Plus petit ancêtre commun dans des arbres

Épreuve pratique d'algorithmique et de programmation
Concours commun des Écoles normales supérieures

Durée de l'épreuve: 3 heures 30 minutes

Juin/Juillet 2019

ATTENTION !

N'oubliez en aucun cas de recopier votre u_0
à l'emplacement prévu sur votre fiche réponse

Important.

Il vous a été donné un numéro u_0 qui servira d'entrée à vos programmes. Les réponses attendues sont généralement courtes et doivent être données sur la fiche réponse fournie à la fin du sujet. À la fin du sujet, vous trouverez en fait deux fiches réponses. La première est un exemple des réponses attendues pour un $\widetilde{u_0}$ particulier (précisé sur cette même fiche et que nous notons avec un tilde pour éviter toute confusion!). Cette fiche est destinée à vous aider à vérifier le résultat de vos programmes en les testant avec $\widetilde{u_0}$ au lieu de u_0. Vous indiquerez vos réponses (correspondant à votre u_0) sur la seconde et vous la remettrez à l'examinateur à la fin de l'épreuve.

En ce qui concerne la partie orale de l'examen, lorsque la description d'un algorithme est demandée, vous devez présenter son fonctionnement de façon schématique, courte et précise. Vous ne devez en aucun cas recopier le code de vos procédures!

Quand on demande la complexité en temps ou en mémoire d'un algorithme en fonction d'un paramètre n, on demande l'ordre de grandeur en fonction du paramètre, par exemple: $O(n^2)$, $O(n \log n)$,...

Il est recommandé de commencer par lancer vos programmes sur de petites valeurs des paramètres et de ***tester vos programmes sur des petits exemples que vous aurez résolus préalablement à la main ou bien à l'aide de la fiche réponse type fournie en annexe***.

Enfin, il est recommandé de lire l'intégralité du sujet avant de commencer afin d'effectuer les bons choix de structures de données dès le début.

L'objectif de ce sujet est de calculer efficacement les plus petits ancêtres communs de paires de sommets dans un arbre. Après des préliminaires, des algorithmes de plus en plus poussés sont étudiés au fur et à mesure des exercices, de difficulté croissante.
Pour chaque calcul demandé, avec le bon choix d'algorithme le calcul ne devrait demander qu'au plus quelques secondes, jamais plus d'une minute.

1 Préliminaires

On rappelle que pour deux entiers naturels a et b, $a \bmod b$ désigne le reste de la division entière de a par b.
Dans tout le sujet les listes sont indexées à partir de 0.

1.1 Suite de nombres pseudo-aléatoires

On fixe $M = 2^{31} - 1 = 2\,147\,483\,647$.
On définit la suite $\overline{u}(n)$ par récurrence :

$$\overline{u}(0) = u_0, \quad \forall n \in \mathbb{N}, \overline{u}(n+1) = (16\,807 \times \overline{u}(n) + 17) \bmod M$$

u_0 vous ayant été donné, et doit être reporté sur votre fiche réponse.
On définit alors la suite

$$u(n) = \overline{u}(n \bmod 999\,983)$$

Question 1 *Calculer* $u(n) \bmod 997$ *pour*
a) $n = 16$, ***b)*** $n = 1024$, ***c)*** $n = 1\,000\,000$.

1.2 Somme de contrôle de listes de nombres

On définit l'opérateur de somme de contrôle :

$$C(x, y) = (2 \times x + y) \bmod 997$$

On définit la *somme de contrôle* d'une liste de nombres par récurrence :

$$C([]) = 0, \quad C([x_1, x_2, ..., x_{n+1}]) = C(C([x_1, x_2, ..., x_n]), x_{n+1})$$

Question 2 *Calculer*
a) $C(u(16), u(17))$, ***b)*** $C([u(100), u(101), ..., u(1\,000)])$, ***c)*** $C([u(200), u(201), ..., u(1\,000\,000)])$.

