Questions de cours

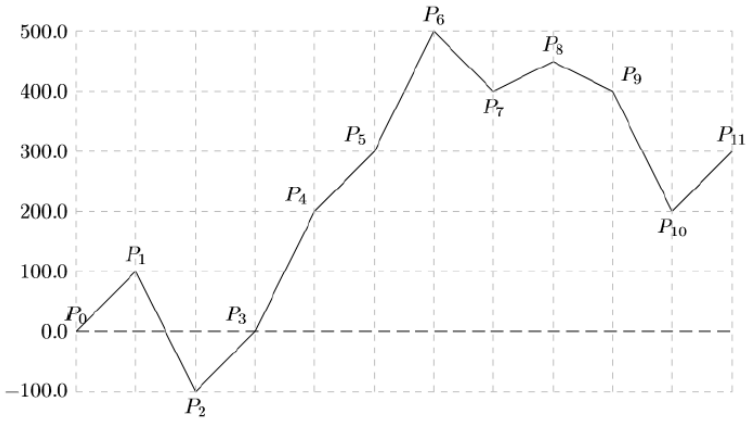
- Écrire une fonction `Occurrences(L)` qui prend comme paramètre une liste et renvoie un dictionnaire dont les clés sont les éléments de `L` et la valeur associée à une clé est le nombre d'occurrences de cette clé dans la liste `L`
- Écrire une fonction `Parcours(G,sommet)` qui code un parcours en largeur ou en profondeur de façon récursive ou avec une pile ou une file du graphe `G` partant de `sommet`. On pourra utiliser librement une fonction `ListeVoisins(G,s)` qui renvoie la liste des voisins du sommet `s` dans le graphe `G`.

Problème : algorithme de pour de poteaux télégraphiques

Phil Amand est ingénieur en BTP (Bâtiment et Travaux Public), cette semaine il a pour mission de choisir où placer des poteaux télégraphiques pour relier le point le plus à gauche d'un paysage unidimensionnel (pose en ligne droite) au point le plus à droite. Pour ce faire, une première partie permet l'étude du terrain alors qu'en deuxième partie, l'installation des poteaux télégraphiques est résolue par deux méthodes différentes.

Étude du terrain

Pour décrire le terrain, Phil Amand s'est fait seconder par des géographes qui ont placé des points P_0, $P_1, \ldots, P_n$ en ligne droite sur ce terrain, convenu que le premier point avait une altitude nulle, puis ont noté les dénivelés (en mètres), c'est à dire les différences d'altitude, entre deux points successifs. Cette succession de mesures est placée dans une liste de nombres, que l'on appelle `denivele`, qui sera passé en argument de chaque fonction. Dans tout le problème, on convient qu'aucun dénivelé mesuré n'est nul. Par exemple, pour le terrain schématisé ci-dessous,



on a rassemblé dans le tableau qui suit les valeurs stockées dans la liste `denivele`, ainsi que l'altitude de chaque point par rapport au point de référence.

indice du point	0	1	2	3	4	5	6	7	8	9	10	11
dénivelé (m)	0.0	100.0	-200.0	100.0	200.0	100.0	200.0	-100.0	50.0	-50.0	-200.0	100.0
altitude (m)	0.0	100.0	-100.0	0.0	200.0	300.0	500.0	400.0	450.0	400.0	200.0	300.0

Pour le calcul de l'altitude au point d'indice 3 on trouve 0.0 comme altitude car c'est le résultat de la somme des dénivelés précédents : $0.0 + 100.0 + (-200.0) + 100.0 = 0.0$.

Même si vous n'avez pas su faire une question, vous pouvez utiliser la fonction que vous étiez censé coder pour répondre aux questions suivantes.

