Feuille d'exercices n°15 - Exercices de l'X session 2023

Exercice 1 : Tri fusion et galop (?)

oral 50', X, 2023

Alice choisit un entier x , et Bob doit essayer de le trouver, pour cela il peut proposer un entier y et demander à Alice si $y \leq x$, et peut recommencer cette opération autant de fois qu'il veut.

Q. 1 Proposer une méthode pour trouver x en utilisant au plus x opérations.

Q. 2 Proposer une méthode pour trouver x en $O(\log(x + 1))$ opérations.

On veut trier un tableau T à n éléments, on peut seulement comparer deux à deux des éléments x_1 et x_2 de T . On s'intéresse uniquement au nombre de comparaisons effectuées.

Q. 3 Donner deux algorithmes de tri qui ont une complexité dans le pire cas en $O(n \log(n))$.

Q. 4 Existe-t-il un algorithme de tri ayant une complexité dans le pire cas en $o(n \log(n))$?

On suppose maintenant que T contient n éléments mais avec seulement σ valeurs distinctes que l'on note $v_1, v_2, \dots, v_\sigma$. On note $r_1, r_2, \dots, r_\sigma$ les multiplicités de ces valeurs, ce qui veut dire que r_i représente le nombre d'occurrences de la valeur v_i dans le tableau T . De plus on définit l'entropie de T par $H(T) = - \sum_{i=1}^{\sigma} \frac{r_i}{n} \log_2 \left(\frac{r_i}{n} \right)$

Q. 5 Proposer une méthode pour trier T en $O(n \log(\sigma) + n)$ opérations.

Q. 6 Montrer que $H \leq \log_2(\sigma)$.

Remarque : il y avait trois questions supplémentaires à cet exercice.

Question subsidiaires

- Le tri rapide est-t-il un tri de complexité pire cas $O(n \log(n))$?
 - Sur quel paradigme repose le tri fusion ?
 - Dans quelle partie du programme a-t-on déjà rencontré l'entropie ?
 - Comment montrer qu'une fonction est concave ? ou convexe ?
-

Révisions dans le thème

- Écrire le pseudo code de la recherche dichotomique d'un entier caché dans un intervalle $[[a, b]]$.
Proposer un invariant pour cet algorithme.
- Quelles sont les opérations du type abstrait ENSEMBLE ?
Proposer une implémentation concrète efficace de ce type.
Quelles sont les complexités associées ?
- Quelles sont les opérations du type abstrait DICTIONNAIRE ?
Proposer une implémentation concrète efficace de ce type.
Quelles sont les complexités associées ?
- Quelles sont les opérations du type abstrait FILE DE PRIORITÉ ?
Proposer une implémentation concrète efficace de ce type.
Quelles sont les complexités associées ?
- Donner un équivalent des sommes classiques suivantes.

$$\sum_{i=1}^n 2^i \qquad \sum_{i=1}^n \frac{1}{2^i} \qquad \sum_{i=1}^n \log(i)$$

Autres développements

- Démontrer la complexité pire cas du tri fusion.
- Redémontrer l'inégalité de Jensen.
- Création d'un tas binaire en temps linéaire

Exercice 2 : Automates d'arbres

oral 50', X, 2023

On fixe Σ un alphabet. On considère des arbres binaires dont les nœuds internes sont étiquetés par Σ et les feuilles par $\perp$ un symbole qui n'est pas dans Σ . On notera $\text{rac}(T)$ la racine d'un tel arbre T . De plus, si n est un nœud interne, on notera $n.a$ son étiquette, $n.g$ son fils gauche et $n.d$ son fils droit.

Définition 0.1

Un **automate montant** sur Σ est un quintuplet $(\Sigma, Q, I, F, \delta)$ où Q est un ensemble fini d'états, I un sous-ensemble de Q , F un sous-ensemble de Q , et δ une fonction de $\Sigma \times Q \times Q$ dans 2^Q .

Un tel automate est dit **déterministe** si I et les ensembles $\delta(a, q)$ pour $(a, q, q') \in \Sigma \times Q \times Q$ sont de cardinal au plus 1.

Une **exécution** d'un tel automate $\mathcal{A}$ sur un arbre T est une fonction e de l'ensemble des nœuds de T dans Q telle que pour toute feuille l de T $e(l) \in I$, et pour tout nœud interne n de T on a $e(n) \in \delta(n.a, e(n.g), e(n.d))$.

Une telle exécution est dite **acceptante** si $e(\text{rac}(T)) \in F$, et dans ce cas T est dit **accepté** par $\mathcal{A}$.

