Ch6

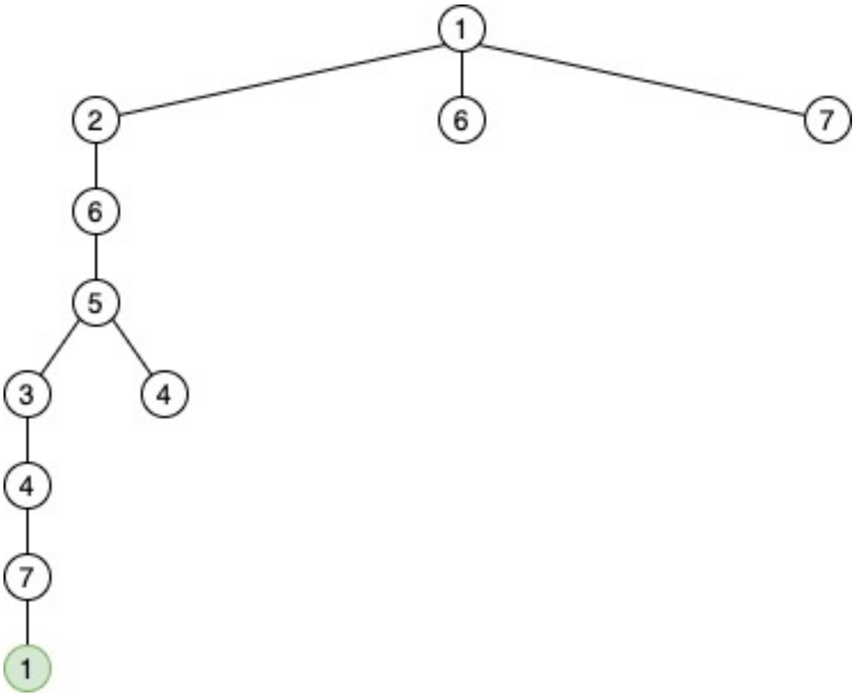
1.算法 DFS_Hamiltonian (Graph)

输入 :无向连通图的邻接矩阵 Graph

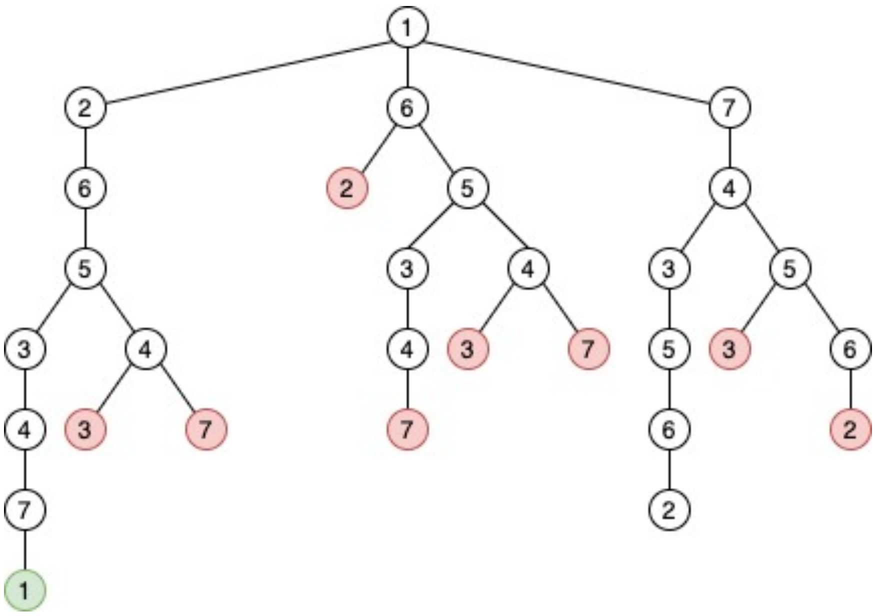
输出 :是否存在哈密顿环

```
For i=1 to n DO //依次判定从每个节点出发有无哈密顿环，n 是节点数
    visit[n]=0 //记录所有节点的访问状态
    path = [] //记录经过的节点
    Stack.Push(i)
    While Stack.empty()==False //如果栈非空
        father=Stack.Top()
        If father 第二次等于 i     then break
        path.append(father)
        Stack.Pop()
        for node in father //遍历该节点的连通子节点
            If visit[node]==False //如果还未被访问
                Stack.Push(node)
        end for
    end While
    if judge(path)==True //判定路径是否是每个节点只到达一次，且包含所有节点
        return True
return false
```

