Projet de TP : Calcul des k plus proches voisins à l'aide d'un KD_Arbre

Introduction aux KD_Arbres

Un **arbre kd** (ou **kd-tree**, pour *k-dimensional tree*) est une structure de données de partition de l'espace permettant de stocker des points, et de faire des recherches (recherche par plage, plus proche voisin, etc.) plus rapidement qu'en parcourant linéairement le tableau de points.

Description

Les arbres kd sont des arbres binaires, dans lesquels chaque nœud contient un point en dimension k.

Chaque nœud non terminal divise l'espace en deux demi-espaces. Les points situés dans chacun des deux demi-espaces sont stockés dans les branches gauche et droite du nœud courant.

Par exemple en dimension 2 (Plan), si un nœud donné divise l'espace selon un plan normal à la direction de l’axe des x, tous les points de coordonnée x inférieure à la coordonnée du point associé au nœud seront stockés dans la branche gauche du nœud. De manière similaire, les points de coordonnée x supérieure à celle du point considéré seront stockés dans la branche droite du nœud.

Construction

Afin d'avoir un arbre équilibré, le point inséré dans l'arbre à chaque étape est celui qui a la coordonnée médiane dans la direction considérée.

La contrainte sur la sélection du point médian n'est pas une obligation, mais permet de s'assurer que l'arbre sera équilibré.

Le tri des points à chaque étape a un coût en temps, ce qui peut amener à un temps de création de l'arbre assez long.

Il est possible de sélectionner aléatoirement le prochain point à insérer, ce qui donne en général un arbre globalement équilibré.

Algorithme : À partir d'une liste de n points, l'algortithme suivant construit un arbre kd équilibré:

```
function kdtree (liste de points pointList, int profondeur) retourner node
{
    if pointList est vide
        returner nil;
    else
    {
        // Sélectionne l'axe de comparaison en fonction de la profondeur du nœud
        var int axe := profondeur mod dimension;

        // trie la liste de points et sélectionne le point médian
        selectionner point median à partir de axe sur
```

```
ListPoints;

        // Crée le noeud
courant, et construit
récursivement les deux fils
        var arbre_noeud node;
        node.location :=
median;
        node.gauche :=
kdtree(points in pointList
avant median, depth+1);
        node.droit :=
kdtree(points in pointList
après median, depth+1);
        return node;
    }
}
```

La complexité algorithmique de construction d'un arbre kd est de *O(n logn)*.