实验五　图的最短路径实验

一、实验目的

- 掌握图结构的（邻接矩阵）输入方法
- 掌握图结构的说明、创建以及图的存储表示（邻接矩阵）
- 掌握最短路径算法原理
- 掌握最短路径算法的编程实现方法

实验五　图的最短路径实验

二、实验要求

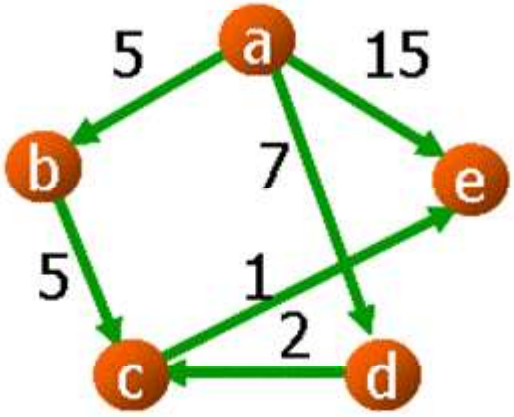
- 熟悉C++语言编程
- 熟悉图的邻接矩阵存储表示
- 熟悉最短路径算法原理
- 熟练使用C++语言，实现最短路径算法

实验五　图的最短路径实验

三、实验内容

1、问题描述

- 给定一个顶点（始点），求该顶点（始点）到（连通）图中其它顶点的最短路径。



$$\begin{pmatrix} \infty & 5 & \infty & 7 & 15 \\ \infty & \infty & 5 & \infty & \infty \\ \infty & \infty & \infty & \infty & 1 \\ \infty & \infty & 2 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

实验五　图的最短路径实验

三、实验内容

顶点	D[i]			
b	5 {a,b}			
c	∞	10 {a,b,c}	9 {a,d,c}	
d	7 {a,d}	7 {a,d}		
e	15 {a,e}	15 {a,e}	15 {a,e}	10 {a,d,c,e}
终点j	b	d	c	e
S	{a,b}	{a,b,d}	{a,b,d,c}	{a,b,d,c,e}

2、图的最短路径算法

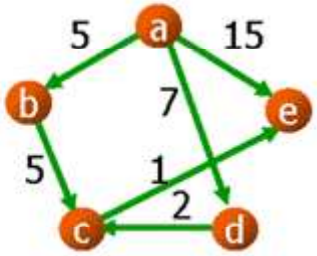
1、初始化：　S ← {v1}；　　// 始点送S

　D[i] ← arc[1][i]，　i = 2, 3, …, n;　// 从v1到vi的距离

　P[i] = {1, i}　　// 从v1到vi的路径

2、求出最短路径的长度：

　D[j] ← min { D[i] }，　i∈ V-S; S←S U {j}；



3、修改：

　if (D[i] > D[j] + arc[j][i]) {D[i] = D[j]+arc[j][i];

　P[i] = P[j] U {i}; }　i∈ V-S　// 更新从v1到vi的路径

4、判断：若 S = V，则算法结束，否则转 2

实验五　图的最短路径实验

三、实验内容

3、输入

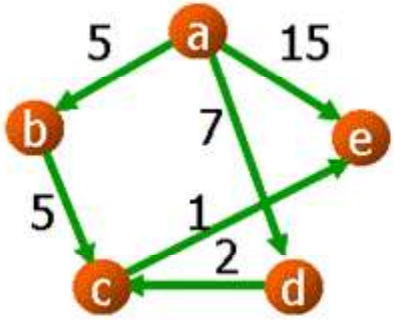
- 第一行：样本顶点个数，假设为n。
- 第二行，n个顶点（用空格隔开）
- 第三行开始到n+2行：每一行是某顶点（按第二行的输入为序）与其它顶点的距离（-1表示无穷大）
- 第n+3行：开始顶点

实验五　图的最短路径实验

三、实验内容

4、输入样本

```
5
a b c d e
-1 5 -1 7 15
-1 -1 5 -1 -1
-1 -1 -1 -1 1
-1 -1 2 -1 -1
-1 -1 -1 -1 -1
a
```



$$\begin{pmatrix} \infty & 5 & \infty & 7 & 15 \\ \infty & \infty & 5 & \infty & \infty \\ \infty & \infty & \infty & \infty & 1 \\ \infty & \infty & 2 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

实验五　图的最短路径实验

三、实验内容

5、输出

- 共计n行（图中顶点数目）
- 每行是（与输入顺序相同）

  某顶点（距离）：路径（顶点序列，用空格隔开，回车前无空格）

实验五　图的最短路径实验

三、实验内容

6、输出样本

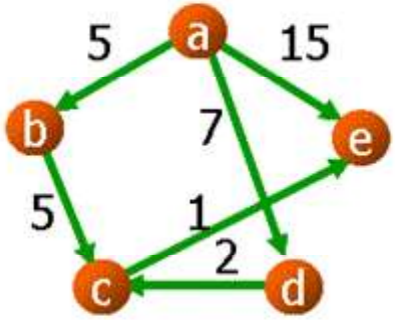
a(0)：

b(5)： a b

c(9)： a d c

d(7)： a d

e(10)： a d c e



$$\begin{pmatrix} \infty & 5 & \infty & 7 & 15 \\ \infty & \infty & 5 & \infty & \infty \\ \infty & \infty & \infty & \infty & 1 \\ \infty & \infty & 2 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