Q. 1 Sur $\Sigma = \{0, 1\}$ donner un automate montant qui reconnaît les arbres ayant au moins un nœud étiqueté par 1.

Q. 2 Montrer que tout automate montant est équivalent à un automate montant déterministe.

Définition 0.2

Un **automate descendant** sur Σ est un quintuplet $(\Sigma, Q, I, F, \delta)$ où Q est un ensemble fini d'états, I un sous-ensemble de Q , F un sous-ensemble de Q , et δ une fonction de $\Sigma \times Q$ dans $2^{Q \times Q}$.

Un tel automate est dit **déterministe** si I et les ensembles $\delta(a, q)$ pour $(a, q, q') \in \Sigma \times Q$ sont de cardinal au plus 1.

Une **exécution** d'un tel automate $\mathcal{A}$ sur un arbre T est une fonction e de l'ensemble des nœuds de T dans Q , telle que $e(\text{rac}(T)) \in I$ et pour tout nœud interne n de T on a $(e(n.g), e(n.d)) \in \delta(n.a, e(n))$.

Une telle exécution est dite **acceptante** si pour toute feuille l de T $e(l) \in F$, et dans ce cas T est dit **accepté** par $\mathcal{A}$.

Q. 3 Montrer que tout automate descendant est équivalent à un automate descendant.

Q. 4 Un automate descendant est-il nécessairement équivalent à un automate montant ?
Un automate descendant est-il nécessairement équivalent à un automate montant déterministe ?

Définition 0.3

Un ensemble d'arbres **rationnel** est l'ensemble des arbres reconnus par un automate montant.

Q. 5 Montrer que la classe des ensembles arbres rationnels est stable par union, intersection et complémentaire.

Remarque : il y a des questions supplémentaires à cet exercice.

Question subsidiaires

- A-t-on un résultat analogue de la question ?? pour les automates sur les mots? Est-ce qu'imposer le caractère déterministe est "couteux"?
 - Sur $\Sigma = \{0, 1\}$, peut-on donner un automate montant qui reconnaît les arbres ayant au plus un nœud étiqueté par 1? Un automate montant déterministe?
 - Même question avec descendant?
-

Révisions dans le thème

- Définition d'un automate produit.
- Exemple d'automate dont le déterminisé est de taille exponentielle.

Bob l'écureuil

Bob est un écureuil qui vit le long d'une ligne de chemin de fer. Il a caché ses réserves de noisettes dans les différentes gares, et cherche maintenant à établir son nid dans l'une de ces gares. Pour des raisons pratiques, il souhaite construire son nid dans une gare où il a déjà caché des noisettes, tout en minimisant la durée moyenne entre son nid et ses réserves de noisettes. Comment choisir dans quelle gare établir son nid ?

Question 1. La ligne est composée de 9 gares, numérotées de 0 à 8 ; Bob a mis une cachette dans chacune de ces gares. On met k minutes pour aller de la gare ℓ à la gare $k + \ell$. Quelle gare Bob choisira-t-il d'établir son nid ?

Question 2. On suppose maintenant que les gares ne sont plus régulièrement espacées. Désormais, il y a n gares, toujours numérotées de 0 à $n - 1$, toujours en ligne droite. Bob connaît des entiers $t_0, t_1, \dots, t_{n-2}$ pour lesquels aller de la gare ℓ à la gare $\ell + 1$ requiert t_ℓ minutes ; ainsi, aller de la gare ℓ à la gare $\ell + 2$ requiert $t_\ell + t_{\ell+1}$ minutes. Donner un algorithme qui permettra à Bob de choisir, en temps $\mathcal{O}(n)$, dans quelle gare installer son nid.

Question 3. Désormais, le réseau prend la forme d'un graphe connexe acyclique. Bob dispose, pour chaque gare k , d'une liste des gares ℓ auxquelles la gare k est reliée directement, ainsi que de la durée $d_{k,\ell}$, en nombre de minutes, du trajet entre les gares k et ℓ .

Donner un algorithme qui permettra à Bob de choisir, en temps $\mathcal{O}(n)$, dans quelle gare installer son nid.

Bob déménage à Manhattan, là où toutes les rues sont orientées selon un axe Nord-Sud ou Est-Ouest, et espacées de 100 mètres l'une de l'autre. Il a caché ses n réserves de noisettes à n croisements de rue, chaque croisement étant identifié par des coordonnées entières. Par exemple, le croisement $(2, 3)$ se trouve 400 mètres à l'Ouest et 500 mètres au Sud du croisement $(6, 8)$, donc Bob doit marcher au minimum 900 mètres pour aller d'un croisement à l'autre.

