Prim算法

#include<stdio.h>

#include <windows.h>

#define INF 999

#define MAXV 100

int A[MAXV][MAXV] = {

{0,5,8,7,INF,3},

{5,0,4,INF,INF,INF},

{8,4,0,5,INF,9},

{7,INF,5,0,5,6},

{INF,INF,INF,5,0,1},

{3,INF,9,6,1,0}

};

typedef struct

{

int edges[MAXV][MAXV];

int n;

int e;

}MGraph;

typedef struct

{

int first;

int last;

int weight;

}Path;

void color(int x) //设置字体颜色

{

if (x >= 0 && x <= 15)

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), x);

else

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 7);

}

//创建图

void CreateMGraph(MGraph& g, int A[][MAXV], int n, int e)

{

for (int i = 0;i < n;i++)

{

for (int j = 0;j < n;j++)

{

g.edges[i][j] = A[i][j];

}

}

g.n = n;

g.e = e;

}

//Prim算法

void Prim(MGraph g, int v)

{

int lowcost[MAXV];

int closest[MAXV];

Path path[MAXV];

int k;

for (int i = 0;i < g.n;i++)

{

lowcost[i] = g.edges[v][i];

closest[i] = 0;

}

for (int i = 1;i < g.n;i++)

{

printf("第%d个步骤:\n", i );

int min = INF;

for (int j = 0;j < g.n;j++)

{

if (lowcost[j] != 0 && lowcost[j] < min)

{

k = j;

min = lowcost[j];

}

}

path[i].first = closest[k];

path[i].last = k;

path[i].weight = lowcost[k];

lowcost[k] = 0;

for (int t = 0;t < g.n;t++)

{

if (lowcost[t] > g.edges[k][t])

{

lowcost[t] = g.edges[k][t];

closest[t] = k;

}

}

printf("lowcost:[");

for (int j = 0;j < g.n;j++)

{

printf("%4d", lowcost[j]);

}

printf("]\nclosest:[");

for (int j = 0;j < g.n;j++)

{

printf("%4d", closest[j]);

}

printf("]\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

for (int h = 1;h < g.n;h++)

{

printf("边(%d,%d):%d\n", path[h].first, path[h].last, path[h].weight);

}

}

int main()

{

MGraph g;

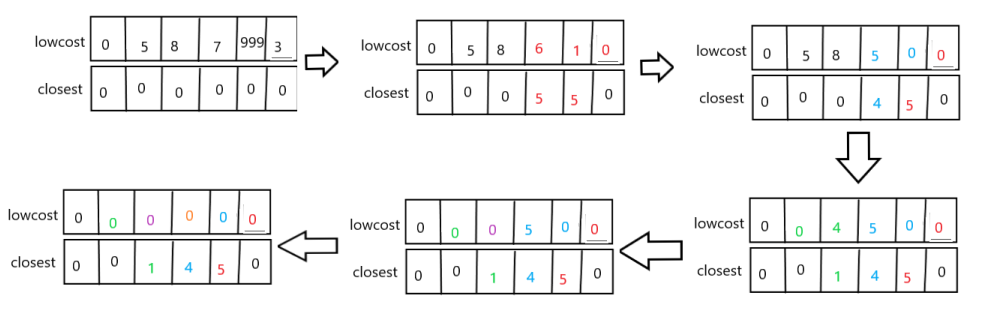
CreateMGraph(g, A,6,10);

Prim(g, 0);

