人工智能

沙瀛

信息学院
2020.3

ARTIFICIAL INTELLIGENCE

与或树的搜索策略

- 解树
- 与或树的一般搜索过程
- 与\或树广度优先搜索算法
- 与\或树深度优先搜索算法
- 与\或树的有序搜索

启发式搜索方法

解树

- 与或图搜索是寻找解树的过程
- 解树是由初始节点及其下属的可解节点构成的

与或树的一般搜索思想

- 可解标示过程
  - 由可解节点来确定父节点、祖父节点等为可解节点的过程
- 不可解标示过程
  - 由不可解节点来确定父节点、祖父节点等为不可解节点的过程

与或树的一般搜索过程

- 把原始问题作为初始节点S0，并把它作为当前节点
- 应用分解或等价变换算符对当前节点进行扩展

与或树的一般搜索过程

- 为每个子节点设置指向父节点的指针
- 选择合适的节点作为当前节点，反复执行前两步，并多次调用可解标示过程和不可解标示过程，直到初始节点被标示为可解或不可解节点。

与或树的一般搜索过程

- 标示可解与不可解的过程是自下而上进行的。
- 如果已确定某个节点为可解节点，则其不可解的后继节点可以从搜索树中删除。
- 如果确定某个节点是不可解节点，则其全部后继节点都可以从搜索树中删除。

•算法

与／或树广度优先搜索算法

① 将初始节点S_0放入OPEN表中。

② 将OPEN表中第1个节点N放入CLOSED表中，并按顺序编号为n。

③ 若N有子节点存在，则执行下述操作：

- 生成N的所有子节点，将其放入OPEN的尾部，并填入其父节点编号n。

•算法

与／或树广度优先搜索算法

- 考察这些子节点中是否有终止节点，若有，则标记这些终止节点为可解节点，并将其放入CLOSED表中，然后由它们的可解性返回去推断其先辈的可解性，并对可解节点进行标记，若S0为可解节点，则搜索成功，退出。
- 从OPEN表中删除其先辈为可解的所有节点，转（2）。

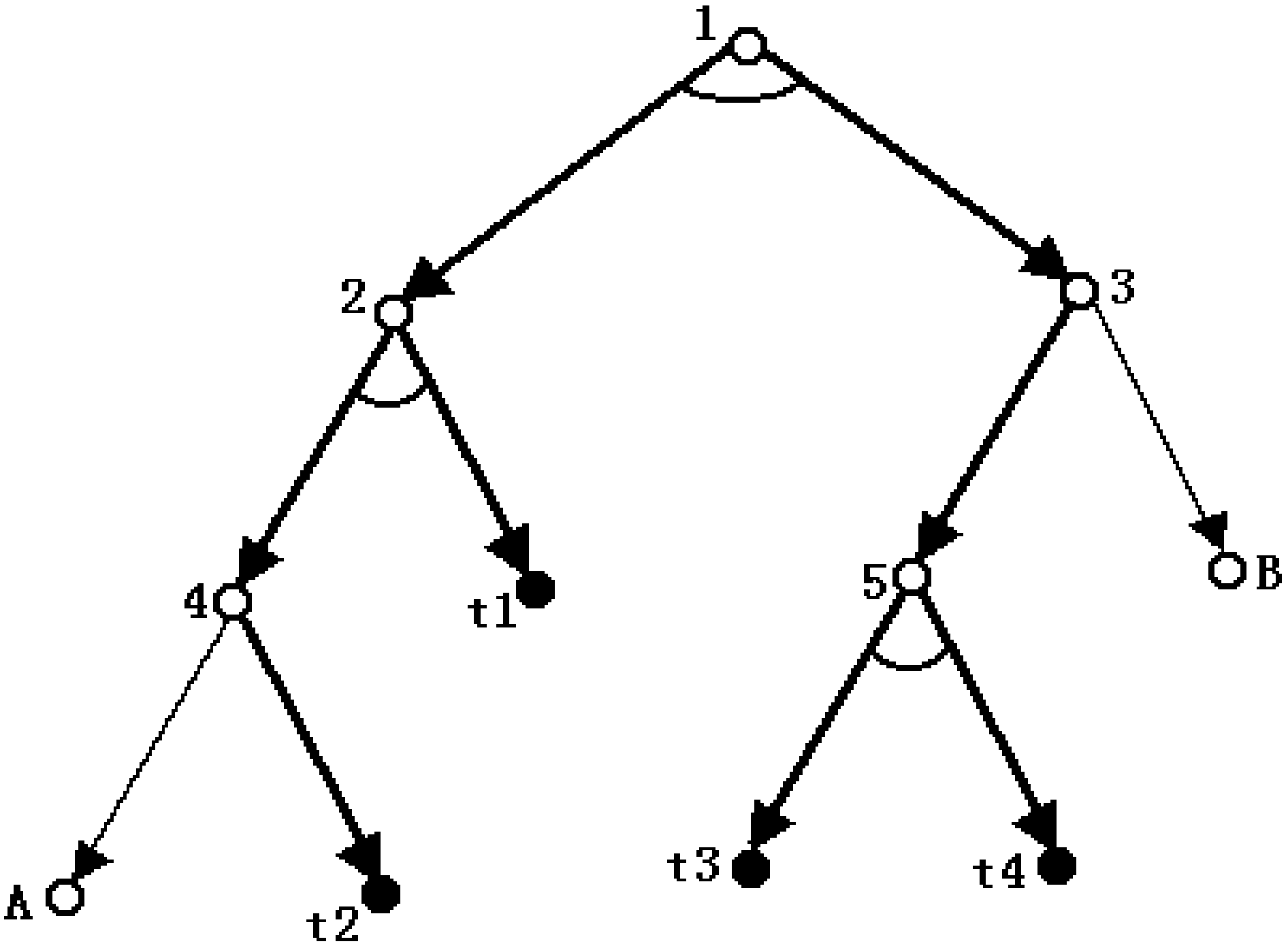
与／或树广度优先搜索算法

④ 若N不可扩展，则执行下述操作：

- 标记N为不可解节点，
- 由N的不可解性返回去推断其先辈的不可解性，并对不可解节点进行标记，若S0为不可解节点，则搜索失败，退出。
- 从OPEN表中删除其先辈为不可解的所有节点，转（2）。

•例子

- 例、求下述与或树的解树。树中，t1,t2,t3,t4为终止节点，A和B为不可解的端节点。


1
2
3
4
5
A
B
t1
t2
t3
t4

•算法

与／或树深度优先搜索算法

① 将初始节点S_0放入OPEN表中。

② 将OPEN表中第1个节点N放入CLOSED表中，并按顺序编号为n。

③ 若N有子节点存在，则执行下述操作：

- 生成N的所有子节点，将其放入OPEN的首部，并填入其父节点编号n。

•算法

与／或树深度优先搜索算法

- 考察这些子节点中是否有终止节点，若有，则标记这些终止节点为可解节点，并将其放入CLOSED表中，然后由它们的可解性返回去推断其先辈的可解性，并对可解节点进行标记，若S0为可解节点，则搜索成功，退出。
- 从OPEN表中删除其先辈为可解的所有节点，转（2）。

