Stanford CS106X Debug Assign4

Assign 3-Disaster Planning

The decomposition of original problem is very important!
We can decompose the problem **recursively**

First we find generate our ideas and implement the whole structure of the program.

```
bool canBeMadeDisasterReady(const Map<string, Set<string>>& roadNetwork, int
numCities, Set<string>& locations) {
    //base case
    if (cities were all covered) {
        return true;
        //used up all cities
    } else if (numCities <= 0) {
        return false
    } else {
        1. find a uncovered city
        //choosing
        2. stock resources at the city
        3. stock resources at the cities aroung the uncovered city.
        //undo choices
    }
}
```

Then write a function to implement `cities were all covered`

```
bool allCitiesCovered(const Map<string, Set<string>>& roadNetwork, Set<string>&
locations) {
    //loop
    if (every single city is covered)
        return true;
    else
```

```
        return false;
}
```

Then write a function to determine whether `single city is covered`

```
bool isCityCovered(string city, const Map<string, Set<string>>& roadNetwork,
Set<string>& locations) {
    if (!roadNetwork.containsKey(city))
        return false;
    // the city stores the resources
    if (locations.contains(city)) {
        return true;
    } else { // adjacent cities stored resources
        for (string surrounding : roadNetwork[city]) {
            if (locations.contains(surrounding)) {
                return true;
            }
        }
    }
    return false;
}
```

It seems that we reached our base case, the return back to implement previous function.

Back to the main function, we found that we also need to implement a function to `find a uncovered city`

```
string findUncoveredCity(const Map<string, Set<string>>& roadNetwork,
Set<string>& locations) {
    //loop over roadNetwork
    if (a city in the roadNetWork is not covered by locations)
        return this city
}
```

Back to our base case again `a city in the roadNetWork is not covered by locations` can be realized by `!isCityCovered(string city, const Map<string, Set<string>>& roadNetwork, Set<string>& locations)`

By the decomposition above, we basically solved the problem.

When design the program, the general structure instead of the detail of implementation should be considered first! We can use human language to write things we need to implement as function. And we can use pseudocode to assume those function is implemented by someone else—leap of faith—.

#CS106X