CentraleSupélec

1re année 2018–2019
ST2, campus de Saclay

Vendredi 25 janvier 2018, matin

Exemple d'examen écrit de

Algorithmes et Compléxité

Durée: 3 heures

Numéro étudiant:	
Nom:	Prénom:

— Tous les documents sont autorisés.
— Les outils électroniques (calculatrices, ordinateurs et téléphones) sont interdits.
— Tous vos algorithmes peuvent être écrits en pseudo-code ou en Python, selon ce que vous préférez.
— Le barème est donné à titre indicatif
— Ce sujet comporte 6 pages
— Vous pouvez utiliser les pages supplémentaires en fin de sujet si vous manquez de place.

Ce sujet d'examen fictif a été composé à partir d'extraits des sujets des 3 dernières années dans les cursus Centrale et Supelec/Gif

Il pour objectif de donner une idée du type de sujet qui sera proposé en 2018-2019.

Il ne s'agit pas d'un véritable sujet d'annales.

Exercice 1 : Algorithme de Huffman **5 point(s)**

On s'intéresse à la compression sans perte de séquences de caractères. En informatique, chaque caractère est représenté à l'aide de n bits suivant une table de codage. La plus connue est la table ASCII, sur 7 bits. Par exemple, le symbole "a" est codé `1100001` en ASCII.

L'idée de Huffman pour compresser une séquence de caractères est de coder les caractères sur un nombre de bits variable selon la fréquence d'occurrence du caractère. Par exemple, la lettre la plus fréquente en français est la lettre "e" : elle devrait être codée avec le moins de bits possible, contrairement au "w", peu fréquent, qui peut être codé avec plus de bits.

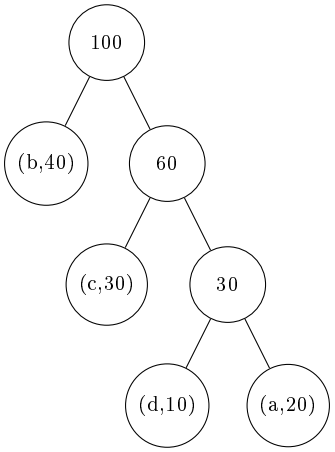
L'algorithme de compression de Huffman repose sur l'utilisation d'un arbre binaire de compression qui permet de calculer le code binaire de chaque caractère. La construction de l'arbre est la suivante :

1. Calculer la fréquence d'apparition de chaque caractère et construire une liste de couples (c, f_c) où c est un caractère et f_c sa fréquence d'occurrence ;
2. Trier la liste par f_c croissant ;
3. Tant que la liste n'est pas réduite à un seul élément :
   (a) Retirer de la liste les deux plus petits éléments c_1 et c_2 ;
   (b) Construire un arbre binaire a de hauteur 1 dont les deux feuilles sont c_1 et c_2 et dont la racine est étiquetée par la somme des fréquences de ces éléments : $f_a = f_{c_1} + f_{c_2}$;
   (c) Insérer (a, f_a) dans la liste (en maintenant le tri portant sur les valeurs de $f\_$).

Supposons par exemple les fréquences d'occurrence suivantes :

caractère	a	b	c	d
fréquence	20%	40%	30%	10%

L'algorithme prendra tout d'abord les caractères "a" et "d" pour les regrouper. Il obtiendra un arbre de fréquence "30%" qu'il mettra dans la liste. Au tour suivant, il assemblera cet arbre avec le caractère "c" pour faire un arbre de fréquence "60%", etc. L'arbre de compression obtenu sera le suivant :



Le code binaire de chaque caractère est alors défini par son parcours dans l'arbre de compression, en mettant un 0 lorsqu'on prend la branche de gauche et un 1 lorsqu'on prend la branche de droite. Dans notre exemple, le caractère "b" est codé par "0" alors que le caractère "a", moins fréquent, est codé par "111".

Question 1 1 point(s)

Proposez une structure de données pour représenter les données dans l'arbre de codage construit. Vous indiquerez précisément comment la structure peut être utilisée.

Décodage

Question 2 1 point(s)

Pour commencer, nous allons décoder un seul caractère. Écrivez, en pseudo-code ou en Python, une fonction de décompression qui prend en entrée une liste de bits et un arbre de compression et qui renvoie le caractère représenté par cette liste de bits, ou None si la liste de bits n'est pas un code de Huffman valide pour cet arbre de compression.

Question 3 1 point(s)

En vous inspirant du code précédent, écrivez, en pseudo-code ou en Python, une fonction de décompression qui prend en entrée une liste de bits représentant une liste de plusieurs caractères et un arbre de compression, et qui renvoie la liste de caractères initiale.

Question 4 0.5 point(s)

Quelle est la complexité de cet algorithme ?

Encodage

Question 5 1.5 point(s)

Écrivez une fonction qui prend en entrée un arbre de compression et qui renvoie une structure de données permettant d'associer chaque caractère à son code de Huffman.

Exercice 2 : Problème du chemin le plus sûre **5 point(s)**

Notre objectif est de calculer le trajet le plus sûre pour un transport de fonds du point S au point T. On considère un graphe orienté modélisant le réseau routier, les nœuds sont les carrefours et les arcs représentent les routes. Un arc $A \xrightarrow{p} B$ représente une route permettant de circuler du carrefour A au carrefour B avec un risque de braquage correspondant à une probabilité $0 < p < 1$.

Question 1 1 point(s)

Quelle est la probabilité qu'un chemin soit emprunté sans rencontrer de braquage en fonction des probabilités de braquage portées par ses arcs.

