TP3 - Ensembles et tables associatives
Isotoponymes et villes confondues

Yonatan DELORO

Pour le 14 décembre 2016

Pour le fichier "villes.txt", on dispose des noms et des positions géographiques de 35180 villes de France. Dans ces données, certaines villes ont des noms identiques. De plus, du fait des approximations, certaines villes ont également des coordonnées identiques. Le but du TP consiste à calculer efficacement pour combien de villes il est ambigu de parler d' "une ville A toute proche d'une ville B", où "toute proche" est à prendre au sens de "mêmes coordonnées".

1 Nombre d'occurrences des noms et des partages de coordonnées de villes

1.1 Histogramme des répétitions des noms de villes (Question 1)

Pour construire l'histogramme des répétitions des noms de villes, c'est-à-dire connaitre le nombre N_k de noms de villes partagés par k villes exactement, on construit d'abord une première table associative, nommé *occ_noms* dans le code, qui associe à chaque nom de ville son nombre d'occurrences dans le vecteur *towns* stockant les 35180 villes. L'utilisation d'une telle structure de données $map < string, int >$ est possible grâce à l'ordre lexicographique défini dans les standards C++ sur les *string* (alphabet latin). Pour remplir cette table, il suffit de parcourir linéairement *towns* et pour chaque nom de ville rencontré : si il n'y apparait pas encore comme clé ajouter une nouvelle clé dans la table et initialiser sa valeur à 1, ou sinon incrémenter de 1 la valeur associée à la clé. La recherche du nom dans la table s'effectue à l'aide de la fonction $find$ implémentée dans la STL.

Une fois cette table d'occurrences construite, il suffit, pour déterminer l'histogramme, de créer une seconde table associative, nommé *histo_noms*, qui à un entier k associe le nombre de noms de villes partagés par exactement k villes de *towns* (k n'apparaîtra come clé que si ce nombre est plus grand que 1). Là encore, les entiers sont ordonnés ce qui autorise la construction d'une table du type $map < int, int >$. Pour remplir cette table, on n'a qu'à parcourir linéairement *occ_noms* et pour chaque nom de ville : si le nombre d'occurrences de ce nom n'est pas encore présent comme clé ajouter une nouvelle clé dans la table correspondant à ce nombre et initialiser sa valeur à 1, ou sinon incrémenter de 1 la valeur associée à la clé (c'est-à-dire le nombre de noms de villes qui sont partagées le même nombre de fois que le nouveau nom rencontré).

1.2 Histogramme du nombre de villes de mêmes coordonnées (Question 2)

On construit l'histogramme de la même manière que dans la question 1 : on crée d'abord une table associative *occ_coords* qui associe à chaque position géographique (point2D ou paire de coordonnées) le nombre de villes partageant cette position, ou autrement dit le nombre d'occurrences de cette paire de coordonnées dans *towns*. On peut créer une telle $map < Point2D, int >$ en munissant les $Point2D$ d'un ordre total, par exemple de l'ordre lexicographique : $P1(x, y) < P2(x', y')$ si et seulement si ($x < x'$ ou $x = y, y = y'$) (x et y sont du type "float").

Puis on crée une deuxième table associative *histo_coords* qui, qui à un entier k associe le nombre de paires de coordonnées partagées par exactement k villes de *towns* (k n'apparaîtra comme clé que si ce nombre est plus grand que 1).

2 Villes homonymes, isotopes, et isotoponymes (Question 3)

Grâce à la table associative *occ_noms* on peut alors construire facilement l'ensemble N des villes qui admettent une autre ville de même nom comme l'ensemble des villes telles que le nombre d'occurrences de leur nom dans *towns* est strictement plus grand que 1. N est dénommé *homonymes* dans le code pour plus de clarté.

De la même manière, grâce à la table associative *occ_coords* on peut construire aisément l'ensemble C des villes qui admettent une autre ville de mêmes coordonnées comme l'ensemble des villes telles que le nombre d'occurrences de leurs paires de coordonnées dans *towns* est strictement plus grand que 1. C est dénommé *isotopes* dans le code.

On peut définir de telles structures de type $set < Town >$ en munissant la classe $Town$ d'une relation d'ordre total. On a choisi le plus simplement : $Town1(name1, point1) < Town2(name2, point2)$ si et seulement si $name1 < name2$ ou $name1 = name2, point1 < point2$, aux sens respectifs de l'ordre total sur les "string" défini par C++ et de l'ordre lexicographique sur les points2D défini plus haut.

Une fois les ensembles N et C déterminés, on utilise pour déterminer $N \cap C$ la méthode *set_intersection* du module "algorithm" de la STL qui permet de construire l'intersection algébrique de deux ensembles ordonnées dans un *vector* dont on doit préparer la taille au cardinal maximal de l'intersection ($min(|N|, [C|)$). Dans le code, l'intersection des ensembles *homonymes* et *isotopes* est le vecteur *isotoponymes*.

Avec les données de "villes.txt", on obtient comme cardinaux respectifs de C, N et $N \cap C$: 3515 villes homonymes, 2150 villes isotopes, et 207 villes isotoponymes.

3 Nombre de villes mal identifiées dans la proposition : "une ville A toute proche d'une ville B" (Question 4)

On souhaite enfin calculer efficacement pour combien de villes on peut se tromper en entendant parler d' "une ville A toute proche d'une ville B", où "toute proche" est à prendre au sens de "a mêmes coordonnées". Plus formellement, il s'agit de calculer le nombre de villes

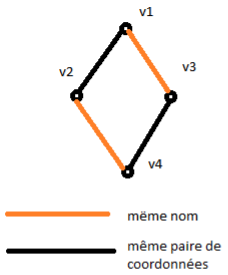

FIGURE 1 – Représentation d'un quatuor (v1,v2,v3,v4) vérifiant les 4 conditions.

