第六章作业

**思考题CBCCD BABBB ABAAA BDCBD ABBB**

图

1、在一个图中，所有顶点的度数之和等于所有边数的( )倍。

(A)1/2 (B) 1 (C) 2 (D) 4

2、在一个有向图中，所有顶点的入度之和等于所有顶点的出度之和的( )倍。

(A) 1/2 (B) 1 (C) 2 (D) 4

3、一个有n个顶点的无向图最多有( )条边。

(A) n (B) n(n-1) (C) n(n-1)/2 (D) 2n

4、有8个结点的无向连通图最少有 C 条边。

(A)5 (B) 6 (C) 7 (D) 8

5、对于一个具有n个顶点的无向图，若采用邻接矩阵表示，则该矩阵的大小是( )

(A) n (B) (n-1)^2 (C) n-1 (D) n^2

6、用邻接表表示图进行广度优先遍历时，通常是采用( ) 来实现算法的。

(A)栈        (B) 队列        (C)  排序         (D)  查找

7、用邻接表表示图进行深度优先遍历时，通常是采用 ( )来实现算法的。

(A)栈        (B) 队列        (C)  排序         (D)  查找

8、如果从无向图的任一顶点出发进行一次深度优先搜索即可访问所有顶点，则该图一定是（ ）。

(A) 完全图 (B) 连通图 (C) 有回路 (D) 一棵树

9、带权有向图G用邻接矩阵A存储，则顶点i的入度等于A中（ ）。

(A) 第i行非无穷的元素之和　 (B) 第i列非无穷的元素个数之和

(C) 第i行非无穷且非0的元素个数 (D) 第i行与第i列非无穷且非0的元素之和

10、采用邻接表存储的图，其深度优先遍历类似于二叉树的（ ）。

(A) 中序遍历 (B) 先序遍历 (C) 后序遍历 (D) 按层次遍历

11、无向图的邻接矩阵是一个（ ）。

(A) 对称矩阵 (B) 零矩阵 (C) 上三角矩阵 (D) 对角矩阵

12、邻接表是图的一种（ ）。

(A) 顺序存储结构 (B) 链式存储结构 (C) 索引存储结构 (D) 散列存储结构

13、在无向图中定义顶点vi与vj之间的路径为从vi到vj的一个（ ）。

(A) 顶点序列 (B) 边序列 (C) 权值总和 (D) 边的条数

14、在有向图的逆邻接表中，每个顶点邻接表链接着该顶点所有（ ）邻接点。

(A) 入边 (B) 出边 (C) 入边和出边 (D) 不是出边也不是入边

15、设G1=(V1,E1)和G2=(V2,E2)为两个图，如果V1⊆V2,E1⊆E2则称（ ）。

(A) G1是G2的子图 (B) G2是G1的子图

(C) G1是G2的连通分量 (D) G2是G1的连通分量

16、已知一个有向图的邻接矩阵表示，要删除所有从第i个结点发出的边，应（ ）。

(A) 将邻接矩阵的第i行删除 (B) 将邻接矩阵的第i行元素全部置为0