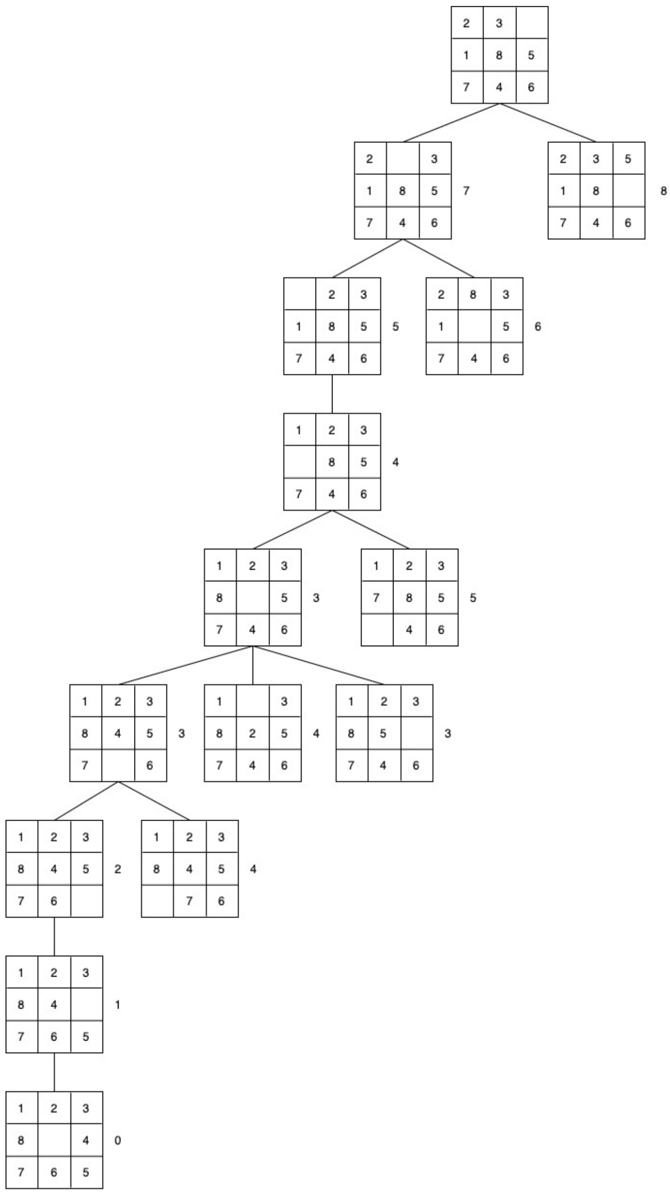
将解空间表示成树如下，深度优先的遍历顺序是 1-2-6-5-3-4-7-1，类似于前序遍历。



BFS 和 DFS 过程类似，算法详细过程省略，把栈换为队列即可。将解空间表示成树如下，广度优先的遍历顺序是 1-2-6-7-6-2-5-4-5-3-4-3-5-3-4-4-3-7-5-3-6-4-3-7-7-6-2-7-2-1，类似于层序遍历。



2.由于算法和课件一样，省略。爬山法和最佳优先方法搜索的过程一样，如下所示:



3. 方法是采用分支界限策略的深度优先搜索，依次对每个节点遍历。

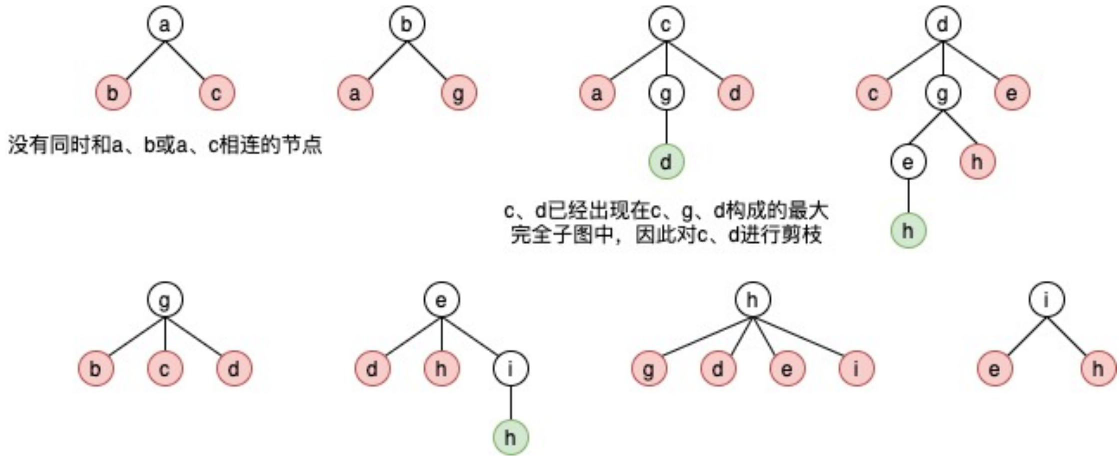
算法 Max_Clique(Graph)

