ACT : TP n°01

Question 1

> un rectangle de surface maximale respectant les contraintes a nécessairement deux sommets de la forme (xi; 0); (xj ; 0) avec 0 <= i < j <= n <= 1 (Pourquoi ?)

En se référençant à l'énoncé, `On souhaite dessiner un rectangle dont la base est sur l'axe des x dans le plan déterminé par le rectangle (0,0) (0,h) (l,h) (l,0)`. Par conséquent, deux des sommets du rectangle de surface maximale trouvé ont une ordonnée égale à 0.

Résultats en O(n³)

- ***ex_N0_res10000:*** Max surface = 10000 find in **0** ms.
- ***ex_N2_res10000:*** Max surface = 10000 find in **0** ms.
- ***ex_N10_res24400144:*** Max surface = 24400144 find in **0** ms.
- ***ex_N100_res5980:*** Max surface = 5980 find in **6** ms.
- ***ex_N100_res6741:*** Max surface = 6741 find in **2** ms.
- ***ex_N500_res7616:*** Max surface = 7616 find in **25** ms.
- ***ex_N500_res7854:*** Max surface = 7854 find in **16** ms. ...
- ***exCodeChef_N5_res49997500:*** Max surface = 49997500 find in **0** ms. ...

Résultats en O(n²)

- ***ex_N0_res10000:*** Max surface = 10000 find in **0** ms.
- ***ex_N2_res10000:*** Max surface = 10000 find in **0** ms.
- ***ex_N10_res24400144:*** Max surface = 24400144 find in **0** ms.
- ***ex_N100_res5980:*** Max surface = 5980 find in **1** ms.
- ***ex_N100_res6741:*** Max surface = 6741 find in **1** ms.
- ***ex_N500_res7616:*** Max surface = 7616 find in **9** ms.
- ***ex_N500_res7854:*** Max surface = 7854 find in **2** ms.
- ***ex_N100000_res100000:*** Max surface = 100000 find in **24847** ms.
- ***ex_N200000_res75141975:*** Max surface = 75141975 find in **139374** ms.
- ***ex=_N100000_res10000000:*** Max surface = 10000000 find in **5911** ms.
- ***ex=_N200000_res20000000:*** Max surface = 20000000 find in **32523** ms.
- ***exCodeChef_N5_res49997500:*** Max surface = 49997500 find in **0** ms.
- ***exT_N100000_res30011389:*** Max surface = 30011389 find in **20883** ms.
- ***exT_N200000_res75141975:*** Max surface = 75141975 find in **137502** ms.

Constatation

On remarque que l'algorithme en O(n²) est nettement plus performant.

Compatible avec Codechef

Pour 100 000 éléments, nous pensons que ces deux algorithmes ne sont pas assez performants.

Question 2 (diviser pour régner)

Explication

L'algorithme selon le paradigme "diviser pour régner" trouvé pour résoudre le problème posé a été pensé de la manière suivante :

Cas de base

Dans le cadre de l'algorithme, le cas de base est défini par `n = 0` où n est le nombre de points de l'ensemble. Dans ce cas-ci, l'aire maximale du sous-rectangle vaut l'aire totale du plan que l'on analyse.

Diviser l'ensemble

La division de l'ensemble se fait à l'abscisse du point dont l'ordonnée est minimale. On obtient donc deux sous-ensembles dont la taille n'est pas égale, contrairement à ce que l'on a vu dans les exemples en cours théoriques.

- **Note importante :** dans notre algorithme, la recherche du minimum sélectionne toujours le premier minimum trouvé. Cependant, lorsque plusieurs points possèdent l'ordonnée minimale, on ne cherchera pas à diviser les ensembles de manière plus équitable en choisissant le point d'ordonnée minimum le plus central.
- **Note importante :** pour certains échantillons, l'exécution de notre algorithme entraînait une "StackOverflow Error". Cependant, cette erreur ne provenait pas réellement de notre algorithme mais du langage (java). En effet, la pile d'exécution allouée à la JVM par défaut est de 512k bytes. Cette taille n'est pas assez grande pour gérer tous les appels récursifs dans certains cas (notamment, le cas où il y a une grande quantité de points de même ordonnées). En augmentant la taille de la pile, le problème a été résolu.

Suite de l'algorithme

Il reste à calculer l'aire maximale des rectangles à droite et à gauche du minimum.

Fusion

Un rectangle d'aire maximale peut avoir des sommets se trouvant à la fois à gauche et à droite du minimum. Ce rectangle aurait d'office comme hauteur, la valeur de l'ordonnée du point minimum et, comme largeur, la distance entre les bornes de gauche et de droite de l'ensemble calculé.