1.3 Indexation de listes de nombres contigus

On définit une liste $D_n^m(i)$ $(i \in [\![0, m-1]\!])$ ainsi :

$$D_n^m(0) = 0, \quad \forall i \in [\![0, m-2]\!], D_n^m(i+1) = \begin{cases} D_n^m(i) - 1 \text{ si } u(n+i) \text{ est pair,} \\ D_n^m(i) + 1 \text{ sinon} \end{cases}$$

Par exemple, pour $u_0 = \widetilde{u_0}$, on obtient $\widetilde{D_{16}^4} = [0, -1, -2, -1]$
Ces listes seront utilisées plus tard à la section 3.

Plus généralement, pour une liste $L(i)$ $(i \in [\![0, m-1]\!])$ de nombres telle que

$$\forall i \in [\![0, m-2]\!], L(i+1) - L(i) \in \{-1, 1\} \quad (1)$$

on définit son *index* $I(L)$ ainsi :

$$I(L) = \sum_{i=0}^{m-2} \left(\frac{L(i+1) - L(i) + 1}{2} \right) \times 2^i$$

qui caractérise ainsi la liste, à sa première valeur $L(0)$ près.
Par exemple, $I\left(\widetilde{D_{16}^4}\right) = 4$.

Question à développer pendant l'oral 1 *Montrer que pour un m fixé, I est injectif sur l'ensemble des listes L de longueur m vérifiant la propriété (1) et $L(0) = 0$, et donner l'image de cet ensemble par I.*

Question 3 *Calculer*
a) $I(D_{100}^{10}) \bmod 997$, ***b)*** $I(D_{200}^{32}) \bmod 997$.

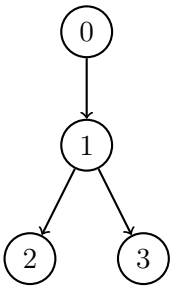
1.4 Parcours d'arbres

On fabrique des arbres T_n^m dont les m nœuds sont d'arité au plus 2 ainsi :

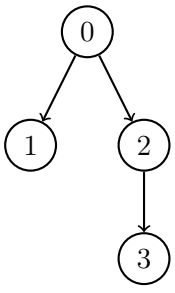
$T_n^m = T_{n,0}^m$, où
- La racine de $T_{n,i}^m$ a pour étiquette i.
- Si $m = 1$, $T_{n,i}^m$ n'a pas de sous-arbre fils.
- Sinon, si $m = 2$ ou $u(n)$ est divisible par 3, $T_{n,i}^m$ a comme seul sous-arbre fils $T_{n+1,i+1}^{m-1}$.
- Sinon, $T_{n,i}^m$ a comme sous-arbres fils gauche et droite respectifs

$$T_{n+1,i+1}^{\lfloor \frac{m-1}{2} \rfloor} \text{ et } T_{n+1+\lfloor \frac{m-1}{2} \rfloor, i+1+\lfloor \frac{m-1}{2} \rfloor}^{\lceil \frac{m-1}{2} \rceil}$$

Ainsi, par construction, les étiquettes des m nœuds de l'arbre T_n^m sont les nombres de 0 à $m-1$.
Par exemple, pour $u_0 = \widetilde{u_0}$, on obtient les arbres



$\widetilde{T_{16}^4}$



$\widetilde{T_{17}^4}$

Le *chemin d'Euler* d'un arbre T de taille m est la liste $E(T) = [v_1, ..., v_{2m-1}]$ des étiquettes des nœuds de l'arbre rencontrés lors d'un parcours en profondeur de gauche à droite. La liste commence et termine ainsi par la racine, et l'on parcourt les feuilles une seule fois, les nœuds ayant un sous-arbre deux fois, et les nœuds ayant deux sous-arbres trois fois. Par exemple, le chemin d'Euler de l'arbre $\widetilde{T_{16}^4}$ est la liste $E\left(\widetilde{T_{16}^4}\right) = [0, 1, 2, 1, 3, 1, 0]$

La *hauteur* d'un nœud est sa distance à la racine de l'arbre, la racine étant donc à la hauteur 0, ses fils à la hauteur 1, etc. Pour un arbre T de taille m, on note les hauteurs des nœuds rangés dans l'ordre du chemin d'Euler $H(T) = [h_1, ..., h_{2m-1}]$, ainsi $H(T)(i)$ est la hauteur du nœud d'étiquette $E(T)(i)$. Par exemple, $H\left(\widetilde{T_{16}^4}\right) = [0, 1, 2, 1, 2, 1, 0]$.

