云南大学数学与统计学院
《运筹学通论实验》上机实践报告

课程名称： 运筹学实验	**年级：** 2015 级	**上机实践成绩：**
指导教师： 李建平	**姓名：** 刘鹏	**专业：** 信息与计算科学
上机实践名称： Dijkstra 算法求最短路径	**学号：** 20151910042	**上机实践日期：** 2018-07-08
上机实践编号： 5	**组号：**	

一、 实验目的

1. 学习 Dijkstra 算法的使用；
2. 了解 Dijkstra 算法作为贪心算法能达最优的理论证明。

二、 实验内容

1. 写出 Dijkstra 算法[1]的伪码描述[2]；
2. 用 C 语言[3]编程实现 Dijkstra 算法，找出一幅图的给定两个节点间的最短路径；

三、 实验平台

Microsoft Windows 10 Pro Workstation 1803；
Cygwin GCC 与 G++编译器

四、 算法设计

4.1 算法背景

Dijkstra 算法解决的是带权重的有向图$\boldsymbol{G}_{\text{in}} = (\boldsymbol{V}, \boldsymbol{E})$上单源最短路径问题，该算法要求所有边的权重都为非负值。假定所有的边$(u, v) \in \boldsymbol{E}$，都有权重$w(u, v) \geq 0$。Dijkstra 算法在运行过程中维护的关键信息是一组节点集合$\boldsymbol{S}$，从源节点s到该节点集合中每个节点之间的最短路径都已经被找到。算法重复地从节点集合$\boldsymbol{V} - \boldsymbol{S}$中选择最短路径估计最小的节点$u$，将$u$加到集合$\boldsymbol{S}$中去，然后对所有从$u$发出的边进行松弛（当然也包括$\boldsymbol{S}$中已经存在的边）。所谓的松弛，指的是对初态的一种调整。若存在一个从u到v的有向边，那么令v中保留的前驱信息$v.\pi = u$。

松弛：第一次全体松弛（或称初始化）是指把源点s的属性设置为0，其他节点的属性设置为无穷大；其他过程中的松弛为：从集合$\boldsymbol{S}$中的通过某种方法制定的顶点v开始出发，找到顶点v的所有邻边，然后把v邻点的属性都改变，如将一个邻点u的属性改为$w(v, u)$，其他邻点的改法完全类似。

Algorithm **INITIALIZE-SINGLE-SOURCE**, 初始化
Input 图$\boldsymbol{G}$，初始节点s

Output 已经改变过图$\boldsymbol{G}$
Begin

Step 1 **for** each vertex $v \in \boldsymbol{G}.\boldsymbol{V}$
let $v.property = \infty$
let $v.\pi = \mathbf{None}$
GOTO **End**
End

Algorithm **RELAX**, 对边$(u,\ v)$在$O(1)$时间内进行松弛操作
Input 节点$u,\ v$，两个节点之间存在由前者指向后者的边
已知$w(u,\ v)$
Output 已经改变过的节点$u,\ v$
Begin
Step 1: if $v.\text{property} > u.\text{property} + w(u,\ v)$
let $v.\text{property} = u.\text{property} + w(u,\ v)$
let $v.\pi = u$
GOTO **End**
End

根据上面给出的两个子函数，加上一个优先队列，可以实现一个复杂度比较低的 Dijkstra 算法。

Algorithm **DIJKSTRA**, 生成一个图，该图包含任意两点间的最短路径
Input 图$\boldsymbol{G}$，初始节点s，权重函数w

Output 图$\boldsymbol{S}$，其中任何两点间的仅有一条路，且为这两点在原来图$\boldsymbol{G}$中的最短路
Begin
Step 1 **INITIALIZE-SINGLE-SOURCE**($\boldsymbol{G}$, s)

Step 2 let $\boldsymbol{S} = \phi$

Step 3 把节点集合加入到优先队列$\boldsymbol{Q}$中

Step 4 **while** $\boldsymbol{Q} \neq \phi$
令u为从$\boldsymbol{Q}$中弹出的property最小的顶点
let $\boldsymbol{S} = \boldsymbol{S} \cup \{u\}$
对于图$\boldsymbol{G}$中所有的与u相邻的顶点v:
RELAX(v)
GOTO **End**
End

