
Exercice 1 : Quelques applications immédiates du cours

Q 1. Pour chaque affirmation suivante, dire si elle est vraie ou fausse. Justifiez briévement.

1. Tout langage reconnu par un automate peut être décidé par une Machine de Turing.
2. $\{a^n / n \ est \ premier\}$ peut être décidé par une Machine de Turing.
3. L'intersection d'un langage récursif et d'un langage algébrique est un langage récursif.
4. Le complémentaire d'un langage algébrique est récursif.
5. Le complémentaire d'un langage récursif est r.e. .
6. Le complémentaire d'un langage r.e. est récursif.

Q 2.*La classe NP*

La propriété de décision Dec a été prouvée NP-complète. Que pensez-vous de chacune des trois affirmations suivantes? Justifiez briévement.

1. Il existe un algorithme non déterministe polynômial pour décider Dec.
2. Il n'existe pas d'algorithme déterministe polynômial pour décider Dec.
3. La propriété Dec est décidable.
4. Il existe un algorithme déterministe pour décider Dec.

Exercice 2 : Un peu de programmation dynamique

Un mot v est sous-mot d'un mot u si il peut être obtenu à partir de u en effaçant éventuellement certaines lettres. Formellement soit un mot $u = u_1 ... u_n$. Un sous-mot v de u est de la forme $u_{i_1} ... u_{i_k}$ avec $1 \leq i_1 < i_2 < ... < i_k \leq n$. Par exemple, *aac* est un sous-mot de *avance* -avec $i_1 = 1, i_2 = 3, i_3 = 5$ - et d'*anaconda* -avec $i_1 = 1, i_2 = 3, i_3 = 4$, mais pas de *chacal*. Un mot peut apparaître plusieurs fois comme sous-mot dans un autre mot. Par exemple, dans le mot *avalanche*, *aac* apparaît trois fois comme sous-mot, aux positions $(i_1 = 1, i_2 = 3, i_3 = 7)$,$(i_1 = 1, i_2 = 5, i_3 = 7)$ et $(i_1 = 3, i_2 = 5, i_3 = 7)$.

Q 1. Combien de fois *lus* apparaît-il comme sous-mot dans *illusions*? dans *libellules*? dans *hurluberlus*?

Q 2. Proposez un algorithme en $O(|u| * |v|)$ pour compter le nombre d'occurrences d'un mot comme sous-mot d'un autre:

Entrée: deux mots u et v, avec $|v| \leq |u|$.

Sortie: le nombre de fois que v apparaît comme sous-mot de u.

Aide: on pourra par exemple appliquer le principe de programmation dynamique. Pour cela, on pourra introduire **par exemple** $nb(i, j)$, le nombre d'occurrences de $v_1 ... v_j$ dans $u_1 .. u_i$.

Q 3. On cherche maintenant seulement à savoir si le mot v est sous-mot du mot u, sans chercher à compter le nombre d'occurrences.

Entrée: deux mots u et v, avec $|v| \leq |u|$.

Sortie: Oui, si v apparaît comme sous-mot de u. Non, sinon.

Proposez un algorithme linéaire (en $O(|u|)$) pour ce problème.

Exercice 3 : Cadres emboîtés

Soit C un ensemble de n cadres rectangulaires. Un cadre est caractérisé par sa largeur l et sa hauteur h. Un cadre (l_1, h_1) encapsule un cadre (l_2, h_2) si $l_1 \geq l_2$ et $h_1 \geq h_2$. Le problème est d'extraire de C, un sous-ensemble C' de cardinal minimal tel que tout cadre de C soit encapsulé par (au moins) un cadre de C'.

Exemple: $(7,3),(5,4),(6,9),(3,9),(2,12),(6,10)$ est couvert de façon optimale par $(7,3),(2,12),(6,10)$.

Formellement le problème est donc:

Entrée: n un nombre de cadres, $(l_1, h_1)..., (l_n, h_n)$ les dimensions des cadres

Sortie Un sous-ensemble J de $[1..n]$ de cardinal minimal tel que $\forall i, 1 \leq i \leq n, \exists j \in J, l_i \leq l_j \wedge h_i \leq h_j$

Q 1. Quelle est la solution optimale pouur $(7,8),(5,4),(10,9),(3,10),(2,12),(6,15),(7,11)$?

