IFT339 Automne 2014 — Travail pratique 3

Recherche d'information bi-demensionnelle

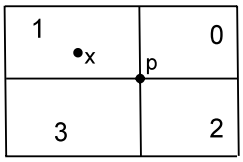
Objectifs de ce laboratoire

Dans ce laboratoire, nous allons utiliser les fichiers de description de graphes du tp2 et du labo2 pour faire un exercice de recherche dans un système d'objets multidimensionnels.

Description de la tâche à réaliser

Dans un système de gestion d'information géographique, on peut chercher des informations de différentes façons. Par exemple, les points ont en général un nom, que ce soit une ville, un code postal ou un lieu historique, ou une adresse précise. Il existe aussi des points sans nom ou adresse, qui ne sont identifiés que par leurs coordonnées. Ainsi, dans le fichier de 2^20 nœuds que je vous ai fournis, il n'y a pas de nœud aux coordonnées (31.61,-93.59). Mais il y a un nœud tout près, le nœud 1048563, zwolle-oil-field-la, aux coordonnées (31.6168,-93.5935). Un bon système de recherche devrait nous permettre de localiser de cette façon le nœud le plus proche d'un point identifié par ses coordonnées.

C'est l'objectif de ce travail. Vous devez ouvrir un graphe décrit par un fichier binaire du format donné pour le tp2, puis lire au clavier des coordonnées (latitude puis longitude) et retrouver dans le graphe le point le plus proche.

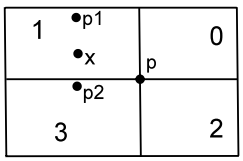


Il n'est pas question bien sûr de lire tout le graphe pour trouver le point. Il faut faire une recherche en temps logarithmique. Nous allons utiliser un Quad-tree pour cette recherche. Il s'agit d'un arbre quaternaire, où on divise l'espace de recherche en quatre portions. À partir d'un point p, la zone 0 contient tous les points dont la latitude et la longitude sont supérieures à celles de p. La zone 1 contient les points de latitude supérieure à celle de p, et de longitude inférieure ou égale à celle de p. La zone 2 contient les points de latitude inférieure ou égale à celle de p et de longitude supérieure à celle de p. La zone 3 contient les autres points. Dans l'illustration ci-contre, le point x est dans la zone 1 de p, puisque sa latitude est supérieure à celle de p et sa longitude inférieure

Dans les fichiers binaires fournis, le Quad-tree est déjà construit. Pour chaque point, on vous donne un autre point de chacune des zones de ce point (ou 0 si cette zone est vide). Si les zones valent (23-0-13-232), c'est que le point 23 est dans la zone 0, le point 13 dans la zone 2 et que la zone 1 est vide. Notez que les zones sont limitées en hauteur et en largeur.

Comment s'effectue la recherche dans un Quad-tree? Cette recherche est toujours relative à un nœud du graphe (disons p). La racine de cet arbre sera le nœud 0. Cela nous donne un premier candidat au point le plus proche, soit p. Les recherches futures nous permettront d'améliorer cette approximation, en cherchant un autre point du graphe plus proche de x que p. On identifie la zone de p où se trouve le point recherché (disons x comme dans l'illustration). On peut aller dans cette zone et y faire la recherche récursivement (inutile d'y aller si la zone est vide). Cette recherche prendra un temps logarithmique, le temps de se rendre jusqu'à une zone vide. Au retour, on dispose d'une meilleure approximation, mais il n'est pas certain que ce soit le nœud recherché. En effet, comme on le voit ci-dessous, on peut avoir trouvé le point p1, plus près de x que p, mais il peut exister un point p2 encore plus près mais dans une autre zone. Si la recherche s'est effectuée d'abord dans la zone 1 (la zone de x), il faudra aussi aller chercher dans la zone 3 et dans la zone 0. Inutile d'aller chercher dans la zone 2, puisque tous les points de cette zone sont plus loin de x que p. Cela peut nous amener à aller voir les trois quarts des points. A-t-on perdu notre borne logarithmique? En général non, surtout s'il y a beaucoup de points. En effet, il est inutile d'aller chercher dans la zone 3 si la frontière de la zone est plus loin de x que le point p1 trouvé.

C'est ici que l'on voit l'intérêt de la numérotation des zones proposée ici. Si on a cherché dans la zone 1, il faut aller aussi chercher (au besoin) dans la zone 3, puis dans la zone 0. On peut généraliser : si on a cherché dans la zone Z, il faut aller aussi dans la zone Z1=(Z+2)%4 puis dans la zone Z2=3-Z1. C'est une autre illustration du principe de faire travailler les données pour nous. Disons que la meilleure distance trouvée à date est DIST. On ne va dans Z1 que si la distance entre la latitude de x et la latitude de p est inférieure à DIST. On ne va dans la zone Z2 que si la distance entre la longitude de x et la longitude de p est inférieure à DIST.



