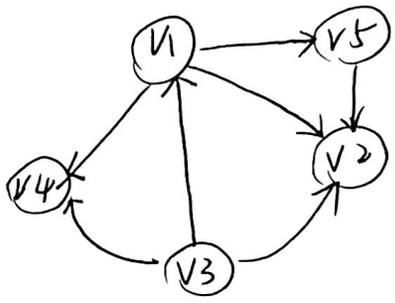
(1) 邻接矩阵表示如下



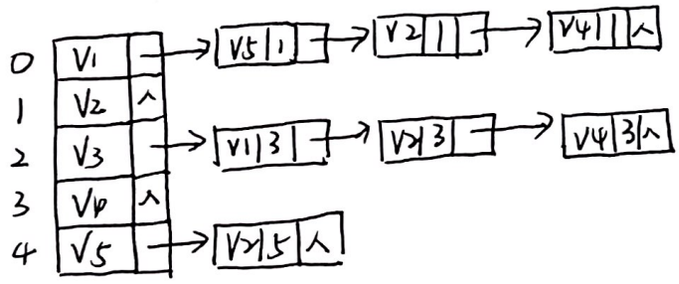
$$A=\begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \end{bmatrix}$$

1 2 3 4 5 (列)；1 2 3 4 5 (行)

该邻接矩阵代码定义如下

```
typedef struct{
    VertexType vertex[5];   //顶点序号
    int vexNum;             //顶点数
    int edgeNum;            //边数
    int matrix[5][5];       //邻接矩阵
}MGraph;
```

邻接表表示如下所示



邻接表代码如下

```
typedef struct edgeNode{    //边表(弧式结构)
    VertexInfoType vertexInfo;   //边所指向顶点位置信息.
    struct edgeNode *nextEdge;   //指向下一条边指针信息
    edgeWeightType weight;       //网的边权值
}edgeNode;
```

```
typedef struct vertex{    // 顶点表信息
    vertexType value;     // 顶点信息
    edgeNode *firstEdge;
}vertex, adjList[5];  指向第一条依附于该顶点的指针

typedef struct{           // ALGraph 用邻接表存图信息
    adjList vertexList;   // 邻接表
    int verNum;           // 图顶点数
    int edgeNum;          // 图的边数
}ALGraph;
```

(2) 有向图不能用多邻接多重表表示，邻接多重表常用于无向图

有向图可采用十字链表表示

(3) 从顶点 V_1 开始的深度优先遍历序列如下

① $V_1 \to V_2 \to V_4 \to V_5 \to V_3$

② $V_1 \to V_2 \to V_5 \to V_4 \to V_3$

③ $V_1 \to V_4 \to V_5 \to V_2 \to V_3$

④ $V_1 \to V_5 \to V_2 \to V_4 \to V_3$

注 $V_1 \to V_5 \to V_4 \to V_2 \to V_3$ 并不是该图深度优先遍历序列 因存在弧 <V_5, V_2>

且 V_2 未访问 则访 V_5 后必访 V_2

深度优先 DFS 递归的代码如下

408 则可理解就行，不需求全部掌握，自命题必须背诵

DFS:

```
bool visited[MAX_VERTEX_NUM];          //访问标记数组
void DFSTraverse(Graph G){             //对图G深度优先遍历
    for(v=0; v<G.vexnum; ++v)
        visited[v]=FALSE;
    for(v=0; v<G.vexnum; ++v){         //连通分量数为调用次数
        if(!visited[v])                //从V=0开始遍历
            DFS(G,v);
    }
}

void DFS(Graph G, int v){              //从顶点出发深度优先遍历图G
    visit(v);                          //访问顶点
    visited[v]=TRUE;                   //标记
    for(w=FirstNeighbor(G,v); w>=0; w=NextNeighbor(G,v,w)){
        if(!visited[w]){               //w为v的尚未访问邻接顶点
            DFS(G,w);                  //用栈实现递归
        }
    }
}
```

空 O(|V|)　总时为 O(|V|²) 邻接矩阵；O(|V|+|E|) 邻接表

(4) 从V1开始的所有深度优先遍历如下

① $V_1 \to V_2 \to V_4 \to V_5 \to V_3$

② $V_1 \to V_2 \to V_5 \to V_4 \to V_3$

③ $V_1 \to V_4 \to V_2 \to V_5 \to V_3$

④ $V_1 \to V_4 \to V_5 \to V_2 \to V_3$

⑤ $V_1 \to V_5 \to V_2 \to V_4 \to V_3$

⑥ $V_1 \to V_5 \to V_4 \to V_2 \to V_3$

BFS 伪代码如下：　　类似树层次遍历

```
bool visited[MAX_VERTEX_NUM];   //访问标记数组

void BFSTraverse (Graph G){     //对图G广度优先遍历

    for (int i=0; i<G.vexnum; ++i)
        visited[i]=FALSE;

    InitQueue (Q);              //初始化辅助队列Q

    for ( i=0; i<G.vexnum; ++i){    //开始探测有几个连通分量

        if (!visited [i])
            BFS (G,i);          //vi未访问过从vi开始BFS

    }

}

void BFS (Graph G, int v){      //从v出发广度优先遍历图G

    visit(v);
    visited[v]=TRUE;
    EnQueue (Q,v);              //v入队
    while (!isEmpty (Q)){
        DeQueue (Q,v);          //v出队
        for (w=FirstNeighbor(G,v); w>=0; w=NextNeighbor (G,v,w)){
            if (!visited[w]){   //w为v尚未访问邻接顶点，
                visit(w);
                visited[w]=TRUE;    //w标记
                EnQueue (Q,w);      //w入队
            } //if
        } //for
    } //while
} //BFS
```

