Devoir maison : moindres carrés par morceaux

*Ce devoir est **individuel** : s'il est autorisé de discuter entre vous du sujet, chaque rédaction doit être indépendante. L'évaluation portera principalement sur la **qualité de la rédaction**.*

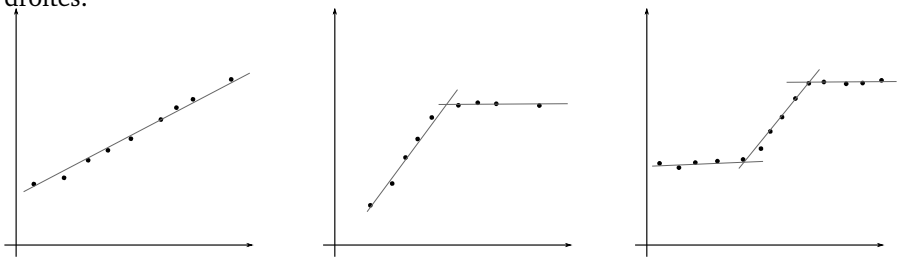
On considère un ensemble P de n points dans le plan, notés (x_1, y_1), …, (x_n, y_n). Étant donné une droite Δ d'équation $y = ax + b$, l'*erreur* de Δ par rapport à P est la somme des carrés des distances entre chaque point et la droite :

$$\textsc{Erreur}(\Delta, P) = \sum_{i=1}^{n} (y_i - (ax_i + b))^2 .$$

La méthode des moindres carrés permet de trouver, étant P, la droite Δ qui minimise l'erreur. On sait calculer son équation explicitement, à savoir

$$a = \frac{n\sum_i x_i y_i - \left(\sum_i x_i\right)\left(\sum_i y_i\right)}{n\sum_i x_i^2 - \left(\sum_i x_i\right)^2} \text{ et } b = \frac{1}{n}\left(\sum_i y_i - a\sum_i x_i\right).$$

Dans certains cas, les points ne sont pas approximativement alignés sur une droite, mais ils le sont *par morceaux*, c'est-à-dire qu'on peut séparer l'ensemble P en sous-ensembles qui sont chacun approximativement alignés sur une droite. Dans les exemples ci-dessous, on *voit* assez facilement qu'il faut une, deux ou trois droites.



L'objectif du devoir est de trouver un algorithme qui, étant donné l'ensemble P de n points, calcule les équations des droites qui approchent au mieux l'ensemble de points. Pour cela, on partitionne P en *segments* de points, c'est-à-dire en sous-ensembles $P_{i,j} = \{(x_i, y_i), (x_{i+1}, y_{i+1}), \ldots, (x_{j-1}, y_{j-1}), (x_j, y_j)\}$ où $i < j$. (Chaque point de P appartient à un et un seul segment $P_{i,j}$.) On cherche pour chaque segment la droite $\Delta_{i,j}$ qui minimise $\textsc{Erreur}(\Delta_{i,j}, P_{i,j})$. On note $E_{i,j}$ l'erreur minimale obtenue pour le segment $P_{i,j}$.

Une solution au problème des moindres carrés par morceaux est donc une partition de P en segments, avec pour chaque segment l'équation de la droite qui minimise l'erreur. Pour analyser les complexités, on considère que chaque opération sur des entiers ou des nombres réels coûte $O(1)$.

1. **i.** La première étape est le cas d'une seule droite : étant donné P, quel est le coût du calcul des coefficients de la droite Δ qui minimise l'erreur et de ERREUR(Δ, P), à l'aide des formules ci-dessus ?
   ii. En supposant qu'on connaît une partition optimale de P en k segment, quel est le coût du calcul des k des k équations de droites et des erreurs associées ?
2. Pour calculer une partition, il faut que les points soient triés par abscisses croissantes. Quel(s) algorithme(s) peut-on utiliser ? Quel est la complexité obtenue ?

On suppose dorénavant que $x_1 < x_2 < \ldots < x_n$.

3. Proposer une structure de données (vue en cours) pour représenter une solution S, c'est-à-dire une partition de P en segment avec pour chaque segment l'équation de la droite associée.
4. Montrer que si n est pair, il existe une solution S avec $n/2$ droites, dans laquelle l'erreur pour chaque segment est nulle.

Le cas précédent montre que ne prendre en compte que les erreurs de chaque droite n'est pas très intéressant. On cherche maintenant à limiter le nombre de segments (et donc de droites), tout en gardant une erreur faible pour chaque segment. Pour cela, on définit la valeur d'une solution comme l'erreur totale des segments plus un multiple du nombre de segments : si une solution S est constituée de k segments P_{i_1,i_2-1}, P_{i_2,i_3-1}, …, $P_{i_k,n}$, la valeur de la solution est

$$\text{VALEUR}(S) = C \cdot k + \sum_{\ell=1}^{k-1} E_{i_\ell, i_{\ell+1}-1} + E_{i_k,n}.$$

On cherche une formule récursive pour trouver la valeur de la solution optimale. Pour tout $i > 0$, on note OPT(i) la valeur d'une solution optimale pour l'ensemble $P_{1,i} = \{(x_1, y_1), \ldots, (x_i, y_i)\}$.

5. **i.** Supposons qu'on ait l'information que le dernier segment est $P_{i,n}$. Exprimer OPT(n) en fonction de $E_{i,n}$, C et OPT$(i-1)$.
   ii. En déduire une formule récursive générale pour OPT(j) en fonction des OPT(i), $i < j$. *Indication. On cherche à trouver quel point (x_i, y_i) fournit le « meilleur » dernier segment.*
6. **i.** Écrire un algorithme de programmation dynamique pour calculer la valeur d'une solution optimale et analyser ses complexités en temps et en espace.
   ii. Peut-on minimiser la mémoire ? *Justifier.*
   iii. Écrire un algorithme de reconstruction qui calcule une solution optimale.