Q 2. Si on suppose que deux cadres différents ont au moins une dimension différente, (c.à.d. si $i \neq j$, $l_i \neq l_j$ ou $h_i \neq h_j$) peut-il y avoir deux solutions optimales différentes pour la même donnée? Justifiez.

Q 3. Proposez un algorithme (par exemple de type "glouton") pour résoudre le problème. Justifiez sa correction et analysez sa complexité.

Exercice 4 : Score

Rappel: Un graphe pondéré est un graphe tel qu'à chaque arc a est associé un poids $p(a)$. Le poids d'un chemin est la somme des poids des arcs qui le composent.

Soit le problème *Score*:

Entrée:

$G(S, A)$ un graphe pondéré non orienté, dont les poids des arcs sont positifs ou nuls.

u, v deux sommets du graphe.

s un entier.

Sortie : Oui, si il existe un chemin sans boucle de u à v de poids supérieur ou égal à s. Non, sinon.

Q 1. Montrez que le problème *Score* est NP.

Q 2. *Rappel:* Un circuit hamiltonien dans un graphe est un chemin qui passe une et une seule fois dans chaque sommet puis revient au sommet de départ. Le problème de l'existence d'un circuit hamiltonien dans un graphe est $NP-$complet.

Montrez que le problème de l'existence d'un circuit hamiltonien se réduit polynômialement dans le problème *Score*.

Q 3. Qu'en déduire pour Score?

Q 4. On suppose maintenant que le graphe est orienté et sans cycle. Pensez-vous que cela change la complexité du problème? Justifiez votre réponse.

Exercice 5 : Affectation Pondérée

Soit une matrice carrée $n \times n$, dont les coefficients sont des entiers positifs ou nuls. Une affectation complète est un sous-ensemble de ses coefficients qui contient un et un seul élément par ligne et par colonne. Son poids est la somme de ses éléments. Le problème est de trouver une affectation complète de poids maximal.

Exemple: soit la matrice

$$\begin{pmatrix} 1 & 0 & 3 \\ 8 & 2 & 6 \\ 4 & 7 & 5 \end{pmatrix}$$

$\{1,2,5\}$ est une affectation complète de poids 8, $\{4,2,3\}$ est une affectation complète de poids 9; $\{1,7,5\}$ n'est pas une affectation complète.

Q 1. Dans l'exemple, donner une affectation complète de poids maximal.

Q 2. Quel est le cardinal d'une affectation complète? Soit un sous-ensemble de n coefficients qui contient un élément *au plus* par ligne et par colonne. Montrer qu'il définit une affectation complète.

Q 3. Si tous les coefficients de la matrice sont distincts, combien existe-t-il d'affectations complètes?

Q 4. Soit l'heuristique suivante:

```
Initialiser Aff à l'ensemble vide;
tant qu'il y a des éléments dans la matrice faire
  choisir l'élément maximal de la matrice;
  l'ajouter à Aff;
  éliminer dans la matrice la ligne et la colonne de l'élément;
fin tant que;
```

Dans l'exemple, l'heuristique choisit d'abord 8 et élimine la première colonne et la deuxième ligne:

$$\begin{pmatrix} 0 & 3 \\ 7 & 5 \end{pmatrix}$$

puis elle ajoute 7 à `Aff` et donc la matrice se réduit à 3 qui est enfin ajouté à `Aff`: `Aff` vaudra donc $\{8, 7, 3\}$.
Q 4.1. Montrer que l'heuristique construit toujours une affectation complète.
Q 4.2. Par un contre-exemple, montrer que l'affectation complète donnée par cette heuristique n'est pas toujours de poids maximal.
Q 4.3. Soit une affectation complète de poids maximal; montrer que ce poids est au plus égal à deux fois le poids de l'affectation produite par l'heuristique;
Q 4.4. Par un exemple, montrer que l'affectation complète de poids maximal peut avoir un poids égal à exactement deux fois le poids de l'affectation produite par l'heuristique;
Q 4.5. Déduire des questions précédentes la garantie de l'heuristique.