# Poteaux télégraphiques (X PSI 2013)

Durée : 2 heures

---

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 $\Delta$ (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 $\ell = 2.0$.

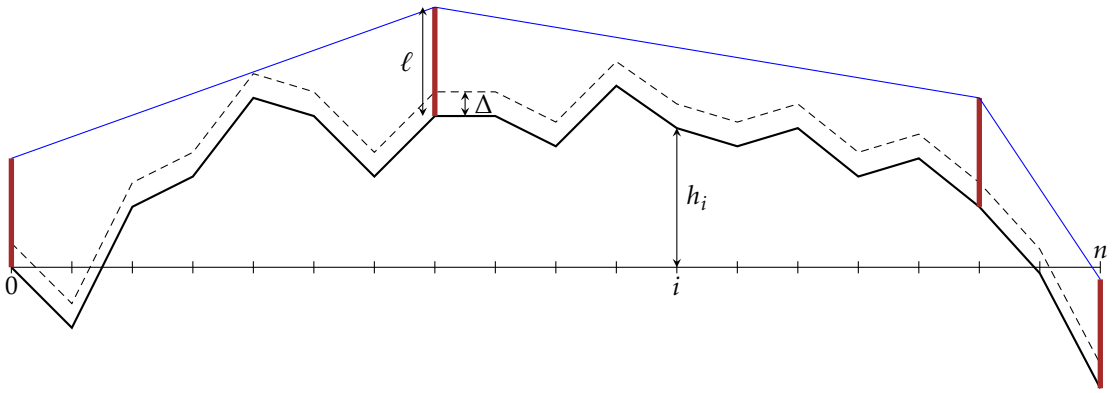


Figure 1 – Un exemple de poteaux où le fil est en tout point au moins $\Delta$ au-dessus du sol.

Dans tout le sujet, la fonction `sqrt` qui calcule la racine carrée d'un nombre positif ou nul pourra être utilisée.

*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 définissons le paysage à partir d'une suite de relevés de dénivelés stockés dans une liste `deniveles` de nombres flottants de taille $n+1$. L'entier `n` et la liste `deniveles` sont supposés être des variables globales.

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 :

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

**1.1.** Écrire un script Python qui stocke les hauteurs des points dans une variable globale `hauteurs` représentant une liste de taille $n+1$.

La variable globale `hauteurs` pourra désormais être utilisée dans la suite du sujet.

**1.2.** Écrire un script Python qui calcule les hauteurs minimale et maximale d'un point du paysage et les stocke dans deux variables globales `hMin` et `hMax`. On stockera également dans deux variables globales `iMin` et `iMax` les indices des points les plus à gauche de hauteur minimale et maximale respectivement.

Pour tout $0 \leqslant i \leqslant j \leqslant n$ on appelle *distance au sol* de $i$ à $j$, la longueur de la ligne brisée allant du point $P_i$ au point $P_j$.

**1.3.** 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 ainsi que 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.

**1.4.** Écrire une fonction `longueurDuPlusLongBassin()` qui calcule et renvoie la distance 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 $\ell \geqslant \Delta$.
$\ell$ et $\Delta$ seront respectivement représentées par les variables globales `l` et `delta`.

La législation impose que le fil doit rester à une distance supérieure ou égale à $\Delta$ (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 $i < j$) respecte la législation, notons

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

les pentes des fils tirés depuis le poteau en $P_i$ jusqu'à une hauteur $\Delta$ au-dessus du point $P_k$ d'une part et jusqu'à une hauteur $\ell$ au-dessus du point $P_j$ d'autre part (voir la Figure 2).

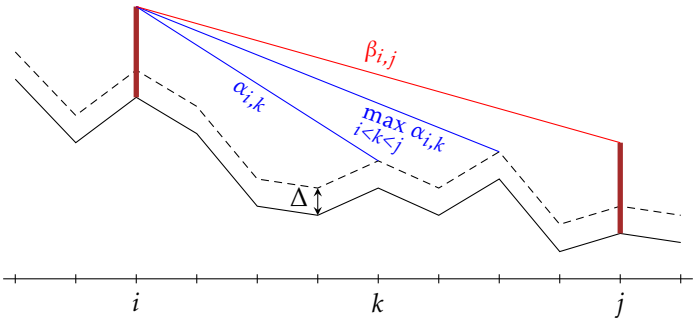


Figure 2 – Condition sur les pentes pour pouvoir tirer un fil.

On admet que le fil tiré d'un poteau en $P_i$ à un poteau $P_j$ en respecte la législation si et seulement si :

$$\beta_{i,j} \geqslant \max_{i<k<j} \alpha_{i,k}$$

**2.5.** En utilisant cette méthode, écrire une fonction `estDeltaAuDessusDuSol(i, j)` 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.

