ÉCOLE NORMALE SUPÉRIEURE DE LYON

Concours d'admission session 2019

Filière universitaire : Second concours

COMPOSITION D'INFORMATIQUE

Durée : 3 heures

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

⋆ ⋆ ⋆

Ce sujet comporte quatre parties. Il est recommandé de lire l'ensemble du sujet avant de commencer la rédaction. Il est également conseillé de traiter les questions dans l'ordre de l'énoncé. On pourra cependant aborder une question en admettant les résultats des questions précédentes. Les algorithmes demandés seront écrits dans un langage de programmation au choix du candidat ou en pseudo-code.

Préliminaires et définitions

On s'intéresse dans ce sujet à deux problèmes : calculer le **plus proche ancêtre commun** dans un arbre binaire ainsi que le **minimum sur un intervalle** dans un tableau. On commence par définir ces deux problèmes.

Plus proche ancêtre commun. On considère un arbre binaire A dont les nœuds sont définis de la façon récursive suivante : un nœud x est soit une feuille de l'arbre, soit un nœud interne, muni de deux nœuds fils (gauche et droit). On note n le nombre de nœuds de l'arbre. Les nœuds sont numérotés de 0 à $n-1$. On dispose des types `arbre` et `nœud` ainsi que des fonctions suivantes, définies pour x de type `nœud` et A de type `arbre` :

- `racine`(A) : renvoie le nœud racine de l'arbre A ;
- `est_feuille`(x, A) : renvoie vrai si et seulement si x est une feuille de A,
- `fils_gauche`(x, A), `fils_droit`(x, A) : renvoient les fils de x dans A (si x n'est pas une feuille),
- `est_racine`(x, A) qui renvoie vrai si et seulement si x est la racine de l'arbre A,
- `père`(x, A) qui renvoie le père de x dans l'arbre A, si x n'est pas la racine.

On appelle descendants de x, l'ensemble des fils de x, des fils des fils de x, etc. On dit que x est un ancêtre de y si et seulement si y est un descendant de x. On appelle **Plus Proche Ancêtre Commun** de x et y dans l'arbre A (noté $PPAC(x, y, A)$) le nœud z qui est ancêtre à la fois de x et de y et dont aucun fils n'est un ancêtre à la fois de x et de y. Sur l'arbre binaire de la figure 1, les ancêtres de 14 sont les nœuds 10, 5, 2 et 0, et le PPAC de 12 et 14 est 2.

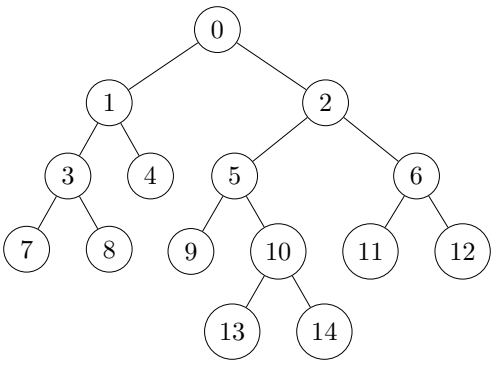


FIGURE 1 – Exemple d'arbre binaire

Minimum sur un intervalle. On considère des tableaux d'entiers, indexés à partir de 0. Pour un tel tableau `tab` de taille m, et pour deux indices i, j tels que $0 \leqslant i \leqslant j \leqslant m-1$, le **Minimum du Tableau sur l'Intervalle** $[i, j]$ (noté $MTI(\texttt{tab}, i, j)$) est égal à l'**indice** de la plus petite valeur du tableau sur cet intervalle, c'est-à-dire

$$\min_{i \leqslant k \leqslant j} \texttt{tab}[k] = \texttt{tab}[MTI(\texttt{tab}, i, j)]].$$

S'il existe plusieurs éléments de valeur minimale dans cet intervalle, MTI est égal au plus petit indice d'un tel élément. Lorsque le tableau sera clair par le contexte, on pourra noter simplement $MTI(i, j)$. On notera également $\texttt{tab}[i, j]$ le tableau de taille $j-i$ des éléments $\texttt{tab}[i], \ldots, \texttt{tab}[j]$.

