Poteaux télégraphiques

(d'après X PSI/PT 2013)

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 $\ell = 5\Delta$).

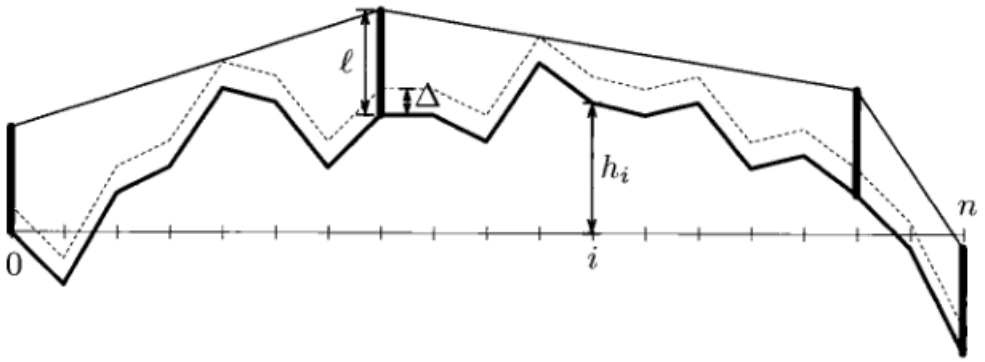

fig1 : Un exemple de poteaux où le fil est en tout point au moins Δ au-dessus du sol.

Dans tout le sujet, la fonction `sqrt` qui calcule la racine carrée d'un nombre positif ou nul pourra être utilisée, ainsi que les fonctions `min` et `max` qui renvoient le minimum et le maximum d'une liste de réels.
La complexité, ou le temps d'exécution, d'une fonction `P` 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. Lorsqu'il est demandé de garantir une certaine complexité, le candidat devra justifier cette dernière si elle ne se déduit pas directement de la lecture du code.

Nous attacherons la plus grande importance à la lisibilité du code produit par les candidats ; aussi, nous encourageons les candidats à introduire des fonctions intermédiaires lorsque cela en simplifie l'écriture.

Partie I. Planter le paysage

Nous supposons que le paysage est représenté par une liste de points P_i, $0 \leqslant i \leqslant n$, de coordonnées (i, h_i) où h_i est l'altitude du point P_i. Le paysage est alors représenté par une ligne brisées de $n+1$ points.

1. Écrire une fonction `Fenetre(H:list)->tuple` qui prend en argument une liste `H` contenant les altitudes des points P_i (`H[i]` contient h_i, l'altitude du point P_i) et qui renvoie les hauteurs minimale et maximale d'un point du paysage et les indices des points les plus à gauche de hauteur minimale et maximale respectivement.

2. On appelle *distance* au sol de i à j, la longueur de la ligne brisée allant du point P_i au point P_j.

Écrire une fonction `distanceAuSol(H:list,i:int,j:int)->float` qui prend en argument une liste `H` contenant les altitudes des points du paysage, deux indices `i` et `j` valides de `H` et qui calcule et renvoie la distance au sol de P_i à P_j.

3. On appelle un *pic* un point P_i d'indice $0 < i < n$ 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.

Écrire une fonction `Remarquables(H:list)->list` qui prend en argument une liste `H` contenant les altitudes des points du paysage et qui calcule et renvoie la liste des points remarquables.

4. Écrire une fonction `PlusLongBassin(H:list)->float` qui prend en argument une liste `H` contenant les altitudes des points du paysage et 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 :

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

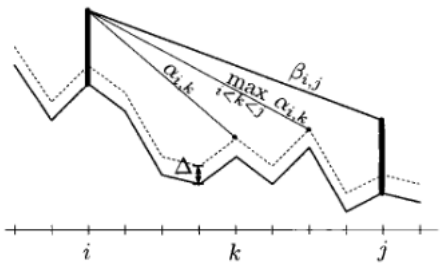

fig 2 : Condition sur les pentes pour pouvoir tirer un fil

1. En utilisant cette méthode, écrire une fonction `Legal(H:list,i:int,j:int:l:float,delta:float)->bool` qui prend en argument une liste `H` contenant les altitudes des points du paysage, deux indices i et j valides de `H`, et deux flottants `l` et `delta` (représentant respectivement les réels ℓ et Δ) et 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.

