与或图搜索

Outline

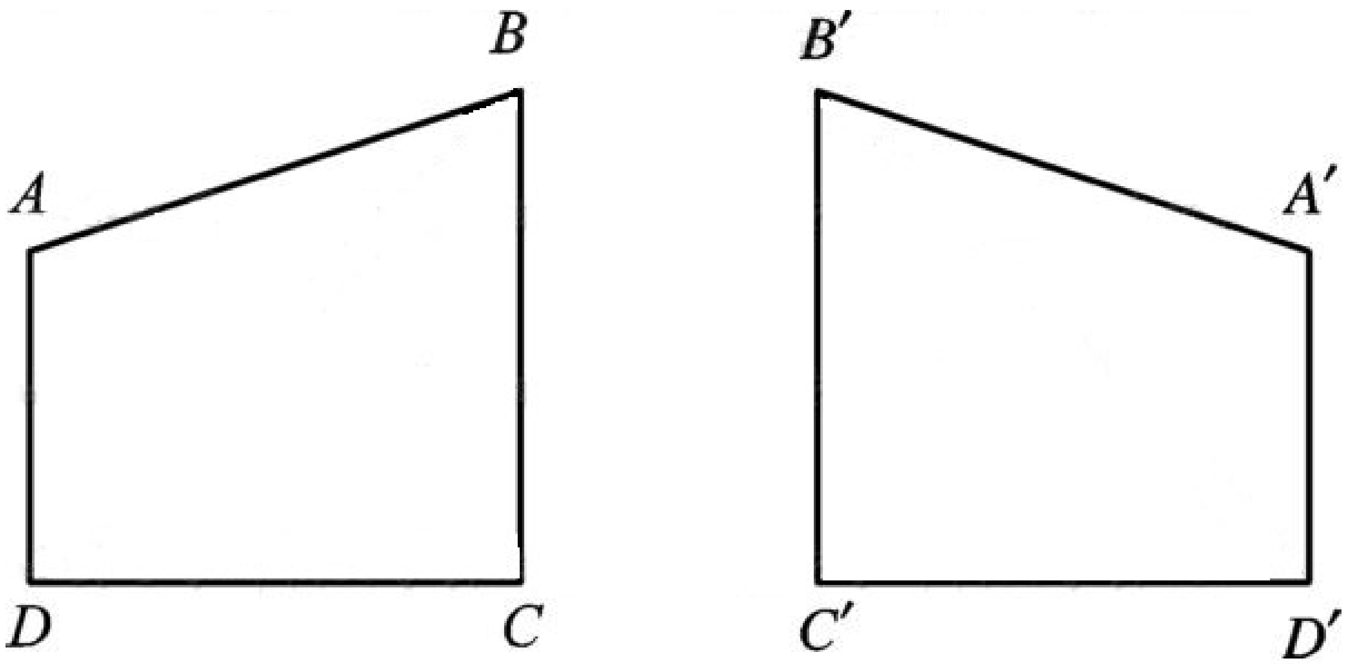
- 问题规约
- 与或图的概念
- 与或图的搜索过程
- 启发式与或图搜索

与或图搜索

- 同状态图一样，与或图也是问题的一种抽象表示。
- 许多问题的求解过程都可以用与或图搜索来描述。如梵塔问题、猴子摘香蕉问题、博弈问题、求不定积分问题、定理证明问题等等。
- 研究与或图搜索具有普遍意义。

