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)