(C) 将邻接矩阵的第i列删除 (D) 将邻接矩阵的第i列元素全部置为0

17、任一个有向图的拓扑序列（ ）。

(A)不存在 (B) 有一个 (C) 一定有多个 (D) 有一个或多个

18、下列关于图遍历的说法不正确的是（ ）。

(A) 连通图的深度优先搜索是一个递归过程

(B) 图的广度优先搜索中邻接点的寻找具有“先进先出”的特征

(C) 非连通图不能用深度优先搜索法

(D) 图的遍历要求每一顶点仅被访问一次

19、带权有向图G用邻接矩阵A存储，则顶点i的入度为A中：（ ）。

(A) 第i行非∞的元素之和 (B) 第i列非∞的元素之和

(C) 第i行非∞且非0的元素个数 (D) 第i列非∞且非0的元素个数

20、采用邻接表存储的图的广度优先遍历算法类似于二叉树的（ ）。

(A) 先序遍历 (B) 中序遍历 (C) 后序遍历 (D) 按层次遍历

21、关键路径是事件结点网络中（ ）。

(A) 从源点到汇点的最长路径　 (B) 从源点到汇点的最短路径

(C) 最长的回路 (D) 最短的回路

22、下面（ ）可以判断出一个有向图中是否有环（回路）。

(A) 广度优先遍历 (B) 拓扑排序

(C) 求最短路径 (D) 求关键路径

23、下面有向图所示的拓扑排序的结果序列是（ ）。

(A) 125634　 (B) 516234 (C) 123456 (D) 521643



24、任何一个无向连通图的最小生成树（ ）种。

(A) 只有一棵　 (B) 有一棵或多棵

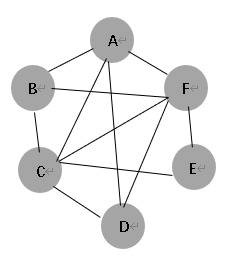
(C) 一定有多棵 (D) 可能不存在

**应用题**

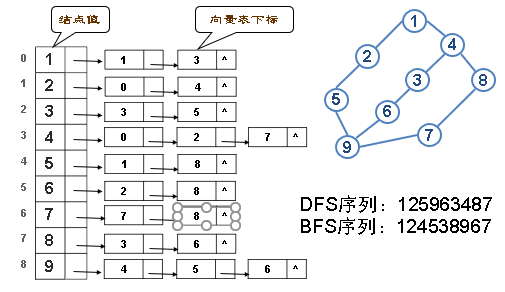
1. 某田径比赛中各选手的参赛项目表在表6.8中。项目A至F各表示一数据元素，若两个项目不能同时举行，则将其连线（约束条件）。（1）根据此表及约束条件画出相应的图状结构模型，并画出此图的邻接表结构；（2）写出从元素A出发按“广度优先搜索”算法遍历此图的元素序列。

|  |  |
| --- | --- |
| 表6.8 | |
| 姓名 | 参赛项 |
| ZHAO | A B C |
| QIAN | C D |
| SUN | C E F |
| LI | D F A |
| ZHOU | B F |

1. （2）**ABCDFE**



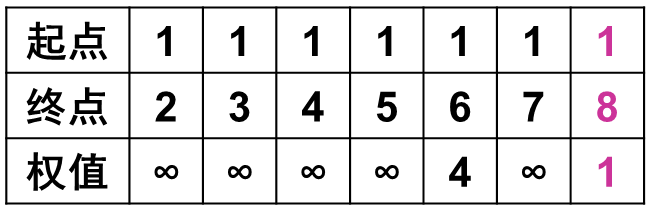
3.给定一无向图，画出它的邻接表，写出用深度优先搜索和广度优先搜索遍历该图时，从顶点1出发所经过的顶点和边序列。



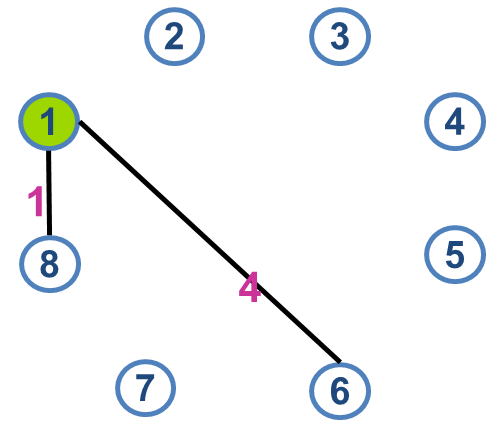
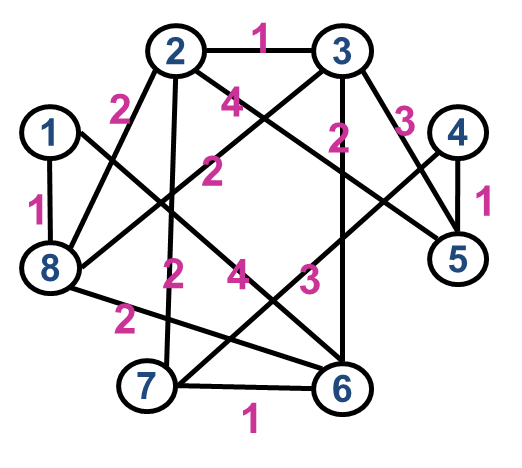
4.对连通网络，分别用Prim算法Kruskal算法构造该网络的最小生成树。