问题的引出——全等四边形的证明

如图所示，设有四边形*ABCD*和 *A ' B ' C ' D '* ,要求证明它们全等。



分析：连接B、D和B'、D', 则原问题可分解为两个子问题

Q1：证明$\triangle ABD \cong \triangle A'B'D'$

Q2：证明$\triangle BCD \cong \triangle B'C'D'$

问题Q1可进一步被分解为

Q11：证明$AB = A'B'$

Q12：证明$AD = A'D'$

Q13：证明$\angle A = \angle A'$

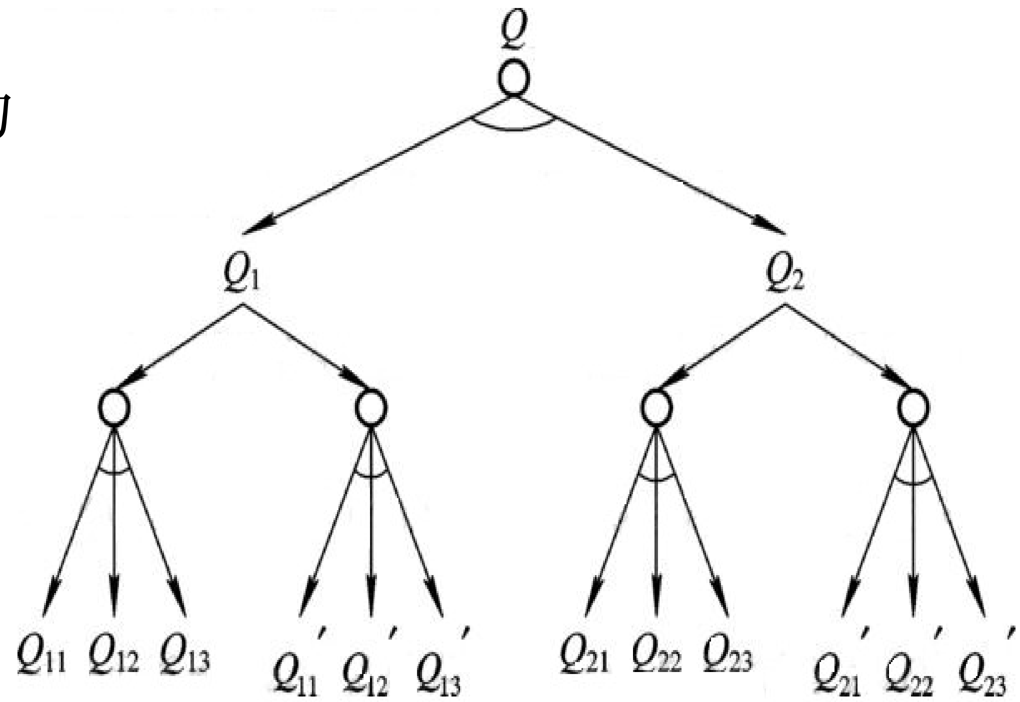
或

Q11′：证明$AB = A'B'$

Q12′：证明$AD = A'D'$

Q13′：证明 $BD = B'D'$

……



全等四边形证明

问题的分解与变换

问题归约的描述

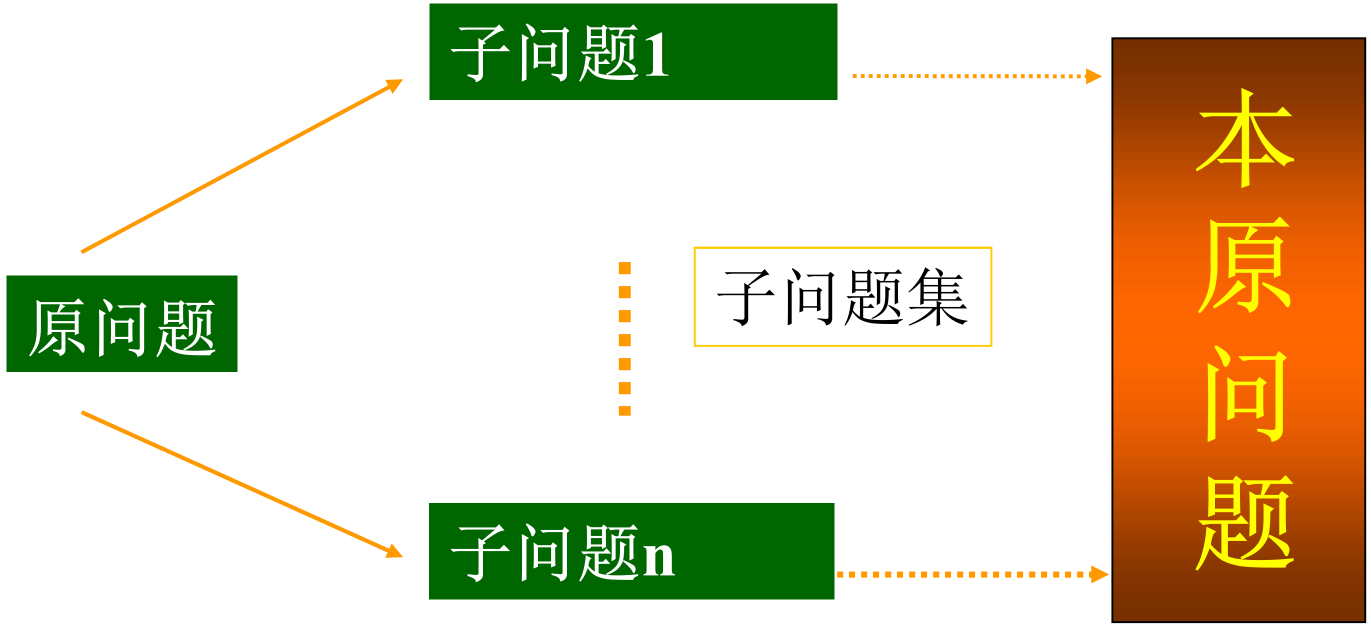
- 基本思想：通过问题的分解或变换，将复杂的问题转化为一系列简单的问题，然后通过对这些简单问题的求解来实现对原问题的求解。

- 若一个问题P可以归约为一组子问题，并且只有当所有子问题都有解时原问题P才有解，任何一个子问题无解都会导致原问题P无解，则称这种归约为问题的分解 。

——分解所得的子问题的“与”与原问题等价

- 若一个问题P可以归约为一组子问题，并且这些子问题中只要有一个有解时P就有解，只有当所有子问题都无解时原问题P才无解，则称这种归约为问题的等价变换。

——变换所得的子问题的“或”与原问题等价



本原问题：不能（或不需要）再进行分解或变换即可直接解答的子问题

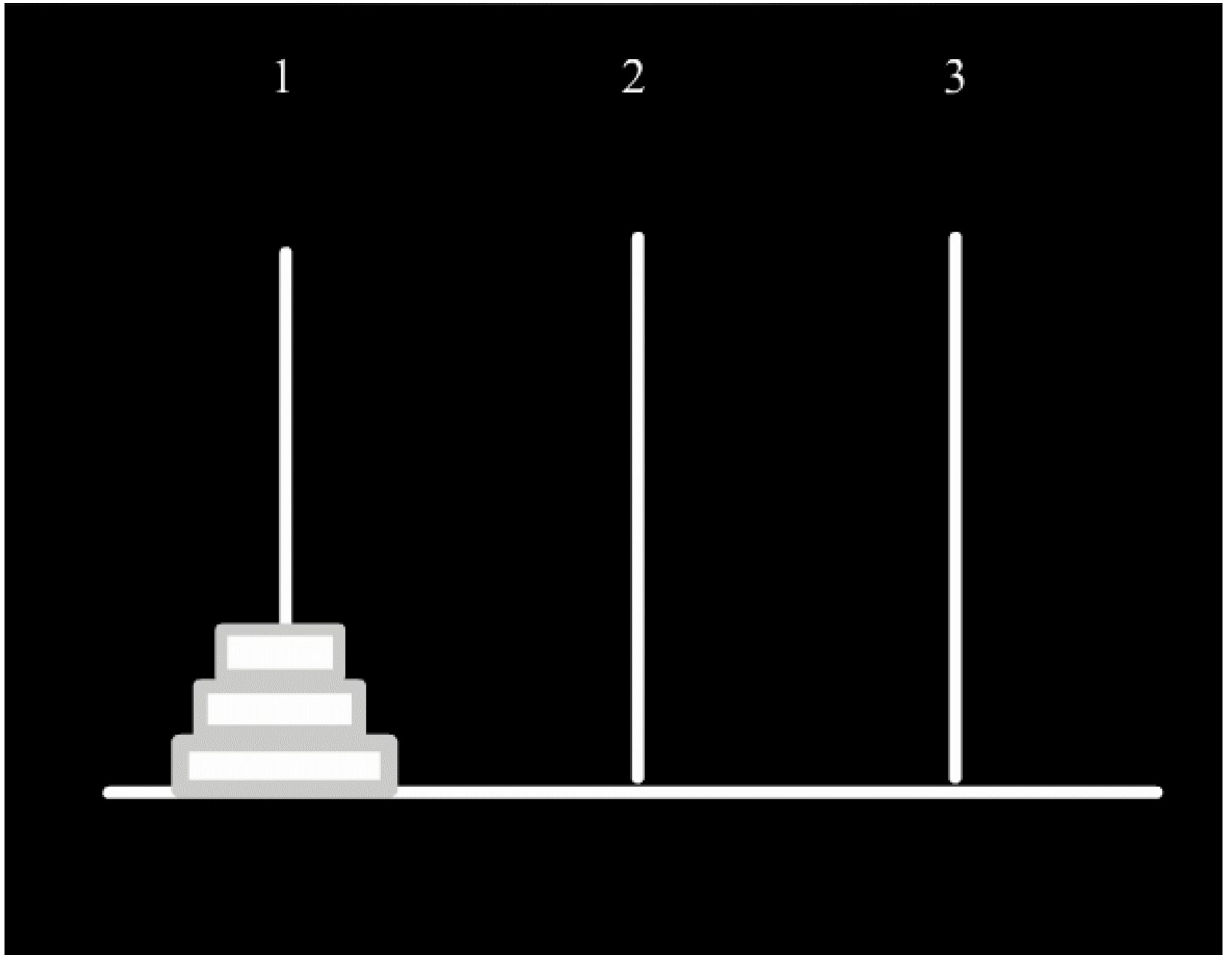
例：梵塔问题

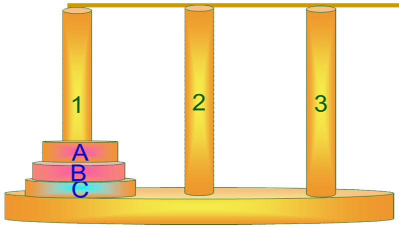
传说在印度的贝那勒斯的圣庙中,主神梵天做了一个由64个大小不同的金盘组成的“梵塔”，并把它穿在一个宝石杆上。另外，旁边再插上两个宝石杆。 然后，他要求僧侣们把穿在第一个宝石杆上的64个金盘全部搬到第三个宝石杆上。 搬动金盘的规则是：一次只能搬一个；不允许将较大的盘子放在较小的盘子上。