空为 $O(|V|)$

时为
- 邻接表 $O(|V|+|E|)$
- 邻接矩阵 $O(|V|^2)$

(5) 每个顶点各自为一个强连通分量 共5个

(6) 首先对有向图拓扑排序 所有顶点排在一个拓扑排序序列中，按该序列顺序所有顶点重新编号使每条边的起点编号小于终点编号就可以把所有边集中到邻接矩阵按组上三角部分

(7) 有向环无环必有拓扑序列

① $V_3 \to V_1 \to V_4 \to V_5 \to V_2$

② $V_3 \to V_1 \to V_5 \to V_4 \to V_2$

③ $V_3 \to V_1 \to V_5 \to V_2 \to V_4$

(8) 该图无环且有向图 存在关键路径

V_3 入度为0 为源点 V_2 V_4 出度为0 V_2 V_4 为汇点

故关键路径为 V_3 V_1 V_5 V_2 关键路径长为 9

AOE只有一条关键路径 V_3为源点，V_2或V_4为汇点

$V_3 \to V_1 \to V_4$ $V_3 \to V_1 \to V_2$ 长度均为4

因此缩短 关键路径 (V_3, V_1, V_5, V_2) 上 V 活动使该关键路径的路径长度 < 后路长度且 >4 均可使整体项目提前完成

(9) 事实不对 权值为正时也可以用 Dijkstra 来求最短路径

有负权值则不可以 (如弧 <V_1, V_4> 权值为4) 则：

用 Dijkstra 可得 从 $V_3 \to V_5$ 最短路径为 $V_3 \to V_1 \to V_5$ 长度为 $6=1+5$

(10) 该图无负圈权值，可以使用Floyd求最短路径

若负权圈则不可以

①求得到所有的顶点路径

1> V1→V2 最短为 V1，V2 长度为2

2> V1→V4 最短为 V1，V4 4

3> V1→V5 最短为 V1，V5 5

4> V3→V1 最短为 V3，V1 1

5> V3→V2 最短为 V3，V2 2

6> V3→V4 最短为 V3，V4 4

7> V3→V5 最短为V3，V1，V5 6

8> V5→V2 最短为 V5，V2 2

(11) 该图为无向有权图时

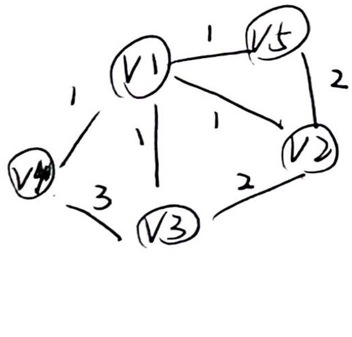
① 不存在拓扑序列 且此时有环

② 不存在边权为负的边 可用 Dijkstra 求解

③ 不存在 负权圈 可用 Floyd 求解

④ 不存在 关键路径

(12) 无向有权图时



以V1为起点，DFS如下

① V1→V2→V3→V4→V5

② V1→V2→V5→V3→V4

③ V1→V3→V2→V5→V4

④ V1→V3→V4→V2→V5

⑤ V1→V4→V3→V2→V5

⑥ V1→V5→V2→V3→V4

若为无向有权图则 V1为起点，广度优先遍历 V1在第1层 其余均在第二层

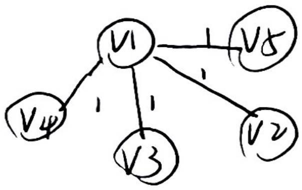
广度优先遍历序列个数为

$A_4^4 = 24$个

(13) 使用Prim或kruskal算法构造出生成树相同且唯一

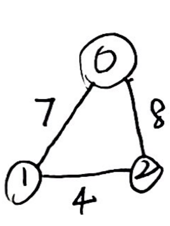
点集{V_1, V_2, V_3, V_4, V_5} 边集为 E={(V_2, V_1), (V_1, V_3), (V_1, V_4), (V_1, V_5)}

即

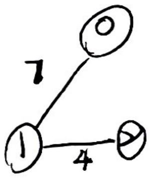


(14) 不对 如图所示带权连通图，从顶点0出发的最小生成树如图(b)所示

而从顶点0到顶点2最短路径为 0->2 不是 0->1->2



(a)



(b)