输入 :无向连通图的邻接矩阵 Graph

输出 :所有最大完全字团

For *i*=1 to *n* **DO** //依次判定从每个节点出发有无最大完全子团，*n* 是节点数
对该节点做 DFS 搜索
只访问那些同时和该节点及其所有父亲有公共边的节点
如果该节点及其父亲已经出现在某个最大完全子团中，则剪枝
记录最大完全字团
return 所有最大完全子团

计算过程如下，得到 c、g、d，d、g、e、h 和 e、i、h 三个最大完全子团。



4.(1)类似于汉明距离，将 g(n)定义为初始状态到 n 增加的匹配数字个数

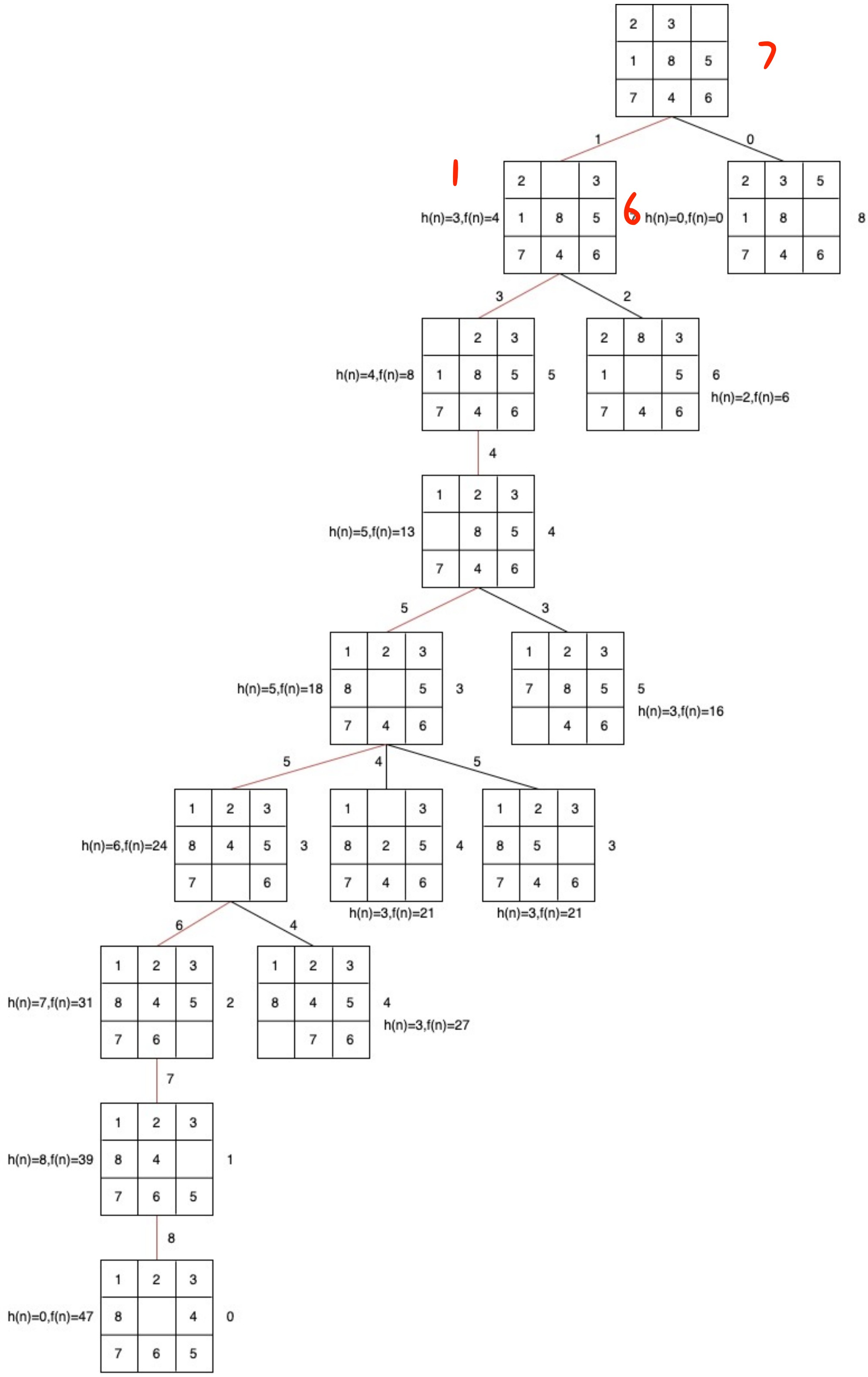
h*(n)=从 n 到目标状态增加的匹配数字个数

h(n)=从 n 到下一状态增加最多的匹配数字个数

f(n)=g(n)+h(n)

每次选择全局最大的 f(n)分支进行搜索，直到达到目标状态。

(2)计算步骤如下，剪枝掉先前已出现的状态，红色边展示了每次的选择分支:



5.(1)g(n)=从树根到 n 的代价