Constatation

Par rapport à des

Résultats de l'algorithme

- *ex_N0_res10000:* Max surface = 10000 find in **0** ms.
- *ex_N2_res10000:* Max surface = 10000 find in **0** ms.
- *ex_N10_res24400144:* Max surface = 24400144 find in **0** ms.
- *ex_N100_res5980:* Max surface = 5980 find in **0** ms.
- *ex_N100_res6741:* Max surface = 6741 find in **0** ms.
- *ex_N500_res7616:* Max surface = 7616 find in **0** ms.
- *ex_N500_res7854:* Max surface = 7854 find in **0** ms.
- *ex_N100000_res100000:* Max surface = 100000 find in **72** ms.
- *ex_N200000_res75141975:* Max surface = 75141975 find in **126198** ms.
- *ex=_N100000_res10000000:* Max surface = 10000000 find in **5098** ms.
- *ex=_N200000_res20000000:* Max surface = 20000000 find in **27379** ms.
- *exCodeChef_N5_res49997500:* Max surface = 49997500 find in **0** ms.
- *exT_N100000_res30011389:* Max surface = 30011389 find in **16126** ms.
- *exT_N200000_res75141975:* Max surface = 75141975 find in **116034** ms.

Complexité

La division de l'algorithme n'est pas une division constante (comme pour le tri dichotomique où les deux parties valent exactement n/2). Néanmoins, on peut tirer, en moyenne, que l'ensemble des n points est divisé par 2 puisque les sous-ensembles se complètent tous les deux.

Nous obtenons donc que `T(n) = 2*T(n/2) + n` et `T(0) = 0` où - n est le nombre de points dans le plan - T(n) est le nombre d'opérations pour la recherche de l'aire du plus grand rectangle dans le plan de n points

Note : Le `+ n` *provient de la recherche du minimum.*

Nous pouvons appliquer le Master Theorem avec les valeurs : - k = 2 - b = 2 - d = 1

Donc : **O(n*log n)**

Question 3

Réduire la hauteur à une petite valeur peut améliorer les performances de la recherche du minimum.

Question non implémentée.

Question 4

L'algorithme consiste à parcourir, une seule et unique fois, tous les points de gauche à droite par rapport à l'axe des abscisses.

On suppose que le tableau de points en paramètre de cette fonction est trié par rapport aux

abscisses ces points.

Lors du parcours, on utilisera une pile pour parvenir à résoudre ce problème avec une complexité temporelle en O(n).

Premièrement, on calcule le rectangle de hauteur h et de longueur égale à la distance du point courant avec le point précédent.

Si la surface calculée est la plus grande qu'on a pu apercevoir durant le parcours, on le mémorise dans une variable qui sera le résultat final de notre fonction. Puis, si la pile est vide ou que l'ordonnée du point au sommet de la pile est plus grande que celle du point courant, on empile celui-ci.

En revanche, lorsque le point courant ne respecte pas cette dernière condition, on dépile le point au sommet de la pile puis on calcule la surface du rectangle avec comme hauteur l'abscisse du point qu'on vient juste de retirer et comme la longueur la distance entre le point courant et le nouveau sommet de pile.

Puis de nouveau, on vérifie que la surface calculée est la plus grande qu'on a calculée durant ce parcours.

Résultats de l'algorithme

- *ex_N0_res10000:* Max surface = 0 find in **0** ms.
- *ex_N2_res10000:* Max surface = 10000 find in **0** ms.
- *ex_N10_res24400144:* Max surface = 24400144 find in **0** ms.
- *ex_N100_res5980:* Max surface = 5980 find in **1** ms.
- *ex_N100_res6741:* Max surface = 6741 find in **0** ms.
- *ex_N500_res7616:* Max surface = 7616 find in **2** ms.
- *ex_N500_res7854:* Max surface = 7854 find in **1** ms.
- *ex_N100000_res100000:* Max surface = 100000 find in **74** ms.
- *ex_N200000_res75141975:* Max surface = 75141975 find in **64** ms.
- *ex=_N100000_res10000000:* Max surface = 10000000 find in **3** ms.
- *ex=_N200000_res20000000:* Max surface = 20000000 find in **18** ms.
- *exCodeChef_N5_res49997500:* Max surface = 49997500 find in **0** ms.
- *exT_N100000_res30011389:* Max surface = 30011389 find in **13** ms.
- *exT_N200000_res75141975:* Max surface = 75141975 find in **25** ms.

Question 5

La difficulté était de savoir lorsqu'il était plus intéressant de paralléliser nos appels récursifs (à partir de quelle valeur de n) et lorsqu'il fallait plutôt exécuter l'appel de fonction en monothread.

Dans notre version de l'algorithme, chaque appel récursif est paralléliser. De ce fait, les performances ne sont pas bonnes. L'algorithme est donc à améliorer.

Nombre de coeurs = 2.