1. Écrire une fonction `calcul_altitude` prenant en argument la liste `denivele` et renvoyant une liste de nombres formée par les altitudes des points successivement repérés sur le terrain. Ainsi, la fonction écrite, appelée avec la liste décrivant le terrain donné en exemple, doit renvoyer :
`[0.0, 100.0, -100.0, 0.0, 200.0, 300.0, 500.0, 400.0, 450.0, 400.0, 200.0, 300.0]`.
2. Écrire une fonction `denivele_moyen` qui prend en argument la liste `denivele` et renvoie l'entier correspondant à la moyenne des valeurs absolues des dénivelés (en m). On pourra utiliser `abs(x)` pour calculer la valeur absolue de x.
On utilise la valeur absolue pour éviter la compensation des dénivelés positifs et négatifs et que ainsi la valeur renvoyée corresponde mieux au relief du terrain.
3. Écrire une fonction `calcul_fenetre` qui prend en argument la liste `denivele` et qui renvoie l'altitude maximale `hMax` et l'altitude minimale `hMin` ainsi que les indices `iMax` et `iMin` correspondant.
Ainsi, la fonction `calcul_fenetre` appelée pour notre exemple renvoie les deux tuples `(500.0,6)` et `(-100.0, 2)`.
4. Écrire une fonction `calcul_longueur` prenant en argument la liste `denivele` et renvoyant un nombre égal à la longueur en kilomètres du trajet parcouru «en ligne droite» par un piéton marchant du point P_0 jusqu'au point P_n, en suivant bien évidemment le relief du terrain. Autrement dit, ce nombre est la longueur de la ligne brisée passant par les points $P_0, P_1, \dots, P_n$ dans cet ordre.
Les points sont relevés tous les 100 mètres.

Le point d'indice $i \in [\![1\,;n-1]\!]$ est dit **remarquable** si et seulement c'est un **pic**, c'est à dire son altitude est plus élevée que celles des points d'indice $i-1$ et $i+1$. On convient de plus que les points d'indice 0 et n sont remarquables.

On appelle alors **bassin**, toute suite finie de points consécutifs dont le premier et le dernier éléments sont des points remarquables et dont aucun autre point n'est remarquable. Si on reprend le profil de terrain donné au début de cette partie :

- Les points remarquables sont ceux d'indice 0, 1, 6, 8 et 11.
- Les bassins sont formés par les points dont les indices forment les listes $[0,1]$, $[1,2,3,4,5,6]$, $[6,7,8]$ et $[8,9,10,11]$. La longueur d'un bassin est le nombre d'éléments de la liste des points appartenant au bassin. Dans notre exemple, le bassin le plus long est le deuxième, et sa longueur est 6.

5. Écrire une fonction `remarquable` prenant en argument la liste `denivele` et l'indice `i` d'un point et renvoyant un booléen égal à `True` si et seulement si le point d'indice `i` est remarquable. On rappelle qu'il n'est pas demandé de vérifier si l'indice `i` est valide. Préciser la complexité de la fonction.
6. Écrire une fonction `bassin_max` prenant en argument la liste `denivele` et renvoyant un entier égal à la longueur du plus long bassin du terrain étudié. Préciser la complexité de la fonction.

Installation de poteaux

Phil Amand souhaite relier le point le plus à gauche au point le plus à droite du terrain. Il doit donc choisir à quels points parmi les $n+1$ sélectionnés, il va planter des poteaux télégraphiques, sachant qu'il y en a évidemment un placé au point d'indice 0, et un placé au point d'indice n. On suppose que :

- Les poteaux ont tous la même hauteur, qui sera appelée h dans le problème,
- Les fils sont sans poids et tendus, donc relient en ligne droite les sommets de deux poteaux consécutifs.
- Les normes de sécurité imposent que les fils soient en tout point à une distance minimale du sol (mesurée verticalement), que l'on notera Δ et que l'on suppose bien évidemment plus petite que h.

Pour tout $i \in [\![0\,;n]\!]$, on note h_i l'altitude du poteau d'indice i. Pour tout $(i,j) \in [\![0\,;n]\!]^2$ tel que $i \neq j$, on pose alors :

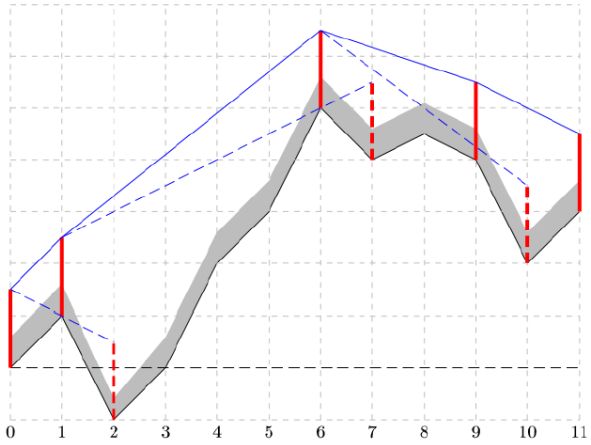
$$\alpha_{i,j} = \frac{(h_j+\Delta)-(h_i+h)}{j-i} \qquad \text{et} \qquad \beta_{i,j} = \frac{h_j-h_i}{j-i}$$

