实验-Dijkstra算法

#include <iostream>  
#include "Adjgraph.h"  
using namespace std;  
void Dijkstra(AdjGraph G, int v0, int \*dist, int \*path)  
{  
 int n = G.n;  
 bool \*S = new bool[n];  
 for (int i = 0; i < n; i++)  
 {  
 dist[i] = G.edges[v0][i];  
 S[i] = false;  
 if (dist[i] < **INT\_MAX**)  
 path[i] = v0;  
 else  
 path[i] = -1;  
 }  
 dist[v0] = 0;  
 S[v0] = true;  
 for (int i = 1; i < n; i++)  
 {  
 int mindist = **INT\_MAX**;  
 int u = v0;  
 for (int j = 0; j < n; j++)  
 if (!S[j] && dist[j] < mindist)  
 {  
 u = j;  
 mindist = dist[j];  
 }  
 S[u] = true;  
 for (int j = 0; j < n; j++)  
 if (!S[j] && G.edges[u][j] < **INT\_MAX**)  
 {  
 int newdist = dist[u] + G.edges[u][j];  
 if (newdist < dist[j])  
 {  
 dist[j] = newdist;  
 path[j] = u;  
 }  
 }  
 }  
 delete[] S;  
}  
int main()  
{  
 AdjGraph G;  
 const int N = G.n;  
 int dist[N], path[N];  
 G.CreateAdjGraph();  
 G.DisplayAdjGraph();  
 Dijkstra(G, 0, dist, path);  
 for (int i = 0; i < G.n; i++)  
 {  
 cout << "从顶点0到顶点" << i << "的最短路径长度为：" << dist[i] << endl;  
 cout << "路径为：";  
 int j = i;  
 while (j != 0)  
 {  
 cout << j << "<-";  
 j = path[j];  
 }  
 cout << "0" << endl;  
 }  
 return 0;  
}

#include <iostream>  
#include "Adjgraph.h"  
using namespace std;void Floyd(AdjGraph G, int path[][MAXVEX], int dist[][MAXVEX])  
{  
 int i, j, k;  
 for (i = 0; i < G.numVertexes; i++)  
 {  
 for (j = 0; j < G.numVertexes; j++)  
 {  
 dist[i][j] = G.arc[i][j];  
 path[i][j] = j;  
 }  
 }  
 for (k = 0; k < G.numVertexes; k++)  
 {  
 for (i = 0; i < G.numVertexes; i++)  
 {  
 for (j = 0; j < G.numVertexes; j++)  
 {  
 if (dist[i][j] > dist[i][k] + dist[k][j])  
 {  
 dist[i][j] = dist[i][k] + dist[k][j];  
 path[i][j] = path[i][k];  
 }  
 }  
 }  
 }  
}  
*//打印路径*void PrintPath(AdjGraph G, int path[][MAXVEX], int dist[][MAXVEX])  
{  
 int i, j, k;  
 for (i = 0; i < G.numVertexes; i++)  
 {  
 for (j = 0; j < G.numVertexes; j++)  
 {  
 if (i != j)  
 {  
 cout << "v" << i << "-->v" << j << " weight: " << dist[i][j] << " path: " << i;  
 k = path[i][j];  
 while (k != j)  
 {  
 cout << "-->" << k;  
 k = path[k][j];  
 }  
 cout << "-->" << j << endl;  
 }  
 }  
 }  
}  
int main()  
{  
 AdjGraph G;  
 CreateGraph(G);  
 int path[MAXVEX][MAXVEX], dist[MAXVEX][MAXVEX];  
 Floyd(G, path, dist);  
 PrintPath(G, path, dist);  
 return 0;  
}