梵天预言：一旦64个盘子都搬到了3号杆上，世界将在一声霹雳中毁灭。

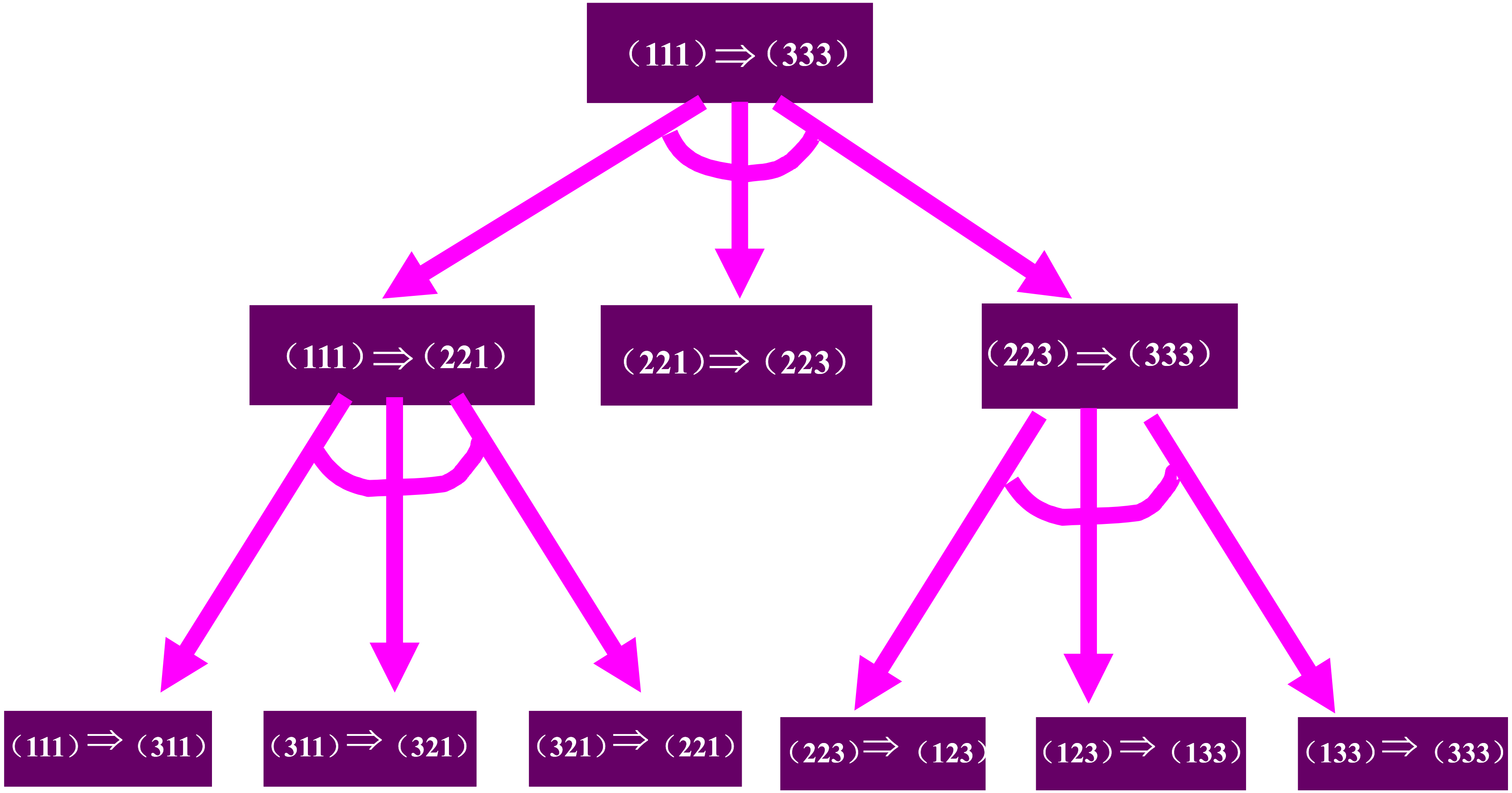
把64个盘子全部搬到3号杆上，需要穿插搬动盘子次数：$2^{64}-1=18\ 446\ 744\ 073\ 709\ 511\ 615$ 。

梵塔问题的解题过程




1
2
3
A
B
C

3阶梵塔问题的规约图


（111）⇒（333）
（111）⇒（221）
（221）⇒（223）
（223）⇒（333）
（111）⇒（311）
（311）⇒（321）
（321）⇒（221）
（223）⇒（123）
（123）⇒（133）
（133）⇒（333）

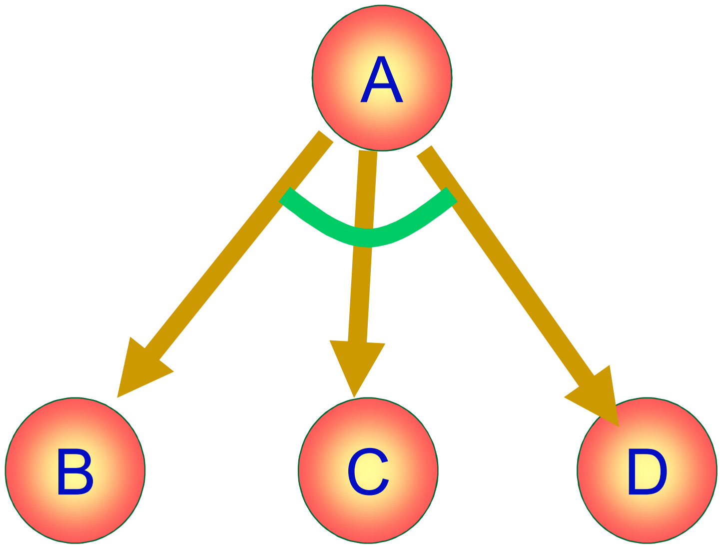
例：Fibonacci数列

- 1202年，意大利家斐波那契提出了一个关于兔子繁殖的问题：
  如果一对兔子每月能生一对小兔（一雄一雌），而每对小兔在它出生後的第三个月里，又能开始生一对小兔。
  假定在不发生死亡的情况下，由一对出生的小兔开始，50個月后会有多少对兔子？