Bob souhaite maintenant établir son nid à l'un des n croisements de rue où se trouve déjà une réserve de noisettes, mais tout en minimisant la distance moyenne entre son nid et des réserves de noisettes. À quel croisement établir son nid ?

Question 4. Bob a disposé ses 6 réserves aux points de coordonnées $(0, 0)$, $(1, 1)$, $(2, 3)$, $(3, 1)$, $(3, 2)$ et $(4, 3)$. Auquel de ces six endroits choisira-t-il d'établir son nid ?

Question 5. Bob a simplement noté les coordonnées entières $(x_0, y_0), (x_1, y_1), \dots, (x_{n-1}, y_{n-1})$ des n réserves qu'il a choisies pour entreposer ses noisettes.

Donner un algorithme qui permettra à Bob de choisir, en temps $\mathcal{O}(n \log(n))$, à quelle intersection installer son nid.

Question 6. On suppose désormais que chacune des coordonnées x_i et y_i est comprise entre 0 et $n^2 - 1$. Comment adapter l'algorithme précédent pour s'assurer qu'il fonctionne désormais en temps $\mathcal{O}(n)$?

Question 7. Pour fluidifier le trafic, des urbanistes ont ajouté des routes orientées du Nord-Est vers le Sud-Ouest, et du Nord-Ouest vers le Sud-Est. Ainsi, le croisement $(2, 3)$ se désormais à $400\sqrt{2} + 100 \approx 666$ mètres du croisement $(6, 8)$: il suffit d'aller quatre fois vers le Nord-Est puis une fois vers le Nord. On suppose toujours que chacune des coordonnées x_i et y_i est comprise entre 0 et $n^2 - 1$, et que Bob veut installer son nid en l'une des n intersections (x_i, y_i) .

Donner un algorithme qui permettra à Bob de choisir, en temps $\mathcal{O}(n)$, à quelle intersection installer son nid.

Algorithme Union-Find

L'algorithme **Union-Find** a pour but de gérer dynamiquement une partition d'un ensemble $\{1, 2, \dots, n\}$ fixé. Initialement, on part de la partition maximale $\{\{1\}, \{2\}, \dots, \{n\}\}$. En pratique, on représente les partitions de $\{1, \dots, n\}$ comme suit :

- ▷ chaque entier k possède un (seul) *père* $p_k \leq n$, ce que l'on notera $k \rightarrow p_k$; on peut avoir $k = p_k$;
- ▷ chaque entier k possède un *poids* $w_k \in \mathbb{N}$;
- ▷ deux entiers distincts k et ℓ ne peuvent être ancêtres l'un de l'autre, et appartiennent au même sous-ensemble si et seulement s'ils ont un ancêtre commun;
- ▷ si l'entier k appartient à un singleton, $w_k = 0$.

Question 1. On dit que m est le *chef* de k si m est un ancêtre de k tel que $m = p_m$. Démontrer que tout entier a un unique chef, et deux entiers k et ℓ appartiennent au même sous-ensemble si et seulement s'ils ont le même chef.

L'algorithme est censé gérer trois types de requêtes, en procédant comme suit :

1. La requête auxiliaire **Chef**(k) : on recherche le chef de k . Si $k = p_k$, il s'agit de k lui-même. Sinon, il s'agit du chef de p_k , et ce chef devient le nouveau parent de k .
2. La requête **Test**(k, ℓ) : on se demande si k et ℓ appartiennent au même sous-ensemble. Cela revient à identifier les entiers $k' = \text{Chef}(k)$ et $\ell' = \text{Chef}(\ell)$ puis à vérifier si $k' = \ell'$.
3. La requête **Union**(k, ℓ) : on souhaite réunir les sous-ensembles auxquels appartiennent k et ℓ . Cela revient à identifier les entiers $k' = \text{Chef}(k)$ et $\ell' = \text{Chef}(\ell)$, puis :
 - ▷ si $w_{k'} > w_{\ell'}$, l'entier k' devient le nouveau parent de ℓ' ;
 - ▷ si $w_{\ell'} > w_{k'}$, l'entier ℓ' devient le nouveau parent de k' ;
 - ▷ si $k' \neq \ell'$ et $w_{k'} = w_{\ell'}$, l'entier k' devient le nouveau parent de ℓ' et son poids augmente de 1.