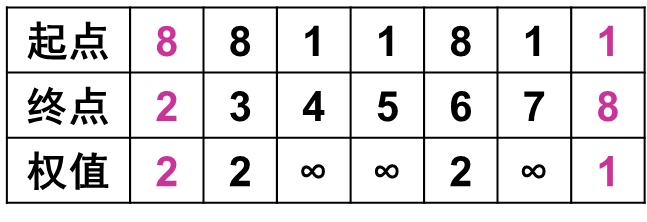
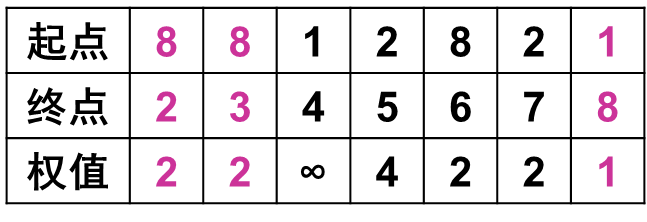
**Prime算法**

**Prime算法**

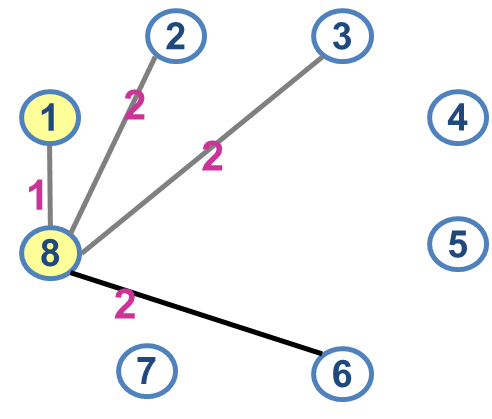
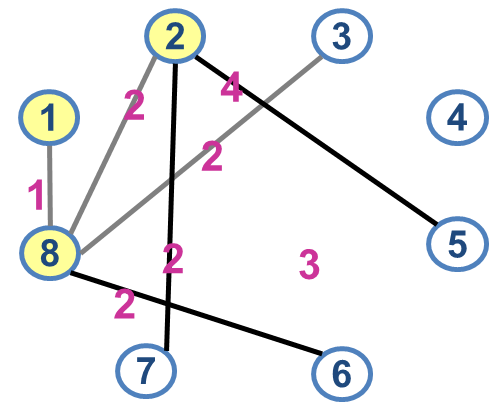


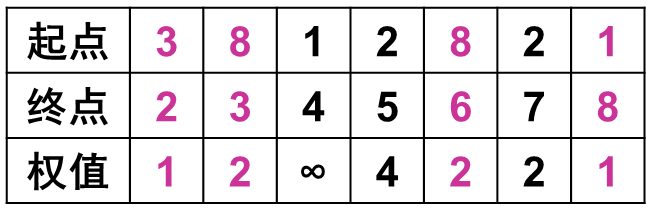
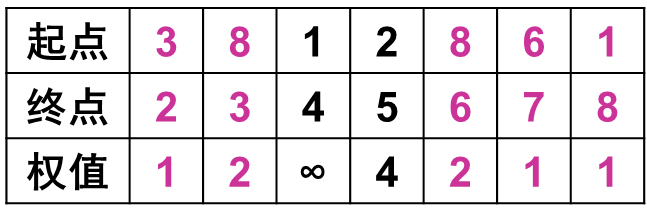
1. 找到最短边(1,8)

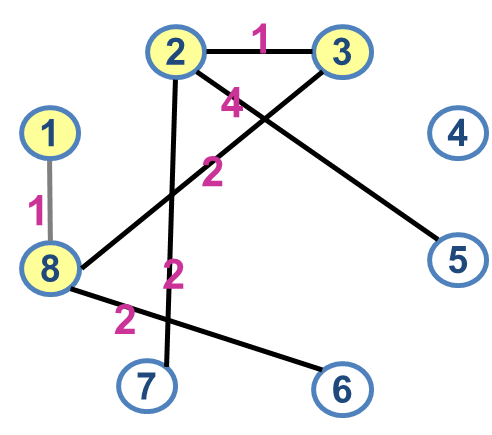
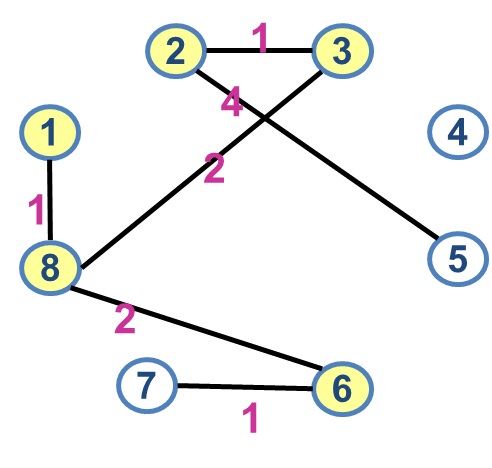
 

