6.8 设图G用邻接矩阵A[n+1,n+1]表示，设计算法以判断G是否是无向图。

思路：无向图邻接矩阵是对角线对称的，即G.arc[i][j] == G.arc[j][i]，所以循环判断若不满足条件则直接return false即可，若全满足则为无向图return true

bool isWuxiang(MGraph G) {

for(int i = 0;i < G.numNodes;i++) {

for(int j = 0;j < G.numNodes;j++) {

if(G.arc[i][j] != G.arc[j][i]) {

return false;

}

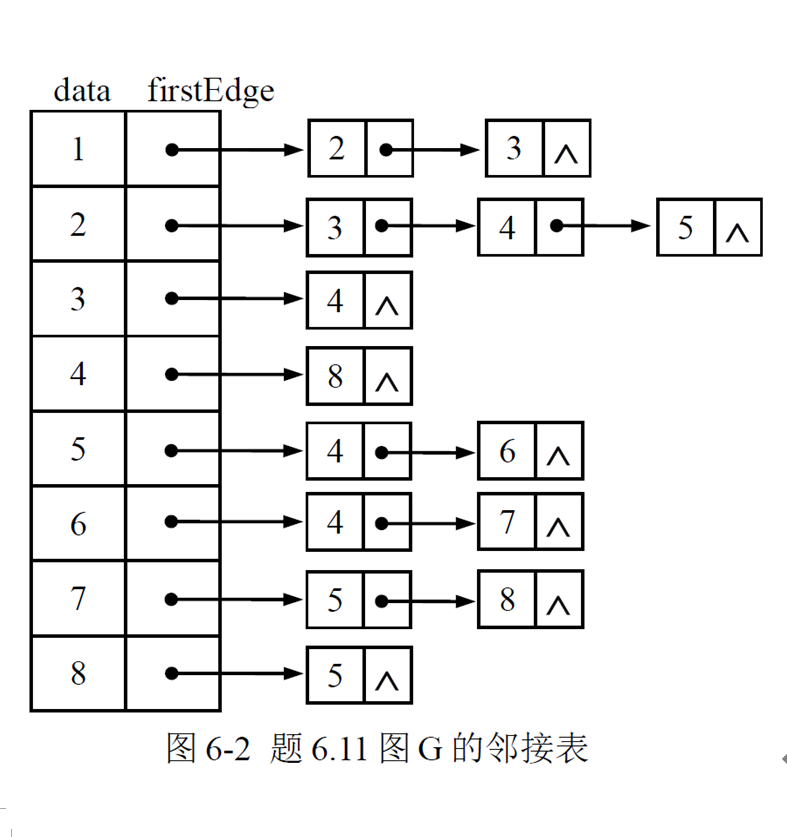
}

}

return true;

}

6.11 对图6-2的邻接表，不用还原出原图，请执行dfs(1)，写出遍历序列，并构造出相应的dfs生成树。



DFS代码如下，用人脑模拟执行DFS(G,1)：

void DFS(GraphAd G,int i) {

visited[i] = TRUE;

printf("%d ",G.adjList[i].data);

EdgeNode \*E = G.adjList[i].firstEdge;

while(E) {

if(visited[E->adjvex] == FALSE) {

DFS(G,E->adjvex);