- 當n＞1時，Fn+2 = Fn+1 + Fn，而 F0=F1=1。

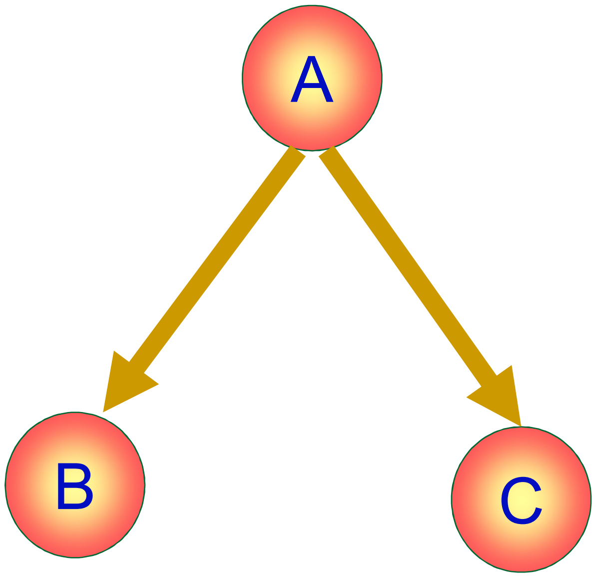
一、与或图的基本概念

1、与图、或图、与或图



1）与图

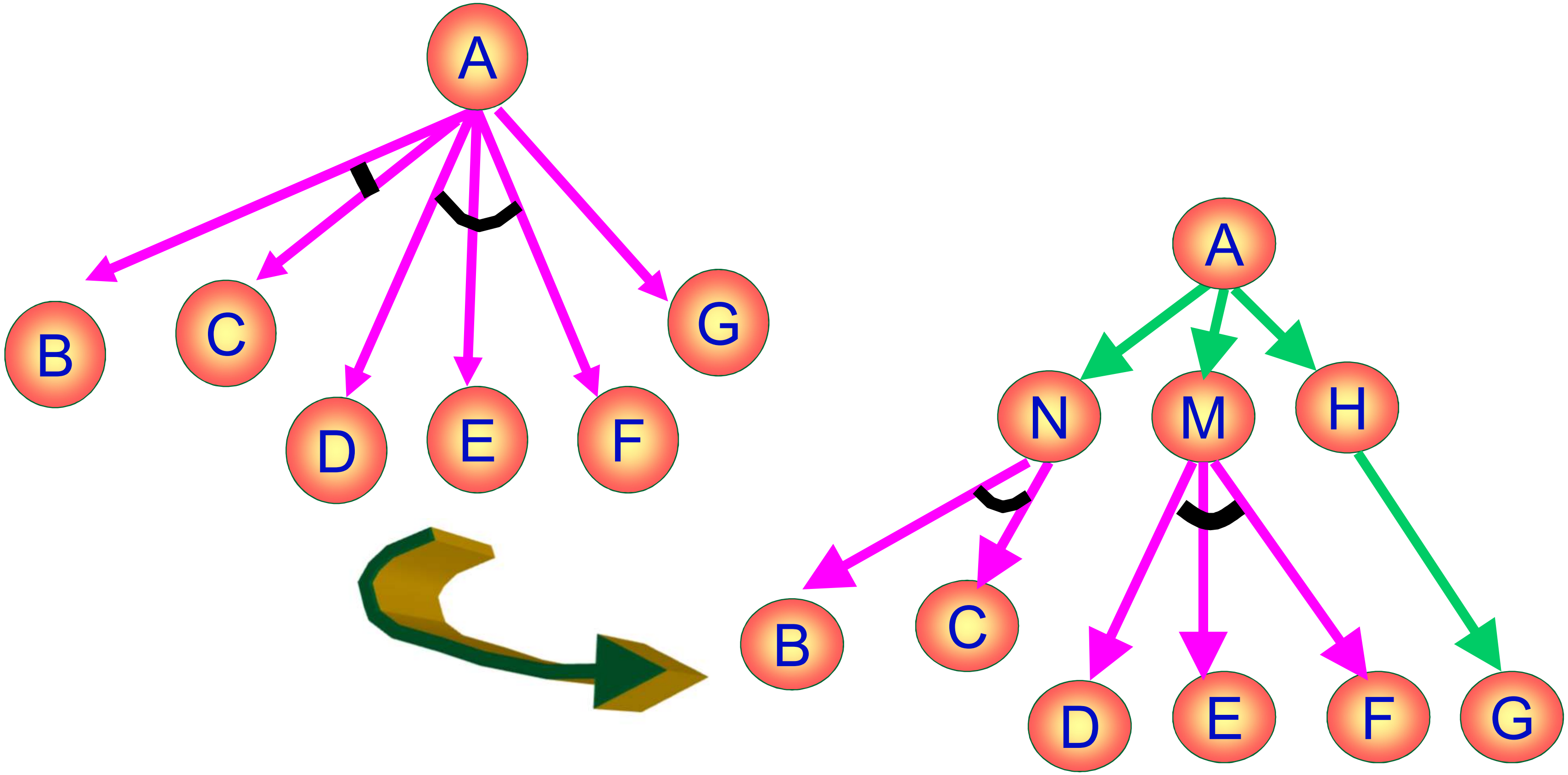
弧线：“与”关系



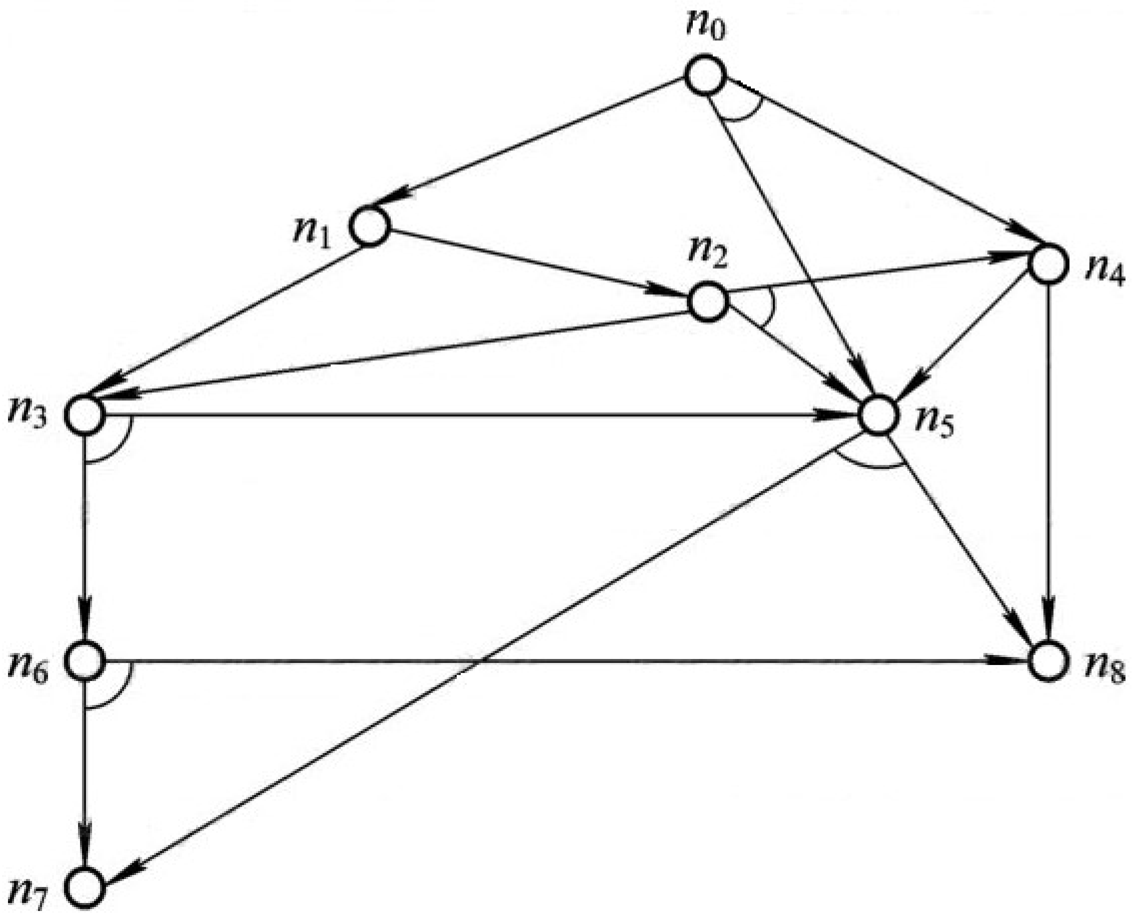
2）或图

不带弧线：“或”关系

3）与或图——既有“与”关系又有“或”关系



典型的与或图



与或图中的一些术语

1）与节点：

子节点的关系为与关系的节点