Question 4 *Calculer*
a) $C(H(T_{16}^4))$, ***b)*** $C(H(T_{100}^{1\,000}))$, ***c)*** $C(H(T_{200}^{10\,000}))$.

On définit la liste $R(T)(v)$ ($v \in [\![0, m-1]\!]$) ainsi : $R(T)(v)$ est l'indice de la première apparition du nœud d'étiquette v dans le chemin d'Euler $E(T)$ (son *représentant*, et l'on a $E(T)(R(T)(v)) = v$). Par exemple, $R\left(\widetilde{T_{16}^4}\right) = [0, 1, 2, 4]$.

Question 5 *Calculer*
a) $C(R(T_{16}^4))$, ***b)*** $C(R(T_{100}^{1\,000}))$, ***c)*** $C(R(T_{200}^{10\,000}))$.

Question à développer pendant l'oral 2 *Expliquer l'algorithme utilisé pour calculer $R(T)$, et évaluer sa complexité en temps en fonction de la taille de T.*

2 Plus petit ancêtre commun (LCA) et minimum d'intervalle (RMQ)

Pour des nœuds u et v d'un arbre T, on appelle *plus petit ancêtre commun (Lowest Common Ancestor)* $LCA(T, u, v)$ le nœud qui est à la fois ancêtre de u et de v et qui n'a pas de descendant qui soit à la fois ancêtre de u et de v (il est donc de hauteur maximum parmi les ancêtres communs à u et v).

Question à développer pendant l'oral 3 *Montrer pourquoi dans le cas d'un arbre le LCA est unique, alors que dans une généralisation aux graphes orientés acycliques, il ne le serait pas forcément, voire n'existe pas forcément.*

Un calcul rapide de LCA permet par exemple de calculer efficacement des distances entre les nœuds d'un arbre, il est également utilisé en bio-informatique ou en algorithmique du texte. Il est courant d'avoir à calculer de nombreuses valeurs de $LCA(T, u, v)$ pour un même arbre T, divers algorithmes ont été proposés pour optimiser ce cas, certaines approches reposent sur une réduction à un autre problème appelé RMQ.

Pour une liste de nombres L et des indices $i \leq j$, on appelle *minimum d'intervalle (Range Minimum Query)* $RMQ(L, i, j)$ l'indice de la première apparition dans L du plus petit élément parmi ceux d'indices compris entre i et j inclus. Autrement dit,

$$\forall k \in [\![i, RMQ(L, i, j) - 1]\!], L(k) > L(RMQ(L, i, j))$$

$$\forall k \in [\![RMQ(L, i, j) + 1, j]\!], L(k) \geq L(RMQ(L, i, j))$$

Pour $i > j$, on pose $RMQ(L, i, j) = RMQ(L, j, i)$.

Par exemple, $RMQ([0, 1, 0, 1, 2, 1, 0], 1, 6) = 2$, correspondant à la première apparition de l'élément 0 entre les indices 1 et 6, 0 étant le minimum des éléments apparaissant entre les indices 1 et 6.

