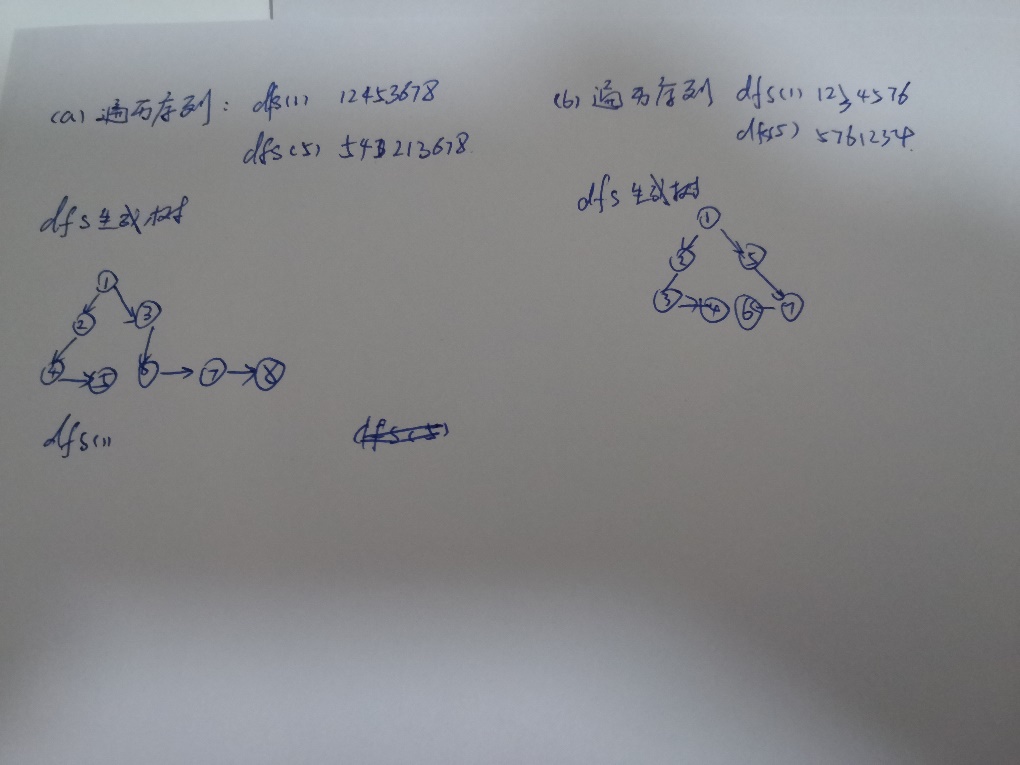
**题6.10**



**题6.12**

bool TF(Graph G, int i, int v)

{

DFS(G, i);

if (visited[v] == false)

return FALSE;

else

return TRUE;

}

**题6.16**

bool Panduan(Graph G, int v)

{

int i; //顶点编号

for (i = 1; i <= G.VerNum; i++) //初始化访问标记数组

visited[i] = false;

DFS(G, v); //遍历指定顶点v所在的连通分量

for (i = 1; i <= G.VerNum; i++)

{ //循环遍历图中所有其它的连通分量

if (!visited[i])

{

return FALSE; //如果未遍历图中所有顶点则不是

}

}

return TRUE;

}

**题6.18**

void DFSTree(Graph& G, int v, csTree& T)

{

int j, w, first = 1; //first标记是否为根节点，初始化为1

csNode\* p, \* q;

q = T;

visited[v] = true;

EdgeNode \*z; //p初始化为顶点v的边链表的头指针

while (z) //还存在临结点

{

if (!visited[ z->adjVer ])

{

p = new csNode;

p->firstChild = NULL;

p->nextSibling = NULL;

p->data = G.Data[w];

if (first) //（生成第一个孩子）

{

T->firstChild = p;

first = 0;

}

else { //（生成节点的兄弟节点）

q->nextSibling = p;

}

q = p;

