CONCOURS COMMUN INP

ÉPREUVE MUTUALISÉE AVEC E3A-POLYTECH

ÉPREUVE SPÉCIFIQUE - FILIÈRE MP

INFORMATIQUE

Durée : 4 heures

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

RAPPEL DES CONSIGNES

- *Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.*
- *Ne pas utiliser de correcteur.*
- *Écrire le mot FIN à la fin de votre composition.*

Les calculatrices sont interdites.

Le sujet est composé de trois parties indépendantes.

Partie I - Programmation en OCaml : sélection du $(\mathbf{k}+\mathbf{1})^e$ plus petit élément

La sélection du $(k+1)^e$ plus petit élément d'une liste d'entiers L, non nécessairement triée, consiste à trouver le $(k+1)^e$ élément de la liste obtenue en triant L dans l'ordre croissant.

Par exemple, si $L = [9; 1; 2; 4; 7; 8]$ le 3^e plus petit élément de L est 4. On pourra remarquer que si la liste L est triée dans l'ordre croissant, le $(k+1)^e$ plus petit élément est l'élément de rang k dans L.

On présente un algorithme permettant de résoudre ce problème de sélection avec une complexité temporelle linéaire dans le pire cas. Celui-ci est basé sur le principe de "diviser pour régner" et sur le choix d'un bon pivot pour partager la liste en deux sous-listes.

Dans cette partie, les fonctions demandées sont à écrire en OCaml et ne doivent faire intervenir aucun trait impératif du langage (références, tableaux ou autres champs mutables ou exception par exemple).

Étant donné un réel a, on note $\lfloor a \rfloor$ le plus grand entier inférieur ou égal à a.

I.1 - Fonctions utiles

Dans cette section, on écrit des fonctions auxiliaires qui sont utiles pour la fonction principale.

Q1. Écrire une fonction récursive de signature :

```
longueur : 'a list -> int
```

et telle que `longueur l` est la longueur de la liste `l`.

Q2. Écrire une fonction récursive de signature :

```
insertion : 'a list -> 'a -> 'a list
```

et telle que `insertion l a` est la liste triée dans l'ordre croissant obtenue en ajoutant l'élément a dans la liste croissante `l`.

Q3. En déduire une fonction récursive de signature :

```
tri_insertion : 'a list -> 'a list
```

et telle que `tri_insertion l` est la liste obtenue en triant `l` dans l'ordre croissant.

Q4. Écrire une fonction récursive de signature :

```
selection_n : 'a list -> int -> 'a
```

et telle que `selection_n l n` est l'élément de rang `n` de la liste `l`.
Par exemple, `selection_n [4;2;6;4;1;15] 3` est égal à `4`.

Q5. Écrire une fonction récursive de signature :

`paquets_de_cinq : 'a list -> 'a list list`

et telle que `paquets_de_cinq l` est une liste de listes obtenue en regroupant les éléments de la liste `l` par paquets de cinq sauf éventuellement le dernier paquet qui est non vide et qui contient au plus cinq éléments. Par exemple :

- `paquets_de_cinq []` est égal à `[]`,
- `paquets_de_cinq [2;1;2;1;3]` est égal à `[[2;1;2;1;3]]`,
- `paquets_de_cinq [3;4;2;1;5;6;3]` est égal à `[[3;4;2;1;5];[6;3]]`.

Q6. Écrire une fonction récursive de signature :

`medians : 'a list list -> 'a list`

et telle que `medians l` est la liste `m` obtenue en prenant dans chaque liste l_k apparaissant dans la liste de listes `l` l'élément médian de l_k. On convient que pour une liste A dont les éléments sont exactement $a_0 \leq a_1 \leq \ldots \leq a_{n-1}$, l'élément médian désigne $a_{\left\lfloor \frac{n}{2} \right\rfloor}$.

Dans le cas où la liste L n'est pas triée, l'élément médian désigne l'élément médian de la liste obtenue en triant L par ordre croissant. Par exemple :

`medians [[3;1;5;3;2];[4;3;1];[1;3];[5;1;2;4]]` est égal à `[3;3;1;2]`.

Q7. Écrire une fonction de signature :

`partage : 'a -> 'a list -> 'a list * 'a list * int * int`

telle que `partage p l` est un quadruplet `l1,l2,n1,n2` où `l1` est la liste des éléments de `l` plus petit que `p`, `l2` est la liste des éléments de `l` strictement plus grand que `p`, `n1` et `n2` sont respectivement les longueurs de `l1` et `l2`.

I.2 - La fonction de sélection et sa complexité

On détaille la fonction de sélection :

Q8. Écrire une fonction récursive de signature :

`selection : 'a list -> int -> 'a`

telle que `selection l k` est le $(k+1)^{\text{e}}$ plus petit élément de la liste `l`. L'écriture de la fonction sera une traduction en OCaml de l'`Algorithme 1` présenté en page 4.