2）或节点：

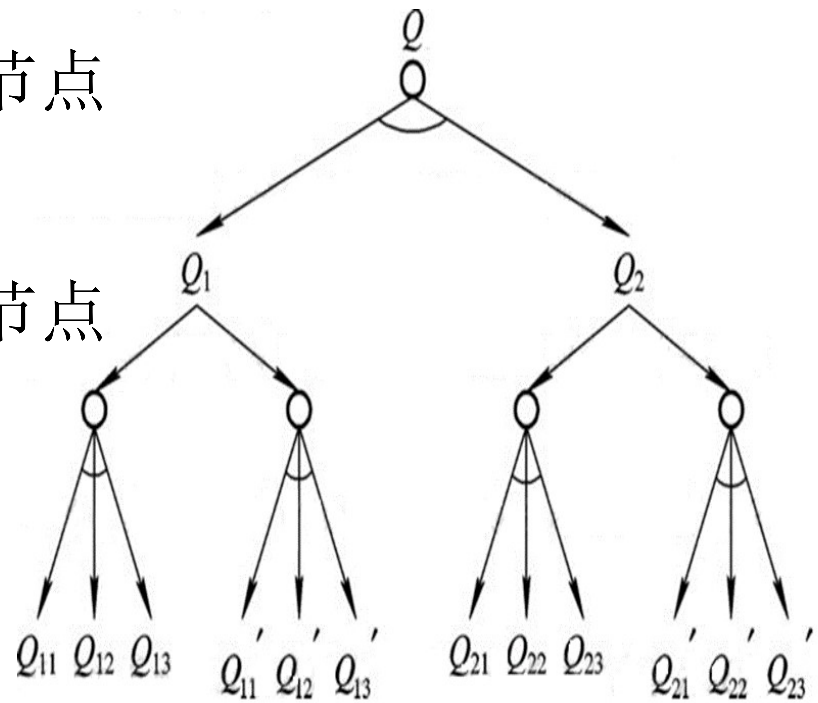
子节点的关系为或关系的节点

3）端节点（叶节点）：

无子节点的节点

4）终止节点：

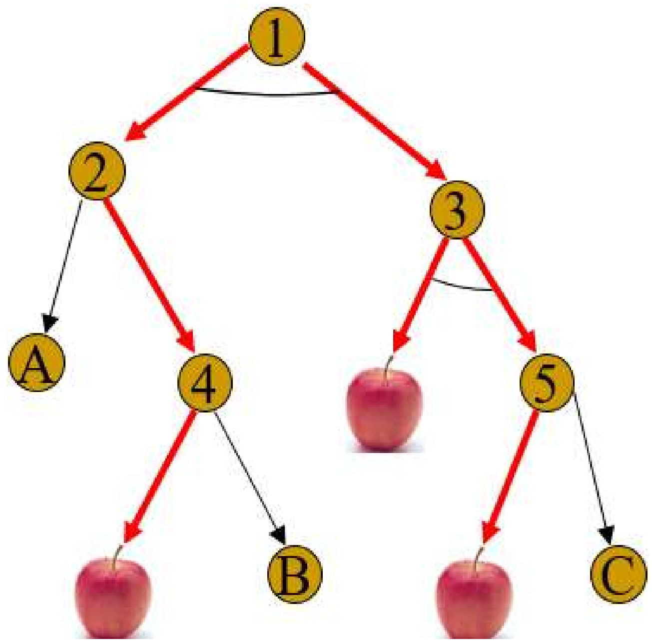
本原问题对应的节点



5) 可解节点：

在与或树中，满足以下三个条件之一的节点称为可解节点：

- 任何终止节点都是可解节点。
- 对“或”节点，当其有一个(或以上)的子节点为可解节点时，则该或节点就是可解节点。
- 对“与”节点，只有当其子节点全部为可解节点时，该与节点才是可解节点。



