

noj实验8报告

0.0 题目//需求分析

Description

给一个赋权图（无向图），求0号结点到其余所有结点的最短路径的长度。

Input

先输入一个小于等于100的正整数n，然后输入赋权图的邻接矩阵（10000表10000）

Output

按结点编号的顺序输出0号结点所有结点的最短路径的长度。

Sample Input

```
6
0?1?4?10000?10000?10000
1?0?2?7?5?10000
4?2?0?10000?1?10000
10000?7?10000?0?3?2
10000?5?1?3?0?6
10000?10000?10000?2?6?0
```

Sample Output

```
0
1
3
7
4
0
```

需求分析：就是利用缔结斯塔拉算法，执行矩阵更新步骤，把最短路径找出来。

1.0 实验思路

就是缔结斯塔拉算法。

通过Dijkstra计算图G中的最短路径时，需要指定起点s(即从顶点s开始计算)。

此外，引进两个集合S和U。S的作用是记录已求出最短路径的顶点(以及相应的最短路径长度)，而U则是记录还未求出最短路径的顶点(以及该顶点到起点s的距离)。

初始时，S中只有起点s；U中是除s之外的顶点，并且U中顶点的路径是"起点s到该顶点的路径"。然后，从U中找出路径最短的顶点，并将其加入到S中；接着，更新U中的顶点和顶点对应的路径。然后，再从U中找出路径最短的顶点，并将其加入到S中；接着，更新U中的顶点和顶点对应的路径。... 重复该操作，直到遍历完所有顶点。

操作步骤是：

- (1) 初始时，S只包含起点s；U包含除s外的其他顶点，且U中顶点的距离为"起点s到该顶点的距离"[例如，U中顶点v的距离为(s,v)的长度，然后s和v不相邻，则v的距离为 ∞]。
- (2) 从U中选出"距离最短的顶点k"，并将顶点k加入到S中；同时，从U中移除顶点k。
- (3) 更新U中各个顶点到起点s的距离。之所以更新U中顶点的距离，是由于上一步中确定了k是求出最短路径的顶点，从而可以利用k来更新其它顶点的距离；例如，(s,v)的距离可能大于(s,k)+(k,v)的距离。
- (4) 重复步骤(2)和(3)，直到遍历完所有顶点。

2.0 代码

```
#include <stdio.h>
#include <stdlib.h>

struct graphList
{
    int vexNum;
    int graph[120][120];
};

struct step
{
    int flags[3000];
    int stepN[3000];
};

void run();
void createNewGraphList (struct graphList *gList);
void DJ(struct step *gStep,struct graphList *gList);
void clearStep(struct step *gStep,struct graphList *gList);
void initializeStep(struct step *gStep,struct graphList *gList);
int judgeStep(struct step *gStep,struct graphList *gList);
int findMinStepN(struct step *gStep,struct graphList *gList);
void updateStepN(struct step *gStep,struct graphList *gList,int min);
void print(struct step *gStep,struct graphList *gList);

int main()
{
    run ();
    return 0;
}

void run()
{
    struct graphList gList;
    struct step gStep;
    createNewGraphList (&gList);
    DJ (&gStep,&gList);
    print (&gStep,&gList);
}
```

```

void createNewGraphList(struct graphList *gList)
{
    int i,j;
    scanf ("%d",&(gList->vexNum));
    for (i=0;i<gList->vexNum;i++)
    {
        for (j=0;j<gList->vexNum;j++)
        {
            scanf ("%d",&(gList->graph[i][j]));
        }
    }
}

void DJ(struct step *gStep,struct graphList *gList)
{
    int min;
    clearStep (gStep,gList);
    initializeStep (gStep,gList);
    while (judgeStep (gStep,gList))
    {
        min=findMinStepN (gStep,gList);
        updateStepN (gStep,gList,min);
    }
}

void clearStep(struct step *gStep,struct graphList *gList)
{
    int i;
    for (i=0;i<gList->vexNum;i++)
    {
        gStep->flags[i]=-1;
        gStep->stepN[i]=0;
    }
}

void initializeStep(struct step *gStep,struct graphList *gList)
{
    int i;
    for (i=0;i<gList->vexNum;i++)
    {
        if (gList->graph[0][i]!=10000)
        {
            gStep->flags[i]=1;
            gStep->stepN[i]=gList->graph[0][i];
        }
    }
}

int judgeStep(struct step *gStep,struct graphList *gList)
{
    int i;
    for (i=1;i<gList->vexNum;i++)
    {
        if (gStep->flags[i]==1)
        {
            return 1;
        }
    }
}

```

```

    }
    return 0;
}

int findMinStepN(struct step *gStep, struct graphList *gList)
{
    int i, min=99999, n=-1;
    for (i=1; i<gList->vexNum; i++)
    {
        if (gStep->flags[i]==1)
        {
            if (gStep->stepN[i]<min)
            {
                min=gStep->stepN[i];
                n=i;
            }
        }
    }
    return n;
}

void updateStepN(struct step *gStep, struct graphList *gList, int min)
{
    int i;
    int minStepN=gStep->stepN[min];
    gStep->flags[min]=0;
    for (i=0; i<gList->vexNum; i++)
    {
        if (gStep->flags[i]==1)
        {
            if (gStep->stepN[i]>gList->graph[min][i]+minStepN)
            {
                gStep->stepN[i]=gList->graph[min][i]+minStepN;
            }
        }
        else
        {
            if (gStep->flags[i]==-1)
            {
                gStep->flags[i]=1;
                gStep->stepN[i]=gList->graph[min][i]+minStepN;
            }
        }
    }
}

void print(struct step *gStep, struct graphList *gList)
{
    int i;
    for (i=0; i<gList->vexNum; i++)
    {
        printf ("%d\n", gStep->stepN[i]);
    }
}

```

3.0 心得

掌握了图的遍历以及缔结斯塔拉这种算法。