DFSTree(G, w, q); //递归处理每个节点

}

z=z->next; //z指向v的下一个邻接点

}

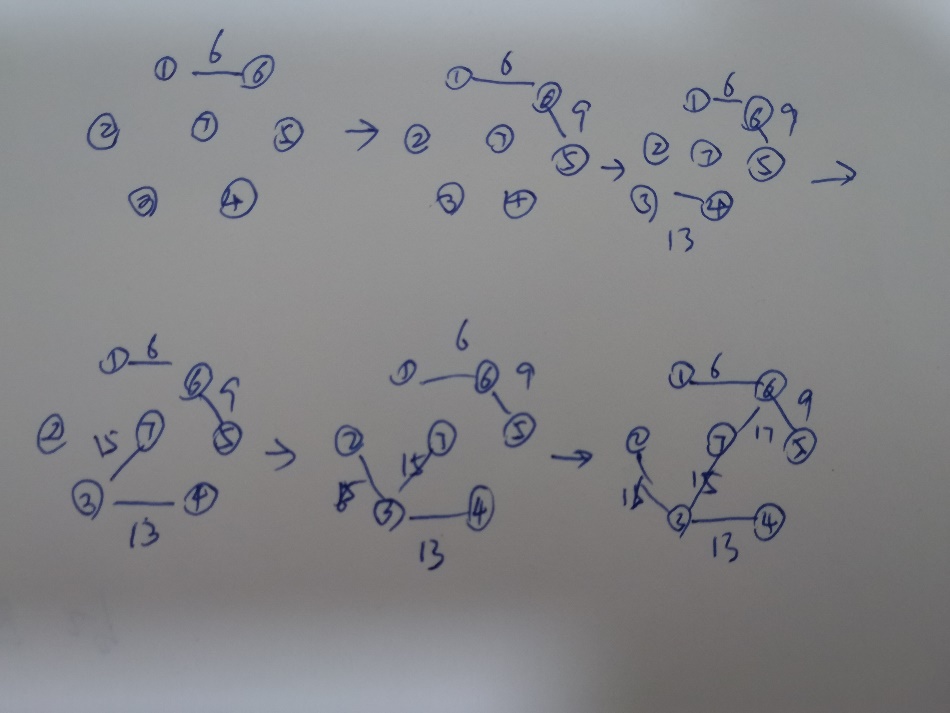
}

**题6.23**

**Prim算法：**

1. S={ }，U={1,2,3,4,5,6,7}，WE={ }，TE={ }
2. 第一轮：顶点1从U移到S，S={1}，U={2,3,4,5,6,7}，WE={(1,2),(1,7)}，TE={(1,6)}
3. 第二轮：顶点6移到S，S={1,6}，U={2,3,4,5,7}，WE={(1,2),(1,7),(6,7) }，TE={(1,6),(6,5)}
4. 第三轮：顶点5移到S，S={1,6,5}，U={2,3,4,7}，WE={(1,2),(1,7),(6,7),(5,4)}，TE={(1,6),(6,5), (5,7)}
5. 第四轮：顶点7移入S，S={1,6,5,7}，U={2,3,4}，WE={(1,2),(5,4),(7,2),(7,4)}，TE={(1,6),(6,5), (5,7),(7,3)}
6. 第五轮：顶点3移入S，S={1,6,5,7,3}，U={2,4}，WE={(1,2),(5,4),(7,2),(7,4),(3,2)}，TE={(1,6),(6,5), (5,7),(7,3),(3,4)}
7. 第六轮：顶点4移到S中，S={1,6,5,7,3,4}，U={2}，WE={(1,2),(7,2)}，TE={(1,6),(6,5), (5,7),(7,3),(3,4),(3,2)}
8. 第七轮：顶点2移到S中，S={1,6,5,7,3,4,2}，U={}，WE={}，TE={(1,6),(6,5), (5,7),(7,3),(3,4),(3,2)}

**Kruskal算法:**



**题6.25**

并查集，选取一条边，那么这两个顶点就是“父子”关系。这样的关系绝对不成环，所以每次选取一条边，它两个顶点所在的两个不同的并查集就形成父子关系。如果它们在同一并查集中，那这条边就不予考虑。

**题6.27**

采用邻接矩阵存储时的拓扑排序算法的时间复杂度为O（n2）

采用邻接表存储时的拓扑排序算法时，若将n个顶点e条弧的有向图采用邻接表存储，则拓扑排序算法的时间复杂度是O（n+e）

**题6.28**

1. 1到1的最短路径为0 路径为1,1
2. 1到2的最短路径为10 路径为1,2
3. 1到3的最短路径为33 路径为1,6,7,3
4. 1到4的最短路径为37 路径为1,6,7,4
5. 1到5的最短路径为15 路径为1,5
6. 1到6的最短路径为20 路径为1,6 (1,5,6)
7. 1到7的最短路径为27 路径为1,6,7 (1,5,6,7)
8. 1到8的最短路径为31 路径为1,6,7,8 (1,5,6,7,8)

**题6.29**

1. 在图采用邻接矩阵存储，Dijkstra 算法的时间复杂为O（n2）
2. 在图采用邻接表存储时，Dijkstra 算法的时间复杂为O（n2）