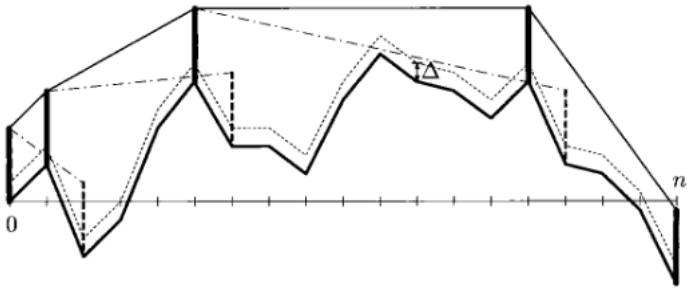

fig 3 : Solution produite par l'algorithme glouton en avant.
Les poteaux et les fils en pointillés sont ceux violant la législation.

2. Écrire une fonction `GloutonEnAvant(H:list,l:float,delta:float)->list` qui prend en argument une liste `H` contenant les altitudes des points du paysage et deux flottant `l` et `delta` (représentant respectivement les réels ℓ et Δ) et qui renvoie une liste `poteaux` telle que `poteaux`$[t]$, pour $t \geqslant 1$, contienne l'indice i indiquant que le t-ème poteau, s'il existe, est planté en P_i.

3. 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 nouvelle 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.

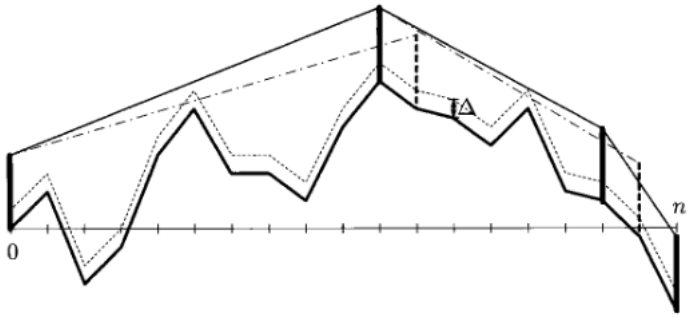

fig 4 : Solution produite par l'algorithme glouton au plus loin.
Les poteaux et fils en pointillés sont ceux violant la législation.

4. Écrire une fonction `GloutonAuPlusLoin(H:list,l:float,delta:float)->list` qui prend en argument une liste `H` contenant les altitudes des points du paysage et deux flottants `l` et `delta` (représentant respectivement les réels ℓ et Δ) et qui renvoie une liste `poteaux` telle que `poteaux`$[t]$, pour $t \geqslant 1$, contienne 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).

1. Pour un paysage à $n+1$ points, combien de placements de poteaux sont possibles ?
En déduire une estimation de la complexité d'un algorithme qui consisterait à calculer la longueur minimale du fil en examinant toutes les configurations possibles. *Aucune implémentation d'une telle fonction n'est demandée.*

2. On note L_i la longueur minimale d'un fil permettant de relier le point P_0 et le point P_i, avec $i \in [\![0, n+1]\!]$ (en respectant la législation).
Justifier que

$$L_0 = 0 \quad \text{et} \quad L_i = \min\{P_jP_i + L_j, \text{ où } 0 \leqslant j < i \text{ et il est possible de tirer un fil de } P_j \text{ à } P_i\},$$

où P_jP_i désigne la longueur du segment $[P_j, P_i]$.

3. Écrire une fonction `Minimale(H:list,l:float,delta:float)->float` qui prend en argument une liste `H` contenant les altitudes des points du paysage et deux flottants `l` et `delta` (représentant respectivement les réels ℓ et Δ) et qui renvoie la longueur minimale de fil pour relier le bord gauche au bord droit, en respectant la législation.
Donner une majoration de la complexité de votre fonction (en fonction de n)

4. Écrire une fonction `PosMinimale(H:list,l:float,delta:float)->list` qui prend en argument une liste `H` contenant les altitudes des points du paysage et deux flottants `l` et `delta` (représentant respectivement les réels ℓ et Δ) et qui renvoie une liste `poteaux` telle que `poteaux`$[t]$, pour $t \geqslant 1$, contienne l'indice i indiquant que le t-ème poteau, s'il existe, est planté en P_i dans une configuration pour laquelle la longueur du fil reliant P_0 et P_n est minimale ; on pourra introduire une liste `Opt` pour la quelle `Opt[i]` contient la position du poteau précédent P_i dans une configuration optimale (en terme de longueur de fil) reliant P_0 et P_i.
Donner une majoration de la complexité de votre fonction (en fonction de n)