorithme Las Vegas simple permettant de déterminer le joueur gagnant.
4. Montrer que l'espérance du nombre de nœuds dont l'état est déterminé par l'algorithme précédent est inférieur ou égal à 3^k.
5. En déduire que la complexité temporelle de cet algorithme est en $\mathcal{O}(n^{0.8})$, où n est le nombre de nœuds.