Report

The goal of this project was to designed to find the optimal path from one place to the other. By doing this, it involves with many algorithms and data structures. For example, for this project, I created my own map instead of using STL map and is called MyMap, basically this works as a binary search tree. It has one key and one value as templates so the user can insert any type of classes or data they want. When I insert an item into the MyMap, it will compare key to the other node in the tree. Notice that smaller items are on the left side of the root, and bigger items are on the right side of the root. This project not only involve with MyMap, but also vector, A\* algorithm. It might be hard to explain the A\* algorithm here but you can look at my code and see how it works.

MyMap class

Since MyMap class works as a binary search tree, therefore:

Associate() function is O(log(n)), n is the number of items in the tree

Find() function is also O(log(n)), n is the number of items in the tree

The reason why they both O(log(n)) is because they search or find an item in the tree by n/2, n/4, n/8, etc.

AttractionMapper class

This class is to store every attractions(value) that are associated with a street name(key) in the mapdata.txt file using MyMap data structure.

Init() function takes O(N+Alog(A)) because I had to run through every streetSegment and some streets have the attractions and some not. N is the number of street segments, and A is the number of attractions that associates on the street.

getGeoCoord() function takes O(log(A)) because I had to call find function is MyMap class in order to get the GeoCoord of each attraction.

SegmentMapper class

This class is to store every streetSegment(value) that are associated with a GeoCoord in the mapdata.txt file using MyMap data structure.

Init() function takes O((N+A)\*log(N+A)) because first I had to loop through every street segment and each street segment, I had to loop through every attraction. Then I check if each GeoCoord is already existing, if it exist, I push a streetSegment that associates with GeoCoord into MyMap data structure. If it does not exist, I create an streetSegment vector to put into GeoCoord into MyMap data structure. N is the number of street segment and A is the number of attractions.

getSegments() function takes O(log(N+A)) because when I found the GeoCoord, every street segment is associate attractions.

Navigator class

Navigate() takes O(log(N+A) + Alog(n)), where N is the number of street segments, A is the number of attractions, and n is the number of navigation segment.

Since I thought we could start and end at both street names and attractions, so my algorithm is a bit different from start and end at attractions. Therefore, the case sensitive does not work. If I know start and end are only attractions, my case sensitive will work since I check it in my attractionsMapper. (See my AttracionsMapper) On the other hand, at least it works with both attraction and street for starting and ending points.

This is just a Pseudocode, I did not know we do not have to create it. But I already created it so I will just leave it here.

Pseudocode:

MyMap class

Public:

Void associate(keyValue, val) {

If (root is nullptr) {

root = new BST(keyValue, val)

return}

BST \*curr = root;

While (1)

{

if (curr->keyMember == keyValue) {

update the valueMember

return }

if (curr->keyMember < keyValue) {

if (curr is nullptr)

create a new node in the BST

else

keep going to the left

return

}

if (curr->keyMember > keyValue) {

if (curr->left is nullptr)

create a new node in the BST

else

keep going to the right

return

}

}

}

const ValueType\* find(keyValue) {

BST \*curr = root;

While (curr is not nullptr)

{

if (curr->keyMember equals keyValue)

return curr->valueMember;

if (curr->keyMember less than keyValue)

curr going to the left

if (curr->keyMember greater than keyValue)

curr going to the right

}

not found return nullptr;

}

private:

Create a struct called BST {

It has left and right child

keyMember and valueMember

construct a BST

BST(key, value){

keyMember = key;

valueMember = value;

left = right = nullptr;

}

}

void clear(BST curr) {

if (curr is nullptr)

return;

clearR(curr left)

clearR(curr right)

delete curr;

}

int sizeR(BST curr) {

if (curr is nullptr)

return 0;

else

return (sizeR(curr left) + 1 + sizeR(curr right))

}

MapLoaderImpl

bool load(string mapFile) {

if (there is no file)

return false;

while (reading the segment’s name is not nullptr)

{

get the streetName

Used delimeter to separate segment for latitude and longitude.

There are 2 coordinates with 2 latitude and longitude

store the name the of street and the segment into streetSegment

temp = {streetName, segment};

get number of how many attractions it has.

run a for loop from 0 to number of attractions – 1 {

used delimeter to the coordinate of each attraction to get

latitude and longitude

store the attraction into streetSegment

temp.attractions.push\_back(attractions)

}

push the streetSegment into the vector

}

bool getSegment(size\_t segNum, streetSegment &seg) {

if (segNum less than numberOfSegment)

store vec[segNum] into seg

return true

return false

}

AttractionMapperImpl

Void init(MapLoader ml)