Dijstra 算法 tsD14536

计 72 陈嘉杰

March 24, 2018

Contents

1 题目说明 **1**

2 实现思路 **1**

3 程序编译环境 **2**

4 实现步骤 **2**
4.1 下标约定 2
4.2 数据的读入 2
4.3 初始化源点和源点连出的边 3
4.4 进行 Dijkstra 算法中的迭代 3
4.5 输出最短路径的长度 3
4.6 完整代码 4

5 遇到的问题和得到的收获 **5**

1 题目说明

文件 `input.txt` 中含有一个图的权矩阵表示，要求输出单源最短路径到 `output.txt` 中。

2 实现思路

实现上课在 PPT 中描述的 $O(n^2)$ 的 Dijkstra 算法，并没有编写带优先队列优化的版本。

3 程序编译环境

1. 操作系统：macOS
2. 编译器：LLVM/Clang 6.0.0

4 实现步骤

4.1 下标约定

输入数据中点和边都从 1 开始，故在我的代码中同样如此。

4.2 数据的读入

考虑到输入的数据中 $0 < n < 32$, 所以可以直接在全局变量中开足够大的空间存放数据。首先是文件重定向，接着，读入数据，保存权矩阵，并初始化距离向量和已访问的点集。

```
#include <stdio.h>

int map[33][33];
int dist[33];
int vis[33];

int main() {
  int n, k;
  freopen("input.txt", "r", stdin);
  freopen("output.txt", "w", stdout);
  scanf("%d", &n);
  for (int i = 1;i <= n;i++) {
    for (int j = 1;j <= n;j++) {
      scanf("%d", &map[i][j]);
    }
    dist[i] = -1;
    vis[i] = 0;
  }
  scanf("%d", &k);
  return 0;
}
```

4.3 初始化源点和源点连出的边

初始化源点的距离和已访问状态，并根据边初始化后继结点的距离。

```
vis[k] = 1;
dist[k] = 0;
for (int i = 1;i <= n;i++) {
  if (map[k][i] != 0) {
    dist[i] = map[k][i];
  }
 }
```

4.4 进行 Dijkstra 算法中的迭代

每次从未访问的点中找到距离最小的，更新距离向量。

```
for (int i = 1;i <= n;i++) {
  int min_dist = 2147483647;
  int min_index = 0;
  for (int j = 1;j <= n;j++) {
    if (vis[j] == 0 && dist[j] != -1 && dist[j] < min_dist) {
      min_dist = dist[j];
      min_index = j;
    }
  }
  if (min_index == 0) {
    break;
  }
  vis[min_index] = 1;
  for (int k = 1;k <= n;k++) {
    if (map[min_index][k] != 0) {
      if (dist[k] == -1 || dist[k] > min_dist +
      ↪  map[min_index][k]) {
        dist[k] = min_dist + map[min_index][k];
      }
    }
  }
 }
```

4.5 输出最短路径的长度

注意要跳过源点本身。

```
  for (int i = 1;i <= n;i++) {
    if (i != k) {
      printf("%d ", dist[i]);
    }
  }
printf("\n");
```

4.6 完整代码

```
#include <stdio.h>

int map[33][33];
int dist[33];
int vis[33];

int main() {
  int n, k;
  freopen("input.txt", "r", stdin);
  freopen("output.txt", "w", stdout);
  scanf("%d", &n);
  for (int i = 1;i <= n;i++) {
    for (int j = 1;j <= n;j++) {
      scanf("%d", &map[i][j]);
    }
    dist[i] = -1;
    vis[i] = 0;
  }
  scanf("%d", &k);
  vis[k] = 1;
  dist[k] = 0;
  for (int i = 1;i <= n;i++) {
    if (map[k][i] != 0) {
      dist[i] = map[k][i];
    }
  }
  for (int i = 1;i <= n;i++) {
    int min_dist = 2147483647;
    int min_index = 0;
    for (int j = 1;j <= n;j++) {
```

```
31        if (vis[j] == 0 && dist[j] != -1 && dist[j] < min_dist)
          ↪  {
32          min_dist = dist[j];
33          min_index = j;
34        }
35      }
36      if (min_index == 0) {
37        break;
38      }
39      vis[min_index] = 1;
40      for (int k = 1;k <= n;k++) {
41        if (map[min_index][k] != 0) {
42          if (dist[k] == -1 || dist[k] > min_dist +
            ↪  map[min_index][k]) {
43            dist[k] = min_dist + map[min_index][k];
44          }
45        }
46      }
47    }
48    for (int i = 1;i <= n;i++) {
49      if (i != k) {
50        printf("%d ", dist[i]);
51      }
52    }
53    printf("\n");
54    return 0;
55  }
```

5 遇到的问题和得到的收获

遇到的问题就是，太长时间没有自己写 Dijkstra ，在编写的时候未能一气呵成，漏写了若干的语句和判断条件。不过，在本地调试以后，很快发现了问题。收获就是，虽然我对这个算法的过程很熟悉了，但是一些细节还是不够清楚，写之前还是需要仔细阅读算法的伪代码，这样可以节省时间。