In this project, I have created one new STL container for storing friendships. I think it was not efficient to use the old container which was used for storing the boss-underlings relationships, consider a situation within a company that we have lots of boss-underlings relations and only one friendship relation and If the friendships relations are stored in boss-underlings, we should search through all the people in the company to find that one pair of friendship. The new container that I have created is like this: unordered\_map<PersonID, Friend\*> in which type of “Friend” is a struct, and this struct is implemented like this:

struct Friend

{

unordered\_map<PersonID, Cost> friends\_map;

};

In this Struct I have defined an unordered map for storing someone’s friends and their friendship costs. Now the complexities for the new functions are listed:

add\_friendship: O(log(V))

remove\_friendship: O(log(V))

get\_friends: O(V)

all\_friendships: O(E) and in the worst case E=V\*V

shortest\_friendpath: O(V + E\*log(V))

check\_boss\_hierarchy: O(E) and in the worst case E=V\*V

cheapest\_friendpath: O(V + E\*log(V))

leave\_cheapest\_friendforest: O(V\*E)