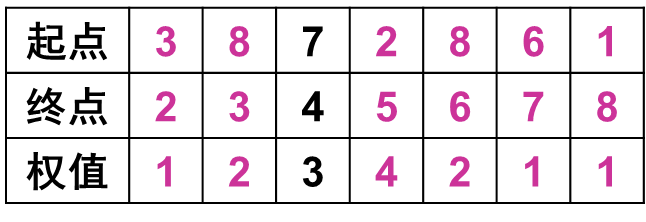
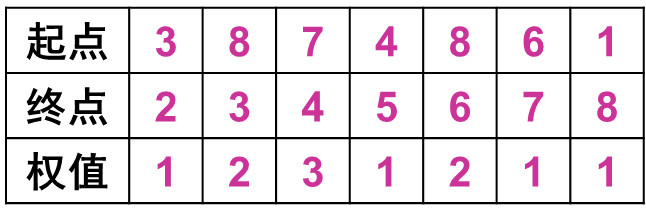
1. 以8作中间点处理后，找到最短边(8,2) (c)以2作中间点处理后，找到最短边(8,3)

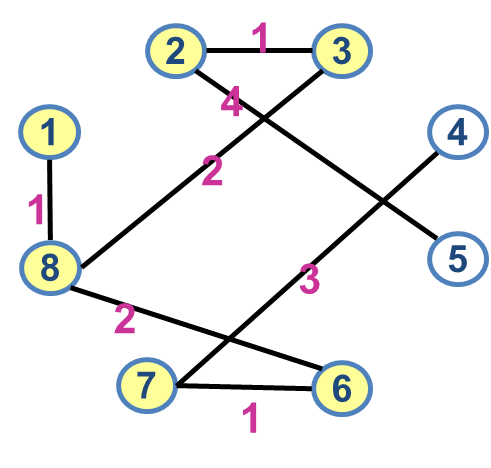
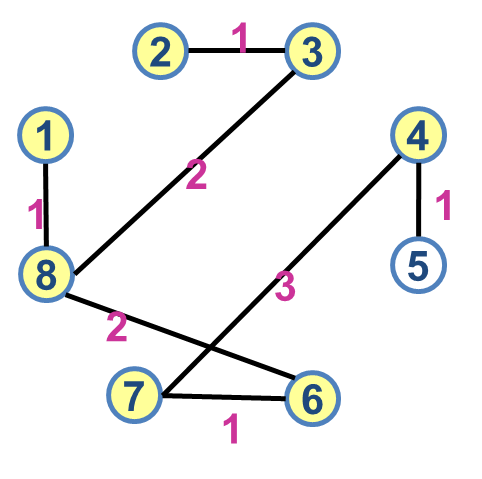
 

（d）以3作中间点处理后，找到最短边(8,6) (e)以6作中间点处理后，找到最短边(6,7)

