TD 8 - Algorithmes gloutons.

Exercice 1.

Étant donné un graphe non-orienté $G = (V, E)$, un *coloriage* de G est une fonction affectant une couleur à chaque sommet de G de sorte que deux sommets adjacents n'aient jamais la même couleur. Dans la suite, on suppose que chaque sommet du graphe est numéroté par un identifiant unique dans $\{1, 2, \ldots, |V|\}$, et on utilisera les mêmes nombres pour désigner les couleurs utilisées dans les algorithmes de coloriage.

On s'intéresse au problème de trouver un coloriage d'un graphe utilisant le moins de couleurs différentes possible. Une approche gloutonne pour résoudre ce problème consiste à prendre les sommets dans l'ordre où ils sont numérotés et à leur affecter la plus petite couleur disponible (c'est-à-dire qui n'est pas déjà affectée à un de ses voisins).

1. Écrivez le pseudocode de l'algorithme correspondant et expliquez pourquoi cette approche est correcte (c'est-à-dire que lorsque l'algorithme se termine, la numérotation trouvée est bien un coloriage ; on ne prétend pas qu'elle est optimale).
2. Montrez sur un exemple que l'ordre choisi pour les sommets a une influence sur le nombre de couleurs utilisées. Cette approche gloutonne n'est donc pas optimale.
3. Raffinons l'approche précédente en imposant de prendre les sommets par degré décroissant. Testez cette stratégie sur quelques exemples.
4. Montrez que cette stratégie utilise au plus $\Delta + 1$ couleurs où Δ est le degré maximal des sommets du graphe.
5. Montrez que cette stratégie n'est toujours pas optimale, en donnant un graphe pour lequel l'algorithme construit un coloriage qui utilise plus de couleurs que nécessaire.

Le nombre minimal de couleurs nécessaires pour colorier le graphe G est appelé nombre chromatique de G. Déterminer le nombre chromatique d'un graphe de manière efficace est un problème difficile étudié depuis longtemps. C'est un des exemples classiques de problème NP*-difficile. Nous en reparlerons en M1.*

Exercice 2.

Un automobiliste veut effectuer un trajet de n kilomètres pour rejoindre un ami. Le réservoir plein, sa voiture peut parcourir k kilomètres. Sa carte lui donne les distances entre les stations-service sur la route. Chaque arrêt faisant perdre du temps, l'automobiliste souhaite planifier les stations où il fera le plein de manière à minimiser le nombre d'arrêts effectués.

Dans la suite, on note $s_0, s_1, \ldots, s_\ell$ les distances des différentes stations-service au point de départ dans l'ordre de rencontre sur le trajet. De plus, on suppose que l'on a toujours une station-service au point de départ — c'est-à-dire que $s_0 = 0$ — et que l'automobiliste s'arrêtera forcément après n kilomètres.

1. Quelle propriété doit vérifier la séquence $(s_0, s_1, \ldots, s_\ell)$ pour que l'automobiliste puisse effectuer son trajet ?
2. En supposant que le trajet est faisable, proposez une stratégie gloutonne pour choisir les stations auxquelles s'arrêter en essayant de minimiser le nombre d'arrêts.
3. Écrivez le pseudocode de l'algorithme implémentant votre stratégie.
   L'algorithme prend en entrée n, k et la séquence $S = (s_0, s_1, \ldots, s_\ell)$ et renverra une sous-séquence de S correspondant aux arrêts à effectuer lorsque le trajet est faisable et "Infaisable" sinon.
4. Votre stratégie est-elle optimale ? Autrement dit, votre algorithme produit-il une solution minimisant le nombre d'arrêts ? Démontrez votre réponse.

Exercice 3.

Lorsqu'on rend la monnaie, la personne qui la reçoit a souvent envie de recevoir le moins de pièces possibles. L'exercice suivant propose de voir si on peut trouver une stratégie gloutonne pour le rendu de monnaie et si une telle stratégie est optimale.

1. Le système de monnaie européen est composé des pièces des valeurs suivantes (données en centimes) : 1, 2, 5, 10, 20, 50, 100, 200. Proposez une stratégie gloutonne pour rendre la monnaie dans ce système monétaire en essayant de limiter le nombre de pièces utilisées. On suppose que le nombre de pièces à disposition n'est pas limité.
2. L'ancien système britannique proposait les pièces suivantes : 1, 3, 6, 12, 24, 30, 60, 240. Votre stratégie gloutonne est-elle optimale pour ce système monétaire ?
3. Écrivez le pseudocode de l'algorithme correspondant à votre stratégie pour un système monétaire quelconque, représenté par un tableau de valeurs triées dans l'ordre décroissant. L'algorithme prend en entrée ce tableau de valeurs P et une somme n et renvoie une liste des pièces représentant cette somme.
   Pour simplifier les choses, c'est-à-dire pour que le rendu soit toujours possible, on suppose que le système contient toujours la pièce de valeur 1.

Exercice 4.

Les problèmes suivants sont des exemples très célèbres de problèmes algorithmiquement difficiles. Pour chacun de ces problèmes, exposez en quelques mots (sans pseudocode) une stratégie gloutonne, et prouvez que votre stratégie n'est pas optimale :

1. INDEPENDENT SET : étant donné un graphe non orienté $G = (V, E)$, trouvez un ensemble $U \subseteq V$ de cardinalité maximale tel qu'aucune paire de sommets de U n'est reliée par une arête.
2. MAX CUT : étant donné un graphe non orienté $G = (V, E)$, trouvez une bipartition $V = V_1 \cup V_2$ maximisant le nombre d'arêtes ayant une extrémité dans chaque sous-ensemble.
3. STRONG CONNECTIVITY AUGMENTATION : étant donné un graphe orienté $G = (V, A)$, trouvez un ensemble d'arcs de taille minimale à rajouter à G pour qu'il devienne fortement connexe.