Comment calculer la distance entre deux points?

La latitude et la longitude ne sont pas des coordonnées cartésiennes usuelles. En effet, la distance entre deux points de même latitude dépend de la latitude. La distance entre les points (45,0) et (45,20) n'est pas la même que la distance entre les points (10,0) et (10,20). La distance entre les longitudes dépend de la latitude! On peut avoir une très bonne approximation en utilisant la formule suivante. Soit les points `p1 (LAT1,LON1)` et `p2 (LAT2,LON2)`. On a alors

```
X2= (LON2-LON1)^2
Y2= (LAT2-LAT1)^2
C2= cos((LAT1+LAT2)/2*pi/180.)^2 //cos prend un angle en radians
D= sqrt(Y2+X2*C2)
```

C2 est une bonne approximation de la correction à apporter en fonction de la latitude. Si on sait que la latitude ou la longitude d'un des points est égale à celle de l'autre, on peut beaucoup simplifier cette formule. Une fois calculé D, si on veut la distance en kilomètres, il faut multiplier le résultat par 111. Les distances des arcs dans les graphes USA sont en kilomètres, calculés par cette formule. La vraie distance serait la distance orthodromique, qui tient aussi compte de la courbure de la terre.

On vous fournit des fichiers binaires de description de graphes. Assurez-vous que le constructeur et les fonctions `size`, `operator[]` et `localiser` ont exactement les mêmes signatures dans votre classe graphe que celles-ci:

```
class graphe{
private:
   ...
public:
   graphe(string);     //constructeur
   size_t size()const;                          //nombre de noeuds dans le graphe
   uint32_t localiser(float LAT,float LON);          //trouver un numero de noeud
   string operator[](uint32_t);               //une chaine qui identifie un noeud
   float distance(uint32_t,float,float);              //distance avec le point i
   ...
};
```

Vous pouvez utiliser les fichier "ibin" pour vos tests sur les machines Intel. Pour la récolte automatique, j'utiliserai les fichiers "sbin" qui sont dans le répertoire public du cours.

Remise du travail

Ce travail doit être complété à 23h59 lundi le 8 décembre au plus tard. Soumettez-le à `turnin`, après vous être assurés qu'il fonctionne bien sur tarin. Ne soumettez pas un répertoire, et attention à la syntaxe. Une seule soumission par équipe. Testez votre code sur tarin avec les fichier ".sbin". Le main.cpp qui testera votre fonction pourrait être aussi simple que ceci, où "s" est une string donnant le nom d'un fichier contenant un graphe, comme pour le labo3.

```
float LAT,LON;
uint32_t i;
cin>>s;
graphe G(s);
while(cin>>LAT>>LON){
   cout<<"LAT:"<<LAT<<" LON"<<LON<<endl;
   i=G.localiser(LAT,LON);
   cout<<i<<G[i]<<" (d="<<G.distance(i,LAT,LON)<<")"<<endl;
   }
```

pour avoir une sortie similaire à celle ci-dessous. Faites votre soumission comme ceci.

```
turnin -c ift339 -p tp3 main.cpp graphe.h graphe.cpp
```

Jean Goulet

Pour ma part, j'ai trouvé dans les fichiers les points suivants comme étant les plus proches de (42.817,-70.517)

```
grUSA_32: 31 (42.0087,-70.8559) [0-0-0-0] winebrook-bags-ma (d=93.9218)
grUSA_1024: 372 (42.9793,-70.6125) [554-951-0-746] gosport-harbor-nh (d=19.6179)
grUSA_32768: 5137 (42.9781,-70.6087) [0-0-0-0] cedar-island-me (d=19.3745)
grUSA_1048576: 920592 (42.7784,-70.4978) [821708-0-0-0] the-stone-wall-ma (d=4.56103)
```

Il n'a fallu lire que 13 nœuds pour trouver le dernier. Sans l'exploration des zones adjacentes, on trouve long-island-pond-ma, à une distance de 9.77491, qui ne réclame de lire que 11 nœuds. À la limite, pour trouver le point le plus proche de (39.1654,-100) qui est tout près de la frontière entre les zones 0 et 2, saline-valley-cemetery-ks, il faut explorer 12 nœuds dans la partie active de la récursivité puis 34 de plus dans les zones adjacentes.