Data Structure Assignment #4

程式相關報告

工資系 113 H54094015 張柏駿

```cpp
#include <iostream>
#include <climits>
#include <iomanip>
#include <string>
#include <vector>
#include <fstream>
using namespace std;
int vertexNum = 8;

int miniDist(int distance[], bool Tset[]) // finding minimum distance
{
    int minimum=INT_MAX;
    int U=0;

    for(int k=0;k<vertexNum;k++)
    {
        if(Tset[k]==false && distance[k]<=minimum)
        {
            minimum=distance[k];
            U=k;
        }
    }
    return U;
}
```

這邊使用助教課提供的程式做模板。

```cpp
void printPath(int route[], int j)
{
  if (route[j] == -1){
    return;   // j be the source
  }
  printPath(route, route[j]); //trace back
  cout << " -> Shop " << j + 1;
}
```

新增一個函數 Printpath，有兩個引數 route ,j，並以遞迴的方式做，從後面回溯走過了那些點，類似 stack 的概念，才能夠印出整個完整路徑，調整輸出以增加使用者易讀性。

```
35  void DijkstraAlgo(int graph[8][8], int src, int dst) // adjacency matrix
36  {
37      int distance[8]; // array to calculate the minimum distance for each node
38      bool Tset[8];    // boolean array to mark visited and unvisited for each node
39      int route[8];    // array to store the route
40
41      //initialize
42      for(int k = 0; k < vertexNum; k++)
43      {
44          distance[k] = INT_MAX;
45          Tset[k] = false;
46          route[src] = -1; // Source vertex  is set -1 in array
47      }
48
49      distance[src] = 0;   // Source vertex distance is set 0
50
51
52      for(int k = 0; k < vertexNum; k++)
53      {
54
55          int m = miniDist(distance, Tset);
56          Tset[m]=true;
57
58          for(int k = 0; k < vertexNum; k++)
59          {
60              int temporaryDist = 0;
61              temporaryDist = distance[m] + graph[m][k];
62
```

使用助教課的 DijkstraAlgo 演算法 function，僅增加了等等為了印路徑用的 route array，size 為 8 因為有八個節點，另外將陣列中的起始店設為-1。

```
64              // updating the distance of neighbor vertex
65              if(!Tset[k] && graph[m][k] && distance[m]!=INT_MAX && temporaryDist<distance[k]){
66                  distance[k]= temporaryDist;
67                  route[k] = m;   // update m into route
68              }
69          }
70      }
71
72      cout<< "\nFrom shop " << src+1 << " to shop " << dst+1 << endl; // print start and end shops
73      for(int k = 0; k < vertexNum; k++)
74      {
75          if (dst==k){
76              cout << "Shortest distance: " << distance[k] <<endl; //Shortest distance
77              cout << "Diliver route: Shop" << src+1; //src+1 == the start shop
78              printPath(route, k);   //call the function PrintPath
79          }
80      }
81
82  }
```

這邊也用助教課的演算法，將相鄰的點 m 加入 route 的陣列。且當呼叫 DijkstraAlgo 同時也會印出起始的分店與結束分店(起始點與終點)，並用一 for 迴圈判斷當值為我們要輸入的終點，則印出在儲存距離陣列中的最短路徑，並且印出其經過的各分店。

```cpp
int main()
{
    ifstream in, inFile;
    ofstream out, outFile;  //file object
    in.open("adjacency_matrix.txt");  //open file

    int A[8][8];
    cout << "\nReading data....." << endl;
    while(!in.eof()){          //eof condition
        for(int i=1;i<=8;i++){
            for(int j=1;j<=8;j++)
            {
                in >> A[i][j]; //append in array
            }
        }
    }
    out.close();  //close file
    //Adjacency matrix
    cout << "Adjacency matrix: ";
    for(int i=1;i<=8;i++){
        cout << "\n";
        for(int j=1;j<=8;j++)
        {
            cout << A[i][j] << " "; //print array A
        }
    }
```

```cpp
    }
    //Adjacency list
    cout << "\n\nAdjacency list:";
    for (int i = 1; i <= 8; i++)
    {
        cout << "\n" << i;
        for(int j = 1; j <= 8; j++)
        {
            if (A[i][j] == 1)
            {
                cout << " -> " << j; //turn A array into list representation
            }
        }
    }
```