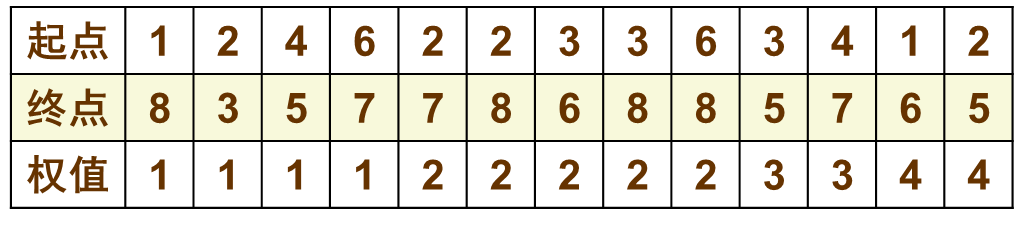
 

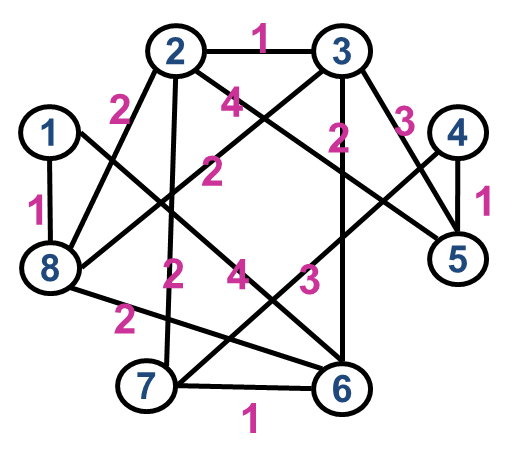
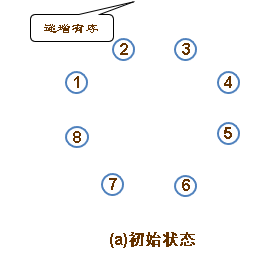
 

(f)以7作中间点处理后，找到最短边(7,4) (g)终态  **权值和=1\*4+2\*2+3=11**

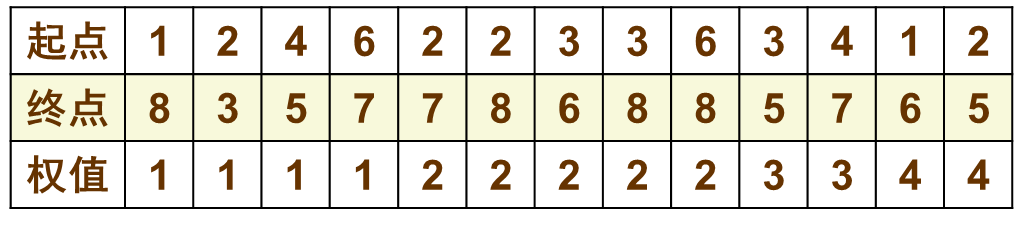
 

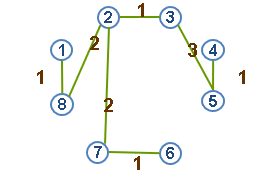




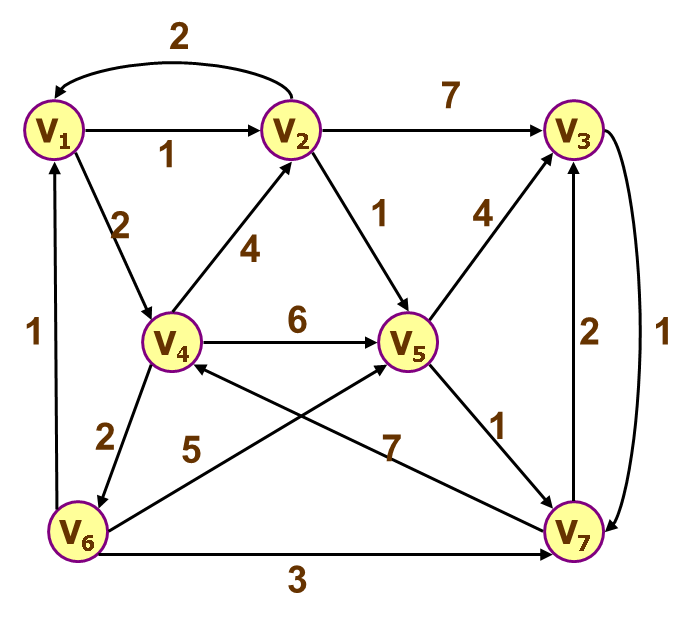


舍去

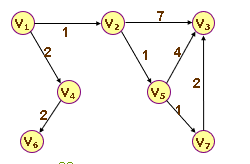
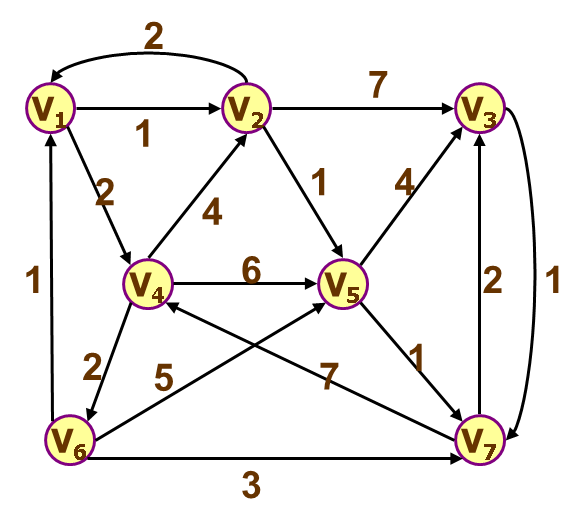


