# 习题4 图结构

4-1 B

4-2 A

4-3 D

4-4 B

4-5

(1) 不存在,因为 <c, e>, <e, d>, <d, c>

(2) ID(d)=2 OD(d)=1 TD(d)=3

(3) `<a>, <b>, <c,e>,<e,d>,<d,c>`

4-6

```
#include <iostream>
#define SIZE 1000
using namespace std;
typedef struct ANode{
    int v;
    int weight;
    struct ANode* next;
}ANode,*ALink;
typedef struct VNode{
    char data;
    ALink Vh;
}VNode;
typedef struct List{
    VNode v[SIZE];
}List;
int main()
{
    List L;
    int n,m;
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        cin>>L.v[i].data;
    for(int i=1;i<=m;i++)
        L.v[i].Vh=NULL;
    for(int i=1;i<=m;i++)
    {
        int v1,v2,weight;
        cin>>v1>>v2>>weight;
        ALink p=new ANode;
        p->v=v2;
        p->weight=weight;
        p->next=L.v[v1].Vh;
        L.v[v1].Vh=p;
    }
    return 0;
}
```

4-7

（1）V1:入度为3，出度为0

V2:入度为2，出度为2

V3:入度为1，出度为1

V4:入度为1，出度为3

V5:入度为2，出度为1

V6:入度为1，出度为3

（2）邻接矩阵为

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

（3）邻接表为

| | | | |
|---|---|---|---|
| $V1$ | | | |
| $V2$ | $V1$ | $V4$ | |
| $V3$ | $V2$ | | |
| $V4$ | $V3$ | $V5$ | $V6$ |
| $V5$ | $V1$ | | |
| $V6$ | $V1$ | $V2$ | $V5$ |

4-8 D

4-9

(1)顶点序列:v1 v2 v3 v6 v4 v5

边序列:e(v1,v2),e(v2,v3),e(v3,v6),e(v6,v4),e(v4,v5)

(2)顶点序列:v1 v2 v3 v4 v5 v6

边序列:e(v1,v2),e(v1,v3),e(v1,v4),e(v1,v5),e(v2,v6)