•算法

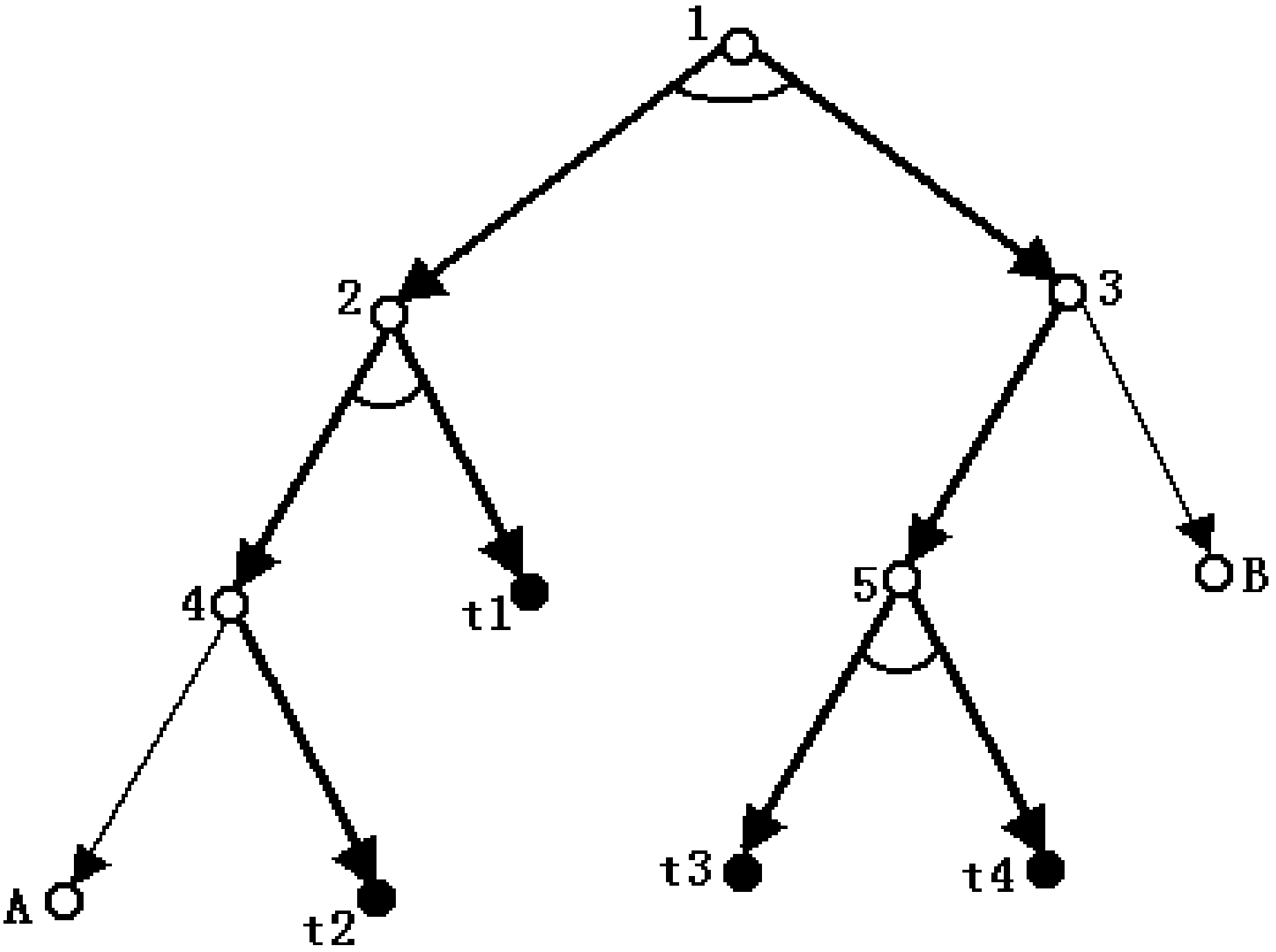
与／或树深度优先搜索算法

④ 若N无子节点存在，则执行下述操作：

- 标记N为不可解节点，
- 由N的不可解性返回去推断其先辈的不可解性，并对不可解节点进行标记，若S0为不可解节点，则搜索失败，退出。
- 从OPEN表中删除其先辈为不可解的所有节点，转（2）。

•例子

- 例、求下述与或树的解树。树中，t1,t2,t3,t4为终止节点，A和B为不可解的端节点。并规定深度界限为4。


1
2
3
4
5
A
B
t1
t2
t3
t4

与\或树的有序搜索

- 盲目搜索
  - 没有考虑代价,所求得的解树不一定是代价最小的解树,即不是最优解树。
- 启发式搜索
  - 要多看几步，计算一下扩展一个节点可能要付出的代价，以选择代价最小的节点进行扩展。