Par exemple, pour le tableau $\texttt{tab} = [1,\ 2,\ -1,\ 10,\ -2,\ 4,\ -2]$, on a $MTI(\texttt{tab}, 1, 3) = 2$, correspondant à l'élément -1, et $MTI(\texttt{tab}, 0, 6) = 4$, correspondant à l'élément -2.

Complexité. Par complexité en temps d'un algorithme $\mathcal{A}$, on entend le nombre d'opérations élémentaires (comparaison, addition, soustraction, multiplication, division, affectation, test, etc) nécessaires à l'exécution de $\mathcal{A}$ dans le cas le pire. Le logarithme et l'exponentielle sont considérés comme des opérations élémentaires.

On rappelle la définition de la notation $O(\cdot)$: lorsque la complexité dépend d'un paramètre n, on dit que $\mathcal{A}$ a une complexité en $O(f(n))$ lorsqu'il existe deux constantes k et n_0 telles que la complexité de A est inférieure ou égale à $k \times f(n)$ pour tout $n \geqslant n_0$.

Partie 1 Algorithmes directs pour MTI et PPAC

Question 1. On considère un tableau `tab` de taille m et deux indices i, j tels que $0 \leqslant i \leqslant j \leqslant m-1$.

(a) Donnez un algorithme `MTI_lin` calculant $MTI(i,j)$ (c'est-à-dire $MTI(\texttt{tab}, i, j)$) et de complexité linéaire en $j - i$.

(b) On dispose d'un second couple d'indices i', j' (avec $0 \leqslant i' \leqslant j' \leqslant m-1$) tel que les deux intervalles s'intersectent : plus précisément, $i \leqslant i' \leqslant j \leqslant j'$. Expliquer comment calculer $MTI(i,j)$ et $MTI(i',j')$ avec un minimum de comparaisons entre les éléments du tableau.

Question 2. On considère l'algorithme `PPAC_naif` suivant, qui utilise la fonction `est_ancêtre`.

```
PPAC_naif(x, y, A) :
si x = y ou est_ancêtre(x, y, A) alors
    retourner x
sinon
    retourner
    PPAC_naif(père(x, A), y, A)
```

```
est_ancêtre(x, y, A) :
si est_racine(y, A) alors
    retourner faux
sinon
    z ← père(y, A)
    si z = x alors
        retourner vrai
    sinon
        retourner est_ancêtre(x, z, A)
```

Justifier la terminaison de `est_ancêtre` et `PPAC_naif`, et montrer que ce dernier calcule bien le PPAC. Donner leur complexité dans le pire cas en fonction du nombre de nœuds n de l'arbre.

Question 3. On considère deux nœuds x et y d'un arbre A.

(a) On suppose que l'on dispose d'un tableau `marquage` tel que `marquage`$[z] = 1$ si z est un ancêtre de x dans A, et 0 sinon. Donner une caractérisation du PPAC de x et y utilisant ce tableau.

(b) En déduire un algorithme `PPAC_lin`(x, y, A) calculant le PPAC de x et y dans A et de complexité linéaire en n.

Partie 2 Résoudre PPAC en utilisant MTI

On appelle **parcours Eulérien** d'un arbre A le tableau composé de tous les nœuds rencontrés lors d'un parcours en profondeur ; en particulier, un nœud interne apparaît plusieurs fois : une première fois lors de la première visite de ce nœud puis après la visite de chacun de ses fils. Le parcours Eulérien de l'arbre binaire de la figure 1 est le tableau suivant, où l'on a mis en évidence les nœuds apparaissant lors du parcours du sous arbre enraciné en 5.

$$E = [0,\ 1,\ 3, 7, 3, 8, 3,\ 1,\ 4,\ 1,\ 0,\ 2,\ \underbrace{5, 9, 5, 10, 13, 10, 14, 10, 5}_{\text{parcours du sous-arbre enraciné en 5}},\ 2, 6, 11, 6, 12, 6,\ 2,\ 0]$$

Question 4. Donner la taille du parcours Eulérien d'un arbre A en fonction du nombre n de nœuds de A. On ne supposera pas que A est un arbre binaire : chaque nœud interne peut avoir un nombre quelconque de fils.