6）不可解节点：

不满足可解节点中给出的全部条件的节点为不可解节点。

7）可解标示过程：

由可解子节点来确定其父节点、祖父节点等为可解节点的过程。

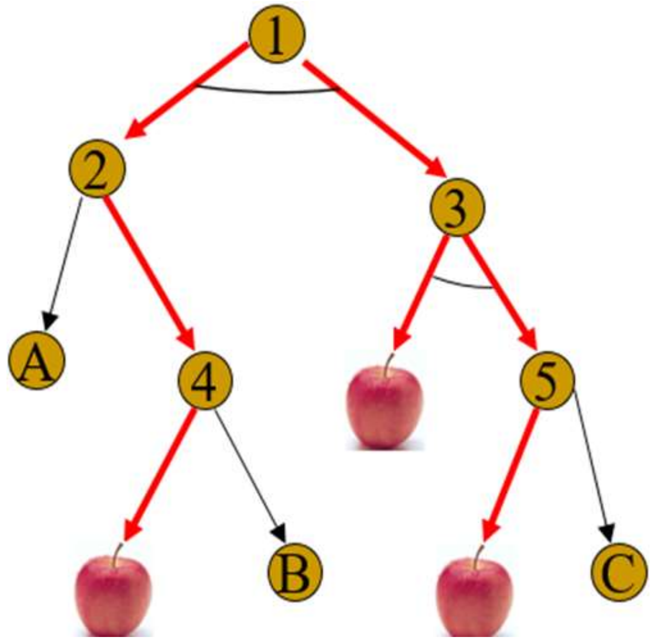
8）不可解标示过程：

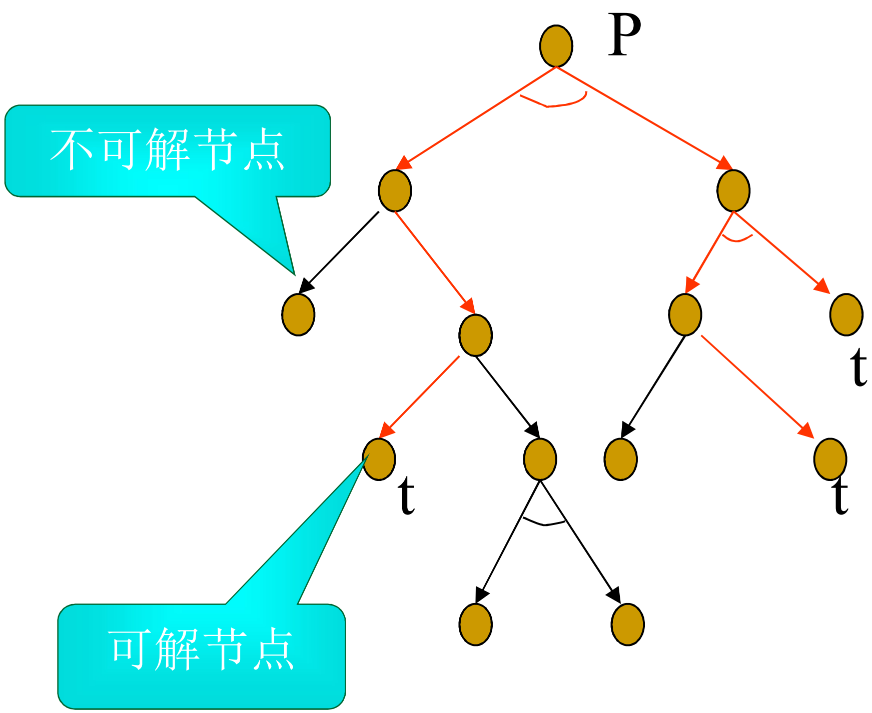
由不可解子节点来确定其父节点、祖父节点等为不可解节点的过程。

9）解树：

由可解节点构成，并且由这些可解节点可以推出初始节点（对应原始问题）为可解节点的子树为解树。

解树中一定包含初始节点。





解树

例：如图给出的与或树中，节点P为原始问题节点，用t标出的节点是终止节点。

用红线表示的子树是一个解树。

问题归约求解过程实际上就是生成解树，即证明原始节点是可解节点的过程。这一过程涉及到搜索的问题。

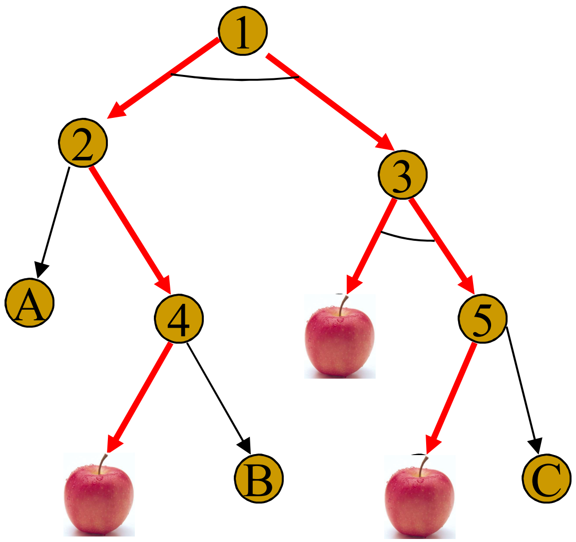
二、与或树的搜索过程

- 与或树的搜索过程实际上是一个不断寻找解树的过程。其一般搜索过程为：
  - (1) 把原始问题（初始节点）作为当前节点；
  - (2) 应用分解或等价变换操作对当前节点进行扩展；为每个子节点设置指向父节点的指针；
  - (3) 选择合适的子节点作为当前节点，返回执行第(2)步，并调用可解标记过程或不可解标记过程，直到初始节点被标记为可解节点或不可解节点。
- 上述搜索过程将形成一颗与或树，这种由搜索过程所形成的与或树称为搜索树。

例：与或树的广度优先搜索

设有如图所示的与或树，节点按图中所标记的顺序号进行扩展，其中标有 t_1，t_2，t_3 的节点是终止节点，A、B、C为不可解的叶节点，

搜索过程：



与或树的广度优先搜索

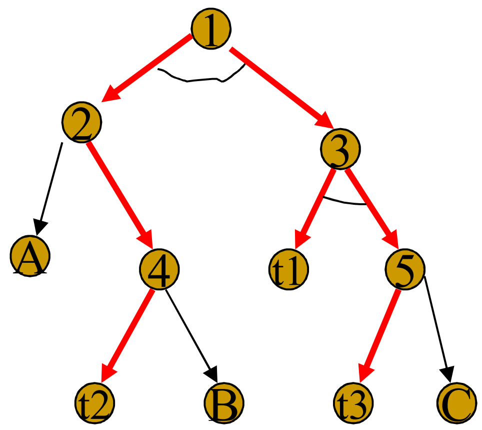
（1）先扩展1号节点，生成2号和3号节点，由于这两个节点都不是终止节点，因此接着扩展2号节点，此时OPEN表中只剩下3号节点。

状态	返回指针
1	

状态	返回指针
2	**1**
3	**1**

状态	返回指针
3	**1**

OPEN表的动态变化过程

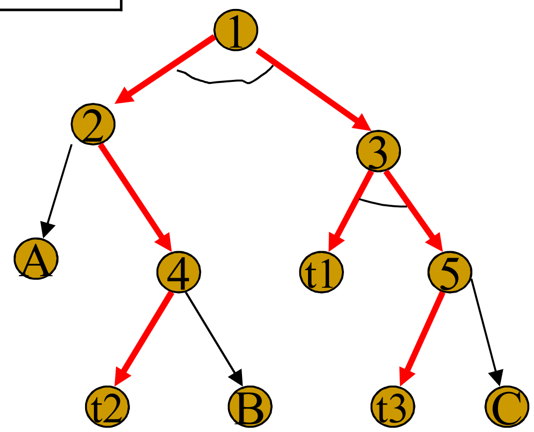


（2）扩展2号节点，生成A节点和4号节点。由于这两个节点均不是终止节点，因此接着扩展3号节点，此时OPEN表中剩下A节点和4号节点。

状态	返回指针
3	1
A	2
4	2

状态	返回指针
A	2
4	2

OPEN表的动态变化过程

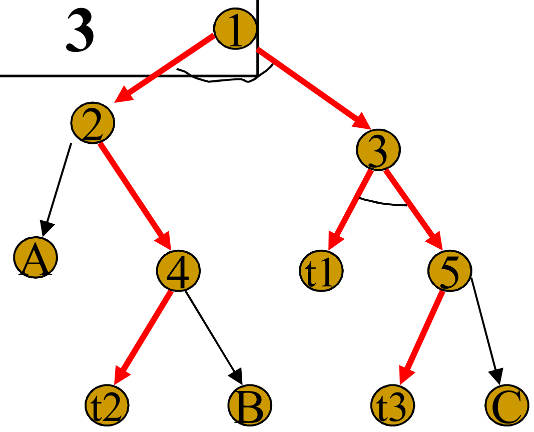


（3）扩展3号节点，生成t1和5号节点。由于t1是终止节点，则标记为可解节点，并应用可解标记过程，对其先辈中的可解节点进行标记，由于t1的父节点3号节点是一个与节点，因此不能确定节点是否可解。下一步扩展A节点，此时OPEN表中剩下4号节点、t1节点和5号节点。