讀取"adjacency_matrix.txt"，用同上次作業的讀取檔案方法並將讀取到的所有資料放入型別為 int 的二維陣列 A。

並將其資料陣列 A 轉為 list，然後再用巢狀 for 迴圈依序判斷，如果鄰接矩陣為 1 代表它跟這個 node 相鄰。以鄰接矩陣的 「0 1 0 0 1 0 0 0」為例，代表它跟分店 2 跟 5 相鄰，那它的鄰接列表就是 1 -> 2 -> 5 若 A[i][j]為 1 代表節點值為 1(即為有相連)，則用箭頭表示轉為 list，

```
124     //distance matrix
125     inFile.open("distance_matrix.txt");
126 
127     int D[8][8];
128     cout << "\n\nReading data....." << endl;
129     while(!inFile.eof()){
130         for(int i=1;i<=8;i++){
131             for(int j=1;j<=8;j++)
132             {
133                 inFile >> D[i][j];
134             }
135         }
136     }
137     cout << "Distance matrix: ";
138     for(int i=1;i<=8;i++)
139     {
140         cout << "\n";
141         for(int j=1;j<=8;j++)
142         {
143             cout << D[i][j] << " ";
144         }
145     }
```

讀取"distance_matrix.txt"，用同上次作業的讀取檔案方法並將讀取到的所有資料放入型別為 int 的二維陣列 D，並列印出陣列。

```
147     cout << "\n";
148     int s, e;
149     cout << "Please input the start shop(1~8):" << endl;  //let user input
150     cin >> s;
151     cout << "Please input the destinate shop(1~8):" << endl;
152     cin >> e;
153     int graph[8][8]={     //distance array to determine shortest path
154     {0, 8, 0, 0, 3, 0, 0, 0},
155     {0, 0, 0, 7, 0, 4, 0, 0},
156     {5, 6, 0, 0, 0, 0, 0, 0},
157     {0, 0, 0, 0, 0, 0, 6, 0},
158     {0, 0, 0, 0, 0, 0, 7, 8},
159     {0, 0, 0, 0, 2, 0, 0, 7},
160     {0, 6, 0, 0, 0, 0, 0, 9},
161     {0, 0, 2, 3, 0, 0, 0, 0}};
162     DijkstraAlgo(graph,s-1,e-1); //call the funtion
163 
164     return 0;
165 
166 }
167 
```

讓使用者輸入起始點和終點(1~8)，將該圖表每個點到點對應的距離用助教課程式的邏輯來做，最後呼叫 DijkstraAlgo 函式。但注意這邊輸入的起終點值在進入函式時會減 1 傳入，因為陣列都是從 0 開始，在函式內才不用再調整對應。

範例輸出結果

```
Reading data.....
Adjacency matrix:
0 1 0 0 1 0 0 0
0 0 0 1 0 1 0 0
1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 1
0 0 0 0 1 0 0 1
0 1 0 0 0 0 0 1
0 0 1 1 0 0 0 0

Adjacency list:
1 -> 2 -> 5
2 -> 4 -> 6
3 -> 1 -> 2
4 -> 7
5 -> 7 -> 8
6 -> 5 -> 8
7 -> 2 -> 8
8 -> 3 -> 4

Reading data.....
Distance matrix:
0 8 -1 -1 3 -1 -1 -1
-1 0 -1 7 -1 4 -1 -1
5 6 0 -1 -1 -1 -1 -1
-1 -1 -1 0 -1 -1 6 -1
-1 -1 -1 -1 0 -1 7 8
-1 -1 -1 -1 2 0 -1 7
-1 6 -1 -1 -1 -1 -1 9
-1 -1 2 3 -1 -1 -1 0
```

```
Please input the start shop:
1
Please input the destinate shop:
7

From shop 1 to shop 7
Shortest distance: 10
Diliver route: Shop1 -> Shop 5 -> Shop 7
```

```
Please input the start shop:
5
Please input the destinate shop:
6

From shop 5 to shop 6
Shortest distance: 17
Diliver route: Shop5 -> Shop 7 -> Shop 2 -> Shop 6
```