On appelle **représentant** du nœud x de l'arbre A dans son parcours Eulérien E l'indice de la première occurence de x dans E, que l'on note $R(x)$. Par exemple, le représentant de 5 dans l'arbre de la figure 1 est 12. On appelle profondeur d'un nœud et on note *prof* sa distance à la racine. Par exemple, ce même nœud 5 a une profondeur 2. On note $PE(A)$ le tableau des profondeurs du parcours Eulérien d'un arbre A :

$$PE(A)[i] = prof(E[i])$$

où E est le parcours Eulérien de A.

Question 5. Étant donné un arbre A, son parcours Eulérien E et deux nœuds x et y de A, montrer que

$$E[MTI(PE(A), R(x), R(y))] = PPAC(x, y, A).$$

Question 6. On suppose que l'on dispose d'un algorithme `MTI_ct` pour résoudre MTI en temps constant (donc de complexité $O(1)$). On cherche à calculer le PPAC de N couples de nœuds sur le même arbre A : $PPAC(x_1, y_1, A), \ldots, PPAC(x_N, y_N, A)$. Donner un algorithme permettant de résoudre ce problème (en utilisant `MTI_ct`) et donner sa complexité.

Partie 3 Série de calculs de MTI

Dans la suite de ce sujet, on suppose que l'on doit effectuer de nombreux calculs de MTI **sur un même tableau**. On va découper le calcul en deux étapes :

Étape 1 : un pré-traitement, exécuté une seule fois, qui ne dépend que du tableau considéré et qui génère des données temporaires ;

Étape 2 : un calcul, exécuté pour chaque intervalle $[i, j]$ demandé, qui dépend donc de ces indices et s'appuie sur les données temporaires générées par l'étape 1.

Pour correctement juger l'efficacité de nos algorithmes, on notera désormais $\langle f(m), t(m), g(m)\rangle$ la complexité totale des solutions proposées, où :

- m est la taille du tableau ;
- $f(m)$ est la complexité de l'étape 1 ;
- $t(m)$ est la taille des données temporaires produite par l'étape 1 ;
- $g(m)$ est la complexité de l'étape 2.

L'algorithme suivant décrit un exemple simple d'une telle stratégie à deux étapes :

Étape 1 Calculer à l'aide de l'algorithme `MTI_lin` de la question 1(a) tous les $MTI(i,j)$ possibles (pour $1 \leqslant i \leqslant j \leqslant m$) et les stocker dans un tableau à deux dimensions de taille m^2, noté `precalcul_MTI`.

Étape 2 Pour chaque calcul $MTI(i,j)$ demandé, renvoyer l'élément `precalcul_MTI`$[i][j]$.

Cette solution est de complexité totale $\langle O(m^3), O(m^2), O(1)\rangle$.

Question 7. On souhaite améliorer la complexité de l'étape 1 de la stratégie ci-dessus.

(a) Donner une expression de $MTI(i,j+1)$ en fonction de $MTI(i,j)$.

(b) En déduire une stratégie à deux étapes et donner sa complexité totale.

Question 8. On fixe un entier K et on suppose m divisible par K. On découpe le tableau initial `tab` en K tableaux de taille m/K, appelés blocs :

$$B_0 = \texttt{tab}[0, m/K-1], \quad B_1 = \texttt{tab}[m/K, 2m/K-1], \quad \ldots \quad B_{K-1} = \texttt{tab}[(K-1)m/K-1, m-1]$$

On décide de pré-calculer dans l'étape 1 l'indice du minimum de chaque bloc avec l'algorithme `MTI_lin` de la question 1(a). On stocke le résultat dans le tableau `min_bloc_ind`, ainsi que le minimum associé dans le tableau `min_bloc` :

$$\begin{aligned} \texttt{min\_bloc\_ind}[k] &= MTI(\texttt{tab}, km/K, (k+1)m/K - 1) \\ \texttt{min\_bloc}[k] &= \texttt{tab}[\texttt{min\_bloc\_ind}[k]] \end{aligned}$$

(a) Donner un algorithme qui utilise cette information pour calculer $MTI(\texttt{tab}, i, j)$ avec des indices i, j quelconques (étape 2).

(b) En déduire la complexité totale de cette méthode (sous la forme $\langle f(m), t(m), g(m)\rangle$).

(c) Trouver la valeur de K qui optimise les complexités en calcul $f(m)$ et $g(m)$.

