| Document Name | | Confidentiality Level |
|---|---|---|
| Scenic Information Management | | Only for Recipients' Reference |
| Template Code | Version | Document Code |
| BP | 1.0 | BP |

# Scenic Information Management System

## ——Operations and Applications of Graph

软酷网 www.RuanKo.com 软酷·卓越实验室 Centre Of Excellence

# Copyright Declaration

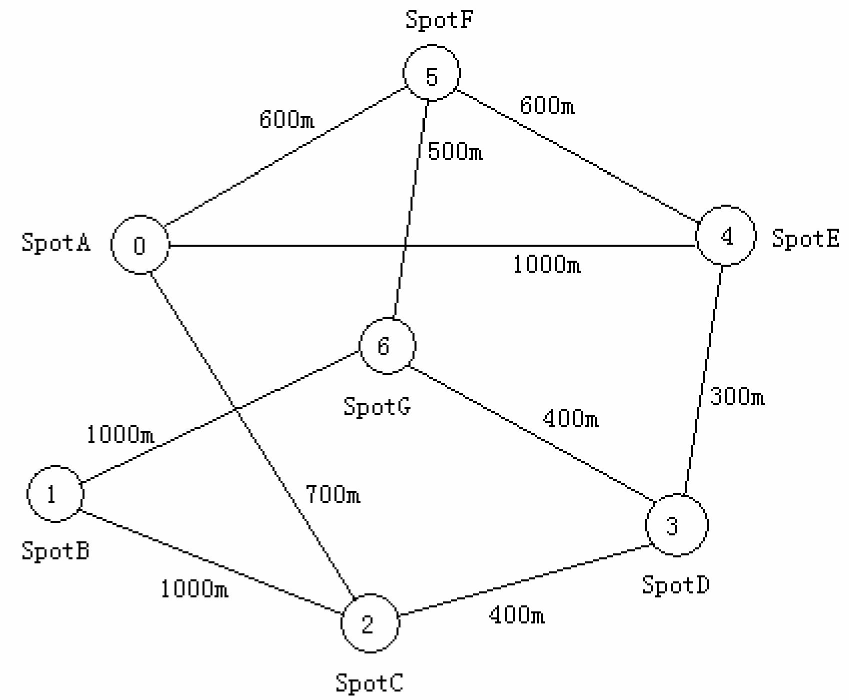
# Contents

# 1 Teaching Tips

(1) Master the definition and storage structure of the graph.
(2) Master how to create the graph.
(3) Master two traversal methods of the graph.
(4) Understand Dijkstra algorithm.
(5) Understand the concept of the minimum spanning tree and Prim algorithm.
(6) Master the file operations.
(7) Use C++ language and the data structure of the graph to develop the scenic information management system.

# 2 Task Requirements

Now, there is a scenic. This scenic contains some spots.



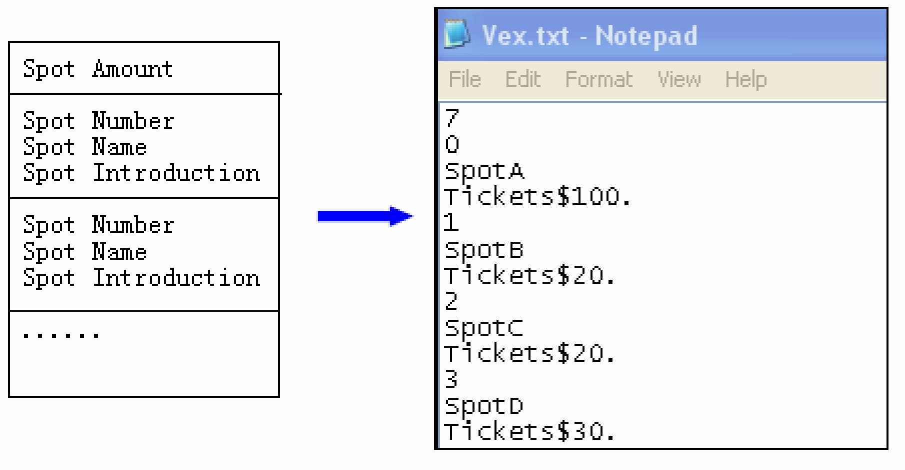
The scenic data contain the spot information and the road information between spots. They are saved in the two text files respectively. Vex.txt file is used to store the spot information. Edge.txt file is used to store the road information.

