Différentes étapes de ma démarche pour la résolution de l'exercice suivant :

Vous trouverez ci-joint un fichier de données comportant 10000 points avec des coordonnées entières.

Le format du fichier est le suivant :

x1 y1

x2 y2

....

etc.

Il s'agit de compter tous les carrés distincts que l'on peut former avec cette liste de points.

Exemple :

[-1,-1][2,1][4,-2][1,-4] est un carré valide.

Note : vous pouvez utiliser n'importe quel outil ou langage de programmation pour résoudre le problème.

Tout d'abord pour des raisons de plus grande maitrise j'ai choisi le Java comme langage.
Il me permettra de rapidement pouvoir lire le fichier source (et gérer les différentes exceptions pouvant survenir).

Ensuite la question que je me suis posée a été : Pour 4 points donnés, quelles sont les conditions à respecter pour qu'ils forment un carré ?
De par les propriétés d'un carré, il n'y a que deux distances possibles, la valeur du côté c et la valeur de la diagonale : $d = c\sqrt{2}$. (Que l'on peut simplifier en 2c si l'on ne travaille pas avec la racine).
Ainsi pour 4 points donnés, nous pouvions vérifier que les 6 distances sont soit c soit d. Si c'est le cas alors nous avons un carré.

Le problème ensuite a été comment tester toutes les combinaisons de 4 points parmi 10.000 ?
Un problème de performance s'est évidemment posé en voulant toutes les tester « naïvement ».
(J'ai essayé avec des méthodes récursives, ou encore une librairie tierce).
J'ai pensé à paralléliser les traitements, mais je me suis dit qu'il y avait forcément un algorithme plus efficace qu'un ayant une complexité factorielle.

En testant les performances des combinaisons, on peut constater facilement que si trouver toutes les combinaisons de 4 parmi 10.000 est couteux, tester toutes les combinaisons de 2 parmi 10.000 l'est évidemment beaucoup moins (complexité $o(n^2)$).

On peut également constater qu'au final pour deux points que l'on fixe arbitrairement, il n'y a que deux carrés existants possible. Pour les trouver il suffit de calculer le vecteur perpendiculaire au segment formé par nos deux points fixés v(x,y) = (y2-y1, -(x2-x1)).
Grâce à ce vecteur on peut ainsi trouver les deux points manquants pour chacun des deux carrés possibles avec les deux points fixés.

L'algorithme consiste donc en un parcours de i allant de 0 à n -1 et de j allant de i+1 à n de l'ensemble des points (duquel on a évidemment enlevé tous les points en double).
Pour chaque combinaison [i][j] de points qui constituent donc les points fixes, on calcule les deux points manquant pour chacun des 2 carrés possibles.
Ensuite il suffit de vérifier pour chacun des deux carrés ainsi formés si les deux points déduits sont effectivement présents dans l'ensemble de départ.
Si c'est le cas alors un carré est trouvé.
Il convient de stocker tous les carrés trouvés afin de ne pas les compter plusieurs fois.

Une implémentation de cet algorithme en java donne, pour le fichier fourni dans le mail, **48 carrés** trouvés en une moyenne de **4700 ms**, pour **10.000 points**

Romain De Oliveira, le 30/10/2015