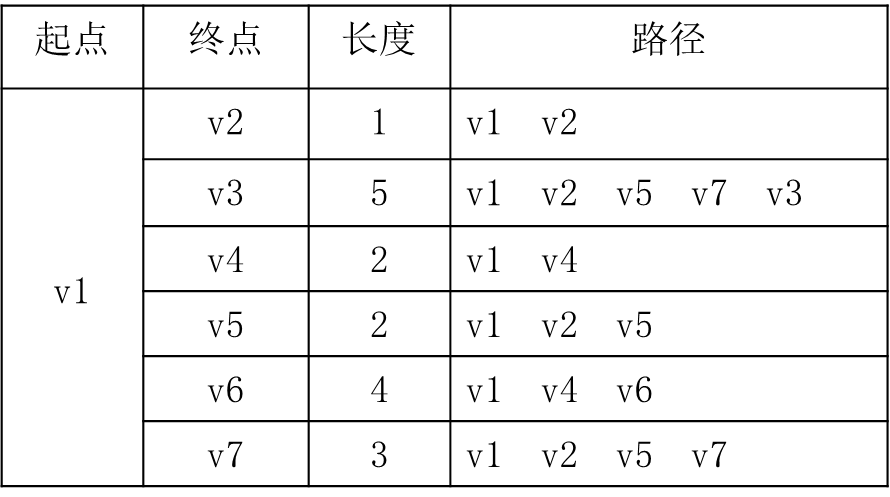
**权值和=1\*4+2\*2+3=11**

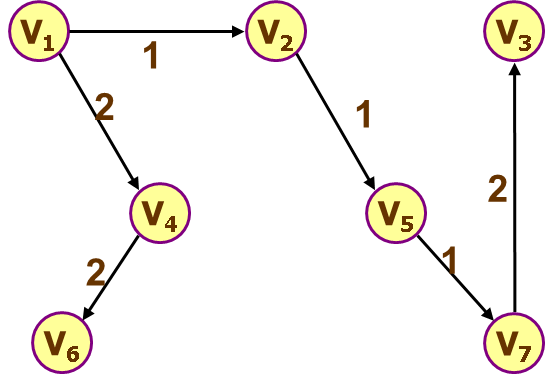
6、用dijkstra算法求v1到其余各顶点的最短距离







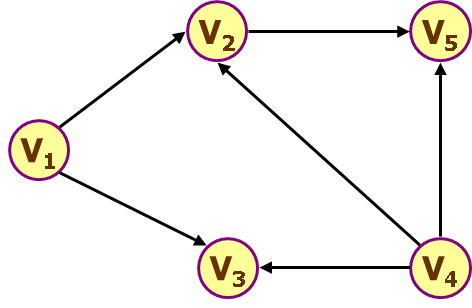




8、给出AOV网的拓扑序列

**拓扑排序条件**

**无环有向图**



(1) V1, V4, V3, V2, V5 (4) V4, V1, V3, V2, V5

(2) V1, V4, V2, V3, V5 (5) V4, V1, V2, V3, V5

(3) V1, V4, V2, V5, V3 (6) V4, V1, V2, V5, V3

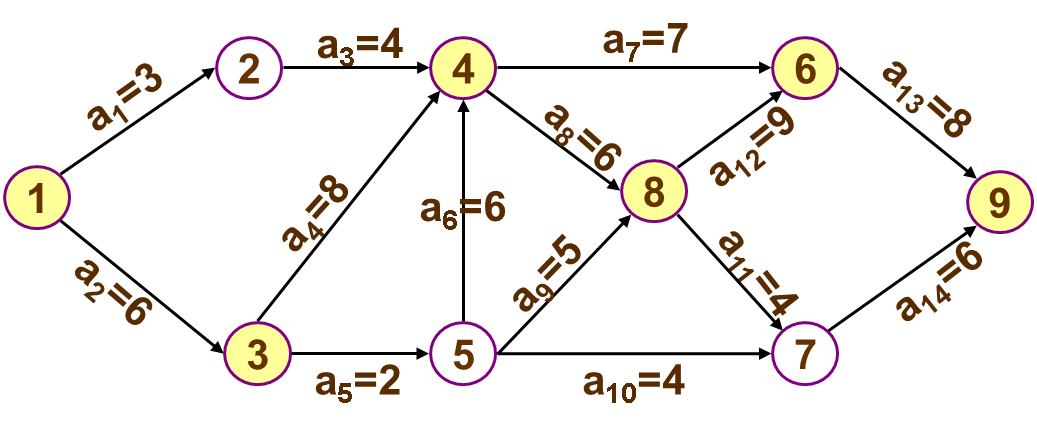
10．对图6.91所示的AOE网求出：

（1）各活动的最早开始时间与最迟开始时间；

（2）所有的关键路径；

（3）该工程完成的最短时间是多少?

（4）是否可通过提高某些活动的速度来加快工程的进度?

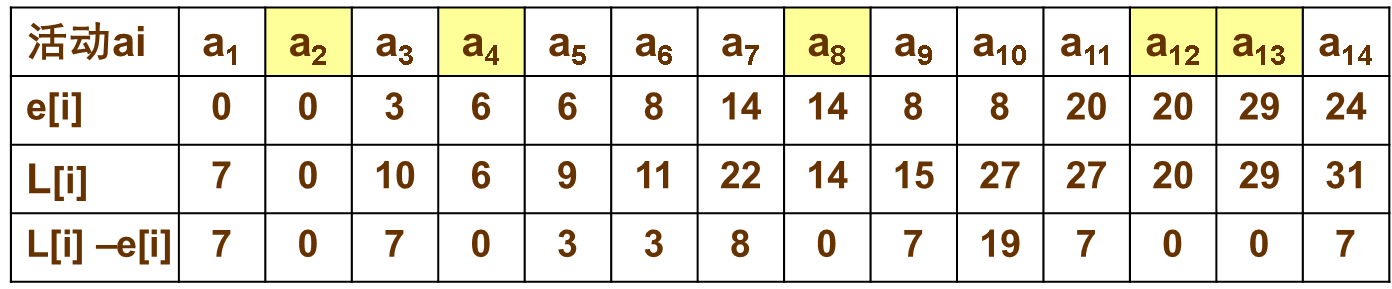


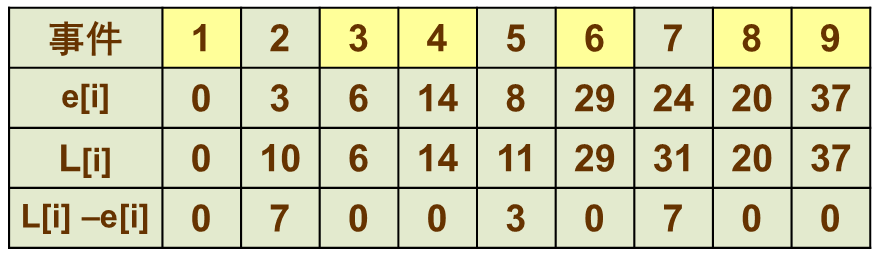
