ÉCOLE NORMALE SUPÉRIEURE DE LYON

# Concours d'admission session 2016

# Filière universitaire : Second concours

# COMPOSITION D'INFORMATIQUE

Durée : 3 heures

*L'utilisation des calculatrices n'est pas autorisée pour cette épreuve.*

⋆ ⋆ ⋆

Ce sujet comporte trois parties. Il est conseillé de traiter les questions dans l'ordre de l'énoncé. On pourra cependant aborder une question en admettant les résultats des questions précédentes. Quand une question demande d'écrire un programme, il est attendu une réponse écrite dans un langage de programmation de haut niveau d'usage largement répandu (Caml, Maple, Pascal, C, Java, Python, etc.) que l'on indiquera.

## Définitions et notations

Dans cet énoncé $n \geq 1$ est un entier et on note $[n] = \{1, \ldots, n\}$ l'ensemble des entiers de 1 à $n$. Pour un ensemble $E$, on note $\mathcal{P}(E)$ l'ensemble des parties (ou sous-ensembles) de $E$. On note $|E|$ pour le nombre d'éléments d'un ensemble $E$. On représentera un ensemble $E \subset [n]$ par un tableau `E` à valeurs dans $\{0, 1\}$ de taille $n$ avec $\texttt{E}[i] = 1$ si et seulement si $i \in E$.

L'objectif de cette épreuve est d'étudier le problème de recherche du maximum

$$\max_{E \in \mathcal{P}([n])} f(E) \tag{1}$$

d'une fonction $f : \mathcal{P}([n]) \to \mathbb{Z}$. On s'intéressera aussi au maximum sur tous les sous-ensembles de taille $k$ pour un entier $k \in [n]$ :

$$\max_{E \in \mathcal{P}([n]) \,:\, |E|=k} f(E) \ . \tag{2}$$

On notera $\binom{n}{k}$ le nombre de sous-ensembles de $[n]$ de taille $k$.

Pour calculer la complexité en temps d'un algorithme, on supposera que les opérations usuelles (affectation $\leftarrow$, comparaison $=, <, >$, addition et soustraction $+, -$) ont un coût unité. Pour les questions de complexité, on pourra donner le résultat sous la forme $O(g)$, où $g$ est une fonction des paramètres du problème.

## 1 Exemples et algorithmes d'énumération

**Question 1.** Calculer à la main $\max_{E \in \mathcal{P}([n])} f(E)$ pour les trois fonctions $f$ suivantes :

1. $f(E) = |E|$
2. $f(E) = \sum_{x \in E} (-1)^x$
3. $n = 4$ et $f(E) = \frac{1}{|E|(|E|-1)} \sum_{(x,x') \in E^2} (x - x')^2$.

**Question 2.** On définit maintenant la fonction $f(E) = \sum_{x \in E} v_x$ où $v_x \in \mathbb{Z}$ pour tout $x \in [n]$. Écrire un programme qui prend en entrée $n, k, \{v_x\}_{x \in [n]}$ et renvoie $\max_{E \in \mathcal{P}([n]) \,:\, |E|=k} f(E)$. Quelle est la complexité en temps de ce programme en fonction de $n$ et $k$ ?

**Question 3.** On s'intéresse maintenant à des algorithmes génériques qui fonctionnent pour toute fonction $f$. On suppose que l'on dispose d'un programme $f$ qui prend en entrée la représentation `E` d'un ensemble et renvoie la valeur $f(E)$. Que calcule le programme `Question3` ? On justifiera la réponse. Pour aider à la compréhension du programme, on pourra le faire tourner explicitement pour $n = 2$ et la fonction $f(E) = |E|$.

**Fonction** `Question3`
`E`$[i] \leftarrow 0$ pour tout $i \in [n]$
`E`$[0] \leftarrow 0$
`max` $\leftarrow -\infty$
$j \leftarrow n$
**Tant que** $j \geq 1$
    `curr` $\leftarrow f($`E`$)$ (a)
    **Si** `curr` $>$ `max`
        `max` $\leftarrow$ `curr`
    $j \leftarrow n$
    **Tant que** `E`$[j] = 1$
        `E`$[j] \leftarrow 0$ (b)
        $j \leftarrow j - 1$ (b)
    `E`$[j] \leftarrow 1$
**Retourner** `max`

**Question 4.** On analyse maintenant la complexité en temps du programme `Question3`.

1. Combien de fois l'instruction (a) du programme est-elle répétée ? En d'autres termes, combien d'appels à $f$ la fonction `Question3` effectue-t-elle ?
2. Combien de fois au total les instructions (b) du programme sont-elles répétées ?
3. En supposant que le coût en temps d'un appel à $f$ est égal à 1 pour tout `E`, déduire la complexité en temps de la fonction `Question3`.