Le premier réel $\alpha_{i,j}$ est la pente d'un fil tendu entre le sommet d'un poteau planté au point d'indice i et le point situé à Δ mètres au-dessus du point d'indice j. Le second réel $\beta_{i,j}$ est la pente d'un fil tendu entre les sommets de poteaux plantés respectivement au point d'indice i et au point d'indice j. On admet que le fil tendu entre deux poteaux placés aux points d'indices i et j tel que $j>i$ vérifie les normes de sécurité si :

$$\beta_{i,j} \geqslant \max\{\alpha_{i,k} \mid k \in [\![i+1\,;j]\!]\}$$

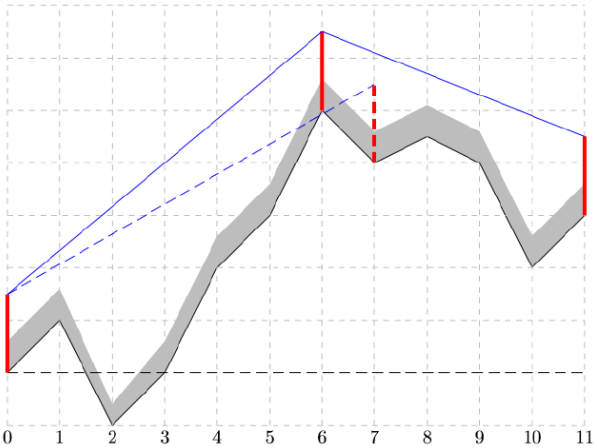
7. Écrire une fonction `correct` prenant en argument la liste `denivele`, la hauteur `h` des poteaux, la distance de sécurité `delta` (Δ), deux indices `i` et `j` de points tels que $j > i$, et renvoyant un booléen égal à `True` si et seulement si un fil tendu entre les poteaux d'indice `i` et `j` vérifie les normes de sécurité.

Considérons une première stratégie pour planter les poteaux, dite algorithme glouton en avant. Le premier poteau est planté en P_0. Lorsqu'un poteau est placé, pour calculer l'emplacement du poteau suivant, on part du dernier poteau planté et on avance (à droite) avec le fil tendu tant que les normes de sécurité sont respectées 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 qui suit illustre la solution produite par cet algorithme, la zone grisée étant celle dans laquelle ne doit passer aucun fil. Les poteaux et les fils en pointillés sont les premiers violant les normes de sécurité lors de la recherche de l'emplacement des poteaux.



8. Écrire une fonction `glouton_avant` prenant en arguments la liste `denivele`, la hauteur `h` des poteaux et la distance de sécurité `delta` (Δ), et renvoyant la liste des indices des points en lesquels doivent être placés des poteaux lorsque l'on suit la stratégie proposée. Avec notre exemple, la liste renvoyée est donc `[0,1,6,9,11]`.

L'algorithme présenté a tendance à placer trop de poteaux, en particulier dans les bassins alors qu'il suffirait d'en relier les deux extrémités par un unique fil. Considérons alors une nouvelle stratégie pour planter les poteaux, dite algorithme glouton en arrière. Le premier poteau est planté en P_0. Lorsqu'un poteau est placé, pour calculer l'emplacement du poteau suivant, on tend un fil entre le dernier poteau planté et un poteau placé en P_n, puis on recule ce dernier poteau jusqu'à ce que les normes de sécurité soient respectées. Un nouveau poteau est alors planté, et on recommence jusqu'à ce que le dernier point soit atteint. La figure qui suit illustre la solution produite par cet algorithme, la zone grisée étant celle dans laquelle ne doit passer aucun fil. Les poteaux et les fils en pointillés les derniers violant les normes de sécurité lors de la recherche de l'emplacement des poteaux.



9. Écrire une fonction `glouton_arriere` prenant en argument la liste `denivele`, la hauteur `h` des poteaux et la distance de sécurité `delta` (Δ), et renvoyant la liste des indices des points en lesquels doivent être placés des poteaux lorsque l'on suit la nouvelle stratégie proposée. Avec notre exemple, la liste renvoyée est donc `[0,6,11]`.