**源点→汇点**

**入度 >1的点 值取最大**

**汇点→源点**

**出度 >1的点 值取最 小**

****

****

**关键路径：1->3->4->8->6->9**

**关键活动：a2,a4,a8,a12,a13**

**最短时间37**

12.表6.9给出某工程之间的优先级和各工程所需时间，要求给出AOE图



**前序工程**

E=15

C=50

G=300

A=15

D=8

I=120

H=15

F=40

F=40

B=10

J=60

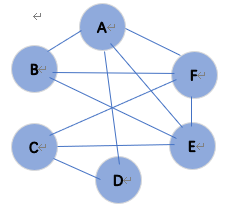
K=15

M=20

L=15

**算法设计题**

5、设田径比赛项目有：A（跳高）、B（跳远）、C（标枪）、D（铅球）、E（100米跑）、F（200米跑）。表6.11为参赛选手的项目表。问如何安排比赛时间，才使得：每个比赛项目无冲突进行；尽量缩短比赛时间。



**数据结构设计：**

将同时举行的比赛放在一个数组node[N]里，数组长度最大为N，数组已有定点的最后未知由尾指针rear记录。次序集OrderSet中次序数最多也为N，即每个顶点的次序不一样。每个次序是否被用由used标记，设0为未用，1为已用。