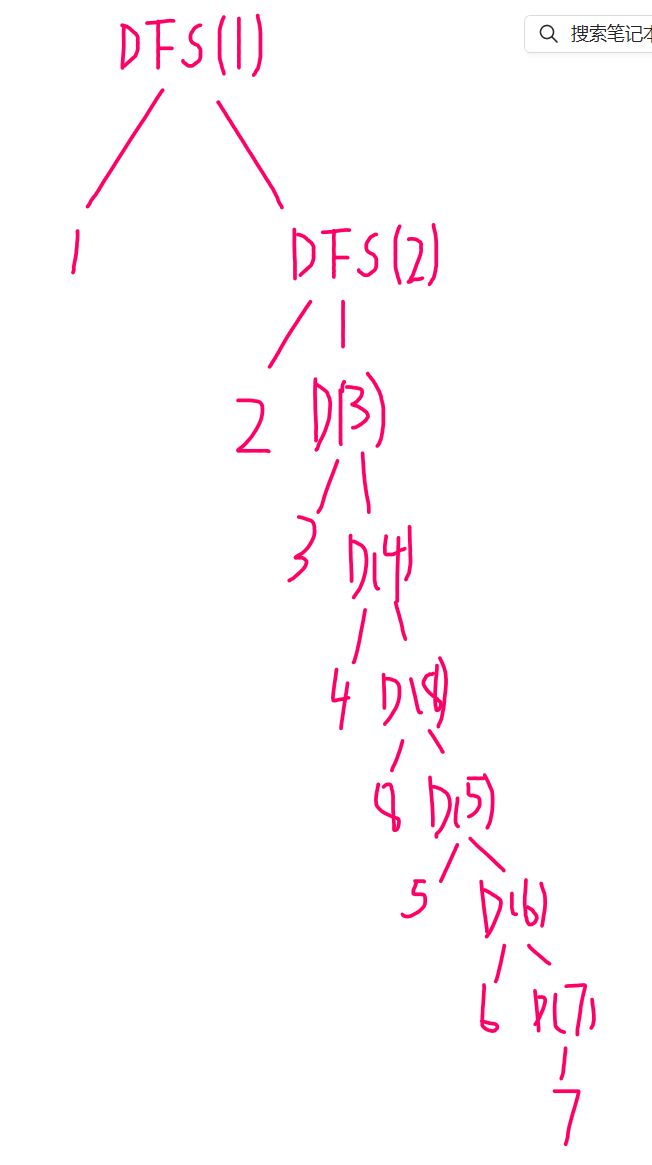
}

E = E->next;

}

}

结合代码和递归知识，推断如下：



即运行结果为：1,2,3,4,8,5,6,7，即DFS(1):1->2->3->4->8->5->6->7

6.15 设G是无向图，设计算法以判断G是否是一棵树，若是树，则返回TRUE，否则返回FALSE；

思路：无向图为树即满足没有环路的连通图；先判断是否存在环：DFSisLoop。是则直接return false，无环则继续判断连通性，不连通则return false，连通则return true

检查环 思路：见DFSisLoop函数，借助DFS和递归，递归调用时传入父结点，若此结点(j)已访问过(visited==true)且不是父结点(j != parent(上一次的i))说明存在环，直接return true

检查连通 思路：检查环完成后，应当无环图才可以执行下面的代码，通过判断是否所有节点都被遍历可判断是否连通

//环

bool DFSisLoop(MGraph G, int i, int parent) {

visited[i] = true;

for (int j = 0; j < G.numNodes; j++) {

if (G.arc[i][j] == 1) {

if (visited[j] == FALSE) {

//子节点中发现环

if (DFSisLoop(G, j, i) == false) {

return true;

}

//已访问过(else)且不是父节点(j != parent(上一次的i))

} else if (j != parent) {

return true;

}

}

}

return false;

}

Status isTreeDFS(MGraph G) {

// 初始化

for (int i = 0; i < G.numNodes; i++) {

visited[i] = false;

}

//检查有没有环

if (DFSisLoop(G, 0, -1) == true) {

printf("IS LOOP\n");

return false;

}

// 检查连通

for(int i = 0;i < G.numNodes;i++) {

if(visited[i] == false) {

return false;