.

$v1$ telles qu'il existe des villes $v2, v3, v4$ vérifiant :

$$\begin{cases} nom(v1) = nom(v2) & (i) \\ coords(v1) = coords(v3) & (ii) \\ nom(v4) = nom(v3) & (iii) \\ coords(v4) = coords(v2) & (iv) \end{cases}$$

Un tel quatuor est représenté en figure 1.

Nommons *villes_trompeuses* le vecteur courant stockant les villes du type $v1$, c'est-à-dire mal identifiées par la proposition "A toute proche de B".

Notons que par définition, si $v1$ appartient à *villes_trompeuses*, il en va de même pour $v2, v3, v4$ qui, en particulier, admettent aussi une autre ville de même nom et une autre ville de même coordonnées. Le périmètre de recherche des villes $v1, v2, v3, v4$ peut donc être réduit à notre ensemble d'isotoponymes.

Comment procéder à la recherche de telles villes ?

Pour chaque ville $v1$ de *isotoponymes*, on cherche donc $v2$ homonyme de $v1$ dans *isotopoymes* (soit dans l'ensemble nommé *homonymes*1 dans le code) et on cherche $v3$ isotope de $v1$ dans *isotoponymes* (dans l'ensemble nommé *isotopes*1). Etant donnés $v2$ et $v3$, il suffit de chercher $v4$ dans *isotoponymes* vérifiant les deux dernières conditions (iii) et (iv). Si on trouve un tel quatuor, on ajoute $v1, v2, v3, v4$ à *villes_trompeuses* et passe directement (d'où la présence du booléen *trouve* et des "break" dans le code) à la ville $v1$ suivante de *isotoponymes* qui n'appartient pas déjà à *villes_trompeuses*.

Avec les données de "villes.txt", on obtient à la fin de la recherche que le vecteur *villes_trompeuses* est vide donc il n'existe pas de villes pour lesquelles on peut se tromper en entendant parler d'une ville A toute proche d'une ville B (voir figure 2).

Le temps de recherche dans le fichier "villes.txt" est de $0,927$ secondes. Par une approche naive (voir code), on obtient comparativement un temps de recherche de $404,826$ secondes. Ce qui correspond donc à un gain en temps supérieur à 400.

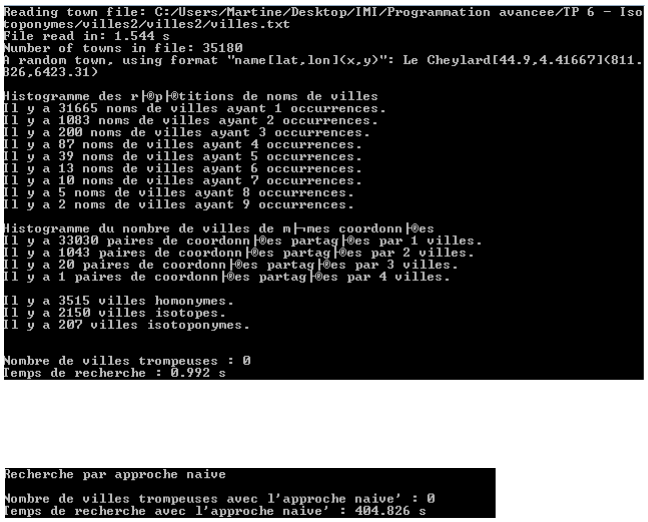

FIGURE 2 – Résultats pour "villes.txt". Histogrammes, nombre d'isotoponymes et nombre de villes "trompeuses" (pour lesquelles on peut se tromper en entendant parler de "A toute proche de B")

.

Avec les données de "villes2.txt" où on a ajouté les villes imaginaires suivantes à la liste des villes de "villes.txt" :

- Machin 50.15 2.733333
- Truc 50.15 2.733333
- Chose 50.15 2.733333
- Machin 48.816667 6.35
- Truc 48.816667 6.35
- Chose 48.816667 6.35

On vérifie que l'on obtient bien un vecteur *villes_trompeuses* de cardinal 6 au terme de la recherche (voir figure 3).

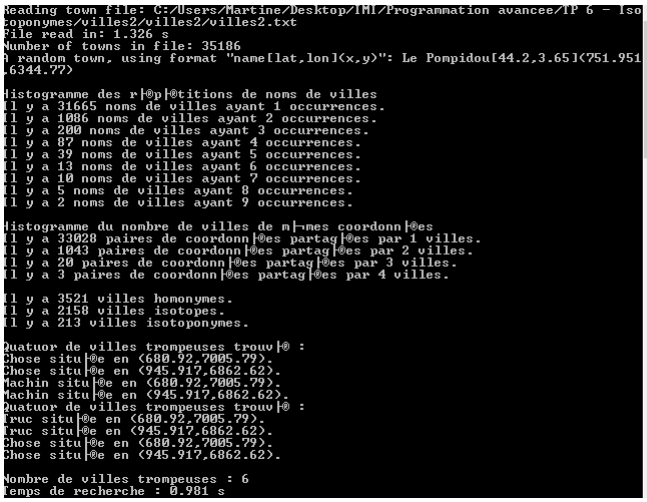

FIGURE 3 – Résultats pour "villes2.txt" dans lequel on a rajouté les 6 villes imaginaires données dans le corps du rapport. Histogrammes, nombre d'isotoponymes et nombre de villes "trompeuses".

.