On cherche à déterminer la complexité en nombre de comparaisons de la fonction `selection`. Pour tout $n \in \mathbb{N}$, on note $T(n)$ le nombre maximum de comparaisons entre éléments lors d'une sélection d'un élément quelconque dans des listes L **sans répétition** de taille n.

En analysant l'`Algorithme 1`, il est possible de démontrer que :

$$\forall n \geq 55, T(n) \leq T\left(\left\lfloor \frac{n+4}{5} \right\rfloor\right) + T\left(\left\lfloor \frac{8n}{11} \right\rfloor\right) + 4n. \qquad \text{(I)}$$

Q9. En admettant la proposition (I), montrer que pour tout entier n supérieur à 1, on a :

$$T(n) \leq (200 + T(55))\, n.$$

Pour l'initialisation, on pourra remarquer que T est une fonction croissante.

Algorithme 1 - Sélection du $(k+1)^e$ plus petit élément

```
SELECTION L k :
/* L est une liste, k est un entier positif                                */
début
    n ← LONGUEUR L
    si n ≤ 5 alors
        M ← (TRI_INSERTION L)
        retourner l'élément de rang k de M
    fin
    sinon
        L_Cinq ← PAQUETS_DE_CINQ L
        M ← MEDIANS L_Cinq
        pivot ← SELECTION M ((n + 4)//5)//2
        /* L'opérateur // désigne le quotient d'entiers. Le rang ((n + 4)//5)//2
           correspond au rang du médian de la liste M                       */
        L1, L2, n1, n2 ← PARTAGE pivot L
        si k < n1 alors
            retourner SELECTION L1 k
        fin
        sinon
            retourner SELECTION L2 (k − n1)
        fin
    fin
fin
```

Partie II - Recherche d'une clique de célébrités

II.1 - Définitions et propriétés

Définition 1 (Graphe)**.** On appelle **graphe** un couple $G = (S, A)$ où S est un ensemble fini appelé ensemble des sommets et A est une partie de $S \times S$, appelée ensemble des arêtes.

On pourra remarquer que, dans cette définition de graphe, les éléments de la forme (s, s) où $s \in S$ sont des arêtes possibles.

Définition 2 (Clique)**.** Soit $G = (S, A)$ un graphe. Soit S' une partie de S. On dit que S' est une **clique** si :

$$\forall (s_1, s_2) \in S' \times S', (s_1, s_2) \in A.$$

Définition 3 (Clique de célébrités, célébrité)**.** Soient $G = (S, A)$ un graphe et C une partie de S. On dit que C est **une clique de célébrités** si C est une clique et :

$$\forall (c, s) \in C \times S, ((s, c) \in A) \wedge ((c, s) \in A \implies s \in C).$$

Un élément de l'ensemble C est alors appelé **célébrité**.

Le terme "célébrité" provient de l'interprétation suivante : l'ensemble des sommets correspond à un ensemble de personnes et une arête (s, c) représente le fait que s connaît c. Ainsi, une célébrité est connue de tous et elle connaît uniquement les autres célébrités.

Q10. Dans cette question, on pose $S = \{0, 1, 2, \ldots, 6\}$. Pour chacun des graphes suivants, préciser s'ils contiennent une clique de célébrités non vide. Dans le cas où il y en a une, l'expliciter.

1. $G_1 = (S, A_1)$ avec $A_1 = \{(1,2), (1,3), (1,5), (2,6)\}$.
2. $G_2 = (S, A_2)$ avec

$$A_2 = \left\{ \begin{array}{l} (0,3), (0,5), (1,2), (1,3), (2,2), (2,3), (3,3), \\ (4,1), (4,3), (4,5), (5,1), (5,3), (6,1), (6,3) \end{array} \right\}.$$

Q11. Soit $G = (S, A)$ un graphe quelconque. Montrer que s'il existe une clique de célébrités non vide C dans G, alors celle-ci est unique.

Dans la suite, on note C_G l'unique clique de célébrités non vide du graphe G. Dans le cas où celle-ci n'existe pas, C_G désigne alors l'ensemble vide qui est noté $\emptyset$.

Q12. Soient $G = (S, A)$ un graphe et p un sommet de G. On note $G' = (S \setminus \{p\}, A \cap (S \setminus \{p\} \times S \setminus \{p\}))$. Montrer les propositions suivantes :

a) Montrer que si $C_{G'}$ est égal à l'ensemble vide, alors $C_G \in \{\emptyset, \{p\}\}$.

b) Montrer que si $C_G \setminus \{p\} \neq \emptyset$, alors $C_{G'} = C_G \setminus \{p\}$.

c) On suppose que $C_{G'}$ n'est pas l'ensemble vide et on fixe c' un élément de $C_{G'}$.

   i) Montrer que si (p, c') n'est pas un élément de A, alors $C_G \in \{\emptyset, \{p\}\}$.

   ii) Montrer que si (c', p) n'est pas un élément de A, alors $C_G \in \{\emptyset, C_{G'}\}$.

   iii) Montrer que si (p, c') et (c', p) sont des éléments de A, alors $C_G \in \{\emptyset, \{p\} \cup C_{G'}\}$.