状态	返回指针
A	**2**
4	**2**
t1	**3**
5	**3**

→

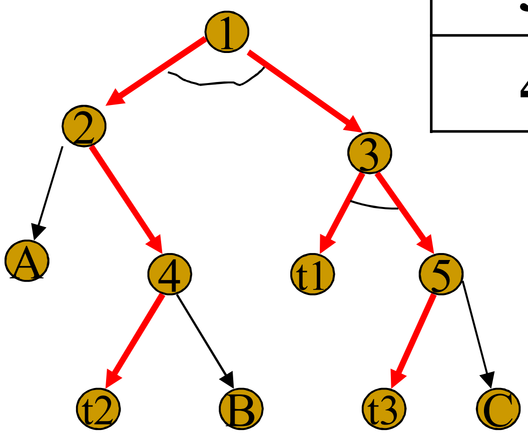
状态	返回指针
4	**2**
t1	**3**
5	**3**



OPEN表的动态变化过程

CLOSED 表的内容

编号	状态	可解标记	返回指针
1	1		
2	2		1
3	3		1
4	A	N	2

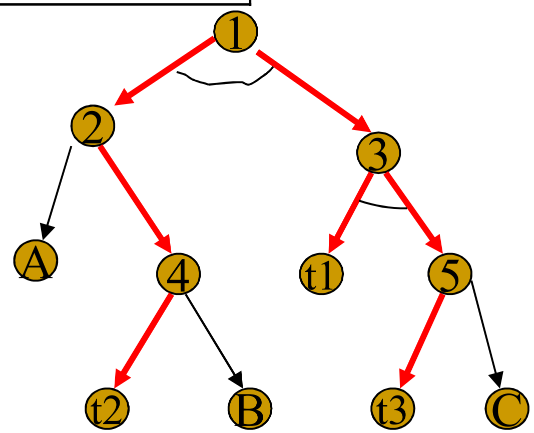


（4）扩展A节点，由于A是端节点，因此不可扩展。调用不可解标记过程，由于2号节点是或节点，因此不能确定2号节点是不可解节点。下一步扩展4号节点，此时OPEN表中剩下t1节点和5号节点。

状态	返回指针
4	**2**
t1	**3**
5	**3**

状态	返回指针
t1	**3**
5	**3**

OPEN表的动态变化过程



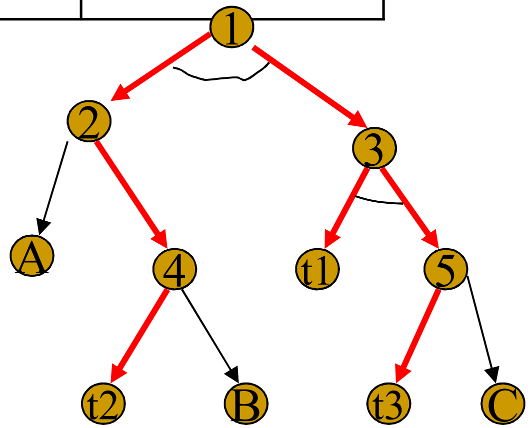
（5）扩展4号节点，生成t2和B节点。由于t2是终止节点，则标记它为可解节点，并应用可解标记过程对其先辈中的可解节点进行标记，由于4号节点是一个或节点，因此可标记它为可解节点(从OPEN表中删除B节点）。继续向上，可标记2号节点为可解节点，但不能标记1号节点为可解节点。下一步扩展5号节点。

状态	返回指针
t1	**3**
5	**3**
t2	**4**
B	**4**

状态	返回指针
5	**3**
t2	**4**

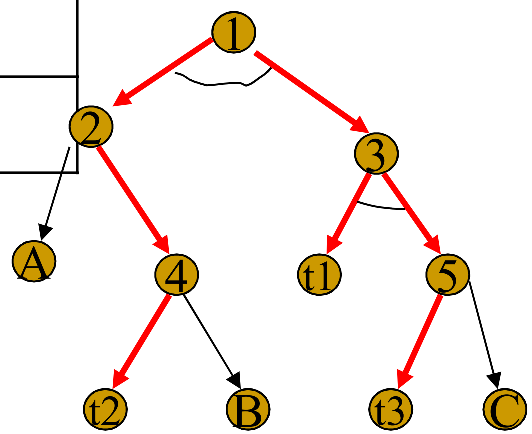
状态	返回指针
t2	**4**

OPEN表的动态变化过程



CLOSED 表的内容

编号	状态	可解标记	返回指针
1	1		
2	2	Y	1
3	3		1
4	A	N	2
5	4	Y	2
6	t1	Y	3
7	5		3

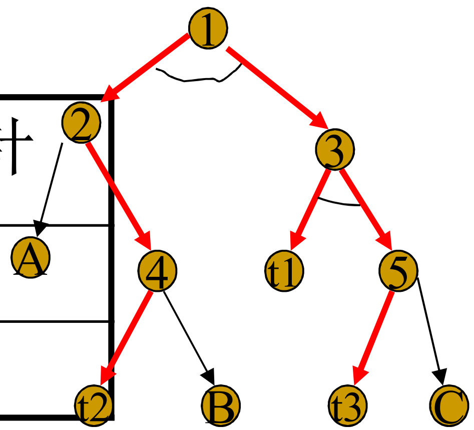


（6）扩展5号节点，生成t3和C节点。由于t3为终止节点，标记它为可解节点，并应用可解标记过程，对其先辈中的可解节点进行标记，由于5号节点是一个或节点，因此可标记它为可解节点(从OPEN表中删除节点C）。继续向上，可标记3号节点为可解节点。由于2号节点和3号节点都为可解节点，因此可标记1号节点为可解节点。

状态	返回指针
t2	**4**
t3	**5**
C	**5**

→

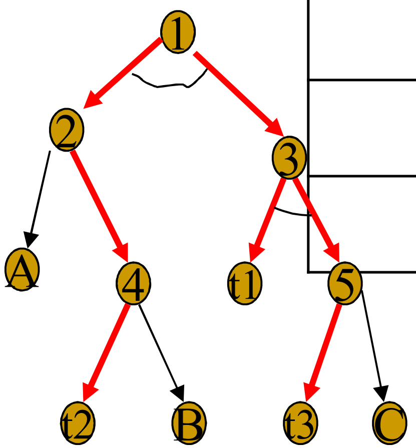
状态	返回指针
t2	**4**
t3	**5**



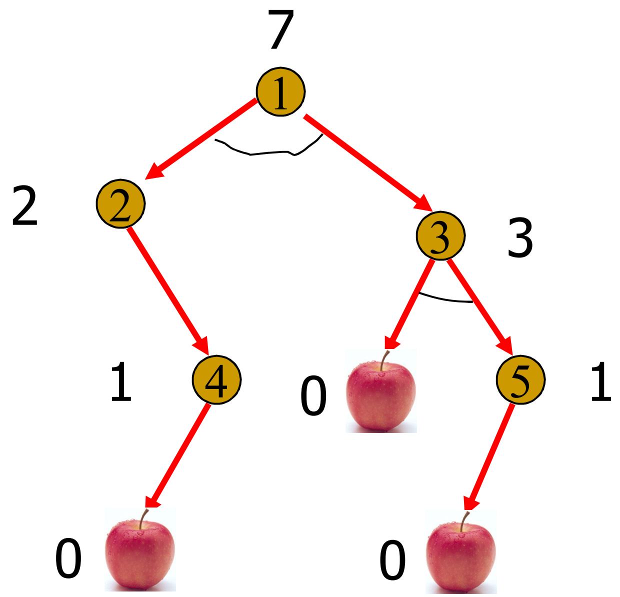
OPEN表的动态变化过程

CLOSED 表的内容

编号	状态	可解标记	返回指针
1	1	Y	
2	2	Y	1
3	3	Y	1
4	A	N	2
5	4	Y	2
6	t1	Y	3
7	5	Y	3
8	t2	Y	4
9	t3	Y	5