Question à développer pendant l'oral 4 *Montrer pourquoi la recherche de $LCA(T, u, v)$ se réduit à la recherche de $RMQ(H(T), R(T)(u), R(T)(v))$.*

On note pour piocher des indices pour une liste L : $U_n(L, x) = u(n + x) \bmod len(L)$, où $len(L)$ désigne la longueur de la liste L.
Par exemple, pour $u_0 = \widetilde{u_0}$, $\widetilde{U_{16}}\left(H\left(\widetilde{T_{16}^4}\right), 0\right) = 6$, $\widetilde{U_{16}}\left(H\left(\widetilde{T_{16}^4}\right), 1\right) = 6$, $\widetilde{U_{16}}\left(H\left(\widetilde{T_{16}^4}\right), 2\right) = 0$, etc.
On note pour piocher aléatoirement des calculs de RMQ pour une liste L :

$$\begin{aligned} CRMQ(L, n, k) = C\,([\quad & RMQ(L, U_n(L, 0), U_n(L, 1)), \\ & RMQ(L, U_n(L, 2), U_n(L, 3)), \\ & \ldots, \\ & RMQ(L, U_n(L, 2 \times (k-1)), U_n(L, 2 \times (k-1) + 1)) \quad]) \end{aligned}$$

2.1 Version naïve

Une recherche naïve de $RMQ(L, i, j)$ consiste à parcourir la tranche $i...j$ de la liste L. Ainsi, par exemple pour $u_0 = \widetilde{u_0}$, $RMQ\left(H\left(\widetilde{T_{16}^4}\right), 2, 4\right) = 3$.

Question 6 *Calculer ainsi*
a) $CRMQ(H(T_{16}^4), 16, 4)$, ***b)*** $CRMQ(H(T_{16}^4), 16, 100)$, ***c)*** $CRMQ(H(T_{100}^{1\,000}), 100, 10\,000)$.

On veut cependant potentiellement calculer les valeurs de $RMQ(L, i, j)$ pour toutes les paires i, j possibles.

Question à développer pendant l'oral 5 *Montrer la complexité totale d'un tel calcul avec cette version naïve.*

2.2 Première version raffinée

En pratique on veut en général effectuer un grand nombre de calculs de valeurs de $RMQ(L, i, j)$, voire tous les calculs pour toutes les valeurs de i et j, pour une même liste L.

Question à développer pendant l'oral 6 *Établir une relation de récurrence entre $RMQ(L, i, j + 1)$ et $RMQ(L, i, j)$.*

Utiliser cette relation pour écrire un algorithme plus efficace que la recherche naïve pour un grand nombre de calculs sur une même liste L.

Question 7 *Calculer en utilisant cette méthode*

a) $CRMQ(H(T_{100}^{1\,000}), 100, 1\,000\,000)$,

b) $CRMQ(H(T_{101}^{1\,000}), 101, 1\,000\,000)$.

Question à développer pendant l'oral 7 *Évaluer la complexité en temps et en espace de cette méthode pour le coût total du calcul des valeurs de $RMQ(L, i, j)$ pour une même liste L de longueur l et pour toutes les paires i, j possibles.*

2.3 Deuxième version raffinée

On veut maintenant améliorer la complexité en espace de notre solution.

Question à développer pendant l'oral 8 *De manière similaire à la question orale 6, établir une relation de récurrence entre $RMQ(L, i, j)$ et $RMQ(L, x, x + 2^k - 1)$ pour $k = \lfloor \log_2(j - i) \rfloor$ et deux valeurs de x bien choisies.*

Utiliser cette relation pour écrire un algorithme plus efficace en espace que celui de la section précédente.

Question 8 *Calculer en utilisant cette méthode*

a) $CRMQ(H(T_{200}^{10\,000}), 200, 1\,000\,000)$,

b) $CRMQ(H(T_{300}^{100\,000}), 300, 100\,000)$,

c) $CRMQ(H(T_{300}^{100\,000}), 300, 1\,000\,000)$.

Question à développer pendant l'oral 9 *Évaluer la complexité en temps et en espace de cette méthode pour le coût total du calcul des valeurs de $RMQ(L, i, j)$ pour une même liste L de longueur l et pour toutes les paires i, j possibles.*

3 Calcul par bloc

Pour préparer une amélioration de la complexité, pour une liste L de longueur l on subdivise le calcul en blocs de longueur k, que l'on fixe à $\lfloor \sqrt{l} \rfloor$ dans cette partie, et l'on applique séparément sur chaque bloc le meilleur algorithme obtenu jusqu'ici.