解树的代价

设g(x)表示节点x的代价，c(x,y)表示节点x到其子节点y的代价。

(1)若x是终止节点，则g(x)=0

(2)若x是或节点，其子节点依次为$y_1,y_2,\cdots,y_n$，则有：

$$g(x)=\min_{1\leqslant i\leqslant n}\{c(x,y_i)+g(y_i)\}$$

解树的代价

(3)若x是与节点，其子节点依次为$y_1,y_2,\cdots,y_n$，则有：

和代价法：$g(x)=\sum_{1\leqslant i\leqslant n}(c(x,y_i)+g(y_i))$

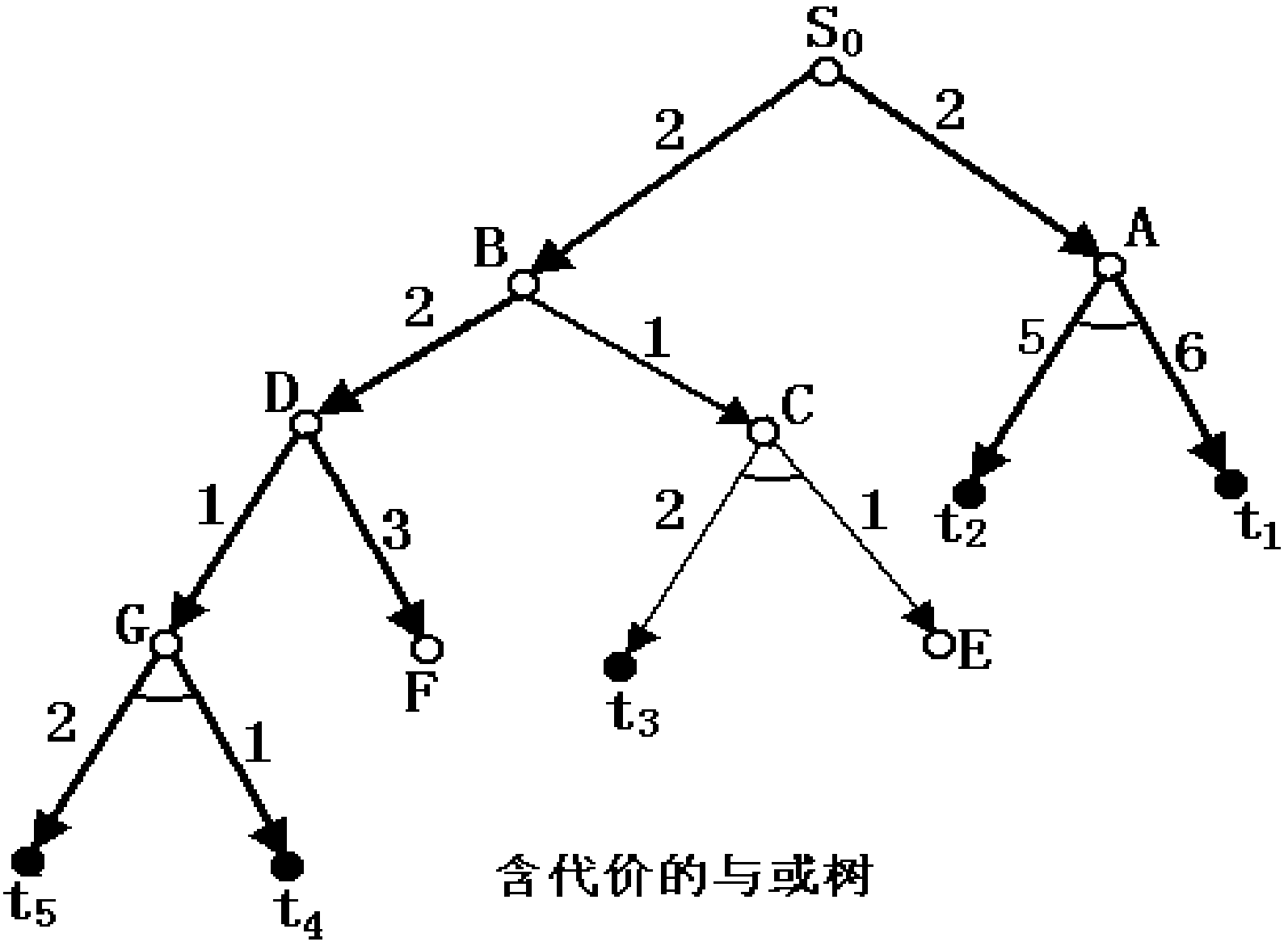
最大代价法：$g(x)=\max_{1\leqslant i\leqslant n}\{c(x,y_i)+g(y_i)\}$

解树的代价

(4)若x不可扩展，则：g(x)=∞

若问题有解的话，则子节点的代价可以推算出父节点的代价，直到初始节点的代价，S0的代价就是解树的代价。

例、根据各个边的代价求各节点的代价。



含代价的与或树

•例子

求解结果

有两棵解树，一棵解树由S0, A, t_1, t_2组成。

按和代价有： $g(A)=11$， $g(S_0)=13$

按最大代价有：$g(A)=6$， $g(S_0)=8$

•例子

求解结果

另一棵解树由S0, B, D, G, t4, t5组成。

按和代价有：

$g(G)=3, g(D)=4, G(B)=6, g(S_0)=8$

按最大代价有：

$g(G)=2, g(D)=3, G(B)=5, g(S_0)=7$

启发式搜索方法

希望树

- 在有序搜索中，应选择那些最有希望成为最优解树一部分的节点进行扩展。我们称这节点构成的树为希望树。

希望树

- 初始节点S0在希望树中。
- 如果节点x在希望树中，则：
  - 如果x是或节点，且其子节点依次为$y_1,y_2,\cdots,y_n$，则具有下述值的子节点也在希望树中：