II.2 - Algorithmique et programmation en Python (Informatique Commune)

Dans la suite, l'ensemble des sommets est de la forme $\{0, 1, \ldots, n-1\}$ où n est un entier supérieur à 1 et un graphe $G = (S, A)$ est représenté en Python par sa liste d'adjacence que l'on note L_G et qui est définie par :

$$\texttt{[[j | } j \in S \texttt{ et } (i, j) \in A \texttt{] } i \in S\texttt{]}.$$

Par exemple, si $G = (\{0, 1, 2, 3\}, \{(0,1), (3,2), (3,1), (1,2)\})$, alors $L_G = [[1], [2], [], [1, 2]]$.

On pourra remarquer que si l'ensemble des sommets d'un graphe G est égal à $\{0, 1, \ldots, n-1\}$, alors la longueur de la liste L_G est égale à n.

Q13. Écrire une fonction Python `est_clique(L,R)` prenant en argument une liste L qui est une liste d'adjacence d'un graphe $G = (S, A)$ et une liste R sans répétition d'éléments de S et qui renvoie `True` si l'ensemble des éléments de R constitue une clique de G et `False` sinon.

Q14. On considère le graphe G ayant comme liste d'adjacence :

$$L_G = [[1, 3, 5], [0, 2], [4, 6], [2, 4, 5, 6], [2], [2, 3, 4], [2, 4, 6]].$$

Décrire l'évolution de la variable C à chaque étape de l'`Algorithme 2` décrit en page 6.

Q15. Écrire une fonction Python `Clique_possible_C(G)` prenant en argument une liste G représentant un graphe et qui renvoie la liste C construite à l'aide de l'`Algorithme 2`.

Q16. Montrer par récurrence sur le nombre de sommets que si G est un graphe où C_G est non vide, alors `Clique_possible_C(G)` est égale à C_G.

Algorithme 2 - Construction d’une clique de célébrités possibles

```
CLIQUE_POSSIBLE_C G :
début
    C ← []
    S ← [0, 1, . . . , n − 1]
    /* n est le nombre de sommet de G                                  */
    pour chaque s élément de S faire
        si C est vide alors
            Ajouter s dans C
        fin
        sinon
            c ← premier élément de C
            t ← FAUX
            /* t permet de vérifier si on a effectué certaines instructions */
            si (s,c) n’est pas une arête de G alors
                C ← [s]
                t ← VRAI
            fin
            si (c,s) n’est pas une arête de G alors
                C ← C
                t ← VRAI
            fin
            si t = FAUX alors
                Ajouter s à la fin de liste C
            fin
        fin
    fin
fin
retourner C
```

Partie III - Étude d'une famille d'automates

Dans cette partie, l'alphabet Σ désigne l'ensemble $\{0, 1\}$, le symbole ε désigne le mot vide et on rappelle que $\Sigma^\star$ désigne l'ensemble des mots sur l'alphabet Σ.

Étant donné un mot w, on rappelle que $|w|$ désigne la longueur du mot w et l'indexation des lettres de w commence par 0. La première lettre de w est donc w_0.

La notation $\mathrm{Card}\,(E)$ désigne le cardinal d'un ensemble E.

Étant donnés un entier n et un entier non nul m, la notation $n \mod m$ désigne le reste de la division euclidienne de n par m.

III.1 - Définitions

Définition 4 (Automate déterministe). Un **Automate déterministe** est un quintuplet $A = (Q, \Sigma, \delta, q_0, F)$ avec :

- Q un ensemble fini non vide appelé ensemble des états,
- Σ est un ensemble fini appelé alphabet,
- $\delta : Q \times \Sigma \to Q$ une application appelée application de transition,
- q_0 un élément de Q appelé état initial,
- F une partie de Q appelée ensemble des états finaux.

Définition 5 (Application de transition étendue aux mots). Soit $A = (Q, \Sigma, \delta, q_0, F)$ un automate déterministe.

On définit de manière récursive $\delta^\star : Q \times \Sigma^\star \to Q$ par :

$$\begin{array}{lrcl} \forall q \in Q, & \delta^\star(q, \varepsilon) & = & q, \\ \forall q \in Q, \forall a \in \Sigma, \forall w \in \Sigma^\star, & \delta^\star(q, aw) & = & \delta^\star(\delta(q, a), w). \end{array}$$

Définition 6 (Reconnaissance d'un mot par un automate). Soient $w = w_0 w_1 \ldots w_n$ un mot sur un alphabet Σ et $A = (Q, \Sigma, \delta, q_0, F)$. On dit que w est **reconnu** par l'automate A si $\delta^\star(q_0, w) \in F$.