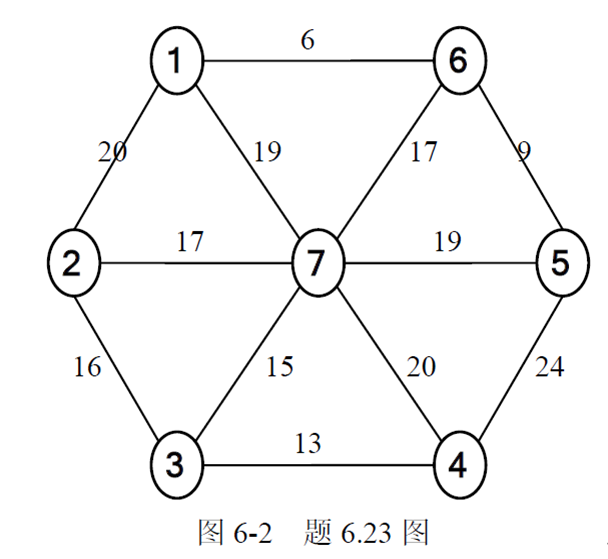
}

}

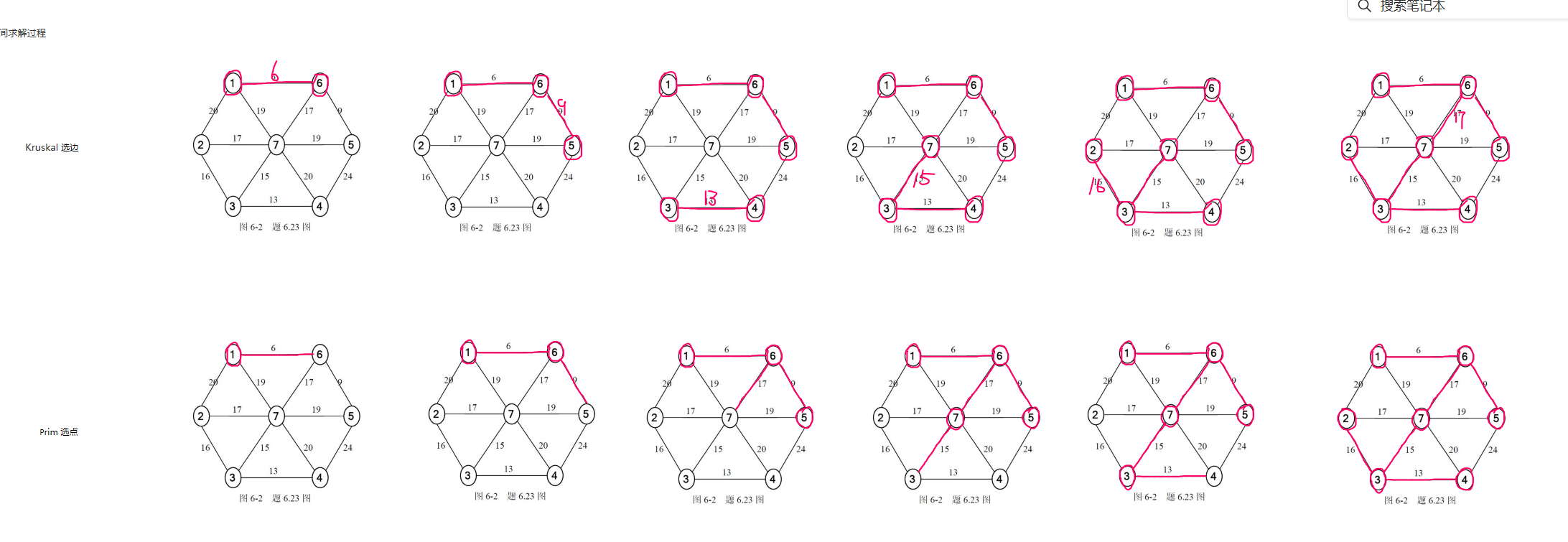
return true;

}

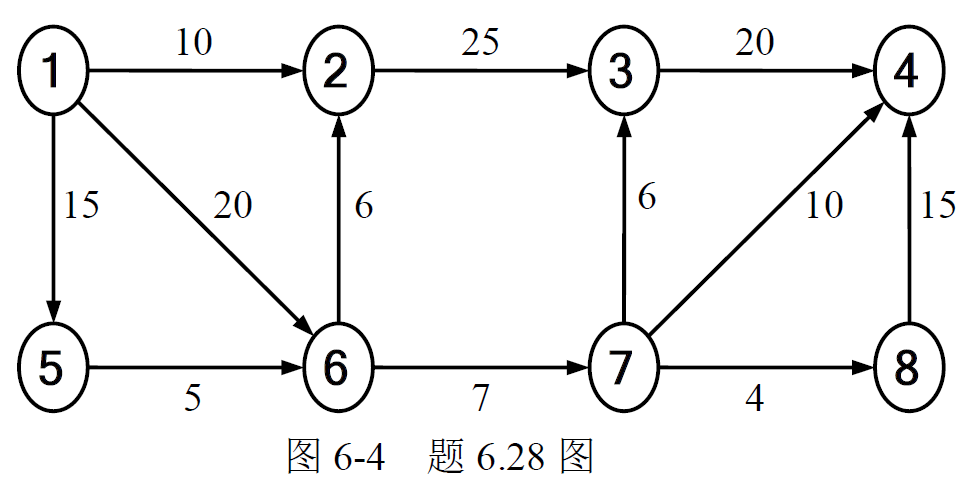
6.23 分别用prim算法和Kruskal算法求解下图的最小生成树，标注出中间求解过程的各状态。

