IFT436 - Série d'exercices #5 : les algorithmes gloutons

Manuel Lafond

Exercice 1: Dans une journée, un certain professeur X a t plages de rendez-vous disponibles, chacune pouvant être allouée à un étudiant qui nécessite un entretien. Chaque étudiant a une plage maximum possible, e.g. un étudiant demande un rendez-vous avant la plage i, avec $1 \leq i \leq t$. Il s'avère que X est une personne ignoble, et affecte une importance différente à chaque étudiant. Le but est d'affecter chacune des t plages de temps à un étudiant de façon à maximiser la somme des importances des étudiants traités.
Formellement, on a le problème suivant:

Problème du professeur discriminant
Entrée: un ensemble $E = \{(x_1, p_1), (x_2, p_2), \ldots, (x_n, p_n)\}$, où chaque (x_i, p_i) représente un étudiant: x_i est le temps maximum possible pour un rendez-vous, et p_i est l'importance de l'étudiant.
Sortie: un tableau $T = [(x'_1, p'_1), \ldots, (x'_t, p'_t)]$ de t étudiants distincts, où $T[i]$ est l'étudiant assigné au temps i. On peut permettre $T[i] = nil$ si on n'affecte personne à la plage i. On exige que pour tout $i \in \{1, \ldots, t\}$, si l'élément $T[i] = (x'_i, p'_i)$ est non-nil, il satisfait $i \leq x_i$. On veut que $\sum_{i=1}^{k} p'_i$ soit maximum parmi tous les choix possibles.

Considérez l'algorithme glouton suivant qui insère les étudiants en ordre d'importance, lorsque c'est possible, le plus tard possible.

fonction
professeurDiscriminant($E = \{(x_1, p_1), (x_2, p_2), \ldots, (x_n, p_n)\}, t$)
Trier E selon les importances p_i, en ordre décroissant
$T = [nil] * t$
pour $i = 1..n$ **faire**
pour $j = x_i..1$ **faire**
si $T[j] = nil$ **alors**
$T[j] = (x_i, p_i)$
break `// Sortir de la boucle pour` j
fin
fin
return T

Montrez que cet algorithme est optimal, c'est-à-dire qu'il maximise la somme des importances.

Exercice 2: Supposons que l'on vit à une époque où les pièces de monnaie sont 25 sous, 10 sous, 5 sous et 1 sou. On doit rendre à un client un certain nombre de sous t en minimisant le nombre de pièces redonnées. Par exemple pour rendre 98 sous, on redonnerait $3 \times 25 + 2 \times 10 + 3 \times 1$ pièces. Plus généralement, on pourrait avoir un ensemble d'entiers P dénotant les valeurs de pièces disponibles (donc dans l'exemple, $P = \{25, 10, 5, 1\}$. L'algorithme glouton fait la chose suivante:

fonction *minPieces*(t, P) *t est la somme cible, P les valeurs de pièces*
$R = ()$
$s = 0$
tant que $s < t$ **faire**
Soit p la valeur maximum de P telle que $s + p \leq t$
$s = s + p$
Ajouter p à R
fin
return R

Montrez que cet algorithme fonctionne correctement avec $P = \{25, 10, 5, 1\}$. Montrez ensuite qu'il ne fonctionne pas sur certaines entrée P.

Exercice 3: Ceci est la question de l'intra 2019. On a un véhicule qui veut traverser un chemin $(s_1, \ldots, s_n)$ avec un réservoir de capacité r, initialement plein. Le parcours de s_i à s_{i+1} nécessite $f(s_i, s_{i+1})$ en essence, et on ne doit jamais tomber en panne d'essence. Chaque fois que le véhicule arrive à un s_i, qui représente une station, il peut s'arrêter pour faire le plein ou bien continuer. Le but est de s'arrêter à un nombre minimum de stations pour passer de s_1 à s_n. Pour $i < j$, on définit $dist(s_i, s_j) = \sum_{k=i}^{j-1} f(s_k, s_{k+1})$.

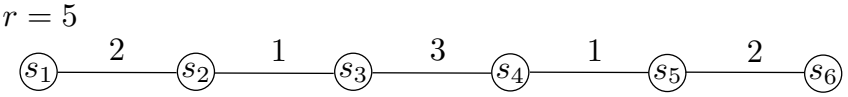

Figure 1: Un exemple d'instance du problème avec $r = 5$. Les arêtes sont étiquetées par les quantités d'essence requises. Une solution possible serait d'arrêter à s_2 et à s_4 pour faire le plein.

Ceci se formalise comme suit.

Entrée: une capacité r, une séquence $(s_1, \ldots, s_n)$ et une fonction f qui attribue un valeur $f(s_i, s_{i+1}) \leq r$ à chaque paire s_i, s_{i+1}, $i \in \{1, \ldots, n-1\}$.
Sortie: une sous-séquence de stations $s_1 = s'_1 < s'_2 < \ldots < s'_k = s_n$ contenant un nombre minimum d'éléments et telle que $dist(s'_i, s'_{i+1}) \leq r$ pour tout $i \in \{1, \ldots, k-1\}$.

Considérez l'algorithme glouton suivant:

fonction *minPlein*$(G = (V, E, f), d)$
  $S = [s_1]$
  $s_p = s_1$ `// ` s_p `est l'emplacement courant`
  tant que $s_p \neq s_n$ **faire**
    $s_p = s_i$, où $i \in \{p+1, \ldots, n\}$ est l'indice maximum tel que
     $dist(s_p, s_i) \leq r$
    $S.append(s_p)$
  fin
  return S

Montrez que l'algorithme glouton *minPlein* retourne effectivement le nombre minimum de stations à utiliser.