(1) Spot information: the spot number, name and introduction

| Number | Name | Introduction |
|---|---|---|
| 0 | Spot A | ... |
| 1 | Spot B | ... |
| 2 | Spot C | ... |
| 3 | Spot D | ... |
| 4 | Spot E | ... |

| | | |
|---|---|---|
| 5 | Spot F | ... |
| 6 | Spot G | ... |

The file format and examples are shown below(Vex.txt) :



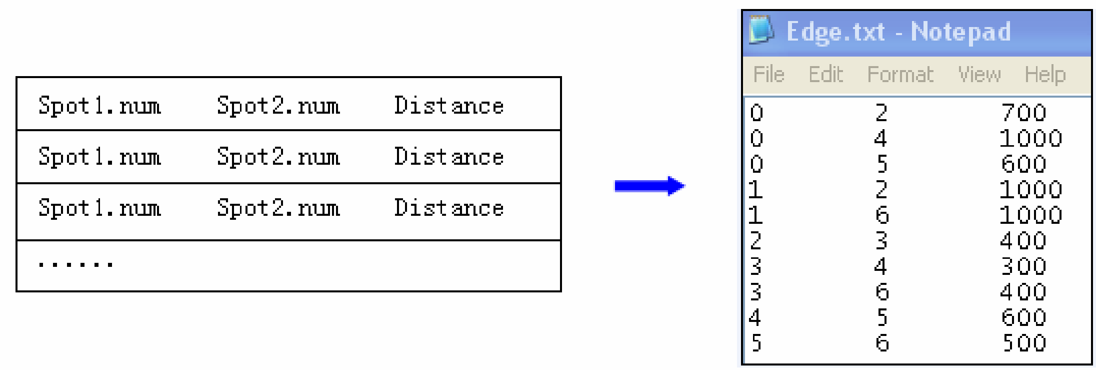
```
Spot Amount

Spot Number
Spot Name
Spot Introduction

Spot Number
Spot Name
Spot Introduction

......
```

```
7
0
SpotA
Tickets$100.
1
SpotB
Tickets$20.
2
SpotC
Tickets$20.
3
SpotD
Tickets$30.
```

(2) Road information: spot 1, spot 2, distance between two spots.

| Spot 1 | Spot 2 | Distance (m) |
|---|---|---|
| A | C | 700 |
| A | E | 1000 |
| A | F | 600 |
| B | C | 1000 |
| B | G | 1000 |
| C | D | 400 |
| D | E | 300 |
| D | G | 400 |
| E | F | 600 |
| F | G | 500 |

The file format and examples are shown below(Edge.txt):

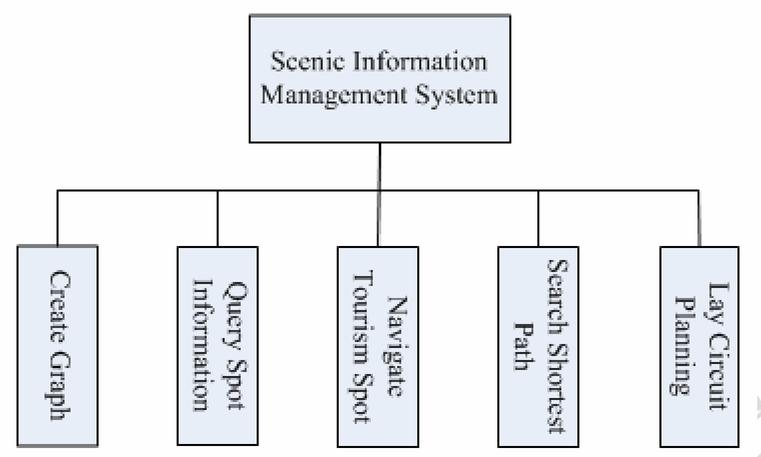


```
Spot1.num     Spot2.num     Distance

Spot1.num     Spot2.num     Distance

Spot1.num     Spot2.num     Distance

......
```

```
0         2         700
0         4         1000
0         5         600
1         2         1000
1         6         1000
2         3         400
3         4         300
3         6         400
4         5         600
5         6         500
```

If there are no information of two certain spots in the road information, it represents there is no direct path between the two spots.