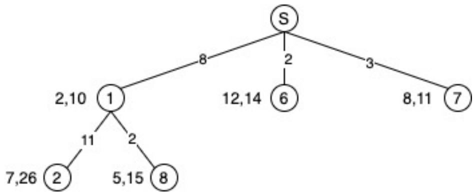
h*(n)=从 n 到目标节点的优化路径的代价

h(n)=从 n 到下一节点路径增长最小的代价

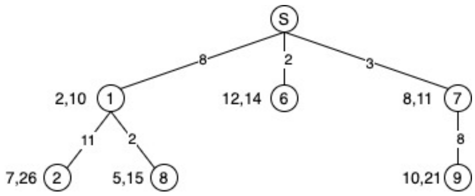
f(n)=g(n)+h(n)

(2)每次选择全局最小的 f(n)分支进行搜索，直到达到目标节点，剪枝掉相同节点中代价函数较大的。计算过程如下，二元组 x,y 分别表示 h(n)和 f(n)，得到最短路径为 S-7-9-11-12-T，距离为 45。

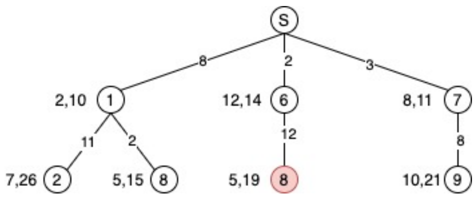
Step1:



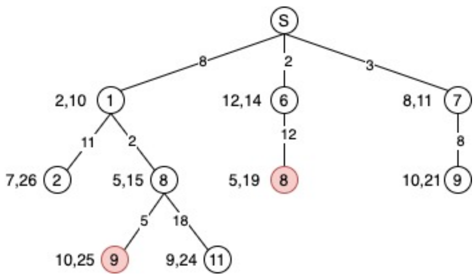
Step2:



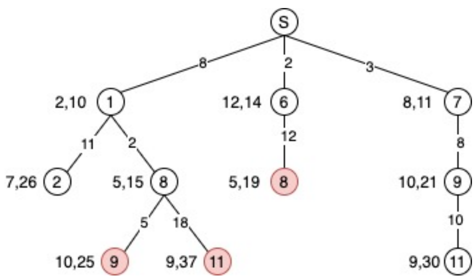
Step3:



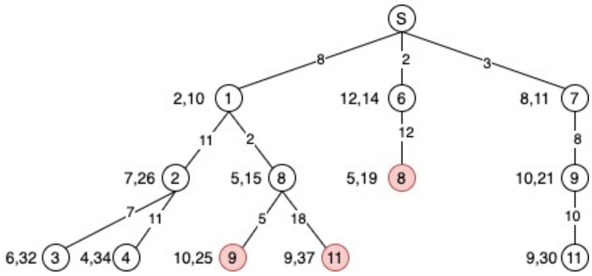
Step4:



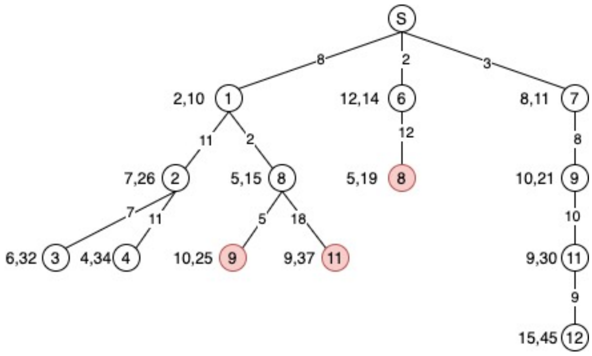
Step5:



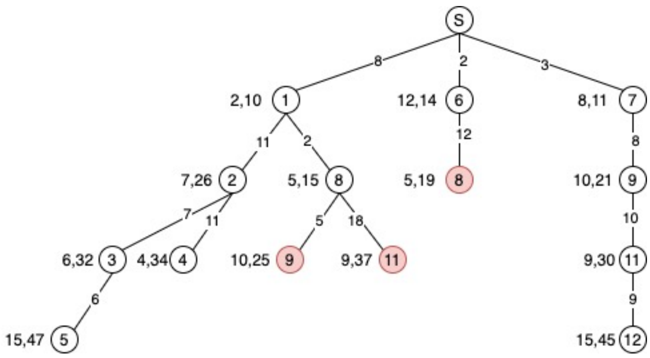
Step6:



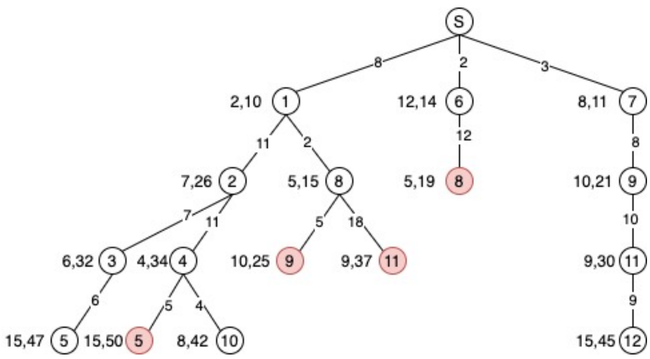
Step7:



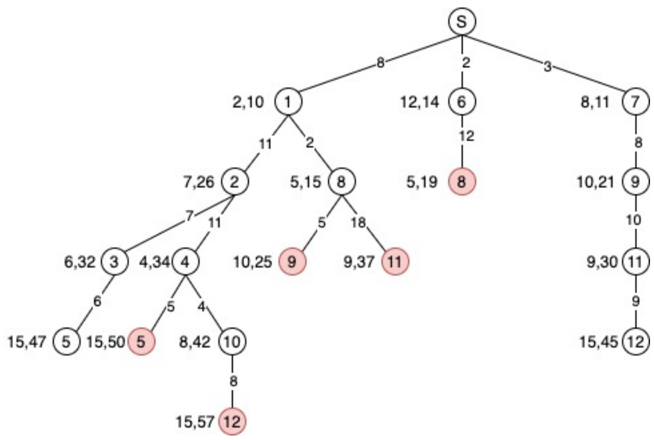
Step8:



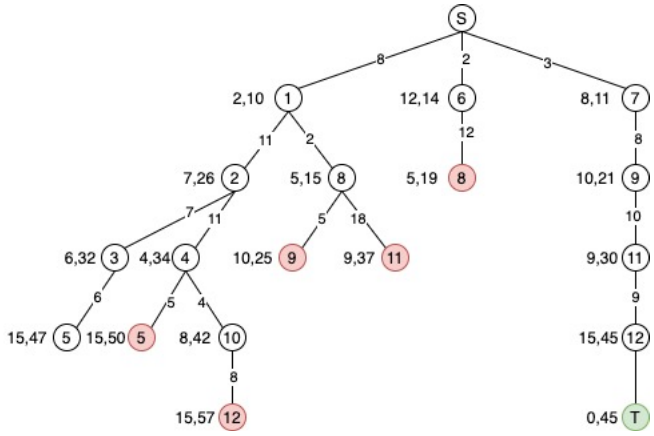
Step9:



Step10:


S
8
2
3
2,10
1
12,14
6
8,11
7
11
2
12
8
7,26
2
5,15
8
5,19
8
10,21
9
7
11
5
18
10
6,32
3
4,34
4
10,25
9
9,37
11
9,30
11
6
5
4
9
15,47
5
15,50
5
8,42
10
15,45
12
8
15,57
12

Step11:


S
8
2
3
2,10
1
12,14
6
8,11
7
11
2
12
8
7,26
2
5,15
8
5,19
8
10,21
9
7
11
5
18
10
6,32
3
4,34
4
10,25
9
9,37
11
9,30
11
6
5
4
9
15,47
5
15,50
5
8,42
10
15,45
12
8
15,57
12
0,45
T