On souhaite démontrer que répondre à m de ces requêtes peut se faire en temps $\mathcal{O}(m \log^*(m))$, où $\log^*$ est la fonction définie par $\log^*(m) = 1$ lorsque $m \leq 1$ et $\log^*(m) = 1 + \log^*(\log_2(m))$ lorsque $m > 1$.

Question 2. En partant de l'ensemble $\{\{1\}, \{2\}, \{3\}, \{4\}\}$, on effectue successivement les requêtes **Union**(1, 2), **Union**(3, 4), **Union**(2, 4) et **Test**(2, 4). Indiquer le père et le poids de chaque entier $k \leq 4$.

Question 3. On note w_k^∞ le poids d'un entier k à la fin de l'algorithme. Démontrer que $w_u^\infty < w_v^\infty$ pour tous les entiers $u \neq v$ tels que v a été le parent de u .

Question 4. Démontrer, pour tout entier $\ell \geq 1$, qu'il y a au plus $m/2^{\ell-1}$ entiers $k \leq n$ pour lesquels $w_k^\infty = \ell$.

Question 5. Démontrer que toute requête **Test** ou **Union** effectuée au cours de l'algorithme a une complexité majorée par $\mathcal{O}(\log_2(\min\{m, n\}))$.

Question 6. Soit $\mathcal{G} = (V, E)$ le graphe dont les sommets sont les entiers de 1 à n et les arêtes sont les paires (u, v) pour lesquelles $u \neq v$ et v a été le parent de u au cours de l'algorithme, mais pas quand celui-ci se termine. Démontrer que la complexité totale de nos m requêtes est majorée par $\mathcal{O}(m + |E|)$.

Question 7. Soit $(a_\ell)_{\ell \geq 0}$ la suite définie par $a_0 = 0$ et $a_{\ell+1} = 2^{a_\ell}$. Pour tout entier $\ell \geq 0$, on note V_ℓ l'ensemble des entiers k tels que $a_\ell \leq w_k^\infty < a_{\ell+1}$. Démontrer que $\mathcal{G}$ contient au plus $4m$ arêtes reliant deux sommets de V_ℓ , et au plus $2m$ arêtes allant d'un sommet de V_ℓ à un sommet en dehors de V_ℓ .

Question 8. Que conclure sur la complexité de l'algorithme **Union-Find** ?

Langages rationnels et lemme de l'étoile

On recherche un langage $\mathcal{L}$ non rationnel mais pour lequel $\mathcal{L}$ et son complémentaire satisfont le lemme de l'étoile.

Question 1. Soit Σ un alphabet fini. Le langage $\mathcal{L}_1$ formé des mots $w \in \Sigma^*$ dont la longueur est paire est-il rationnel ?

Question 2. Le langage $\mathcal{L}_2$ formé des mots $w \in \Sigma^*$ dont la longueur est le carré d'un entier est-il rationnel ?

Soit $\mathcal{L}$ et $\mathcal{L}'$ deux langages sur un alphabet Σ . On dit que $\mathcal{L}$ est $\mathcal{L}'$ -étoilé s'il existe un entier $n \geq 0$ tel que tout mot $w \in \mathcal{L}'$ de longueur $|w| \geq n$ admet une factorisation $w = s \cdot t \cdot u$ pour laquelle $|s| \leq n$, $1 \leq |t| \leq n$ et $s \cdot t^* \cdot u \subseteq \mathcal{L} \cup \{w\}$. Si $\mathcal{L}$ est $\mathcal{L}$ -étoilé, on dit même que $\mathcal{L}$ satisfait le lemme de l'étoile.

Question 3. Démontrer que tout langage rationnel satisfait le lemme de l'étoile.

Question 4. Démontrer que, si $|\Sigma| = 1$, les langages qui satisfont le lemme de l'étoile sont les langages rationnels.

Soit $\mathcal{S}$ une partie finie de Σ^* . On note $\mathcal{M}(\mathcal{S})$ l'ensemble des mots $s_1 \cdot s_2 \cdots s_n$ que l'on peut factoriser en n éléments de $\mathcal{S}$ et pour lesquels il existe deux entiers i et $j = i + 1$ ou $j = i + 2$ tels que $s_i = s_j$.

Question 5. Démontrer que tout ensemble $\mathcal{M}(\mathcal{S})$ est rationnel.

Question 6. Démontrer que, si $|\mathcal{S}| \leq 4$, le langage $\mathcal{M}(\mathcal{S})$ est $\mathcal{S}^*$ -étoilé.

