Licence Informatique ISTIC
Module ALG *2018-2019*

ALG TD Algorithmes Gloutons

Concevoir des solutions basées sur la méthode étudiée

Exercice 1 (Les stations-services)

Le professeur Midas conduit sa voiture de Rennes à Pékin. Il part avec son réservoir d'essence plein et a une carte avec les stations-services le long de la route avec leur distance qui les sépare du point de départ (Rennes).
On supposera que la consommation d'essence du véhicule du professeur Midas est constante (et qu'il n'y a pas de pénurie d'essence). Ceci qui permet de dire que le professeur Midas peut parcourir une distance constante de n kilomètres avec un plein.

1. Donnez une méthode gloutonne, grâce à laquelle le professeur Midas pourra déterminer les stations-service où il peut s'arrêter, sachant qu'il souhaite faire le moins d'arrêts possible.
2. Écrire un algorithme basé sur cette méthode.
3. Montrer sa terminaison et donner sa complexité (indication: on montrera qu'au total la boucle While est exécutée $m - i$ fois dans ITINERAIRE(S, i) où m est la taille de S).
4. Démonter qu'il fournit la solution optimale.
5. Modéliser le problème sous forme d'un graphe. Quels algorithmes vus en cours "Algorithmique des graphes" peuvent être appliqués pour sa résolution?

Exercice 2 (Stockage de fichiers sur bande magnétique)

On souhaite stocker n fichiers sur une bande magnétique sachant que l'accès à l'un des fichiers stockés se fait en parcourant la bande séquentiellement en partant de son début. L'accès à un fichier donné peut prendre un temps significatif si ce fichier est précédé de nombreux fichiers de grande longueur.
On s'intéresse donc au développement d'un algorithme permettant d'organiser les n fichiers sur la bande de manière à minimiser le temps d'accès moyen.
On dispose du tableau $L[1..n]$ des longueurs des fichiers : le fichier i est de longueur $L[i]$.
Si les fichiers sont stockés dans l'ordre de 1 à n, le coût pour accéder au fichier k est

$$cout(k) = \sum_{i=1}^{k-1} L[i]$$

1. En supposant que les fichiers ont la même chance d'être consultés, déterminer l'espérance du coût pour accéder à un fichier arbitraire. Avec $n = 3$, $L[1] = 12, L[2] = 5, L[3] = 8$, donner l'ordonnancement optimal.

2. Décrire une méthode gloutonne pour déterminer une permutation *optimale* π^* des n fichiers, c'est à dire telle qu'en ordonnant les fichiers sur la bande selon π^*, l'espérance du coût est minimale (*i.e.* pour toute permutation π, on a $E[cout_{\pi^*}] \leq E[cout_\pi]$). Montrer sa correction. *Rappel de notation utile : si le k-ème élément du tableau se retrouve à la i-ème position dans la permutation, alors on a* $\pi(i) = k$ *et* $\pi^{-1}(k) = i$.

   On suppose dans les questions suivantes que la bande a une longueur limitée l_{max}.

3. Est-ce qu'un algorithme glouton qui sélectionne les programmes par ordre croissant de longueur maximise le nombre de programmes stockés ? Si oui, le prouver, si non, donner un contre-exemple.

4. Un algorithme glouton qui sélectionne les programmes par ordre décroissant de coût maximise-t-il l'espace utilisé ? Si oui, le prouver, si non, donner un contre-exemple.

Exercice 3 (Algorithme de Prim)
En théorie des graphes, on peut utiliser l'algorithme de Prim afin de calculer un arbre couvrant minimal. Cet algorithme fait partie des algorithmes gloutons. On rappelle l'algorithme de Prim :

Algorithm 1 PRIM$(G = (V, E), s)$

Require: tableau $G = (V, E)$ un graphe connexe, s un sommet de G
Ensure: Retourne E', un arbre couvrant minimal de G.
1: $V' = V \setminus \{s\}$
2: **while** $V' \neq \emptyset$ **do**
3: Choisir $e = (v, v')$ de poids minimal avec $v \notin V'$, $v' \in V'$
4: $E' \leftarrow E' \cup \{e\}$
5: $V' \leftarrow V' \setminus \{v'\}$
6: **end while**
7: **return** E'

1. Rappeler la définition d'un arbre couvrant minimal.

2. Donner le principe glouton derrière l'algorithme de Prim.

3. Montrer que l'algorithme de Prim renvoie bien un arbre et est optimal.

Exercice 4
Dans cet exercice, on souhaite s'intéresser au problème du rendu de monnaie. Ce problème peut s'exprimer sous la forme "comment puis-je rendre l'argent sous une forme optimale", c'est-à-dire en utilisant le moins de pièces/billets possibles. On suppose que l'on ne risque pas de tomber à court de pièce.

1. Donner le rendu optimal pour une somme de 3.79€.

2. Ecrire un algorithme glouton répondant au problème du rendu de monnaie (renvoyer le nombre de pièces et la nature des pièces).