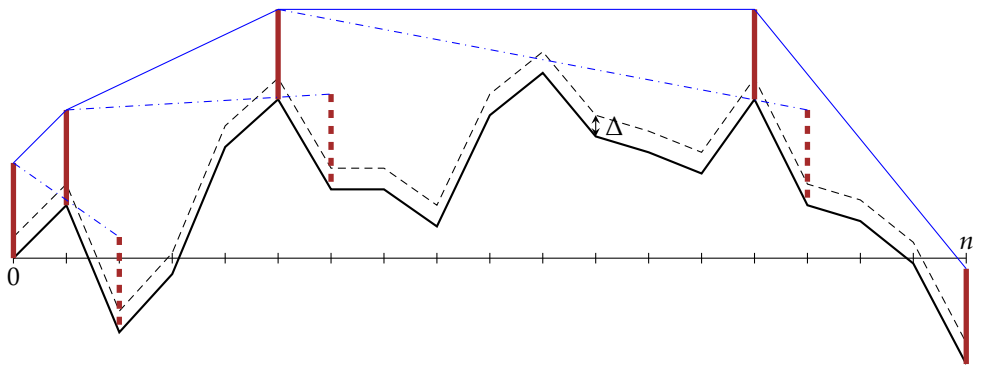


Figure 3 – Solution produite par l'algorithme glouton en avant. Les poteaux et les fils en pointillés sont ceux violant la législation. La fonction `placementGloutonEnAvant()` renvoie ici la liste `[0, 1, 5, 14, 18]`

**2.6.** Écrire une fonction `placementGloutonEnAvant()` qui calcule la disposition des poteaux selon l'algorithme ci-dessus. La solution retournée sera donnée sous la forme d'une liste `poteaux` dans lequel la case `poteaux[t]`, pour $t \geqslant 0$, contiendra l'indice $i$ indiquant que le $(t+1)$-ème poteau, s'il existe, est planté en $P_i$.

**2.7.** 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, et modifier votre fonction (si nécessaire) pour obtenir une complexité en $O(n)$.

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.

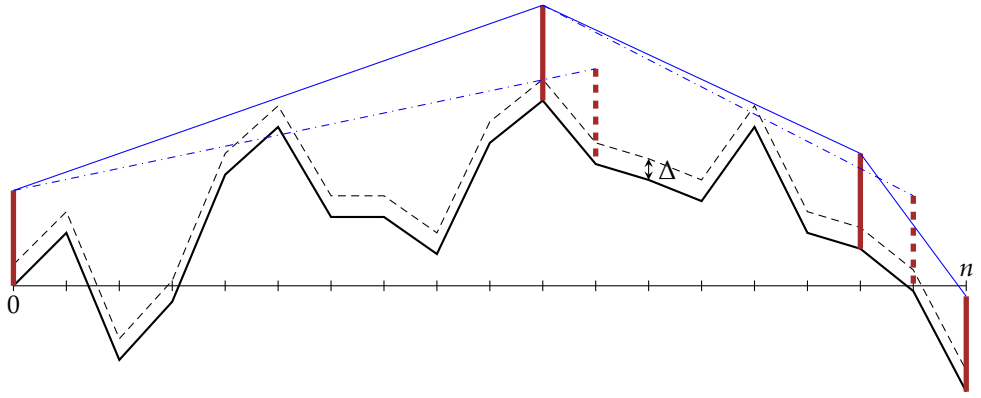


FIGURE 4 – Solution produite par l'algorithme glouton au plus loin. Les poteaux et fils en pointillés sont ceux violant la législation. La fonction `placementGloutonAuPLusLoin()` renvoie ici la liste `[0, 10, 16, 18]`

**2.8.** Écrire une fonction `placementGloutonAuPlusLoin()` qui calcule la disposition des poteaux selon l'algorithme ci-dessus. La solution sera donnée sous la forme d'une liste `poteaux` dans lequel la case `poteaux[t]`, pour $t \geqslant 0$, contiendra l'indice $i$ indiquant que le $(t+1)$-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). Une liste `optL` peut être calculée 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 $P_i$.

**3.9.** 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 $1 \leqslant i \leqslant n$,

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

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

**3.10.** Écrire une fonction `longueurMinimale()` qui renvoie la longueur minimale de fil pour relier le bord gauche au bord droit, en respectant la législation.

**3.11.** Modifier votre fonction `longueurMinimale` pour qu'elle retourne une liste `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]` $= -1$.)

**3.12.** Écrire une fonction qui retourne un placement des poteaux minimisant la longueur du fil de $P_0$ à $P_n$ à partir du tableau `precOptL`. La solution sera retournée sous la forme d'une liste `poteaux` dans lequel la case `poteaux[t]`, pour $t \geqslant 0$, contiendra l'indice $i$ indiquant que le $(t+1)$-ème poteau, s'il existe, est planté en $P_i$.
Quelle est la complexité de votre procédure en fonction de $n$ ?