Question 7. On considère l'alphabet $\Sigma = \{a, b, c, d\}$. Pour tout entier $k \leq 3$, on note σ_k le mot $(abc)^k \cdot (abd)^{3-k}$. Les langages $\mathcal{L}_3 = \{(\sigma_0 \cdot \sigma_1 \cdot \sigma_2)^\ell \cdot (\sigma_0 \cdot \sigma_1 \cdot \sigma_3)^\ell : \ell \geq 0\}$ et $\mathcal{L} = \mathcal{M}(\{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}) \cup \mathcal{L}_3$ sont-ils rationnels ?

Question 8. Démontrer que les ensembles $\mathcal{L}$ et $\Sigma^* \setminus \mathcal{L}$ satisfont le lemme de l'étoile.

Sujets d'oraux d'informatique - 2023

Matthieu XILLO

1 Polytechnique

1.1 Énoncé (définitions / cadre)

Automate bidirectionnel : Pour Σ un ensemble fini de caractères, on appelle *automate bidirectionnel* sur Σ un quintuplet $A = (\Sigma_{\triangleright\triangleleft}, Q, I, F, \delta)$, où $\Sigma_{\triangleright\triangleleft} = \Sigma \sqcup \{\triangleleft, \triangleright\}$, Q est un ensemble fini d'états, $I \subset Q$ est l'ensemble des états initiaux, $F \subset Q$ est l'ensemble des états finaux, et δ est une application de $Q \times \Sigma_{\triangleright\triangleleft}^*$ dans $\mathcal{P}(Q \times \{-1, 1\})$.

Exécution d'un automate bidirectionnel : Pour $w \in \Sigma_{\triangleright\triangleleft}^*$ de taille n , on appelle *exécution de A sur w* un triplet (k, pos, s) , où $k \in \mathbb{N}$, $\text{pos} \in \mathcal{F}(\llbracket 1, k \rrbracket, \llbracket 1, n \rrbracket)$ et $s \in \mathcal{F}(\llbracket 1, k \rrbracket, Q)$, vérifiant $\forall i \in \llbracket 1, n \rrbracket, (s(i+1), w_{\text{pos}(i+1)} - w_{\text{pos}(i)}) \in \delta(s(i), w_{\text{pos}(i)})$. De plus, on doit avoir $\text{pos}(1) = 1$ et $s(1) \in I$. Cette exécution est alors *acceptante* si et seulement si $q_k \in F$.

En somme, les automates bidirectionnels sont des automates avec la capacité de "revenir en arrière" sur le mot.

Langage reconnu par un automate bidirectionnel : On appelle *langage reconnu par l'automate bidirectionnel A* l'ensemble des mots $w \in \Sigma^*$ pour lesquels il existe une exécution acceptante de A sur $\triangleright \cdot w \cdot \triangleleft$. Un langage reconnu par un automate bidirectionnel est dit *bireconnaissable*.

Automate déterministe : Un automate bidirectionnel est *déterministe* si et seulement si I contient au plus un élément et pour tout état $q \in Q$, pour toute lettre $l \in \Sigma_{\triangleright\triangleleft}$, $\delta(q, l)$ contient au plus un élément.

Exécution injective : Une exécution est *injective* si et seulement si $i \mapsto (\text{pos}(i), s(i))$ est injective.

Front : Pour une exécution donnée, on appelle *front d'un état $q \in Q$* la suite (éventuellement finie) $(w_{\text{pos}(i)}, (\text{pos}(\varphi(i)+1) - \text{pos}(\varphi(i))))_{i \in \llbracket 1, k \rrbracket}$, où φ est l'extractrice vérifiant $\forall i, s(\varphi(i)) \in Q$ (en d'autres termes, φ décrit dans l'ordre tous les rangs de l'exécution où l'on est sur l'état q).

1.2 Énoncé (questions)

1. Montrer que tout langage reconnaissable est bireconnaissable.
2. Donner un automate bidirectionnel déterministe reconnaissant tous les mots dont la première et la dernière lettre coïncident.
3. Montrer que l'ensemble des langage bireconnaissables est stable par union et intersection.
4. Montrer que pour tout mot du langage d'un automate bidirectionnel, il existe une exécution injective acceptante pour ce mot.
5. Montrer que le nombre de fronts possibles pour un état donné pour des exécutions injectives est fini et en donner une majoration.
6. Montrer que tout langage bireconnaissable est reconnaissable.

1.3 Corrigé