**函数结构**

①函数功能:找度最大的结点下标

函数输入：顶点度数组

函数输出：度最大的结点下标

int FindMax(int \*a)

{

int i,value,index;

value=-1;

index=0;

for(i=0;i<N;i++)

{

if(value<a[i])

{

value=a[i];

index=i;

}

}

a[index]=-1;

return index;

}

②函数功能：判断K点是否能加入次序集第i种次序顶点集

函数输入：第i种次序，K结点

函数输出：1---可以加入

0---不能加入

int judge(int i,int k)

{

int p,q,m;

p=0;

q=OrderSet[i].rear;

while(AdjMatrix[k][OrderSet[i].node[p]]==0&&p!=OrderSet[i].rear)

p++;

if(p==q)return 1;

return 0;

}

③函数功能：welsh\_Powell图结点染色法

函数输入：无

函数输出：无

屏幕输出：同次序结点集合

int judge(int i,int k)

{

int p,q,m;

p=0;

q=OrderSet[i].rear;

while(AdjMatrix[k][OrderSet[i].node[p]]==0&&p!=OrderSet[i].rear)

p++;

if(p==q)return 1;

return 0;

}

算法伪代码描述一

（1）计算图中各顶点的度

（2）找当前最大节点K

（3）次序集的第colorPtr项次序已经使用过？

{

若K不能用colorPtr项次序，则colorPtr换一个次序

}

（4）将K加入colorPtr项次序结点集

（5）重复步骤（2）至（4），直到所有结点处理完未知

（6）输出同次序结点集合

伪代码描述二

设图的顶点数为N

（1）计算图中各顶点的度，放至数组degree[N]中

（2）在degree[]中找当前度最大结点信息记录在K中，消除degree[]此项结点信息

（3）在次序集OrderSet[]的第orderPtr项次序已经使用过？

{

！（K与OrderSet[orderPtr].node[]中的结点都不相邻）

则orderPtr++

}

（4）将K加入orderPtr项次序结点集OrderSet[colorPtr].node[];

（5）重复步骤（2）至（4），直到所欲结点处理完为止

（6）输出同次序结点集合

**代码实现：**

#include<stdio.h>

#define TRUE 1

#define FALSE 0

#define N 6

int AdjMatrix[N][N]=

{{0,1,0,1,1,1},

{1,0,0,0,1,1},

{0,0,0,1,1,1},

{1,0,1,0,0,1},

{1,1,1,0,0,1},

{1,1,1,1,1,0}};

int degree[N]={0}; //记录顶点的degree数目

char \*order[N]={"第一","第二","第三","第四","第五","第六"};

char competition[N]={'A','B','C','D','E','F'};

struct OrderNode

{

int used; //标记次序是否被用，0代表未用

int rear; //顶点集合尾指针

int node[N]; //同次序顶点集合

}OrderSet[N]={{0,0,0,0}}; //顶点集

int FindMax(int \*a)

{

int i,value,index;

value=-1;

index=0;

for(i=0;i<N;i++)

{

if(value<a[i])

{

value=a[i];

index=i;

}

}

a[index]=-1;

return index;

} //找度最大的结点下标

int judge(int i,int k)

{

int p,q,m;

p=0;

q=OrderSet[i].rear;

while(AdjMatrix[k][OrderSet[i].node[p]]==0&&p!=OrderSet[i].rear)

p++;

if(p==q)return 1;

return 0;

} //判断K点是否能加入次序集中第i中次序顶点集

void Welsh\_Powell()

{

int i,k;

int orderPtr;

for(i=0;i<N;++i)

{

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

{

if(i!=j&&AdjMatrix[i][j])

degree[i]++;

}

}

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

{

k=FindMax(degree);

orderPtr=0;

if(OrderSet[orderPtr].used==1)

{

while(!judge(orderPtr,k))

orderPtr++;

}

OrderSet[orderPtr].node[OrderSet[orderPtr].rear++]=k;

if(OrderSet[orderPtr].used==0) OrderSet[orderPtr].used=1;

}

for(j=0;j<N;++j)

{

if(OrderSet[j].used==1)

{

printf("%s:",order[j]);

for(i=0;i<OrderSet[j].rear;++i)

printf("%c",competition[OrderSet[j].node[i]]);

printf("\n");

}

}

} //Welsh\_Powell图结点染色法

int main()

{

Welsh\_Powell();

return 0;

}