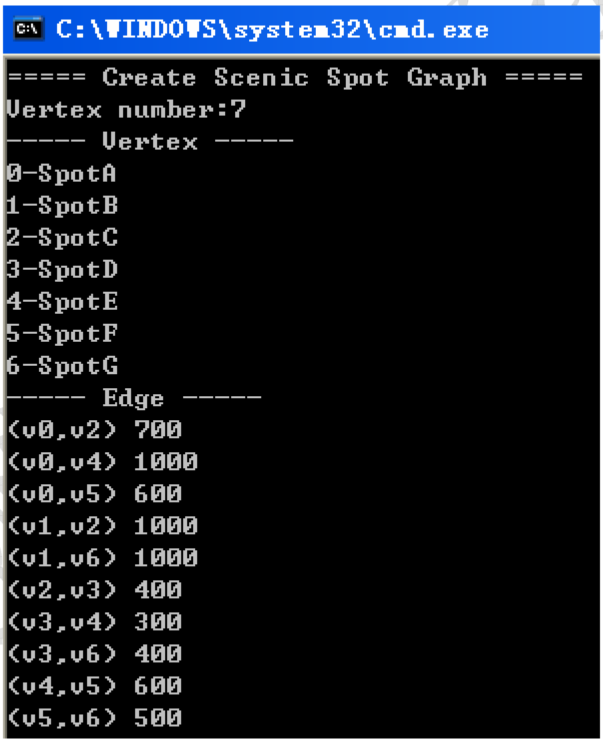
**The following figure shows the application functional structure.**



**(1) Create Graph**

Read the spot information from Vex.txt file, read the road information from Edge.txt file. Create the scenic spot graph according to the read scenic information.

The output format is shown below:

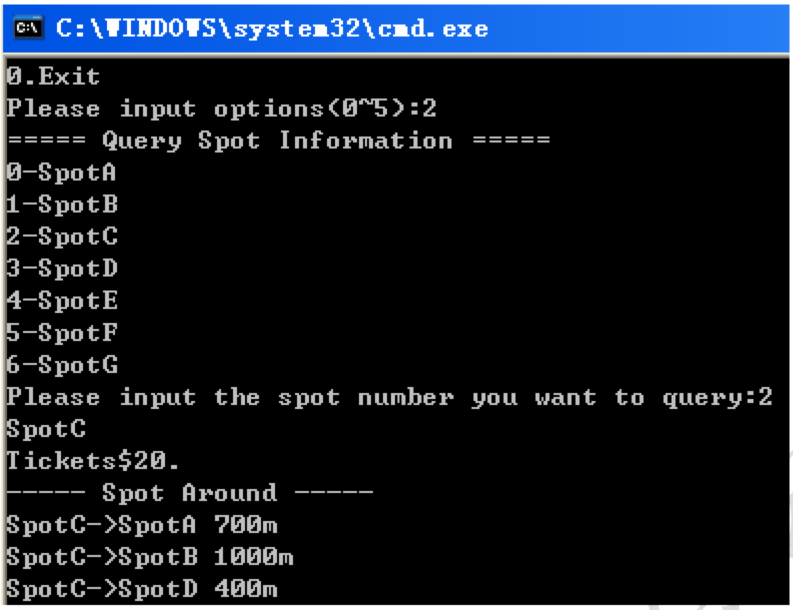


```
===== Create Scenic Spot Graph =====
Vertex number:7
----- Vertex -----
0-SpotA
1-SpotB
2-SpotC
3-SpotD
4-SpotE
5-SpotF
6-SpotG
----- Edge -----
<v0,v2> 700
<v0,v4> 1000
<v0,v5> 600
<v1,v2> 1000
<v1,v6> 1000
<v2,v3> 400
<v3,v4> 300
<v3,v6> 400
<v4,v5> 600
<v5,v6> 500
```

**(2) Query Spot Information**

According to the input spot number, query the information of the spot and the information of surrounding spots that can be directly reached.

