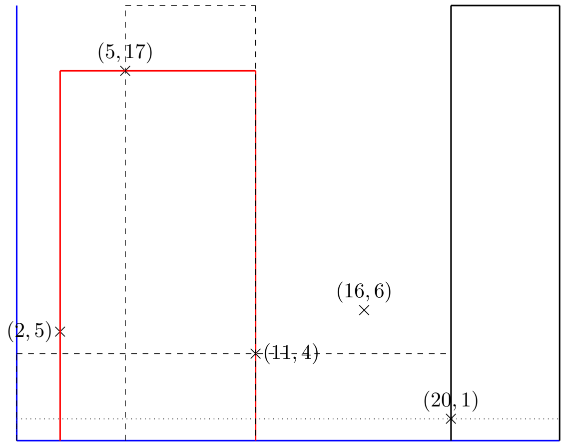
TP4 : Le plus grand rectangle

16 novembre

Le problème : Soit la région rectangulaire du plan déterminée par le rectangle $(0,0)(0,h)(l,h)(l,0)$, h et l éant deux entiers positifs et soient n points à coordonnées entières à l'intérieur de cette région.

On souhaite dessiner un rectangle dont la base est sur l'axe des x, dont l'intérieur ne contienne aucun des n points et qui soit de *surface maximale*.

Par exemple si h=20, l=25, et qu'il y a 5 points (2,5), (5,17), (11,4), (16,6) (20,1), voici quelques exemples de rectangle qu'on peut dessiner : (5,0) (5,20) (11,20) (11,0) de surface 120, (0,0),(0,1),(25,1),(25,0) de surface 25, (0,0),(0,4),(20,4),(20,0) de surface 80, (20,0),(20,20),(25,20)(25,0) de surface 100, ... La surface maximale qu'on peut obtenir est 153 avec le rectangle (2,0) (2,17), (11,17) (11,0).



L'entrée du problème sera :

1. deux entiers strictement positifs l et h.
2. un entier n qui correspond au nombre de points donnés.
3. un tableau bidimensionnel à n lignes et deux colonnes qui contiendra les couples des coordonnées des points. La première colonne contiendra les abscisses et la seconde les ordonnées. On considérera sans perte de généralité que les points $(0,0)$ et $(l,0)$ sont toujours dans le tableau qui a donc au moins deux lignes et que ce dernier est classé par valeurs des abscisses croissante.

Ainsi si on reprend l'exemple ci dessus, les entrées seront :
— $h = 20$ et $l = 25$.
— $n = 7$.
— `int tab[7][2]={{0,0},{2,5},{5,17},{11,4},{16,6},{20,1},{25,0}}`

La sortie du problème sera un entier correspondant à l'aire maximale d'un rectangle satisfaisant les contraintes c'est-à-dire dont l'un des côté est sur l'axe des abscisses et dont l'intérieur ne contient aucun des points.

1 Une première approche

▷ **Question 1.** Expliquer pourquoi les abscisses des côtés latéraux d'un rectangle satisfaisant les contraintes et ayant une aire maximale seront celles de deux points parmi les points passés en arguments.

En déduire une fonction `solve1` de prototype `int solve1 (int l, int h, int n , int tab[n][2])` qui résout le problème. La complexité de votre fonction sera quadratique. ◁

2 DIVISER POUR RÉGNER

▷ **Question 2.** En poursuivant l'approche précédente, concevez un algorithme selon le paradigme "diviser pour régner" qui résout le problème (il conviendra de couper le problème en deux parties astucieusement mais ces parties ne seront pas forcément de mêmes tailles). En déduire une solution `solve2`. Quelle est la complexité dans le meilleur cas ? Dans le pire cas ? ◁

3 UNE SOLUTION PLUS EFFICACE

Pour chaque point d'indice i, on va calculer $L[i]$ qui correspond au plus petit indice j tel que $0 \leq j \leq i$ et $\forall k \in \{j+1, \dots, i\}, tab[k][1] \geq tab[i][1]$ ainsi que $R[i]$ qui correspond au plus grand indice j tel que $i \leq j \leq n-1$ et $\forall k \in \{i, \dots, j-1\}, tab[k][1] \geq tab[i][1]$.

▷ **Question 3.** Justifier que pour tout i, le rectangle démarrant à l'indice `L[i]`, terminant à l'indice `R[i]` et de hauteur `tab[i][1]` satisfait les contraintes. ◁

▷ **Question 4.** Soit un rectangle d'aire maximale répondant au problème. Supposons que ce rectangle commence à l'indice `l`, termine à l'indice `r`, et a pour hauteur `h`. Montrer qu'il existe $i \in \{l, ..., r\}$ tel que `tab[i][1]=h, L[i]=l`, et `R[i]=r`. ◁

Nous allons maintenant voir comment calculer efficacement les valeurs $L[i]$ et $R[i]$.

Pour L on va suivre la stratégie suivante : pour chaque valeur de i, on initialise j à $i-1$ puis si $j=0$ alors $L[i]=0$, sinon si $tab[j][1] < tab[i][1]$ alors $L[i]=j$ et sinon $j=L[j]$.

▷ **Question 5.** Ecrire une fonction `void ell (int tab[][2], int l[],int n)` qui rempli le tableau `l` passé en entrée de telle sorte que `l[i]=L[i]` pour tout i. ◁

▷ **Question 6.** Adapter la méthode précédente pour obtenir une fonction qui remplit un tableau `r` avec les valeurs de $R[i]$. ◁

▷ **Question 7.** En déduire une fonction `solve3` qui résout notre problème. ◁