$$\min\{c(x,y_i)+g(y_i)\}$$

$$1\leqslant i\leqslant n$$

  - 若x是与节点，则其所有子节点都在希望树中。

•算法

与/或树的有序搜索算法

(1) 将初始节点S_0放入OPEN表中。

(2) 求出希望树T，即根据当前搜索树中节点的代价求出以S_0为根的希望树T。

(3) 依次将OPEN表中T的端节点N放入CLOSED表中

与/或树的有序搜索算法

（4）若N为终止节点，则：

① 将它标记为可解节点

② 由N的可解性返回去推断其先辈的可解性，并对可解节点进行标记，若S0为可解节点，则搜索成功，退出。

③ 从OPEN表中删除其先辈为可解的所有节点，转（2）。

与/或树的有序搜索算法

(5) 若N不是终止节点，且不可扩展，则：

① 标记N为不可解节点。

② 由N的不可解性返回去推断其先辈的不可解性，并对不可解节点进行标记，若S0为不可解节点，则搜索失败，退出。

③ 从OPEN表中删除其先辈为不可解的所有节点，转（2）。

•算法

与/或树的有序搜索算法

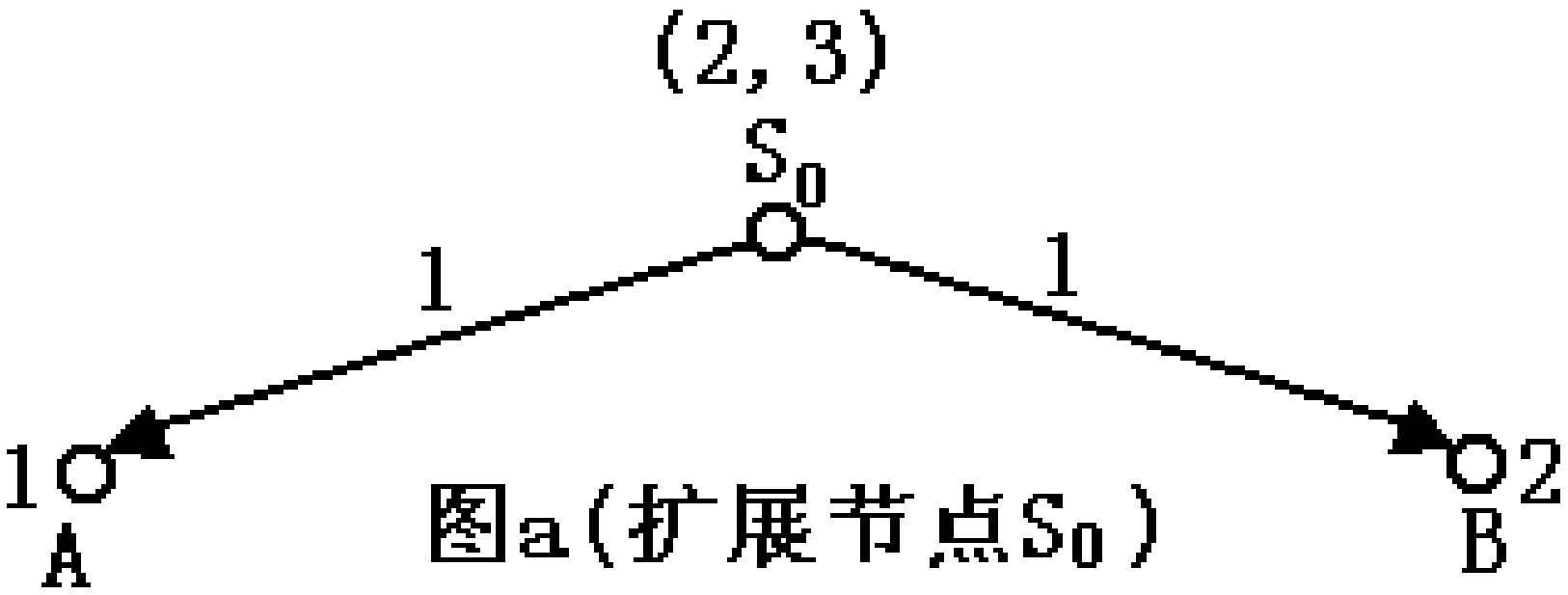
(6) 若N不是终止节点，且可扩展，则：

① 生成N的所有子节点。

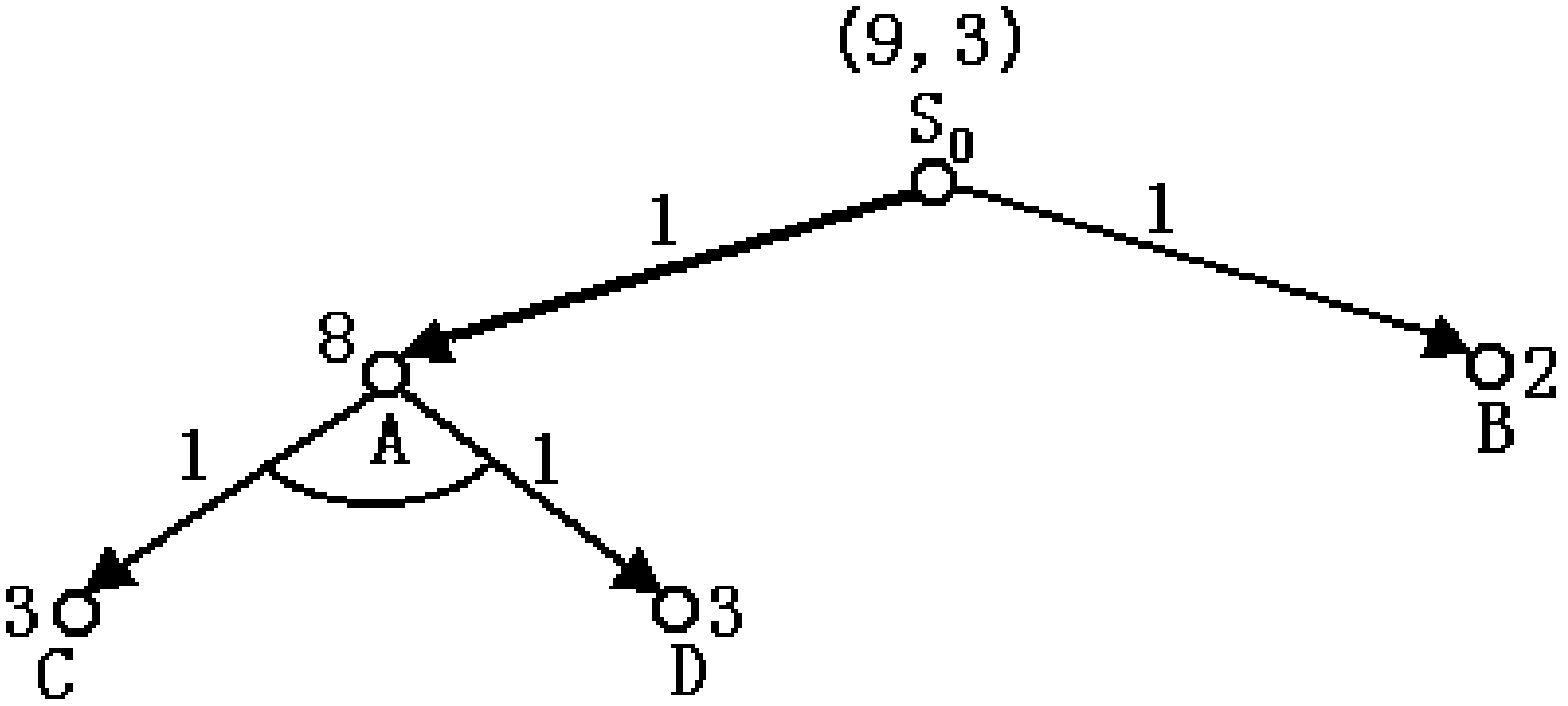
② 将其放入OPEN中，并填入其父节点编号n。

③ 计算所有子节点及其先辈节点的代价值，转（2）。

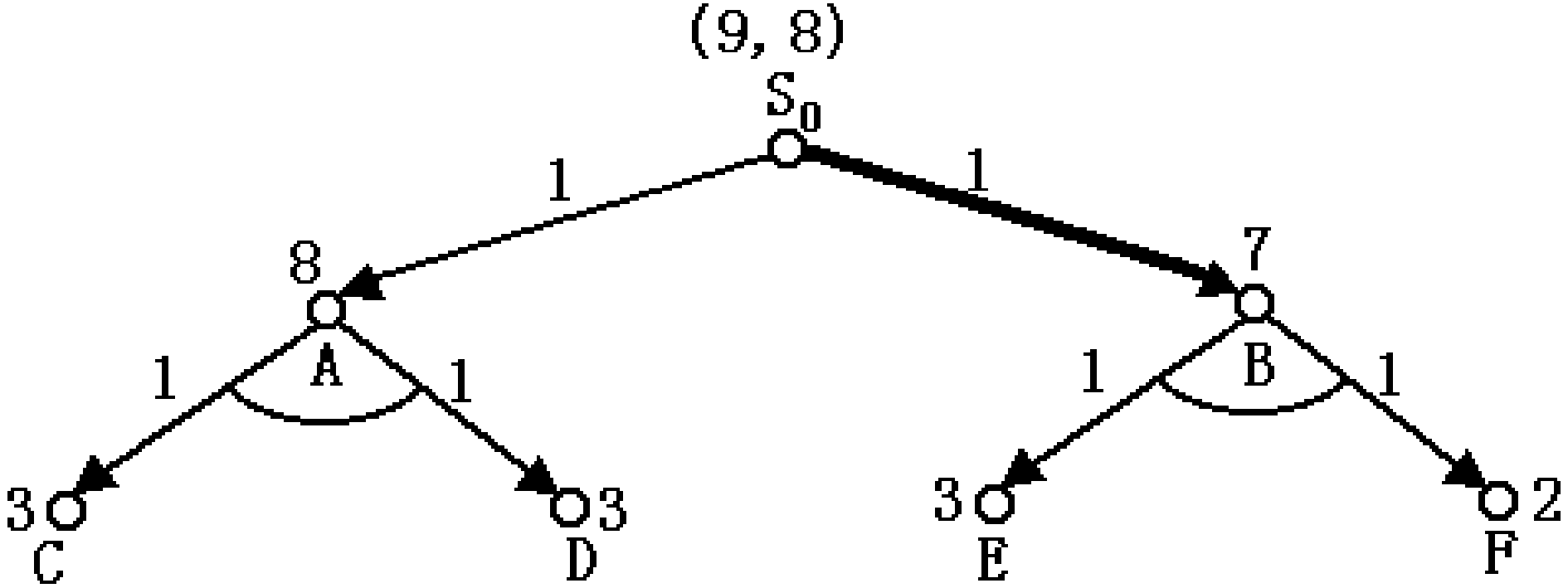
例、与或树有序搜索示例



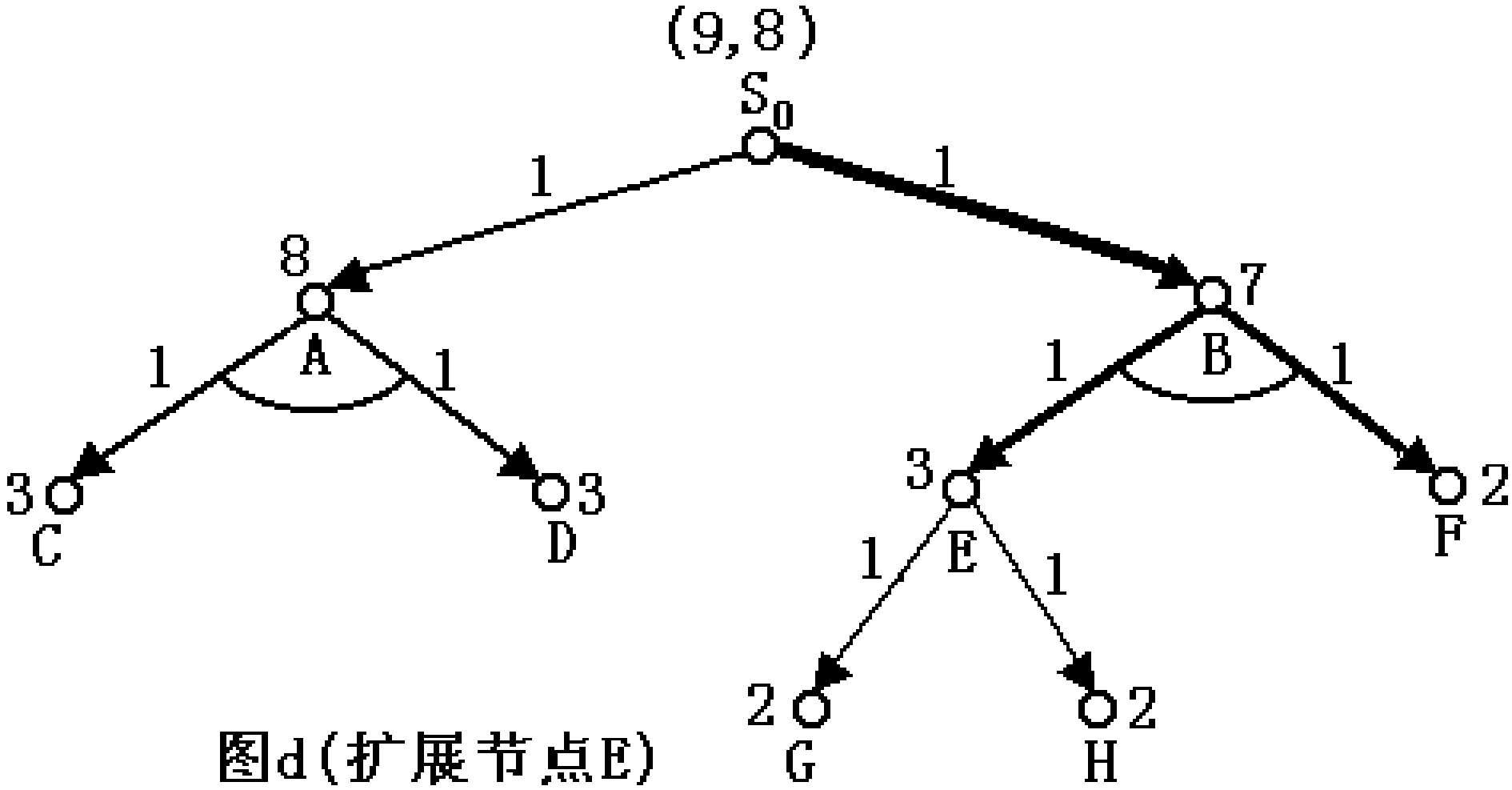
图a(扩展节点S_0)



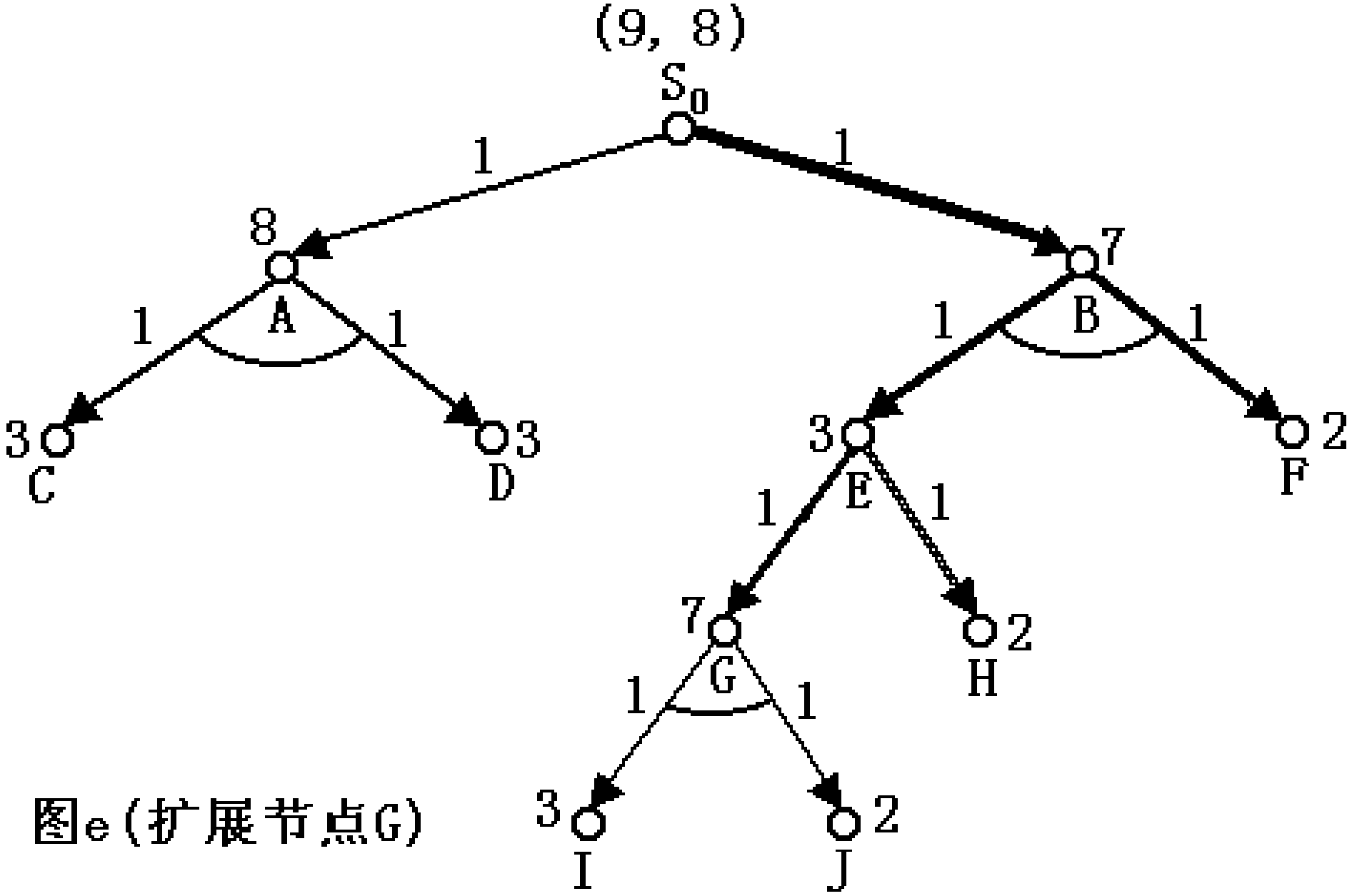
图b(扩展节点A)