The output format is shown below:



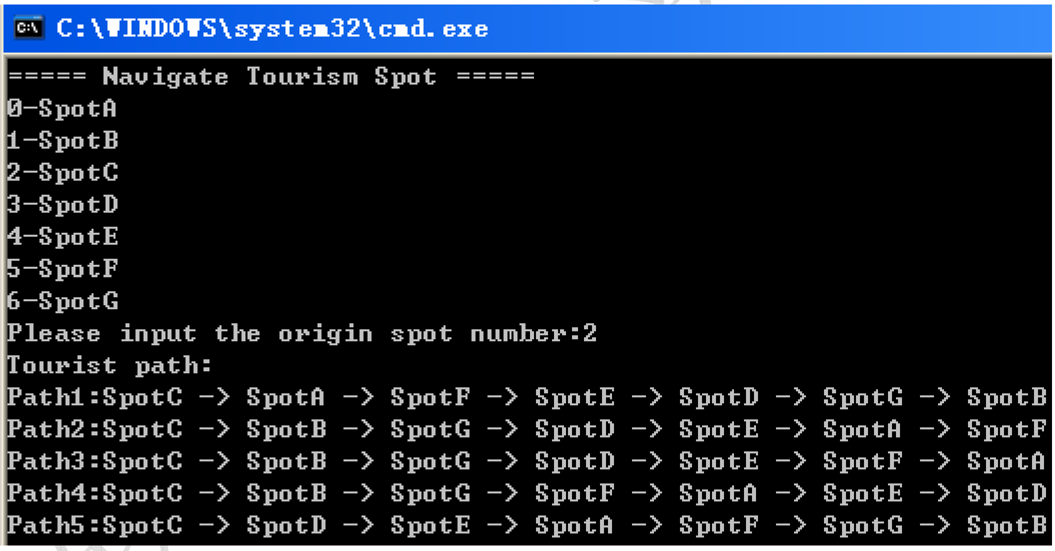
C:\WINDOWS\system32\cmd.exe

```
0.Exit
Please input options(0~5):2
===== Query Spot Information =====
0-SpotA
1-SpotB
2-SpotC
3-SpotD
4-SpotE
5-SpotF
6-SpotG
Please input the spot number you want to query:2
SpotC
Tickets$20.
----- Spot Around -----
SpotC->SpotA 700m
SpotC->SpotB 1000m
SpotC->SpotD 400m
```

### (3) Navigate Tourism Spot

Input the starting spot number. Use depth first search (DFS) algorithm to query the acyclic path that can visit the whole scenic from the spot.

The output format is shown below:



C:\WINDOWS\system32\cmd.exe

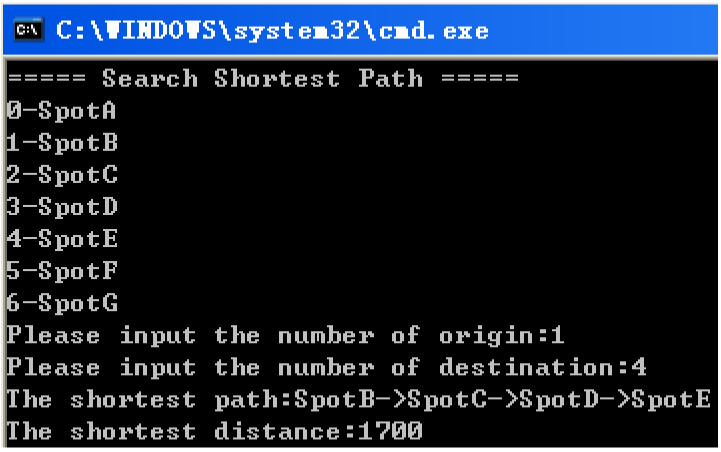
```
===== Navigate Tourism Spot =====
0-SpotA
1-SpotB
2-SpotC
3-SpotD
4-SpotE
5-SpotF
6-SpotG
Please input the origin spot number:2
Tourist path:
Path1:SpotC -> SpotA -> SpotF -> SpotE -> SpotD -> SpotG -> SpotB
Path2:SpotC -> SpotB -> SpotG -> SpotD -> SpotE -> SpotA -> SpotF
Path3:SpotC -> SpotB -> SpotG -> SpotD -> SpotE -> SpotF -> SpotA
Path4:SpotC -> SpotB -> SpotG -> SpotF -> SpotA -> SpotE -> SpotD
Path5:SpotC -> SpotD -> SpotE -> SpotA -> SpotF -> SpotG -> SpotB
```

### (4) Search Shortest Path

**Input:**

Input the number of the starting spot and the number of the ending spot. Use Dijkstra algorithm to get the shortest path and the total distance between the starting spot and the ending spot.