五、 程序代码

5.1 程序描述

时间关系，4 号报告中的一些 bug 还没有调完，但是提交在即。这段 C++程序从开源社区获得。

5.2 程序代码

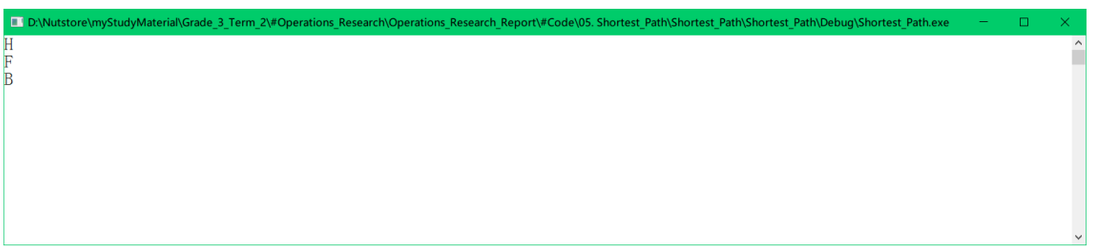
```
/*
* Copyright (c) 2018, Liu Peng, School of Mathematics and Statistics, YNU
* Apache License.
*
* 文件名称：Shortest_Path.cpp
* 文件标识：见配置管理计划书
* 摘 要：寻找给定顶点间的最短路
*
* 当前版本：1.0
* 作 者：刘鹏
* 创建日期：2018 年 7 月 7 日
* 完成日期：2018 年 7 月 7 日
*
* 取代版本：
* 原作者 ：刘鹏
* 完成日期：
*/

/*
* A function based on Dijkstra Algorithm to find the Shortest Path
* in a undirected graph.
*/

#include <unordered_map>
#include <vector>
#include <limits>
#include <algorithm>
#include <iostream>

using namespace std;

class Graph {
    unordered_map<char, const unordered_map<char, int>> vertices;

public:
    void add_vertex(char name, const unordered_map<char, int>& edges) {
        vertices.insert(unordered_map<char, const unordered_map<char, int>>::value_type(name,
edges));
    }

    vector<char> shortest_path(char start, char finish) {
        unordered_map<char, int> distances;
        unordered_map<char, char> previous;
```

```
44          vector<char> nodes;
45          vector<char> path;
46
47          auto comparator = [&](char left, char right) { return distances[left] > dis-
    tances[right]; };
48
49          for (auto& vertex : vertices) {
50              if (vertex.first == start) {
51                  distances[vertex.first] = 0;
52              }
53              else {
54                  distances[vertex.first] = numeric_limits<int>::max();
55              }
56
57              nodes.push_back(vertex.first);
58              push_heap(begin(nodes), end(nodes), comparator);
59          }
60
61          while (!nodes.empty()) {
62              pop_heap(begin(nodes), end(nodes), comparator);
63              char smallest = nodes.back();
64              nodes.pop_back();
65
66              if (smallest == finish) {
67                  while (previous.find(smallest) != end(previous)) {
68                      path.push_back(smallest);
69                      smallest = previous[smallest];
70                  }
71
72                  break;
73              }
74
75              if (distances[smallest] == numeric_limits<int>::max()) {
76                  break;
77              }
78
79              for (auto& neighbor : vertices[smallest]) {
80                  int alt = distances[smallest] + neighbor.second;
81                  if (alt < distances[neighbor.first]) {
82                      distances[neighbor.first] = alt;
83                      previous[neighbor.first] = smallest;
84                      make_heap(begin(nodes), end(nodes), comparator);
85                  }
86              }
87          }
88
89          return path;
90      }
91  };
```

```
92
93  int main() {
94      Graph g;
95      g.add_vertex('A', { { 'B', 7 },{ 'C', 8 } });
96      g.add_vertex('B', { { 'A', 7 },{ 'F', 2 } });
97      g.add_vertex('C', { { 'A', 8 },{ 'F', 6 },{ 'G', 4 } });
98      g.add_vertex('D', { { 'F', 8 } });
99      g.add_vertex('E', { { 'H', 1 } });
100     g.add_vertex('F', { { 'B', 2 },{ 'C', 6 },{ 'D', 8 },{ 'G', 9 },{ 'H', 3 } });
101     g.add_vertex('G', { { 'C', 4 },{ 'F', 9 } });
102     g.add_vertex('H', { { 'E', 1 },{ 'F', 3 } });
103
104     for (char vertex : g.shortest_path('A', 'H')) {
105         cout << vertex << endl;
106     }
107
108     return 0;
109 }
```

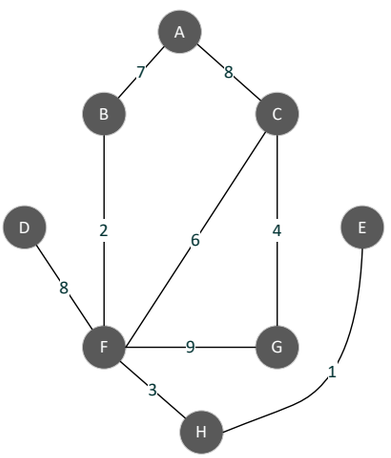
程序代码 1

六、 运行结果



运行结果 1

6.1 代码分析



分析上图之后可以看出，运行结果没有问题。

七、 实验体会

实验比较简单，但是时间不够了。感谢开源社区[4]提供的高质量程序

八、 参考文献

[1] HILLIER F S, LIEBERMAN G J. 运筹学导论 [M]. 9th ed. 北京: 清华大学出版社, 2010.

[2] CORMEN T H, LEISERSON C E, RIVEST R L, et al. 算法导论 [M]. 3rd ed. 北京: 机械工业出版社, 2013.

[3] 林锐. 高质量 C++/C 编程指南 [M]. 1.0 ed., 2001.

[4] https://github.com/mburst/dijkstras-algorithm