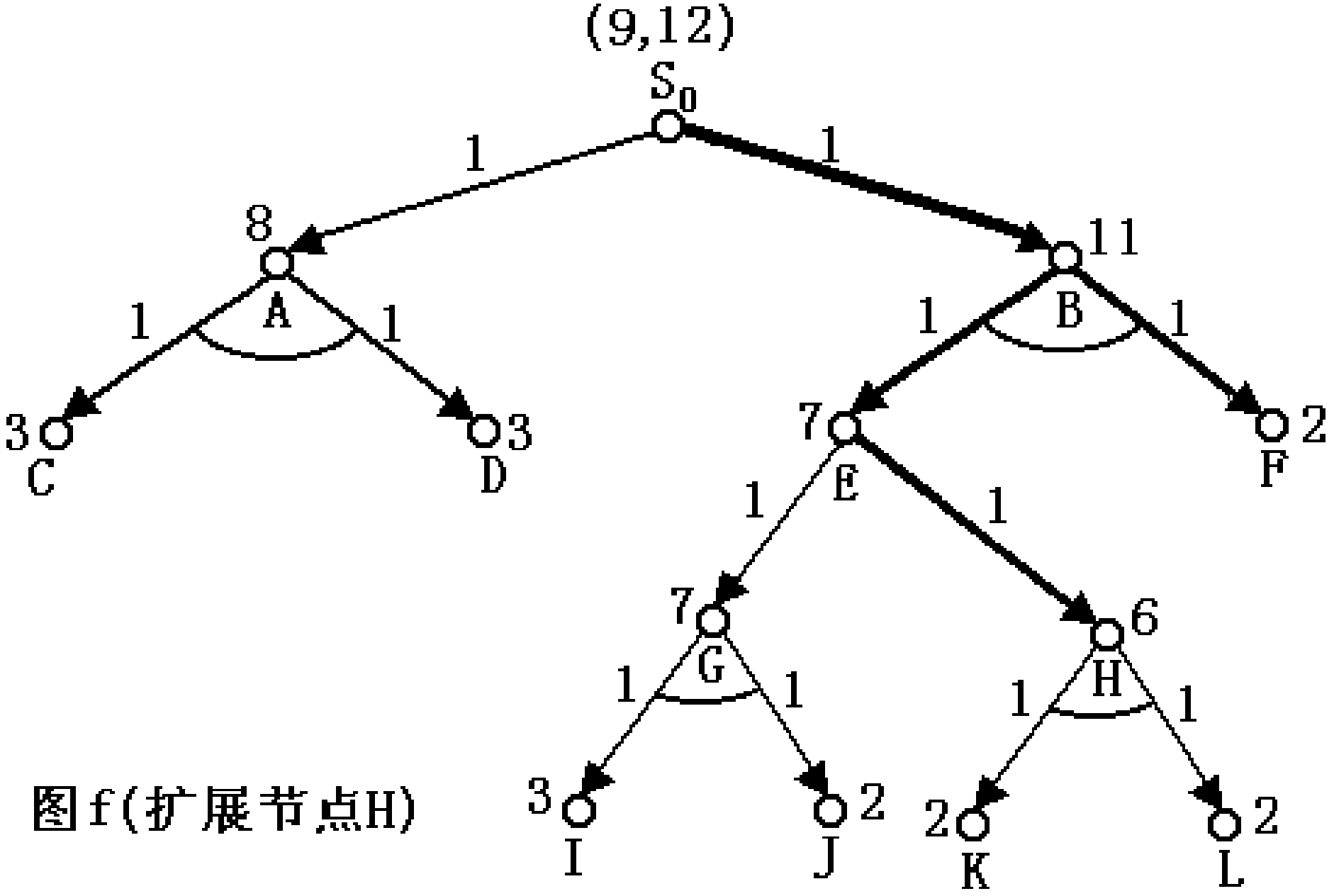
图c(扩展节点B)



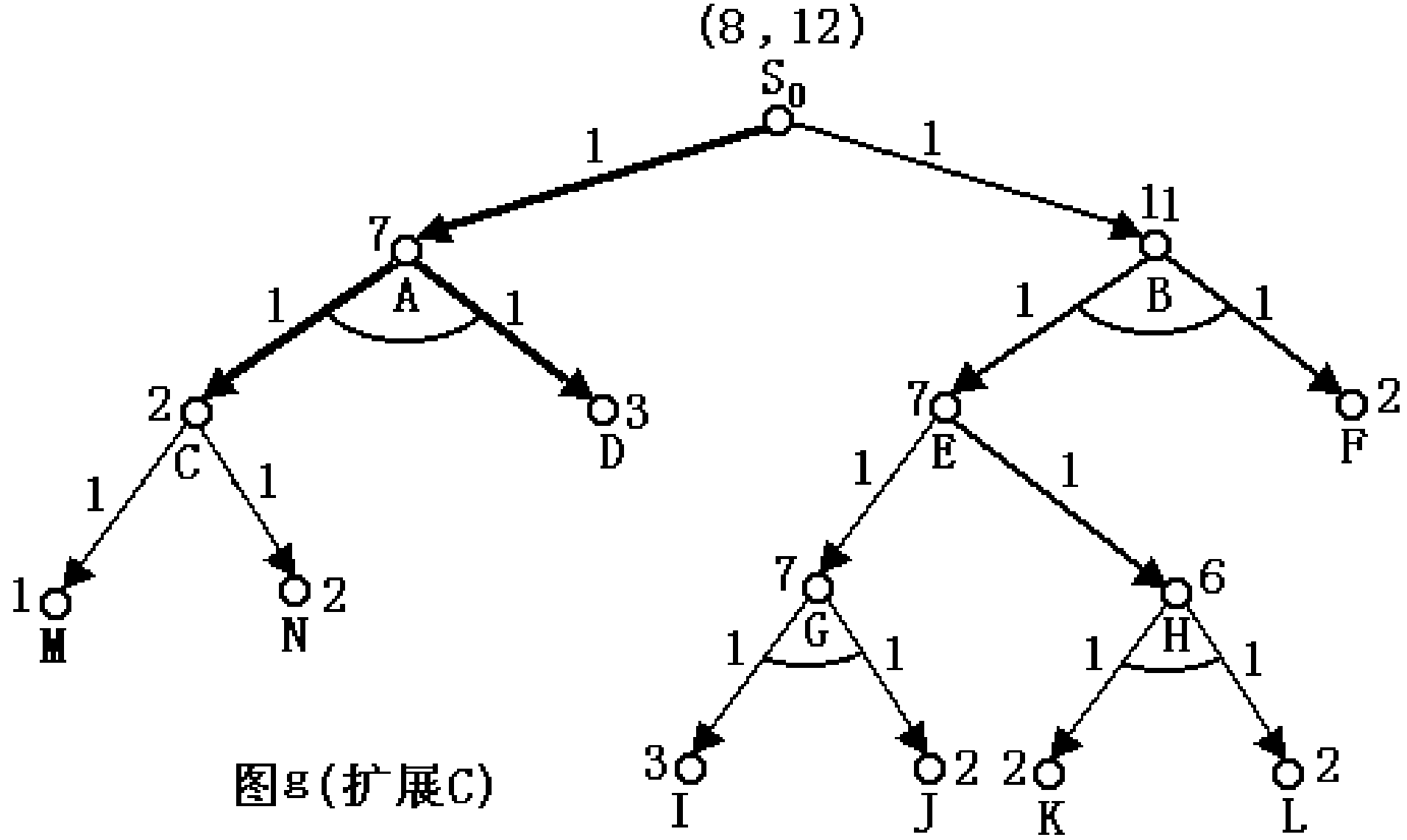
图d(扩展节点E)