The output format is shown below:

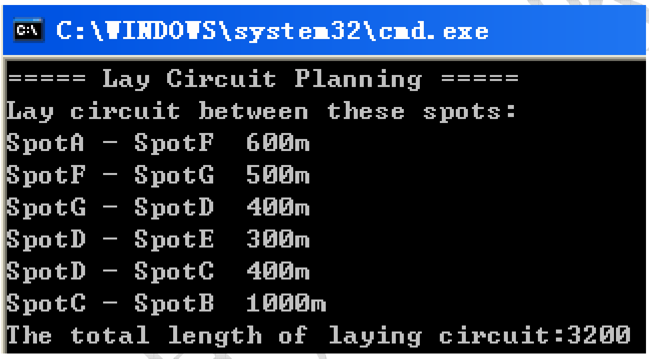


```
===== Search Shortest Path =====
0-SpotA
1-SpotB
2-SpotC
3-SpotD
4-SpotE
5-SpotF
6-SpotG
Please input the number of origin:1
Please input the number of destination:4
The shortest path:SpotB->SpotC->SpotD->SpotE
The shortest distance:1700
```

**(5) Lay Circuit Planning**

According to the scenic spot graph, use Prim algorithm to construct the minimum spanning tree and design a scheme that requires laying the shortest path and each spot must have electricity.

The output format is as below:



```
===== Lay Circuit Planning =====
Lay circuit between these spots:
SpotA - SpotF  600m
SpotF - SpotG  500m
SpotG - SpotD  400m
SpotD - SpotE  300m
SpotD - SpotC  400m
SpotC - SpotB  1000m
The total length of laying circuit:3200
```

# 3 Design Ideas

This program will directly read data from the two files of Vex.txt and Edge.txt. Develop “scenic information management system” with the storage structure of the graph, depth first search (DFS) algorithm, Dijkstra algorithm, Prim algorithm and so on.

## 3.1 Program Design

The project name is GraphCPro.

(1) Add CGraph class to define the data structure of the graph and implement the relative operations of the graph.

(2) Add CTourism class to implement the relative functionalities of the scenic information management system.

(3) Add Main.cpp file and create the entry function of the program int main(void) in the file.

## 3.2 Interface Design

Output the menus in int main(void) function to list the system functionalities for users.

| Input Number | Functionality |
|---|---|
| 1 | Create Graph |
| 2 | Query Spot Information |
| 3 | Navigate Tourism Spot |
| 4 | Search Shortest Path |
| 5 | Lay Circuit Planning |
| 0 | Exit |

## 3.3 Algorithm Design

**(1) Navigate Tourism Spot**

① Use the depth first search (DFS) algorithm to traverse the scenic spot graph and get a navigation path.

② Improve the depth first search algorithm to get multiple navigation paths.

**(2) Search Shortest Path**

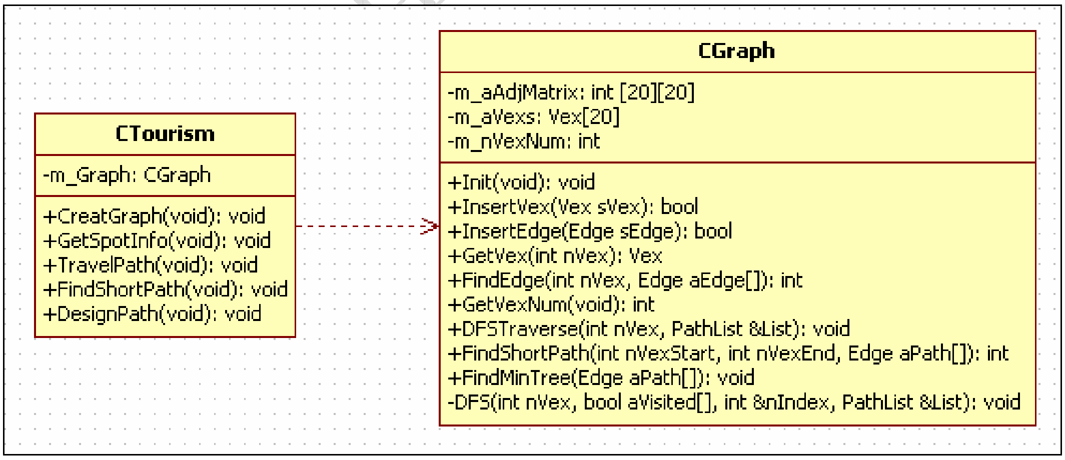
Use Dijkstra algorithm to get the shortest paths between 2 vertexes.

