# Homework 4.2 Report

## 实现代码

## A*搜索

```
vector<string> ASearch(string question, string start, string end)
{
    vector<string> solution;
    map<pair<string, string>, int> pathMap;
    map<string, int> pathSLD;
    chooseMap(question, pathMap, pathSLD);
    map<string, int> frontier;
    vector<string> explored;
    frontier[start] = 0;

    while(1)
    {
        if(frontier.empty())
            return solution;

        for(map<string, int>::iterator iter = frontier.begin(); iter != frontier.end(); iter++)
            iter->second += pathSLD[iter->first];
        map<string, int>::iterator minNode = frontier.begin();
        for(map<string, int>::iterator iter = frontier.begin(); iter != frontier.end(); iter++)
            if(iter->second < minNode->second)
                minNode = iter;
        string current = minNode->first;
        int d = minNode->second - pathSLD[minNode->first];
        frontier.erase(minNode);
        for(map<string, int>::iterator iter = frontier.begin(); iter != frontier.end(); iter++)
            iter->second -= pathSLD[iter->first];
        explored.push_back(current);
        solution.push_back(current);

        if(current == end)
            return solution;

        for(map<pair<string, string>, int>::iterator iter = pathMap.begin(); iter !=
pathMap.end(); iter++)
            if(iter->first.first == current)
            {
                if(frontier.find(iter->first.second) == frontier.end() &&
                    find(explored.begin(), explored.end(), iter->first.second) == explored.end())
                        frontier[iter->first.second] = d + iter->second;
                else
                {
                    map<string, int>::iterator child = frontier.find(iter->first.second);
                    if(child != frontier.end() && child->second > d + iter->second)
                        child->second = d + iter->second;
                }
            }
    }
}
```

该算法具有如下的输入

- string question：解决的问题，本例中为"Romania"或"HIT"
- string start：搜索的起始点
- string end：搜索的结束点

该算法返回一个vector

- 向量内包含BFS搜索过程中每步寻找到的点

# A*遍历

```
vector<string> ATraversal(string question, string start)
{
    vector<string> solution;
    map<pair<string, string>, int> pathMap;
    map<string, int> pathSLD;
    chooseMap(question, pathMap, pathSLD);
    map<string, int> frontier;
    vector<string> explored;
    frontier[start] = 0;

    while(1)
    {
        if(frontier.empty())
            return solution;

        for(map<string, int>::iterator iter = frontier.begin(); iter != frontier.end(); iter++)
            iter->second += pathSLD[iter->first];
        map<string, int>::iterator minNode = frontier.begin();
        for(map<string, int>::iterator iter = frontier.begin(); iter != frontier.end(); iter++)
            if(iter->second < minNode->second)
                minNode = iter;
        string current = minNode->first;
        int d = minNode->second - pathSLD[minNode->first];
        frontier.erase(minNode);
        for(map<string, int>::iterator iter = frontier.begin(); iter != frontier.end(); iter++)
            iter->second -= pathSLD[iter->first];
        explored.push_back(current);
        solution.push_back(current);

        if(explored.size() == pathSLD.size())
            return solution;

        for(map<pair<string, string>, int>::iterator iter = pathMap.begin(); iter !=
pathMap.end(); iter++)
            if(iter->first.first == current)
            {
                if(frontier.find(iter->first.second) == frontier.end() &&
                    find(explored.begin(), explored.end(), iter->first.second) == explored.end())
                        frontier[iter->first.second] = d + iter->second;
                else
                {
                    map<string, int>::iterator child = frontier.find(iter->first.second);
                    if(child != frontier.end() && child->second > d + iter->second)
                        child->second = d + iter->second;
                }
            }
    }
}
```

该算法具有如下的输入

- string question：解决的问题，本例中为"Romania"或"HIT"
- string start：搜索的起始点

该算法返回一个vector

- 向量内包含BFS遍历过程中每步寻找到的点

相比于搜索，遍历只是将结束条件由找到搜索点改为所有节点都被遍历过，即

```
if(explored.size() == pathSLD.size())
    return solution;
```

# 实验结果

## 以罗马尼亚为输入，给出不同起点到终点Bucharest的解

1. ASearch:Romania:Arad-Buchares

- Arad Sibiu Rimnicu Pitesti Faragas Bucharest

2. ASearch:Romania:Craiova-Bucharest

- Craiova Pitesti Bucharest

3. ASearch:Romania:Lugoj-Bucharest

- Lugoj Mehadia Dobreta Craiova Timisoara Pitesti Bucharest

## 以哈工大校园导航问题，给出起点为正心楼、终点为诚意楼的解

- ZhengxinBuilding 3GStore ICBC ChengyiBuilding

## 使用A*算法解决从Arad遍历罗马尼亚主要城市

- Arad Sibiu Rimnicu Pitesti Faragas Bucharest Timisoara Zerind Lugoj Giurgui Craiova Oradea Mehadia Urziceni Dobreta Hirsova Eforie Vaslui Iasi Neamt

# 算法特征比较

| | BFS | DFS | IDS | A* |
|---|---|---|---|---|
| 完备性 | 完备 | 图搜索在有限状态空间完备，无限空间不完备 | 完备 | 完备 |
| 最优性 | 耗散随节点深度的非递减则有最优解 | 非最优 | 非最优 | 最优 |
| 时间复杂性 | O(b^d) | O(b^m)，m：最大深度 | O(b^d)，d：深度 | |

| | | | | |
|---|---|---|---|---|
| 空间复杂性 | O(b^(d-1)) | bm+1 | O(bd)，d：深度 | |