Définition 7 (Automate $A_{k,p}$, fonction indicatrice $L_{k,p}$). Soient p et k deux entiers vérifiant $0 \leq k \leq p-1$. L'automate $A_{k,p}$ est défini par :

- $Q = \{0, 1, \ldots, p-1\} \times \{0, 1\}$,
- $\Sigma = \{0, 1\}$,
- $\forall (c, e) \in Q, \delta((c, e), 0) = ((c+1) \mod p, e)$,
- $\forall (c, e,) \in Q, \delta((c, e), 1) = \begin{cases} ((c+1) \mod p, 1-e) & \text{si } c = k \mod p, \\ ((c+1) \mod p, e) & \text{sinon.} \end{cases}$
- $q_0 = (0, 0)$,
- $F = \{0, 1, \ldots, p-1\} \times \{1\}$.

On note $L_{k,p}$ la fonction indicatrice de l'ensemble des mots reconnus par $A_{k,p}$. Soit autrement :

$$\forall u \in \Sigma^\star, L_{k,p}(u) = \begin{cases} 1 & \text{si } A_{k,p} \text{ reconnaît } u \\ 0 & \text{sinon.} \end{cases}$$

III.2 - Exemples et propriétés élémentaires des $A_{k,p}$

Q17. Soit $w \in \Sigma^\star$. Expliciter sans démonstration l'état $\delta^\star(q_0, w)$, la lecture du mot étant effectuée dans l'automate $A_{1,3}$. On pourra s'aider d'une représentation graphique de l'automate.

Dans les questions **Q18 à Q22**, p et k désignent des entiers tels que $p > 2$ et $k \in \{0, 1, \ldots, p-1\}$.

Q18. Soit $w \in \Sigma^\star$. Expliciter l'état $\delta^\star(q_0, w)$, la lecture du mot étant effectuée dans l'automate $A_{k,p}$. On ne demande pas de démonstration.

Un corollaire direct du résultat de la question **Q18** est que l'ensemble des mots reconnu par l'automate $A_{k,p}$ est égal à :

$$\left\{w \in \Sigma^\star \,\middle|\, \mathrm{Card}\left(\left\{m \in \mathbb{N} \,\middle|\, pm + k \leq |w| - 1 \text{ et } w_{pm+k} = 1\right\}\right) \text{ est impair}\right\}.$$

Dans la suite du problème, on admet ce résultat.

Q19. Soit w un mot reconnu par un automate $A_{k,p}$. Montrer que w est reconnu par au moins un autre automate parmi $A_{0,2}, A_{1,2}, A_{l,p}$ avec $l \neq k$.

Définition 8 (Ou exclusif étendu aux mots binaires)**.** On rappelle que le **Ou exclusif** qu'on note $\oplus$ est une opération définie sur $\{0, 1\}$ par :

$$0 \oplus 0 = 1 \oplus 1 = 0 \text{ et } 0 \oplus 1 = 1 \oplus 0 = 1.$$

Soient $n \in \mathbb{N}, u$ et v deux éléments de $\{0, 1\}^n$. On définit le **Ou exclusif** de u et v, noté $u \oplus v$, le mot de longueur n défini par :

$$\forall i \in \{0, 1, \ldots, n-1\}, (u \oplus v)_i = u_i \oplus v_i.$$

Q20. Soient u et v deux mots de $\Sigma^\star$ de même longueur. Montrer que :

$$L_{k,p}(u \oplus v) = L_{k,p}(u) \oplus L_{k,p}(v).$$

Q21. Soit w un mot binaire vérifiant :

$$L_{0,2}(w) = L_{1,2}(w) = 0 \text{ et } \forall k \in \{1, 2, \ldots, p-1\}, L_{k,p}(w) = 0.$$

a) Montrer que $L_{0,p}(w) = 0$.

b) En déduire que pour tout mot $w' \in 0^\star \cdot w$, on a :

$$L_{0,2}(w') = L_{1,2}(w') = 0 \text{ et } \forall k \in \{1, 2, \ldots, p-1\}, L_{k,p}(w') = 0.$$

Q22. Montrer que pour tout $w \in \Sigma^\star$ et $w' \in w \cdot 0^\star$, on a $L_{k,p}(w) = L_{k,p}(w')$.

Remarque. *Ces égalités permettent la construction d'une relation d'équivalence sur les mots qui est utilisée pour montrer que deux mots de longueur N peuvent être séparés par un automate de la forme $A_{k,p}$ ayant $O\left(\sqrt{N}\ln(N)\right)$ états.*

FIN

IMPRIMERIE NATIONALE – 23 1130 – D'après documents fournis