4-10

生成树是含有无向图G的n个顶点，n-1条边的一个连通图。

最小生成树是无向图G的所有生成树中各边权值之和最小的一颗生成树。

4-11 D

4-12 C

4-13

将m条边按权值从小到大排序，从最小的边开始，若加入这条边，图不产生回路，则加入这条边，否则不加入。直至加入n-1条边，算法结束。

4-14 D

4-15 C

4-16

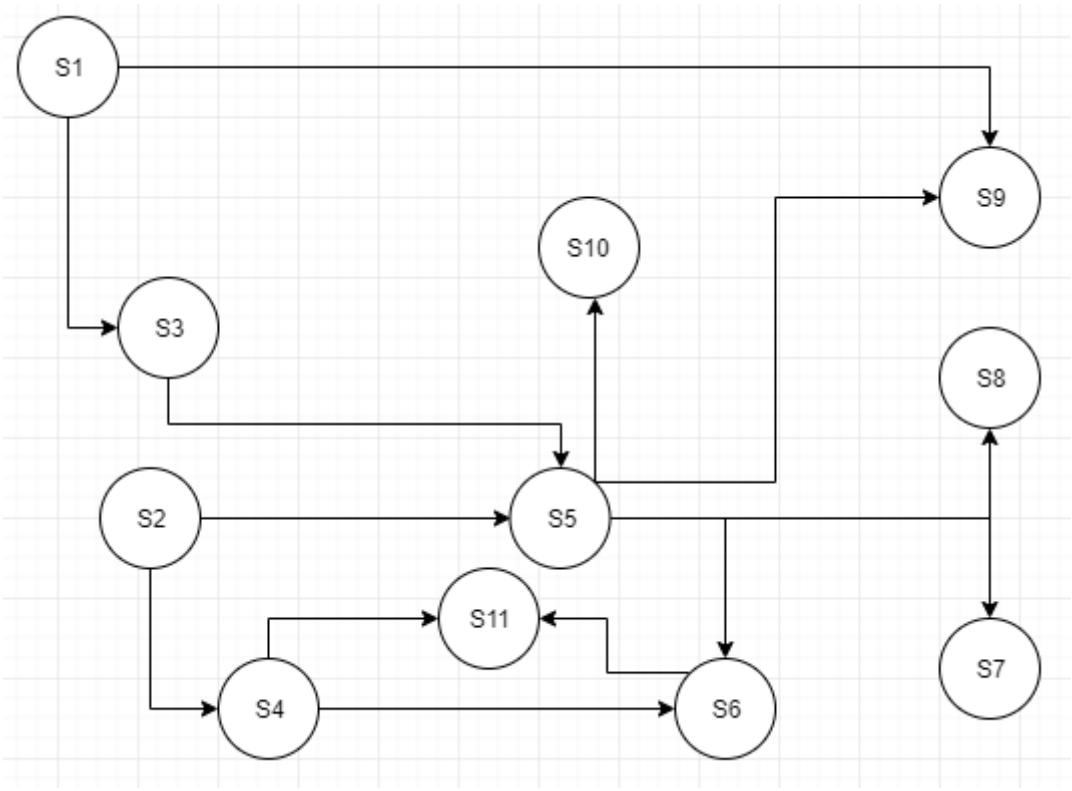
```
void findver(int a,int b,int k,bool &ok)//寻找简单路径
    {
        static int n = 0;//在路径上的顶点
        int j;
        EdgeNode *p;//边结点
        visited[a] = 1;//遍历过
        ++n;//结点加一
        if(a==b && n==k+1)//a与b重合，且路径上顶点足
        {
            ok = true;//找到简单路径
        }
        else
        {
            p = vertexs[a]->first;//取顶点的邻结点
            while(p!=NULL&&!ok)
            {
                if(!visited[p->adjVex])
                {
                    findver(p->adjVex, b, k,ok);//从邻结点遍历
                }
                p = p->next;//下一个邻结点
            }
        }
        visited[a] = 0;
        --n;
    }
```

4-17

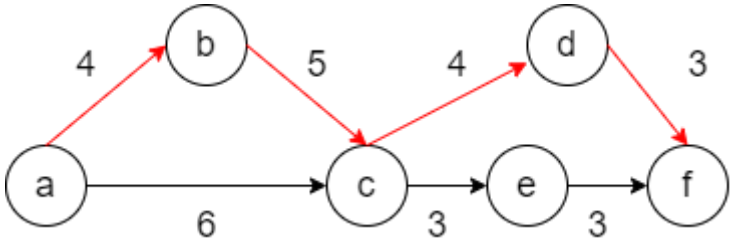
```
void Dijkstra(int v0,int n)
{
   for(int v=0;v<n;++v)
   {
       final[v]=false;
       dis[v]=a[v0][v];//a为v0到v的权值
       //dis[v]表示v0到v的最短路径长度
   }
    dis[v0]=0;
    final[v0]=true;//将v0加入最近点集合
    for(int i=1;i<n;++i)//其余n-1个顶点
    {
        int min=INT_MAX;//当前所知离v0最近的距离
        for(int w=0;w<n;++w)
            if(!final[w])//w在剩余的集合中
                if(dis[w]<min)//w离v0顶点更近
                {
                    int v=w;min=dis[w];//将其设置为当前的最近
                }
        final[v]=true;//将最近的v加入集合
        for(int w=0;w<n;++w)//更新当前最短路径及距离
            if(!final[w]&&(min+a[v][w]<dis[w]))
                dis[w]=min+a[v][w];
    }
```

```
}
```

4-18




4-19




关键路径：a->b->c->d->f 长度为16

4-20 C