TRAVAUX PRATIQUES : Algorithmes gloutons

Le principe d'un algorithme glouton (en anglais *greedy algorithm*) est de faire, à chaque étape de l'algorithme, un choix localement optimal dans l'espoir que ce choix mènera à une solution globalement optimale.

I - Deux exercices d'entrainement

Exercice 1: Rendu de monnaie (Solution)

1. On dispose de pièces (ou billets) de $10, 5, 2, 1$ euros. On doit rendre 29 euros et on souhaite le faire de manière optimale (avec un minimum de pièces).

   Proposez une solution gloutonne c'est-à-dire en commençant par rendre avec des pièces de 10 euros, puis 5 euros, puis 2 puis 1.

2. Imaginons que l'on dispose de pièces de 18, 7 et 1 euros (!). On souhaite rendre 21 euros.

   Proposez une solution gloutonne. Est-elle optimale ?

3. L'objectif est d'écrire une fonction utilisant le principe d'un algorithme glouton dans un problème de rendu de monnaie. Écrire une fonction `Rendu(Montants,Prix)` dont les arguments sont :
   - Une liste `Montants=[M1,...,Mp]` contenant les montants disponibles pour rendre la monnaie. Cette liste est classée par ordre décroissant de montants.
   - Un entier `Prix`.

   Cette fonction renvoie une liste de couples `L=[(M1,n1),...,(Mp,np)]` indiquant pour chaque montant le nombre de pièces à rendre.

4. Soit $p \geqslant 2$ un entier.

   On considère la liste de montants $M = [p^k, p^{k-1}, \dots, p, 1]$. On souhaite montrer qu'avec cette liste de montants, un algorithme de rendu de monnaie glouton est optimal.

   (a) Combien de pièces de p^{k-1} euros peut-on utiliser au maximum dans un algorithme glouton ? de pièces de p^{k-2} euros ?

   (b) Calculer à la main la somme maximale pouvant être rendue en suivant ce principe si l'on n'utilise que les pièces de $p^{k-1}, p^{k-2}, ..., p, 1$ euros.

   (c) Conclure.

Exercice 2: Gestion de salle (Solution)

On reçoit une liste de demande de réservation d'une salle sous la forme d'une liste `[(d1,f1),...(dn,fn)]` composée de couples d'horaires (d_i, f_i). On suppose que la liste est ordonnée suivant les horaires de fin f_i croissants : $f_1 \leqslant f_2 \leqslant \cdots \leqslant f_n$. Écrire une fonction `gestion(L)` qui, étant donnée un telle liste de demandes, renvoie une proposition d'attribution de la salle satisfaisant au maximum de demande à l'aide d'un algorithme glouton.

II - Un sujet d'annales

Exercice 3: Un sujet d'annales (Solution)

Poteaux télégraphiques

Cette épreuve a pour objectif de choisir où placer des poteaux télégraphiques pour relier le point le plus à gauche d'un paysage unidimensionnel au point le plus à droite en fonction de critères de coût. Nous ferons les simplifications suivantes : les fils sont sans poids et tendus ; ils relient donc en ligne droite les sommets de deux poteaux consécutifs. Les normes de sécurité imposent que les fils soient en tout point à une distance d'au moins Δ (mesurée verticalement) au-dessus du sol. Les poteaux sont tous de longueur identique $\ell \geqslant \Delta$. Voici par exemple une proposition valide de placement de poteaux pour le paysage ci-dessous avec $\Delta = 0.5$ et $\ell = 2.0$.

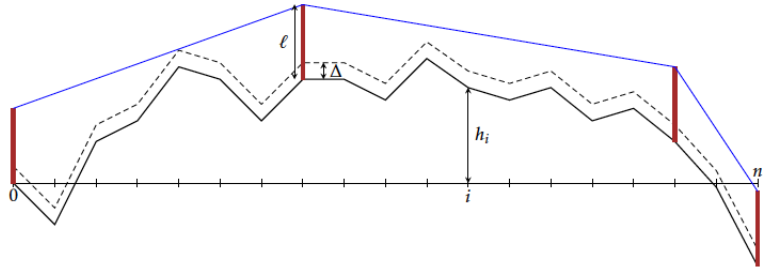

Figure 1 – Un exemple de poteaux où le fil est en tout point au moins Δ au-dessus du sol.