Question 9. (a) Donner une solution de complexité totale $\langle O(m), O(m), O(1)\rangle$ pour calculer des séries de MTI sur des intervalles de type $[0, i]$ ou $[i, m-1]$.

(b) Détailler comment cette solution peut accélérer l'étape 2 de la solution proposée à la question précédente lorsque $i - j > K$.

(c) En supposant que tous les intervalles $[i, j]$ $(0 \leqslant i \leqslant j \leqslant m-1)$ ont la même probabilité d'apparaître dans la série de calcul de MTI, donner l'espérance de la complexité totale de cette solution.

Question 10. On souhaite maintenant développer une autre approche n'utilisant pas le découpage par blocs. On décide de pré-calculer, dans l'étape 1, le MTI de tous les intervalles dont la longueur est une puissance de deux. Plus précisément, on veut calculer `MTI2`$(i,k) = MTI(i, i + 2^k - 1)$ pour $k = 1, \ldots \log_2 m$ et $i = 0, \ldots, m - 2^k$.

(a) Donner un algorithme efficace pour cette étape 1.

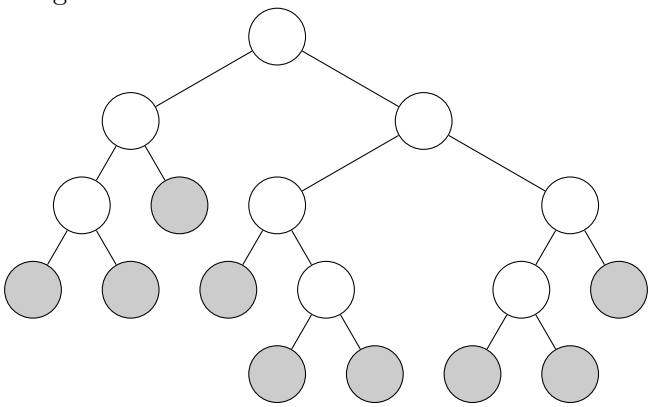
(b) Expliquer comment utiliser cette information dans l'étape 2 et donner la complexité totale de la solution obtenue.

Partie 4 Résoudre MTI en utilisant PPAC

On définit l'**arbre cartésien** d'un tableau `tab` de m entiers positifs distincts entre eux de la façon suivante :

— la racine de l'arbre porte le numéro k qui est l'indice du minimum du tableau `tab` (donc $k = MTI(0, m-1)$),
— le fils gauche de l'arbre est l'arbre cartésien du tableau $\texttt{tab}[0, k-1]$,
— le fils droit de l'arbre est l'arbre cartésien du tableau $\texttt{tab}[k+1, m-1]$,
— si le tableau est vide ($m = 0$), l'arbre est un nœud feuille, numéroté par un entier $l \geqslant m$.

Question 11. Trouver un tableau `tab1` dont l'arbre cartésien a la même structure que l'arbre ci-dessous. On dessinera également cet arbre cartésien avec tous les numéros des nœuds.



Question 12. Soit `tab` un tableau et A son arbre cartésien Montrer que $MTI(\texttt{tab}, i, j) = PPAC(i, j, A)$.

Question 13. Soit C_i l'arbre cartésien de $\texttt{tab}[0, i]$. Donner un algorithme pour calculer C_{i+1} en fonction de C_i ainsi que sa complexité.

On considère le problème d'une série de calculs de PPAC sur un même arbre. Comme on l'a fait pour le problème MTI, on procède en deux étapes :

Étape 1 : un prétraitrement sur l'arbre, effectué une seule fois, qui génère des données temporaires.

Étape 2 : un calcul exécuté pour chaque requête de PPAC, qui dépend des nœuds de la requête.

On adopte la même notation pour la complexité totale d'une série de PPAC que pour celle d'une série de MTI.

Question 14. On suppose qu'on dispose d'une solution de complexité totale $\langle O(m), O(m), 1\rangle$ pour le problème d'une série de calculs de PPAC sur un même arbre. Proposer une solution de même complexité totale pour une série de calculs de MTI sur un même tableau. On pourra adapter l'algorithme de la question précédente et raffiner son analyse de complexité.

⋆ ⋆ ⋆