**Question 5.** Soit $k$ un entier. On cherche maintenant à calculer le maximum $\max_{E \in \mathcal{P}([n]) \,:\, |E|=k} f(E)$ de $f$ sur tous les sous-ensembles de taille $k$.

Écrire un programme qui calcule cette valeur et analyser sa complexité en temps en fonction de $n$ et $k$. Comme pour la fonction `Question3`, on suppose que l'on dispose d'un programme $f$ qui prend en entrée la représentation `E` d'une ensemble et renvoie la valeur $f(E)$ et que le coût d'un appel à $f$ est 1. Lorsque $k$ est constant, on attend un algorithme de complexité polynomiale en $n$.

**Question 6.** Soit `MaxEnsemble` un algorithme qui prend en entrée $n$ et $k$ et fait des appels à une fonction $f$ inconnue. Ceci signifie que le seul moyen pour notre programme de connaître la valeur que $f$ prend sur l'ensemble $E$ est de faire un appel $f($`E`$)$ où `E` représente l'ensemble $E$. Supposons que pour toute fonction $f$, `MaxEnsemble` renvoie $\max_{E \in \mathcal{P}([n]) \,:\, |E|=k} f(E)$ sur l'entrée $(n, k)$ en faisant $r(n, k)$ appels à la fonction $f$. Quelle est la valeur minimale que peut prendre $r(n, k)$ ? On justifiera la réponse.

# 2 Algorithmes d'approximation

Nous allons maintenant considérer une classe de fonctions appelées sous-modulaires. On dit que $f : \mathcal{P}([n]) \to \mathbb{Z}$ est sous-modulaire si pour tous sous-ensembles $E$ et $F$ de $[n]$, on a

$$f(E \cup F) + f(E \cap F) \leq f(E) + f(F). \tag{3}$$

**Question 7.** Montrer que la fonction $f : \mathcal{P}([n]) \to \mathbb{Z}$ définie par $f(E) = |E|$ est sous-modulaire. Plus généralement, montrer que la fonction $f(E) = \sum_{x \in E} v_x$, où $v_x \in \mathbb{Z}$ pour tout $x \in [n]$, satisfait la propriété (3) avec égalité.

**Question 8.** Montrer que $f$ est sous-modulaire si et seulement si pour tous sous-ensembles $E, F \subset [n]$ avec $E \subset F$ et $x \in [n]$ mais $x \notin F$, on a

$$f(E \cup \{x\}) - f(E) \geq f(F \cup \{x\}) - f(F) \ . \tag{4}$$

Nous allons maintenant supposer que $f$ est une fonction sous-modulaire. De plus, on supposera que $f$ est positive, c'est-à-dire $f(E) \geq 0$ pour tout $E \subset [n]$, et croissante, c'est-à-dire $f(E) \leq f(F)$ pour tout $E \subset F$. Pour ce type de fonctions, nous cherchons maintenant un algorithme plus efficace que celui de la Question 5 pour trouver un ensemble $E$ de taille $k$ tel que $f(E)$ n'est pas trop éloignée du maximum $\max\limits_{E \in \mathcal{P}([n]) \ : \ |E|=k} f(E)$.

**Question 9.** On considère un algorithme simple (dit "glouton") pour ce problème. L'algorithme commence avec l'ensemble vide $E_0 = \emptyset$. À l'étape $i$, on rajoute un élément $x \in [n]$ qui maximise $f(E_{i-1} \cup \{x\})$. L'algorithme s'arrête après $k$ étapes et renvoie $E_k$. Écrire un programme `Glouton` pour implémenter cet algorithme glouton et analyser sa complexité en temps. Encore une fois, on supposera que l'appel à $f$ a un coût unité.

**Question 10.** Donner pour $n = 3$, un exemple de fonction sous-modulaire, positive et croissante $f$ et de valeur de $k$ pour lequel `Glouton` renvoie bien un ensemble $E_{glouton}$ de valeur $f(E_{glouton}) = \max\limits_{E \in \mathcal{P}([n]) \ : \ |E|=k} f(E)$ et un exemple pour lequel il retourne un ensemble de valeur strictement plus petite.

Dans les prochaines questions, nous allons montrer que même si l'algorithme `Glouton` ne renvoie pas toujours un ensemble de valeur $\max\limits_{E \in \mathcal{P}([n]) \ : \ |E|=k} f(E)$, il renvoie un ensemble qui a une valeur comparable.