Dans tout le sujet, les tableaux sont indicés à partir de 0. L'accès à la $(i+1)$-ème case d'un tableau t de taille $n+1$, notée $t[i]$, n'est donc valide que pour i entier compris entre 0 et n au sens large. Tous les tableaux définis dans ce problème seront considérés comme des variables globales, accessibles et modifiables directement par toutes les procédures et fonctions.
Les booléens `true` et `false` sont utilisés dans certaines questions de ce problème. La fonction `sqrt` qui calcule la racine carrée d'un nombre positif ou nul pourra être utilisée.

La complexité, ou le temps d'exécution, d'un programme P (fonction ou procédure) est le nombre d'opérations élémentaires (addition, soustraction, multiplication, division, affectation, etc...) nécessaires à l'exécution de P. Lorsque cette complexité dépend d'un paramètre n, on dira que P a une complexité en $O(f(n))$, s'il existe $K > 0$ tel que la complexité de P est au plus $K\,f(n)$, pour tout n. une certaine complexité, le candidat devra justifier cette dernière si elle ne se déduit pas directement de la lecture du code.

Partie I. Planter le paysage

Nous définissons le paysage à partir d'une suite de relevés de dénivelés stockés dans un tableau `deniveles` de nombres flottants (c'est-à-dire des nombres à virgule) de taille $n+1$. Le tableau `deniveles` est supposé être une variable globale.
Le paysage est alors décrit par une ligne brisée de $n+1$ points, dont le i-ème point P_i est de coordonnées (i, h_i) où :

$$h_0 = 0.0 \text{ et } h_i = h_{i-1} + \texttt{deniveles}[i] \text{ pour } i \in \{1, ..., n\}.$$

Question 1 Écrire une procédure `calculHauteurs`(n) qui stocke les hauteurs des points dans une variable globale `hauteurs` représentant un tableau de taille $n+1$.

Question 2 Écrire une procédure `calculFenetre`(n) qui calcule les hauteurs minimale et maximale d'un point du paysage et les stocke dans deux variables globales `hMin` et `hMax`. Cette procédure stockera également dans deux variables globales `iMin` et `iMax` les indices des points les plus à gauche de hauteur minimale et maximale respectivement.
On appelle *distance* au sol de i à j, la longueur de la ligne brisée allant du point P_i au point P_j.

Question 3 Écrire une fonction `distanceAuSol`(i, j) qui calcule et renvoie la distance au sol de P_i à P_j.
On appelle un *pic* un point P_i d'indice $1 \leqslant i < n$ tel que $h_i > \max(h_{i-1}, h_{i+1})$.
On appelle *point remarquable*, les pics et les points des bords gauche (point P_0) et droit (point P_n). On appelle *bassin* toute partie du paysage allant d'un point remarquable au suivant.

Question 4 Écrire une fonction `longueurDuPlusLongBassin`(n) qui calcule et renvoie la longueur au sol maximale d'un bassin dans le paysage.