Question 2 3 point(s)

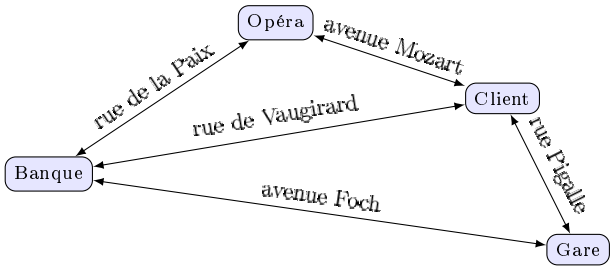
Déterminez puis adaptez l'algorithme (vu en cours) le plus efficace pour résoudre le problème du chemin le plus sûre.

Donnez sa complexité en fonction des entrées du problème.

Question 3 1 point(s)

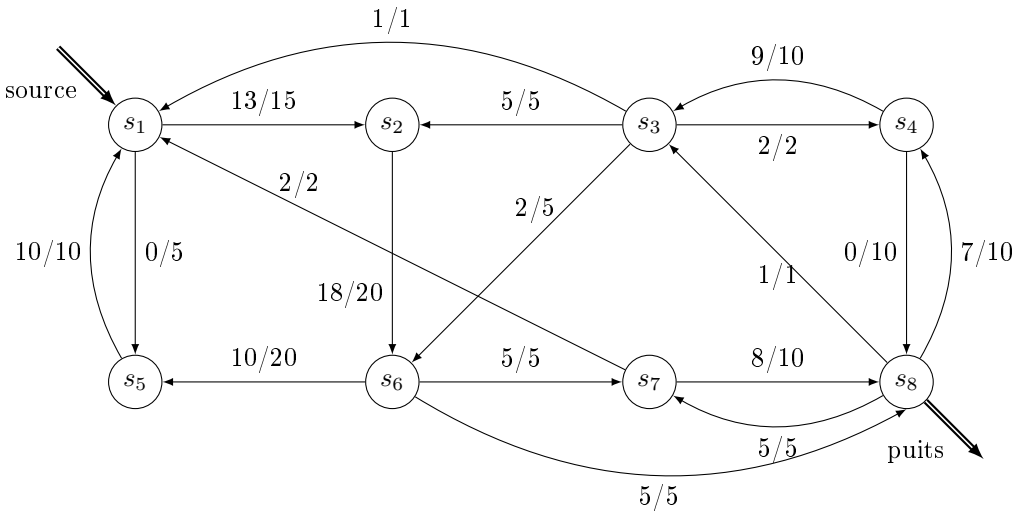
Déroulez cet algorithme sur l'instance suivante en décrivant les étapes intermédiaires (il faut aller de la banque jusqu'au client). On peut emprunter les arcs dans les deux sens avec la même probabilité de braquage.

- Rue de la Paix : $p = 0.1$
- Rue de Vaugirard : $p = 0.1$
- Avenue Mozart : $p = 0.05$
- Rue Pigalle : $p = 0.05$
- Avenue Foch : $p = 0.01$



Exercice 3 : Problème de Flot **3 point(s)**

Le graphe suivant est initialisé avec un flot non nul, l'étiquette $\varphi(a)/c(a)$ sur l'arc a indiquant qu'un flux $\varphi(a)$ circule sur l'arc a qui a pour capacité $c(a)$.



Terminer l'exécution de l'algorithme de Ford-Fulkerson et donner la valeur d'un flot maximal en montrant les étapes intermédiaires effectuées. Donner une coupe de capacité minimale.

Exercice 4 : Programmation Dynamique — 4 point(s)

Soit A une matrice $n \times m$ de nombres entiers. On cherche à trouver la taille d'une sous-matrice carrée $k \times k$ de A composée exclusivement de zéros dont la taille est la plus grande.

Exemple : Pour un tableau

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

un algorithme doit donner la réponse 2.

Question 1 — 1.5 point(s)

Proposez une formule de récurrence.

Question 2 — 1.5 point(s)

Utilisez votre formule pour proposer un algorithme de résolution par programmation dynamique.

Question 3 — 0.5 point(s)

Calculez la complexité en temps.

Question 4 — 0.5 point(s)

Donnez la taille d'une structure dans laquelle vous stockez des résultats intermédiaires.

Exercice 5 : Traitement des problèmes difficiles — 4 point(s)

Le problème de la clique maximum dans un graphe $G = (V, E)$ consiste à trouver un sous-ensemble de sommets V' de taille maximum tel que le sous graphe induit par V' (autrement dit : un sous-graphe $G' = (V', E')$ tel que $E' = E \cap V' \times V'$) est complet.

Le problème de décision associé, déterminer s'il existe une clique de taille au moins k, est un problème *NP*-complet.

Question 1 — 1.5 point(s)

Proposez alors un algorithme glouton pour construire un sous-graphe qui est une clique.

Question 2 0.5 point(s)

Calculez la complexité de votre algorithme.

Question 3 0.5 point(s)

La clique renvoyée par votre algorithme est-elle maximum ? Justifiez.

Question 4 1.5 point(s)

Le problème de la couverture par des cliques est le suivant : étant donnée un graphe $G = (V, E)$ et un entier K, existe-t-il une partition de V en au plus K sous-ensembles V_i tels les sous-graphes induits par chaque V_i (autrement dit : des sous-graphes $G_i = (V_i, E_i)$ tels que $E_i = E \cap V_i \times V_i$) sont des cliques.

Un graphe est coloriable avec au plus K couleurs si l'on peut associer à chaque sommet du graphe une couleur de telle sorte que à ce qu'un sommet ne partage jamais une couleur avec l'un de ces voisins.

Sachant que déterminer si un graphe $G = (V, E)$ est coloriable en K couleurs est un problème *NP*-Complet, montrez que le problème de la couverture par des cliques est également *NP*-Complet.

— fin —

2019-01-17