3. Montrer qu'il est optimal pour le cas d'un sous-système monétaire de l'euro : pièces de 1-2-5-10 (en centimes).

   Cependant, il existe des systèmes monétaires sur lequel l'algorithme glouton n'est pas optimal! (On peut par exemple penser au système monétaire anglais pré-réforme de 1971).

4. Donner un système monétaire et une somme pour lesquels l'algorithme glouton n'est pas optimal.

5. Justifier de l'utilisation de cet algorithme dans la vie de tous les jours, par rapport à d'autres algorithmes toujours exacts.

   Il n'existe actuellement pas de caractérisation des systèmes canoniques, uniquement des moyens de le vérifier.

Le partitionnement de données (data clustering)

Le calcul d'un ACM (Arbre couvrant minimal) joue un rôle important dans le problème du partionnement de données (data clustering) que nous traitons à présent.

Les algorithmes gloutons nommés *Kruskal, et Reverse-Delete* décrits informellement ci-après permettent aussi de calculer un ACM. Nous en donnons une explication informelle.

- Kruskal : L'arbre grandit par l'ajout des arêtes dans l'ordre croissant de leur poids. La seule condition est d'éviter l'apparition des cycles.
- Reverse-Delete : L'arbre est obtenu à partir du graphe complet, duquel sont éliminées des arêtes dans l'ordre décroissant de poids à la seule condition de ne pas déconnecter le graphe (l'approche est justifiée par la "Cycle Property" rappelée ci-dessous).

Proposition 1 (Cycle Property) *Si G contient un cycle, alors une arête de poids maximal dans ce cycle n'appartient à aucun ACM de G.*

Soit $U = \{u_1, u_2, \ldots, u_n\}$ un ensemble fini muni d'une distance $d : U \times U \to \mathbb{R}^+$, c-à-d. (i) $d(u,u) = 0$, pour tout $u \in U$, (ii) $d(u,u') = d(u',u)$, pour tous $u, u' \in U$, et (iii) $d(u,u') \leq d(u,u'') + d(u'',u')$, pour tous $u, u', u'' \in U$.

Définition 1 (Espacement) *Étant donnés $C, C' \subseteq U$ deux sous-ensembles non-vides et disjoints, l'*espacement *entre C est C', noté $\mathtt{esp}(C, C')$, est le minimum des distances $d(u,u')$ où $u \in C$ et $u' \in C'$. Étant donné une partition Π de U, on définit son espacement par*

$$\mathtt{esp}(\Pi) = \min_{C,C' \in \Pi} \mathtt{esp}(C, C')$$

Définition 2 (k-Clustering) *Pour $k \in \mathbb{N}$, un k-*clustering *de U est une partition de U de cardinal k.*

Exercice 5
Étant donnés $k \in \mathbb{N}$ et un ensemble fini U muni d'une distance, on souhaite calculer un k-clustering de U d'espacement maximal.

5.1 *Quelle forme aurait l'algorithme "brute-force" ? Quelle serait sa complexité ?*
Indications : on pourra essayer de dénombrer le nombre de partitions de cardinal k sur un ensemble à n éléments.
Ce nombre noté $\left\{ \begin{array}{c} n \\ k \end{array} \right\}$ *s'appelle le* nombre de Stirling de second type.

5.2 *Adapter l'algorithme de Kruskal écrit comme un Algorithme Glouton (ci-dessous) pour calculer un k-clustering d'espacement maximal.*

Algorithm 2 KruskalGlouton$(G(V,E), w : E \to \mathbb{R}, U, X)$

Require: Un graphe $G(V,E)$ dont les arêtes sont triées par poids croissants, la fonction de poids étant $w : E \to \mathbb{R}$.
Require: Un ensemble $X \subseteq E$ et une structure Union-Find U d'éléments de V.
1: **if** E est vide **then**
2: **return** X
3: **else**
4: Soit (u,v) et E' tels que $E = (u,v) :: E'$.
5: **if** U.`find`$(u) \neq U$.`find`(v) **then**
6: $X \leftarrow X \cup \{(u,v)\}$
7: U.`union`(`find`(u), `find`(v))
8: **end if**
9: **return** KruskalGlouton$((V,E'), w, U, X)$
10: **end if**

Algorithm 3 Kruskal$(G(V,E), w : E \to \mathbb{R})$

Require: Un graphe $G(V,E)$ muni d'une fonction de poids $w : E \to \mathbb{R}$.
1: Trier les arêtes E par poids croissant.
2: $U = \emptyset$
3: **for all** $u \in V$ **do**
4: U.`makeset`(u)
5: **end for**
6: $X \leftarrow \emptyset$
7: Return KruskalGlouton(G, w, U, X)

5.3 *Démontrer la proposition suivante, qui induit un algorithme pour calculer un k-clustering d'espacement maximal. En déduire que l'algorithme de la Question 5.2 est correct.*

Proposition 2 *Les k composantes connexes $C_1, C_2, \ldots C_k$ obtenues par la suppression des $k-1$ arêtes de plus grand poids dans un ACM T représentent un k-clustering d'espacement maximal.*

5.4 *Décrire en pseudo-language une adaption de l'algorithme "Reverse-Delete" qui calcule un k-clustering d'espacement maximal.*