（7）搜索成功，得到1、2、3、4、5号节点及t_1，t_2，t_3 节点构成的解树，该解树如图中的红线所示。



与或树的广度优先搜索解树

三、启发式与或树搜索

- 盲目搜索的共同点
  - 搜索从初始节点开始,先自上而下地进行搜索,寻找终止节点及端节点，然后再自下而上地进行可解性标记,一旦初始节点被标记为可解节点或不可解节点，搜索就不再继续进行；
  - 搜索都是按确定路线进行的,当要选择一个节点进行扩展时,只是根据节点在与或树中所处的位置,而没有考虑要付出的代价,因而求得的解树不一定是代价最小的解树, 即不一定是最优解树。
- 与或树的有序搜索：根据代价决定搜索路线的方法

1. 解树的代价

- 即树根（初始节点）的代价，从树叶开始自下而上逐层计算而求得。
  - 计算节点代价
  - 推出父节点代价，直到找到初始节点s_0的代价
  - s_0的代价即为解树的代价

节点x的代价$g(x)$

- 若x是终止节点, $g(x)=0$
- 对非终止的端节点x, $g(x)=\infty$
- 若x是或节点, $g(x)=\min\limits_{1\le i\le n}\{c(x,y_i)+g(y_i)\}$
- 若x是与节点, 则

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

  ② 最大代价法: $g(x)=\max\limits_{1\le i\le n}\{c(x,y_i)+g(y_i)\}$

其中y_i为x的子节点，c（x，y_i）为x到y_i的代价。

2. 希望树

- 有序搜索的目的是求出最优解树，每次选择扩展节点时，都应选择最有希望成为最优解树的一部分的节点进行扩展——希望树。
- 希望树的定义：
  - 初始节点在希望树T中（ T是对最优树近根部分的某种估计）。
  - 如果节点n在希望树T中，则一定有：

① 如果n是具有子节点$m_1, m_2, \ldots, m_n$的“或”节点，则具有值 $\min_{1\le i\le l}\{c(n,m_i)+g(m_i)\}$ 的那个子节点m_i也应在T中。

② 如果n是“与”节点，则它的全部子节点都应在T中。

3. 与或树的有序搜索

- 与或树的有序搜索过程是一个不断选择、修正希望树的过程。
- 若问题有解，则经有序搜索将找到最优解树。
- 与或树的有序搜索过程：

1） 把初始节点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g求出以S_0为根的希望树T。

3） 依次把OPEN表中T的端节点n选出放入CLOSED表中。

4） 若n是终止节点，则：

- 标示n为可解节点。 对T应用可解标记过程，把n的先辈节点中的可解节点都标记为可解节点。
- 若初始节点S_0能被标记为可解节点，则T就是最优解树,成功退出。否则，从OPEN表中删去具有可解先辈的所有节点。

5）若n不是终止节点，且它不可扩展，则：

- 标示n为不可解节点。对T应用不可解标记过程，把n的先辈节点中的不可解节点都标记为不可解节点。
- 若初始节点S_0也被标记为不可解节点，则失败退出。
- 否则，从OPEN表中删去具有不可解先辈的所有节点。

6） 若n不是终止节点，但它可扩展，则：

- 扩展节点n，产生n的所有子节点。
- 把这些子节点都放入OPEN表中，并为每一个子节点配置指向父节点(节点n)的指针。
- 计算这些子节点的g值及其先辈节点的g值。

7） 转2）步

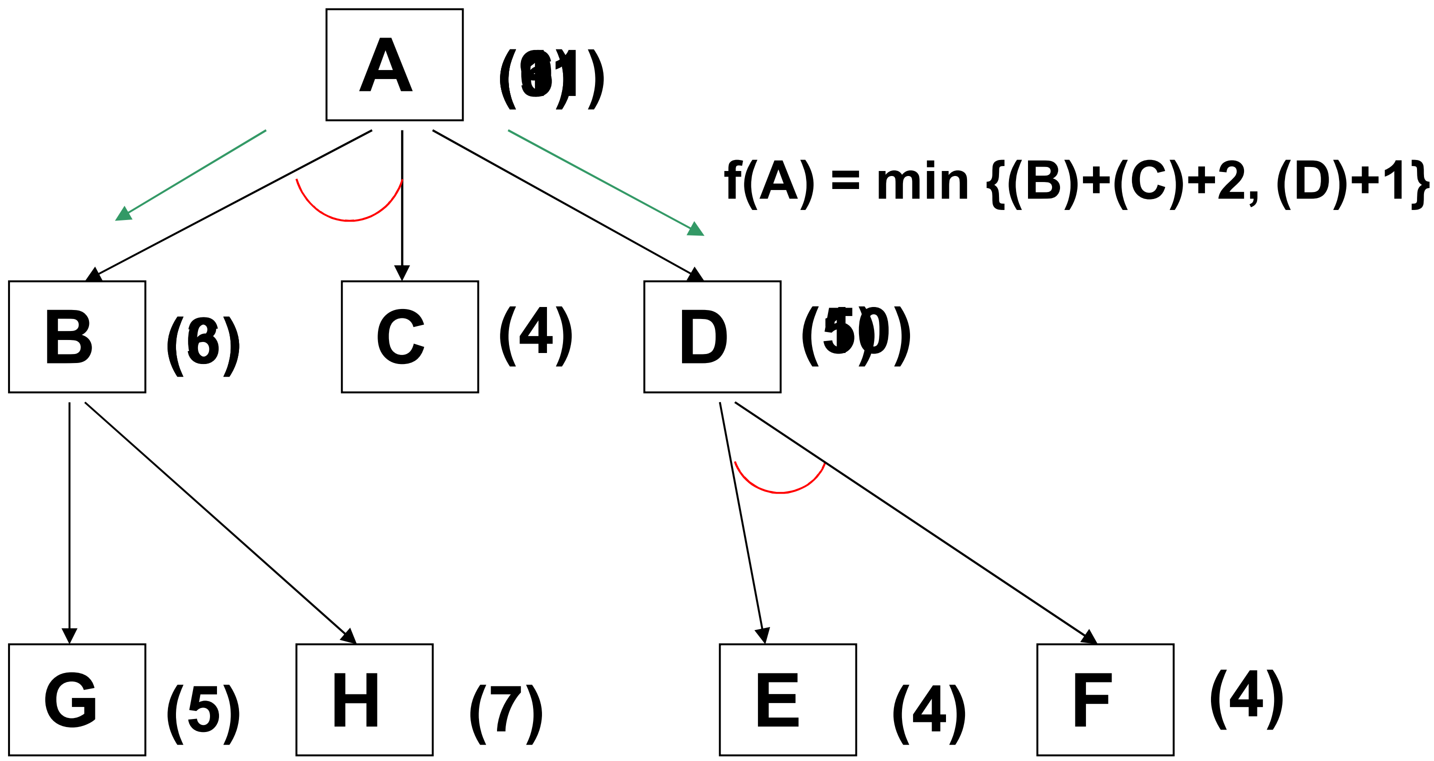
AO* algorithm

- for And/Or Graph search
- Similar with A* algorithm, but the evaluation of the node is the function of the sub-graph, instead of the path.

- Two stages

  1.Top-down graph generation

  2. Bottom-up node evaluation



node Expansion and Evaluation

Q & A