实验五　图的最短路径实验

四、实验步骤

1、图的定义（邻接矩阵）

2、创建图的邻接矩阵

3、显示图的邻接矩阵

4、图的最短路径函数

5、显示图的最短路径

6、主函数

实验五　图的最短路径实验

四、实验步骤

1、图的定义（邻接矩阵）

```
#define MAXVERTEXNUM    100                          // 最大顶点数
struct Graph {
   int  VertexNum;                                   // 顶点数
   char Vertex[MAXVERTEXNUM];                        // 顶点数据
   int  AdjMatrix[MAXVERTEXNUM][MAXVERTEXNUM];//邻接矩阵
};
Graph MGraph;
```

实验五　图的最短路径实验

四、实验步骤

2、创建图的邻接矩阵

```
#define INFINITY        100                          // 无穷大
void CreateGraph(Graph *G)                           // 生成图（采用邻接矩阵）
{  int  i, j;
   cin >> G->VertexNum;                              // 输入顶点样本数目
   for (i=1; i<=G->VertexNum; i++)
        cin >> G->Vertex[i];                         // 输入顶点
   for (i=1; i<=G->VertexNum; i++) {
        for (j=1; j<=G->VertexNum; j++) {
            cin >> G->AdjMatrix[i][j];               // 依次输入邻接矩阵
            if (G->AdjMatrix[i][j] == -1) G->AdjMatrix[i][j] = INFINITY;
}}}
```

实验五　图的最短路径实验

四、实验步骤

3、显示邻接矩阵（调试时用）

```
a b c d e
100 5 100 7 15
100 100 5 100 100
100 100 100 100 1
100 100 2 100 100
100 100 100 100 100
```

```
void ShowGraph(Graph *G)
{  int  i, j;
   for (i=1; i<=G->VertexNum; i++)
        cout << G->Vertex[i] << " ";                    // 输出顶点
   cout << endl;
   for (i=1; i<=G->VertexNum; i++) {
        for (j=1; j<=G->VertexNum; j++) {
             cout << G->AdjMatrix[i][j] << “ ”;}//依次输出邻接矩阵
        cout << endl;}}
```

实验五　图的最短路径实验

四、实验步骤

4、图的最短路径函数

```
char Path[MAXVERTEXNUM][MAXVERTEXNUM];                    // 采用顺序表，存放顶点的最短路径
int Dest[MAXVERTEXNUM];                                   // 存放顶点的最短距离
void ShortestPath(Graph *G, char StartVexChar)
{   int i, j, m, StartVex, CurrentVex, MinDest, Final[MAXVERTEXNUM];
    for (i=1; i<=G->VertexNum; i++) {                     // 找到开始顶点的序号
         if (G->Vertex[i] == StartVexChar) {StartVex = i; break;}}
    for (i=1; i<=G->VertexNum; i++) {
         Path[i][0] = 0;                  // 顺序表的0位置，存放顺序表(路径)的长度
         Dest[i] = INFINITY;              // 所有顶点到开始顶点之间的距离初值设为无穷大
         if (G->AdjMatrix[StartVex][i] < INFINITY) {//在开始顶点与当前顶点之间存在弧
             Dest[i] = G->AdjMatrix[StartVex][i];
             Path[i][1] = G->Vertex[StartVex];
             Path[i][2] = G->Vertex[i];
             Path[i][0] = 2;}
         Final[i] = 'F';}
```

实验五　图的最短路径实验

四、实验步骤

4、图的最短路径函数

```
Dest[StartVex] = 0;                    // 初始化，开始顶点属于S集（已处理过的顶点）
Final[StartVex] = 'T';
for (i=1; i<=G->VertexNum; i++) {
    MinDest = INFINITY;
    for (j=1; j<=G->VertexNum; j++) {//找当前未处理过顶点中到开始顶点最近的顶点
        if (Final[j] == 'F') {
            if (Dest[j] < MinDest) {CurrentVex = j;    MinDest = Dest[j];}}}
    Final[CurrentVex] = 'T';
    for (j=1; j<=G->VertexNum; j++) {              // 更新当前最短路径及距离
        if ((Final[j]=='F')&&(MinDest+G->AdjMatrix[CurrentVex][j]<Dest[j])) {
            Dest[j] = MinDest+G->AdjMatrix[CurrentVex][j];//更新顶点j的最短距离
            for (m=0; m<=Path[CurrentVex][0]; m++)// 更新顶点j到开始顶点的路径
                Path[j][m] = Path[CurrentVex][m];
            Path[j][0]++;
            Path[j][Path[j][0]] = G->Vertex[j];}}}}
```

实验五　图的最短路径实验

四、实验步骤

5、显示图的最短路径

```
void ShowPath(Graph *G)
{  int i, j;
   for (i=1; i<=G->VertexNum; i++) {
        cout << G->Vertex[i] << "(" << Dest[i] << "): ";
        if (Path[i][0] > 0) {
            for (j=1; j<=Path[i][0]; j++) {
                cout << " " << Path[i][j];}}
        cout << Path[i][j] << endl;
}}
```

实验五　图的最短路径实验

四、实验步骤

6、主函数

```
int main( )
{
   char StartVex;

   CreateGraph(&MGraph);                    // 生成图（采用邻接矩阵）
   ShowGraph(&MGraph);
   cin >> StartVex;
   ShortestPath(&MGraph, StartVex);
   ShowPath(&MGraph);
   return 0;
}
```