Question à développer pendant l'oral 10 *Expliquer comment calculer la valeur de $RMQ(L, i, j)$ à partir des valeurs des RMQ au sein des blocs.*

Question 9 *Utiliser cette approche pour calculer (on utilise cette fois-ci les listes de la section 1.3 pour tester sur de très grandes listes) :*

a) $CRMQ(D_{100}^{1\,000\,000}, 100, 100\,000)$,

b) $CRMQ(D_{100}^{1\,000\,000}, 100, 1\,000\,000)$,

c) $CRMQ(D_{100}^{999\,983}, 100, 100\,000)$,

Question à développer pendant l'oral 11 *Expliquer quel pré-traitement votre algorithme effectue, et comment.*

Question à développer pendant l'oral 12 *Évaluer la complexité en temps et en espace de cette méthode pour le coût total du calcul des valeurs de $RMQ(L, i, j)$ pour une même liste L de longueur l et pour toutes les paires i, j possibles, en séparant le temps de préparation du temps de calcul pour toutes les paires.*

4 RMQ±1

Les listes $H(T)$ ne sont pas quelconques : puisqu'elles représentent les hauteurs des nœuds rencontrés lors d'un parcours de l'arbre, les valeurs sont contiguës, elles vérifient la condition de la section 1.3.
Ceci permet de simplifier le calcul de RMQ au sein des blocs.
On choisit dans cette partie comme taille de bloc $k = \left\lfloor \frac{log_2(l)}{2} \right\rfloor$.

Question à développer pendant l'oral 13 *Montrer pourquoi pour deux blocs de même taille ayant le même index tel que défini à la section 1.3, les valeurs des RMQ sont les mêmes.*

Question 10 *Utiliser ces remarques pour améliorer la complexité du calcul au sein des blocs, et calculer en utilisant cette méthode*

a) $CRMQ(D_{200}^{3\,000\,000}, 200, 10\,000)$,

b) $CRMQ(D_{200}^{3\,000\,000}, 200, 1\,000\,000)$.

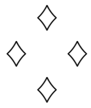
Question à développer pendant l'oral 14 *Expliquer comment votre implémentation stocke ses résultats préparatoires.*

Avec le choix de k précisé plus haut, le nombre d'index différents possibles est en $\mathcal{O}(\sqrt{l})$.

Question à développer pendant l'oral 15 *Évaluer la complexité en temps et en espace de cette méthode pour le coût total du calcul des valeurs de $RMQ(L, i, j)$ pour une même liste L de longueur l et pour toutes les paires i, j possibles, en séparant le temps de préparation du temps de calcul pour toutes les paires.*

5 Épilogue

À la section 2 on a ramené le problème LCA au problème RMQ. Il est également possible de ramener le problème RMQ au problème LCA à l'aide d'un arbre cartésien.



Fiche réponse type: Plus petit ancêtre commun dans des arbres

$\widetilde{u_0}$: 42

Question 1

a) 781

b) 374

c) 805

Question 2

a) 373

b) 167

c) 989

Question 3

a) 55

b) 355

Question 4

a) 82

b) 558

c) 135

Question 5

a) 12

b) 711

c) 534

Question 6

a) 52

b) 117

c) 360

Question 7

a) 227

b) 701

Question 8

a) 158

b) 988

c) 320

Question 9

a) 760

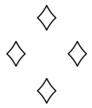
b) 376

c) 890

Question 10

a) 35

b) 380



Fiche réponse: Plus petit ancêtre commun dans des arbres

Nom, prénom, u_0: .....

Question 1

a)

b)

c)

Question 2

a)

b)

c)

Question 3

a)

b)

Question 4

a)

b)

c)

Question 5

a)

b)

c)

Question 6

a)

b)

c)

Question 7

a)

b)

Question 8

a)

b)

c)

Question 9

a)

b)

c)

Question 10

a)

b)