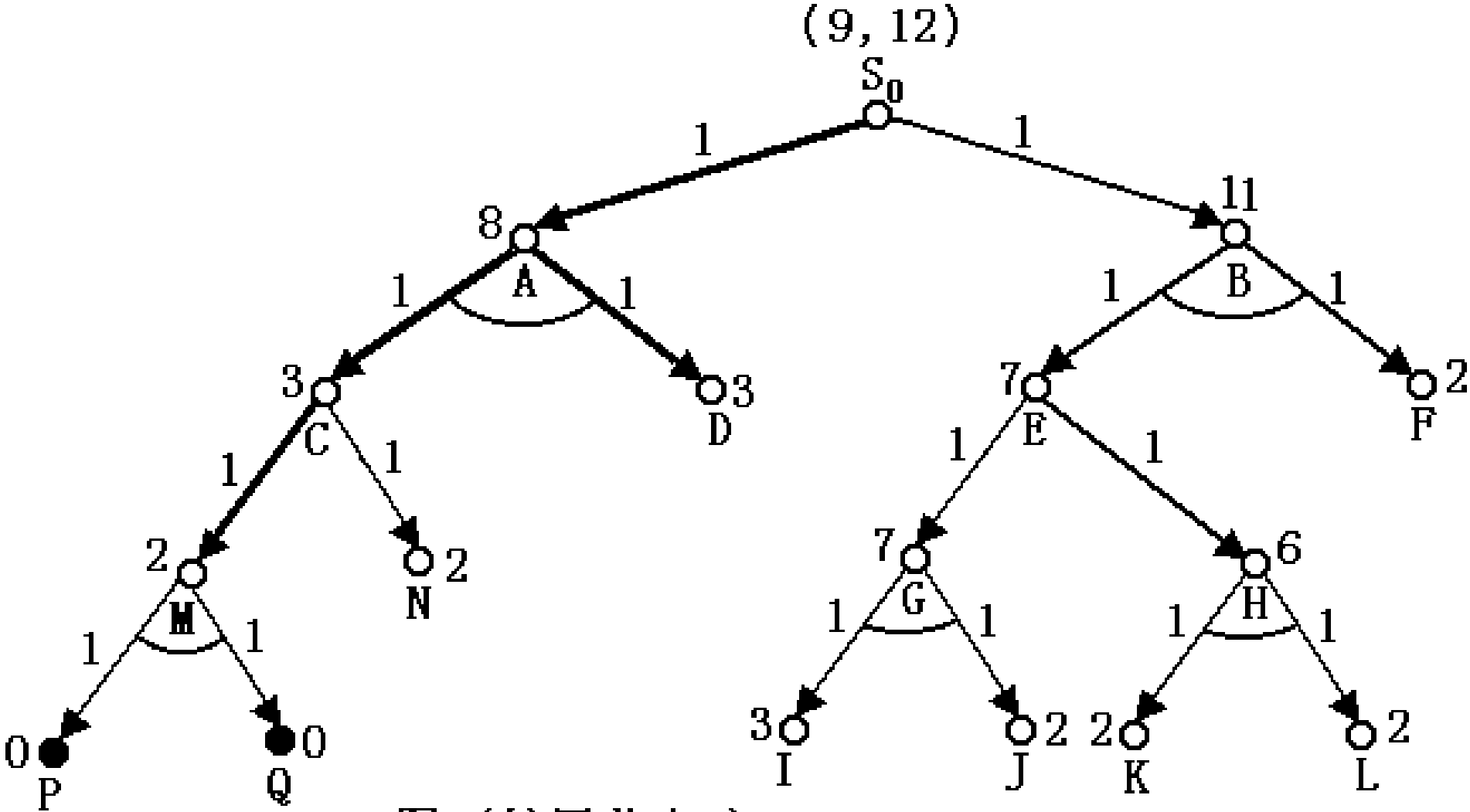
图e(扩展节点G)