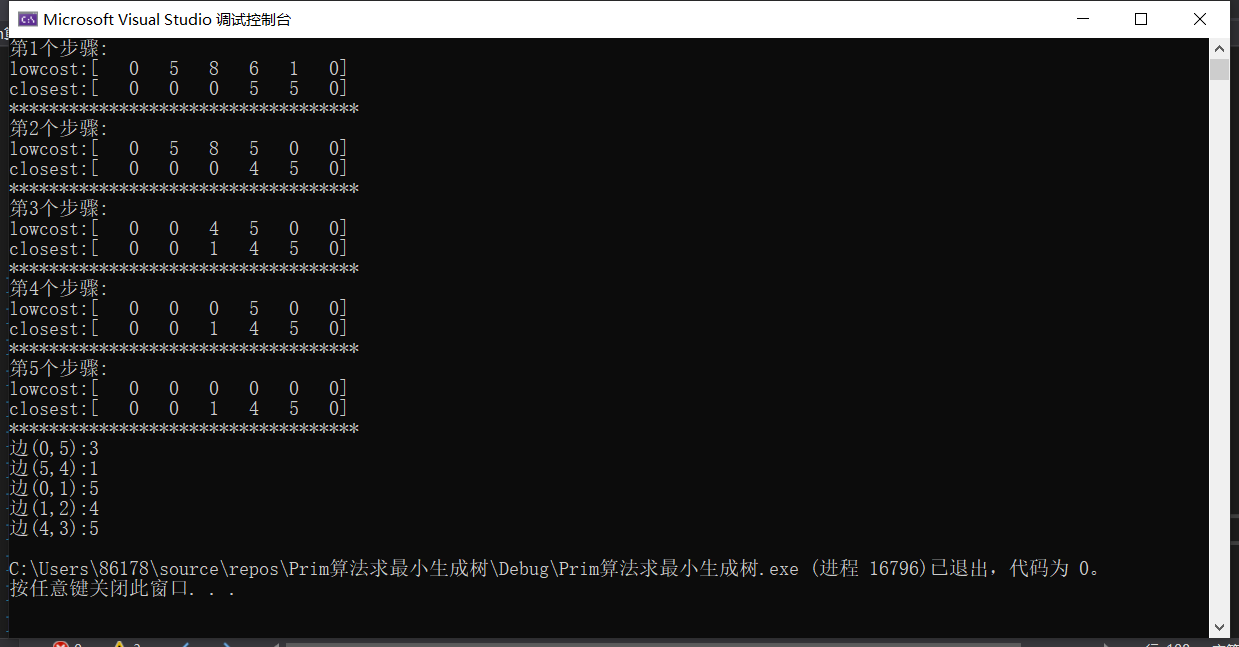
return 0;

}

手工计算：



程序运行：



Dijkstra算法

#include<stdio.h>

#include <windows.h>

#define INF 999

#define MAXV 100

int A[MAXV][MAXV] = {

{0,5,INF,7,INF,INF},

{INF,0,4,INF,INF,INF},

{8,INF,0,INF,INF,9},

{INF,INF,5,0,INF,6},

{INF,INF,INF,5,0,INF},

{3,INF,INF,INF,1,0}

};

typedef struct

{

int edges[MAXV][MAXV];

int n;

int e;

}MGraph;

void color(int x) //设置字体颜色

{

if (x >= 0 && x <= 15)

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), x);

else

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 7);

}

//创建图

void CreateMGraph(MGraph& g,int A[][MAXV],int n,int e)

{

for (int i = 0;i < n;i++)

{

for (int j = 0;j < n;j++)

{

g.edges[i][j] = A[i][j];

}

}

g.n = n;

g.e = e;

}

//Dijkstra算法

void Ppath(int path[],int k,int v)

{

int p=path[k];

if (v == p)

{

return;

}

Ppath(path, p, v);

printf("%2d", p);

}

void Dispath(int s[], int dist[], int path[], int n, int v,int u)

{

printf("从%d到%d的最短路径：", v, u);

printf("%2d", v);

Ppath(path, u, v);

printf("%2d\n", u);

}

void Dijkstra(MGraph g, int v)

{

int u;

int s[MAXV];

int dist[MAXV], path[MAXV];

for (int i = 0;i < g.n;i++)

{

s[i] = 0;

dist[i] = g.edges[v][i];

if (dist[i] < INF)

{

path[i] = 0;

}

else

{

path[i] = -1;

}

}

s[v] = 1;

path[v] = 0;

for (int i = 0;i < g.n-1;i++)

{

int min=INF;

for (int j = 0;j < g.n;j++)

{

if (s[j] == 0 && dist[j] < min)

{

u = j;

min = dist[j];

}

}

s[u] = 1;

color(2);

printf("第%d个步骤:\n", i + 1);

color(16);

printf("s:\t[");

for (int k = 0;k < g.n;k++)

{

if (k == u)

{

color(4);

printf("%4d", s[k]);

color(7);

}

else

{

printf("%4d", s[k]);

}

}

printf("]\ndist:\t[");

for (int j = 0;j < g.n;j++)

{

if (dist[j] > dist[u] + g.edges[u][j])

{

color(6);

printf("%4d", dist[u] + g.edges[u][j]);

path[j] = u;

color(16);

}

else

{

printf("%4d", dist[j]);

}

}

printf("]\npath:\t[");

for (int j = 0;j < g.n;j++)

{

if (dist[j] > dist[u] + g.edges[u][j])

{

color(9);

dist[j] = dist[u] + g.edges[u][j];

path[j] = u;

printf("%4d",path[j]);

color(16);

}

else

{

printf("%4d", path[j]);

}

}

printf("]\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

Dispath(s, dist, path, g.n, v,u);

}

}

int main()

{

MGraph g;

CreateMGraph(g, A, 6, 10);

Dijkstra(g, 0);

return 0;

}