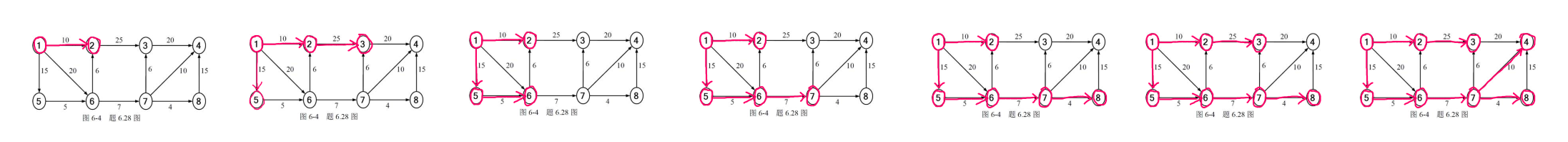


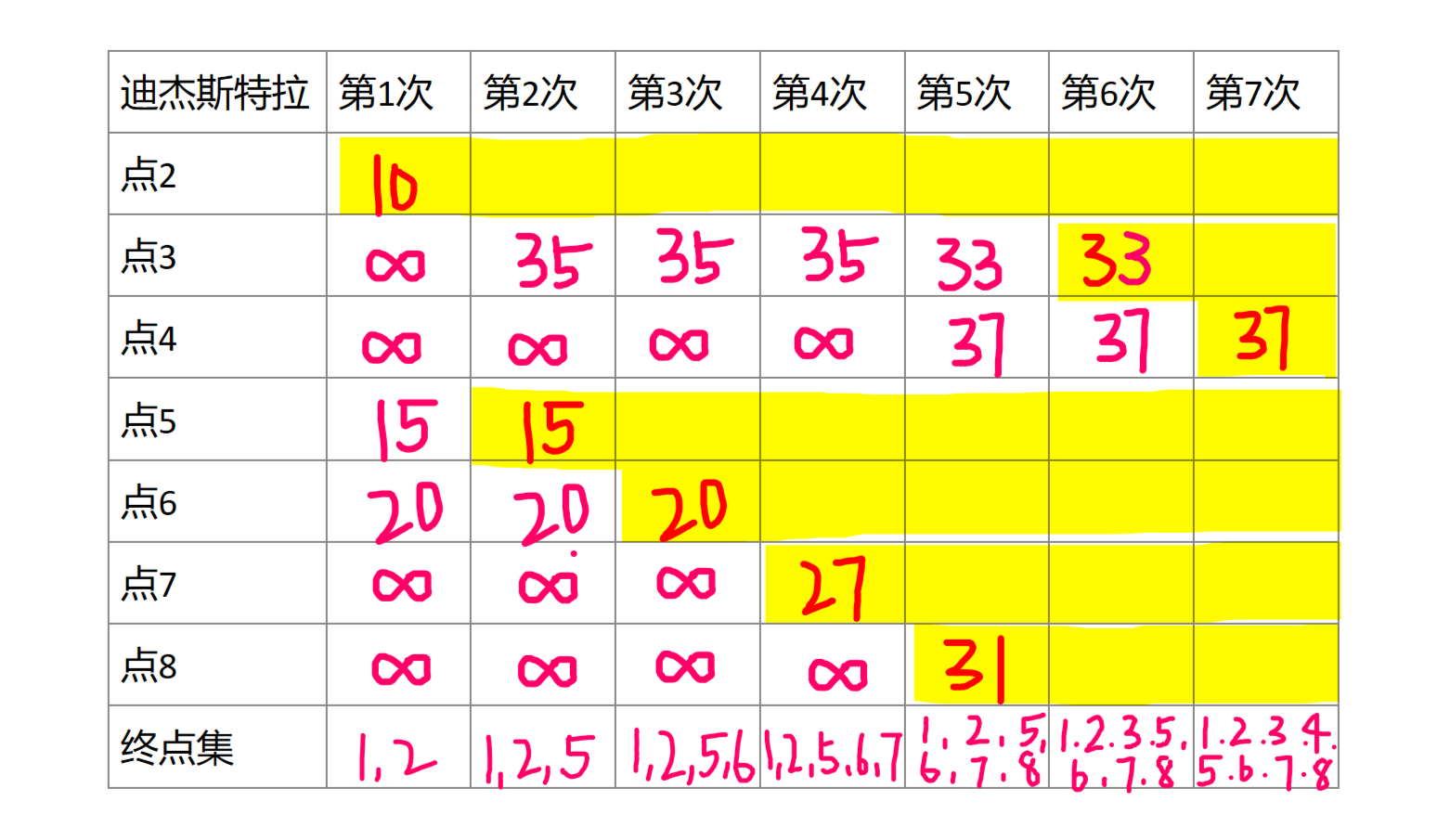
解答如图



6.28 对下面的图，求出从顶点1到其余各顶点的最短路径。







使用迪杰斯特拉算法求得

点1至点2最短路径为1->2，长度10；

点1至点3最短路径为1->5->6->7->3或1-6->7->3，长度33；

点1至点4最短路径为1->5->6->7->4或1-6->7->4，长度37；

点1至点5最短路径为1->5，长度15；

点1至点6最短路径为1->5->6或1->6，长度20；

点1至点7最短路径为1->5->6->7或1-6->7，长度27；

点1至点8最短路径为1->5->6->7->8或1-6->7->8，长度31；