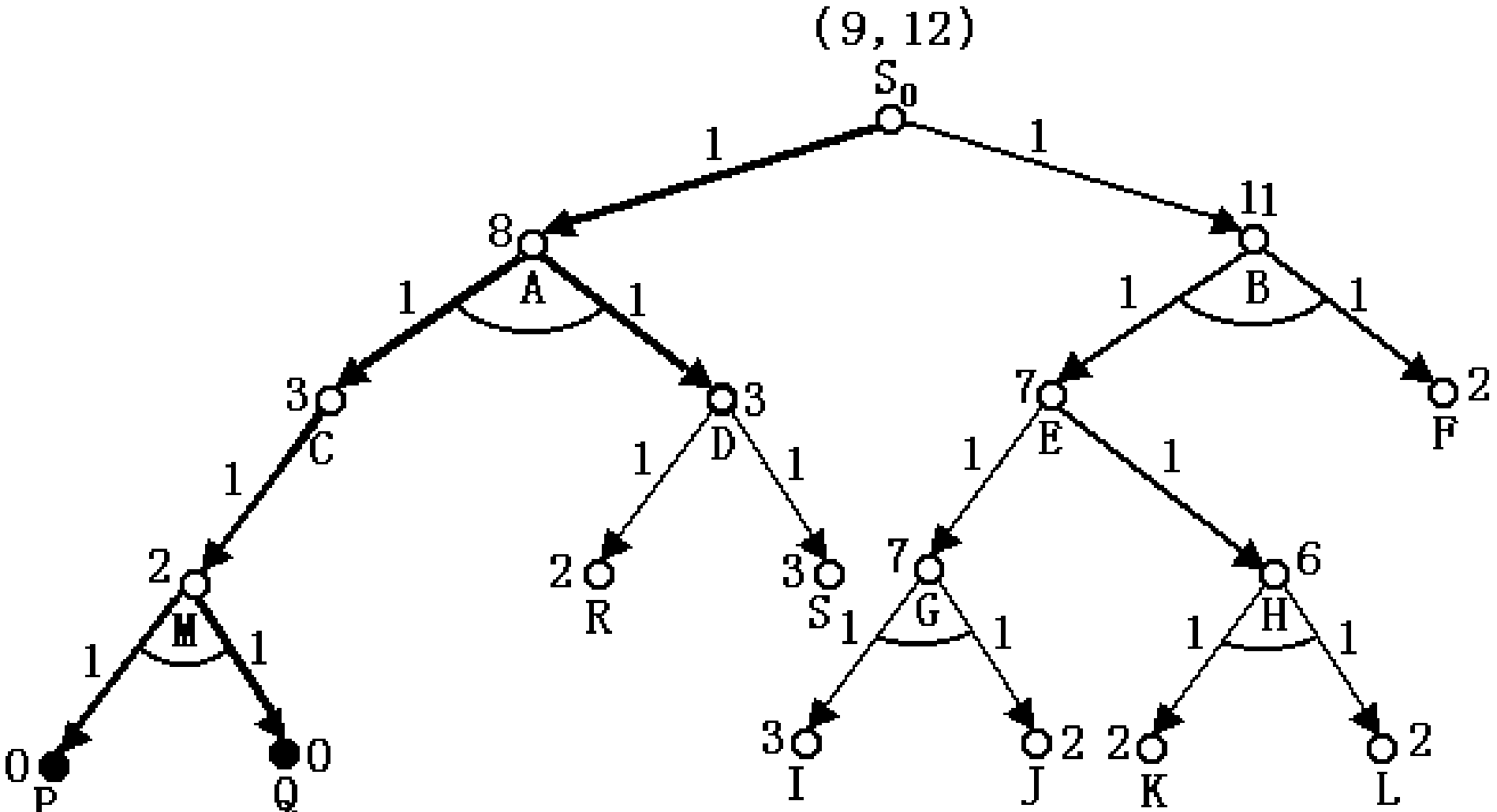
图f(扩展节点H)



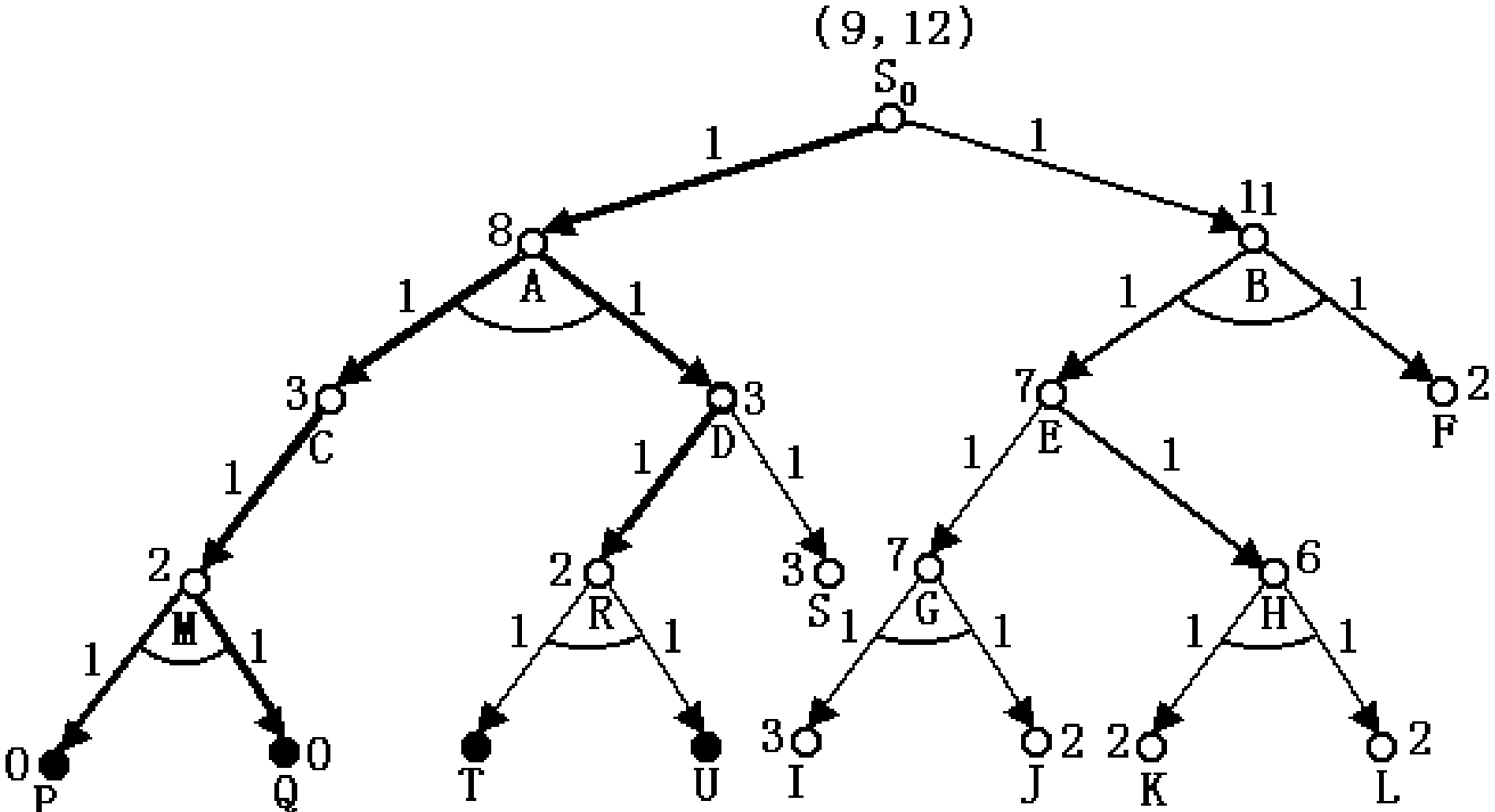
图g(扩展C)


(9, 12)
S0
1
1
8
11
A
B
1
1
1
1
3
7
2
C
E
F
D
1
1
1
1
2
7
6
M
G
H
N
1
1
1
1
1
1
0
0
3
2
2
2
P
Q
I
J
K
L

图h(扩展节点M)


(9,12)
S0
1
1
8
11
A
B
1
1
1
1
3
3
7
2
C
D
E
F
1
1
1
1
2
3
7
6
R
S
G
H
1
1
1
1
1
1
0
0
3
2
2
2
P
Q
I
J
K
L

图i(扩展节点D)


(9, 12)
S0
1
1
8
11
A
B
1
1
1
1
3
3
7
2
C
D
E
F
1
1
1
1
2
2
3
7
6
M
R
S
G
H
1
1
1
1
1
1
1
1
0
0
3
2
2
2
P
Q
T
U
I
J
K
L

图j(扩展节点R)

•例子

求解结果

T,U是终止节点，推得节点R,D,A,S_0都是可解节点。故搜索成功，最优解树为：

S_0ACMPQDRTU：

本节结束！