Partie II. Planter les poteaux

On souhaite relier le point le plus à gauche au point le plus à droite par un fil télégraphique. Pour cela, nous devons choisir à quels points parmi les $(P_i)_{0 \leqslant i \leqslant n}$ planter les poteaux télégraphiques intermédiaires. Rappelons que le fil est supposé sans poids et tendu et qu'il relie donc les sommets de chacun des poteaux en ligne droite. Les poteaux sont plantés verticalement et ont tous une longueur identique $\ell \geqslant \Delta$ (ℓ et Δ sont des nombres flottants).
La législation impose que le fil doit rester à une distance supérieure ou égale à Δ (mesurée verticalement) au-dessus du sol. Pour tester si un fil tiré entre un poteau placé au point P_i et un poteau placé au point P_j (avec $j > i$ ou $j < i$) respecte la législation, notons

$$\alpha_{i,k} = \frac{(h_k + \Delta) - (h_i + \ell)}{k - i} \text{ pour } k \neq i, \text{ et } \beta_{i,j} = \frac{(h_j + \ell) - (h_i + \ell)}{j - i},$$

les pentes des fils tirés depuis le poteau en P_i jusqu'à une hauteur Δ au-dessus du point P_k d'une part et jusqu'à une hauteur ℓ au-dessus du point P_j d'autre part (voir la Figure 2).
On admet que le fil tiré d'un poteau en P_i à un poteau en respecte la législation si et seulement si :

$$\beta_{i,j} \geqslant \max_{i<k<j} \alpha_{i,k}, \text{ lorsque } j > i; \text{ et } \beta_{i,j} \leqslant \min_{j<k<i} \alpha_{i,k}, \text{ lorsque } j < i.$$

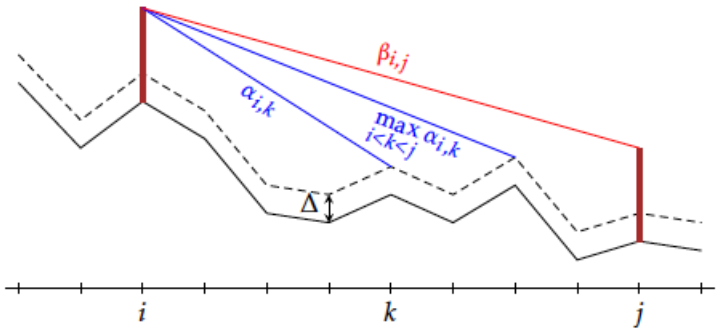

Figure 2 – Condition sur les pentes pour pouvoir tirer un fil.

Question 5 En utilisant cette méthode, écrire une fonction `estDeltaAuDessusDuSol`(i, j, ℓ, Δ) qui renvoie `true` si un fil tiré entre les sommets d'un poteau placé au point P_i et d'un poteau placé au point P_j respecte la législation, et renvoie `false` dans le cas contraire.

Considérons une première stratégie, dite *algorithme glouton* en avant. Le premier poteau est planté en P_0. Pour calculer l'emplacement du prochain poteau, on part du dernier poteau planté et on avance (à droite) avec le fil tendu tant que

la législation est respectée (et que P_n n'est pas atteint). Un nouveau poteau est alors planté, et on recommence jusqu'à ce que P_n soit atteint.
La figure 3 illustre la solution produite par cet algorithme.

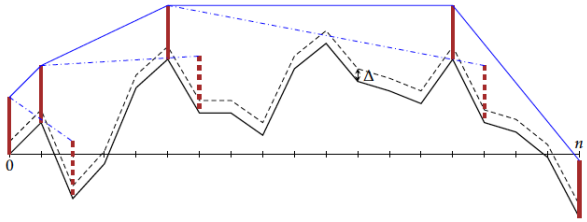

FIGURE 3 – Solution produite par l'algorithme glouton en avant. Les poteaux et les fils en pointillés sont ceux violant la législation.