**Question 11.** On rappelle que $E_i$ est l'ensemble obtenu par l'algorithme glouton à l'étape $i$. Soit $E_{opt}$ un ensemble de taille $k$ tel que $f(E_{opt}) = \max\limits_{E \subset [n] \ : \ |E|=k} f(E)$. Montrer que pour tout $i \in \{0, \ldots, k-1\}$,

$$f(E_{opt}) \leq f(E_i) + k(f(E_{i+1}) - f(E_i)) \ .$$

En déduire que, $f(E_{opt}) - f(E_{i+1}) \leq (1 - \frac{1}{k})(f(E_{opt}) - f(E_i))$.

**Question 12.** Conclure que l'algorithme glouton retourne un ensemble $E_{glouton}$ de taille $k$ tel que

$$f(E_{glouton}) \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) f(E_{opt}) \geq \left(1 - \frac{1}{e}\right) f(E_{opt}) \ ,$$

où $e$ est la base du logarithme népérien ($e$ vaut environ 2.718).

# 3 Applications à deux problèmes de graphes

Dans cette section, on s'intéresse à deux problèmes particuliers du type (1) et (2). Plus précisément, pour les deux prochaines questions, on considère un graphe donné par un ensemble de sommets $S = L \cup R$ avec $L$ et $R$ deux ensembles disjoints de sommets et un ensemble d'arêtes $A \subset S \times S$. On suppose que le graphe contient uniquement des arêtes allant de $L$ vers $R$ : $A \subset L \times R$. On representera un tel graphe par un tableau `G` ayant $|L|$ lignes et $|R|$ colonnes avec $\texttt{G}[l, r] = 1$ si $(l, r) \in A$ et $\texttt{G}[l, r] = 0$ si $(l, r) \notin A$. Notre objectif ici est de trouver un ensemble de sommets $E \subset L$ de taille $k$ qui couvre un maximum de sommets de $R$. Plus précisément, on note $N(E) = \{r \in R \ : \ \text{il existe } l \in E \text{ tel que } (l, r) \in A\}$. On souhaite trouver $E \subset L$ de taille $k$ tel que $N(E)$ est de taille maximale.

**Question 13.** Soit le graphe $L = [n]$ et $R = \{n+1, \ldots, 2n\}$ avec

$$A = \{(i, i+n) \ : \ i \in [n]\} \cup \{(i, i+n-1) \ : \ i \in \{2, \ldots, n\}\} \ .$$

Tracer ce graphe pour $n = 4$ et calculer $\max\limits_{E \subset L \ : \ |E|=2} |N(E)|$.

**Question 14.** Écrire un programme qui prend en entrée la représentation d'un graphe $G = (L \cup R, A)$ et un entier $k \in \{1, \ldots, |L|\}$ et renvoie un ensemble $E$ de taille $k$ tel que $|N(E)| \geq (1 - 1/e) \max_{E' \subset L : |E'| = k} |N(E')|$. Analyser la complexité en temps de cet algorithme en fonction de $|L|, |R|$ et $|A|$. On attend une complexité polynomiale en ces paramètres.

Pour les deux questions suivantes, on considère un graphe $G = (V, A)$ non-orienté avec un ensemble de sommets $V$ et un ensemble d'arêtes $A$. $G$ est représenté par un tableau `G` ayant $|V|$ lignes et $|V|$ colonnes avec $\texttt{G}[x, y] = \texttt{G}[y, x] = 1$ si $\{x, y\} \in A$ et $\texttt{G}[x, y] = \texttt{G}[y, x] = 0$ si $\{x, y\} \notin A$. On cherche à trouver une coupe maximale du graphe. Une coupe est un ensemble de sommets $E \subset V$ et on cherche à maximiser le nombre d'arêtes $\{\{x, y\} \in A \ : \ x \in E, y \notin E\}$ entre $E$ et le complémentaire de $E$ dans $V$. Par exemple, pour le graphe défini par $V = \{1, 2, 3, 4\}$ et $A = \{\{x, y\} : x \neq y \text{ et } x, y \in V\}$, une coupe maximale est donnée par $E = \{1, 2\}$ et le nombre d'arêtes entre $E$ et le complémentaire de $E$ est 4.

**Question 15.** Écrire ce problème sous la forme (1) pour une fonction $f_{coupe}$ qu'on définira. Prouver que $f_{coupe}$ est sous-modulaire mais pas croissante.

**Question 16.** Proposer un algorithme (par exemple de type glouton) en temps polynomial en $|V|$ qui trouve une coupe $E$ telle que le nombre d'arêtes traversées par la coupe vaut au moins la moitié du nombre d'arêtes pour une coupe maximale. Plus précisément, si $E_{opt}$ est une coupe optimale, alors $|\{\{x, y\} \in A \ : \ x \in E, y \notin E\}| \geq \frac{1}{2} |\{\{x, y\} \in A \ : \ x \in E_{opt}, y \notin E_{opt}\}|$.