/\*\*

\* dijkstra.c 迪杰斯特拉算法

\*

\* @author Darbuly 970073804@qq.com

\* @copyright 2018-2019

\*/

#include <stdio.h>

#include <stdlib.h>

#define VRType int

#define InfoType int

#define VertexType char

#define MAX\_VERTEX\_NUM 20

#define MAX\_VALUE 65535

typedef enum{DG,DN,AG,AN}GraphKind;

typedef int Patharc[MAX\_VERTEX\_NUM];//用于储存最短路径下标的数组

typedef int ShortPathTable[MAX\_VERTEX\_NUM];//用于储存到各点最短路径的权值和

typedef struct ArcCell{

VRType adj;

InfoType \*info;

}ArcCell,AdjMatrix[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];

typedef struct

{

VertexType vexs[MAX\_VERTEX\_NUM];

AdjMatrix arc;

int vexnum,arcnum;

GraphKind kind;

}MGraph;

void g\_create(MGraph \*graph)

{

int num;

int i,j,k;

char c;

printf("Please enter the number of vertex:\n");

scanf("%d",&graph->vexnum);

getchar();

printf("Please enter the vertex infomations:\n");

for(i=0;i<graph->vexnum;i++)

{

scanf("%c",&graph->vexs[i]);

getchar();

}

for(i=0;i<graph->vexnum;i++)

for(j=0;j<graph->vexnum;j++)

graph->arc[i][j].adj=MAX\_VALUE;

graph->arcnum=0;

for(i=0;i<graph->vexnum;i++)

{

printf("Please enter vertex nextto the %c , and end by #\n",graph->vexs[i]);

for(j=0;j<graph->vexnum;j++)

{

scanf("%c",&c);

if('#'==c)

{

getchar();

break;

}

scanf("%d",&num);

for(k=0;k<graph->vexnum;k++)

{

if(graph->vexs[k]!=c)

continue;

graph->arc[i][k].adj = num;

graph->arcnum++;

}

getchar();

}

}

graph->arcnum /=2;

printf("\n");

for(i=0;i<graph->vexnum;i++)

{

for(j=0;j<graph->vexnum;j++)

{

printf("%5d\t",graph->arc[i][j].adj);

}

printf("\n");

}

}

void ShortestPath\_Dijkstra(MGraph G,int V0,Patharc \*P,ShortPathTable \*D)

{

int v,w,k,min;

int final[MAX\_VERTEX\_NUM];

for(v=0;v<G.vexnum;v++)

{

final[v]=0;

(\*D)[v] = G.arc[V0][v].adj;

(\*P)[v] = 0;

}

(\*D)[V0] = 0;

final[V0] = 1;

for(v=1;v<G.vexnum;v++)

{

min = MAX\_VALUE;

for(w=0;w<G.vexnum;w++)

{

if(!final[w] && (\*D)[w]<min)

{

k = w;

min = (\*D)[w];

}

}

final[k] = 1;

for(w=0;w<G.vexnum;w++)

{

if(!final[w] && min+G.arc[k][w].adj< (\*D)[w])

{

(\*D)[w] = min + G.arc[k][w].adj;

(\*P)[w] = k;

}

}

}

printf("\n");

for(v=0;v<G.vexnum;v++)

{

printf("%d\t",(\*P)[v]);

}

}

int main()

{

MGraph graph;

Patharc P;

ShortPathTable D;

g\_create(&graph);

ShortestPath\_Dijkstra(graph,0,&P,&D);

printf("Dijkstra.c Test\n");

return 0;

}

代码分析:

1. 时间复杂度：O(n^2)
2. 思路分：
   1. 其实，原理就是普里姆算法，只不过，在判断的时候，通过记录每个顶点的最短路径值，对比多路，淘汰长的路径，选出当前顶点下一个最佳路径的方法
   2. P的作用，就是记录下标顶点的前驱