Question 6 Écrire une procédure `placementGloutonEnAvant`(n, ℓ, Δ) qui calcule la disposition des poteaux selon l'algorithme ci-dessus. La solution sera stockée dans un tableau global `poteaux` de la façon suivante :
- la case `poteaux`[0] contiendra le nombre de poteaux utilisés ;
- la case `poteaux`[t], pour $t \geqslant 1$, contiendra l'indice i indiquant que le t-ème poteau, s'il existe, est planté en P_i

Question 7 Donner une majoration de la complexité du temps de calcul de votre algorithme en fonction de n. Justifier qu'on peut se contenter du calcul de $O(n)$ pentes pour implémenter l'algorithme glouton en avant. Expliquer succinctement comment modifier votre procédure (si nécessaire) pour obtenir une complexité en $O(n)$. *Aucune implémentation n'est demandée dans cette question.*

L'algorithme glouton en avant a tendance à placer beaucoup trop de poteaux, en particulier dans les vallées alors qu'il suffirait de relier les deux extrémités par un unique fil. Nous considérons donc une alternative, dite glouton au plus loin, qui consiste à planter le prochain poteau le plus à droite possible de la position courante. Le premier poteau est toujours planté en P_0.
La figure 4 illustre la solution produite par cet algorithme sur le même paysage que celui de la figure 3.

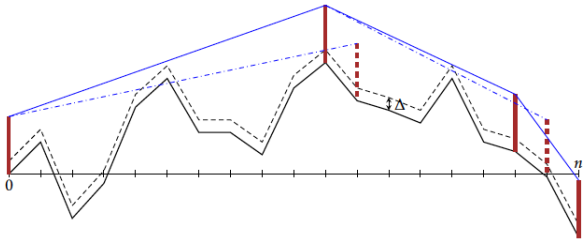

FIGURE 4 – Solution produite par l'algorithme glouton au plus loin. Les poteaux et fils en pointillés sont ceux violant la législation.

Question 8 Écrire une procédure `placementGloutonAuPlusLoin`(n, ℓ, Δ) qui calcule la disposition des poteaux selon l'algorithme ci-dessus. La solution sera stockée dans un tableau global `poteaux` de la façon suivante :
- la case `poteaux`[0] contiendra le nombre de poteaux utilisés ;
- la case `poteaux`[t], pour $t \geqslant 1$, contiendra l'indice i indiquant que le t-ème poteau, s'il existe, est planté en P_i.

Partie III. Minimiser la longueur du fil

L'objectif est maintenant de calculer un placement optimal des poteaux **en terme de longueur totale du fil** (le nombre de poteaux pouvant être maintenant arbitraire). Un tableau `optL` peut être calculé de proche en proche tel que `optL`[i] désigne la longueur minimale de fil à utiliser si l'on souhaitait relier le point P_0 au point P_i.

Question 9 Soit $L_{i,j}$ la longueur du segment d'extrémités P_i et P_j.
Montrer que le tableau `optL` vérifie : `optL`[0] $= 0$, et pour $1 \leqslant i \leqslant n$,

$$\texttt{optL}[i] = \min\{L_{i,j} + \texttt{optL}[j],\ \text{où } 0 \leqslant j < i \text{ et il est possible de tirer un fil de } P_j \text{ à } P_i\}.$$

La case `optL`[n] contient alors la longueur minimale de fil pour relier le bord gauche au bord droit.

Question 10 Écrire une fonction `longueurMinimale`(n, ℓ, Δ) qui renvoie la longueur minimale de fil pour relier le bord gauche au bord droit, en respectant la législation.

Question 11 Modifier votre procédure `longueurMinimale`(n, ℓ, Δ) pour qu'elle remplisse également un tableau `precOptL` de taille $n+1$, où `precOptL`[i] contient l'indice du poteau précédant le poteau placé en P_i dans une solution de longueur minimale reliant P_0 à P_i. (Par convention, `precOptL`[0] = -1.)

Question 12 Écrire une procédure qui produit un placement des poteaux minimisant la longueur du fil de P_0 à P_n à partir du tableau `precOptL`. La solution sera stockée dans un tableau global `poteaux` de la façon suivante :
- la case `poteaux`[0] contiendra le nombre de poteaux utilisés ;
- la case `poteaux`[t], pour $t \geqslant 1$, contiendra l'indice i indiquant que le t-ème poteau, s'il existe, est planté en P_i.

Quelle est la complexité de votre procédure en fonction de n ?

* *
*