**(3) Circuit Layout Planning**

Use Prim algorithm to construct a minimum spanning tree to get the circuit layout planning.

## 3.4 Class Design

This program will use two classes that are CGraph class and CTourism class. CGraph class is used to implement the data structure and the relative operations of the graph. CTourism class is used to implement the main functionalities of the system.


CTourism
-m_Graph: CGraph
+CreatGraph(void): void
+GetSpotInfo(void): void
+TravelPath(void): void
+FindShortPath(void): void
+DesignPath(void): void
CGraph
-m_aAdjMatrix: int [20][20]
-m_aVexs: Vex[20]
-m_nVexNum: int
+Init(void): void
+InsertVex(Vex sVex): bool
+InsertEdge(Edge sEdge): bool
+GetVex(int nVex): Vex
+FindEdge(int nVex, Edge aEdge[]): int
+GetVexNum(void): int
+DFSTraverse(int nVex, PathList &List): void
+FindShortPath(int nVexStart, int nVexEnd, Edge aPath[]): int
+FindMinTree(Edge aPath[]): void
-DFS(int nVex, bool aVisited[], int &nIndex, PathList &List): void

## 3.5 Data Structure Design

When the graph structure is saved, the vertex information and the edge both need to save. The graph can be saved with the array or the linked list. In this program, the array is used to store the graph.

(1) Define Vex structure and store the vertex information of the graph.

```
struct Vex
{
        int num;                // Spot Number
        char name[20];          // Spot Name
        char desc[1024];        // Spot Instruction
};
```

(2) Define Edge structure and store the edge information of the graph

```
struct Edge
{
        int vex1;         // The first vertex of the edge
        int vex2;         // The second vertex of the edge
        int weight;       // Weight
};
```

# 4 Involving knowledge and technology

**1. Graph**

(1) The definition of graph

(2) The storage structure of the graph

The storage of the vertex: one-dimensional array

The storage of the edge: relational matrix

(3) Create the graph (Create Graph and Query Spot Information)

(4) Traverse the graph (Navigate Tourism Spot)

Depth first search (DSF) traversal

(5) The shortest path (Search Shortest Path)

Dijkstra algorithm

(6) The minimum spanning tree (Lay Circuit Planning)

Prim algorithm

**2. Class and Object**

(1) The definition of the class

(2) Member variable and member function

(3) Access Modifier : public,private,protected

**3. File Operations (Create Graph)**

(1) Open the file: fopen()

(2) Read formatted file: fscanf()

(3) Close the file: fclose()

# 5 Implementation Idea

This program will be iteratively developed. The content of each iteration will be explained in some topics. The iterations are shown below:

(1) Create Graph and Query Spots Information.
(2) Navigate Tourism Spot
(3) Search Shortest Path.
(4) Lay Circuit Planning

**1. Iteration Development Ideas**

The incremental method of just designing and implementing a part of the system every time is called the iterative development. The iterations of this project are based on the functionalities. Each iteration can be developed as an entire project development.

**2. Programming and Implementation**

There are four technical subject for the content of each iteration.

**(1) Create Graph and Query Spot Information**

① Create project GraphCPro
② Define graph CGraph class
③ Create the scenic spot graph
④ Query the spot information
⑤ Compile and run

**(2) Navigate Tourism Spot**

① Traverse the scenic spot graph
② Query the navigation path
③ Compile and run

**(3) Search Shortest Path**

① Search for the shortest path
② Query the shortest path
③ Compile and run

**(4) Lay Circuit Planning**

① Construct the minimum spanning tree.
② Query the circuit layout planning graph
③ Compile and run