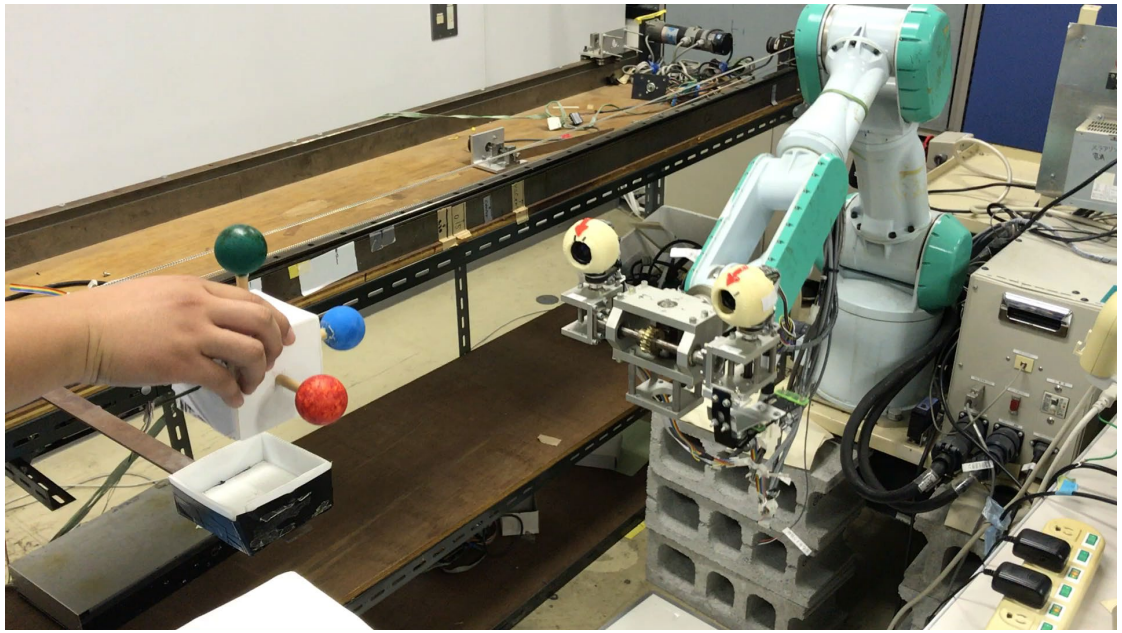
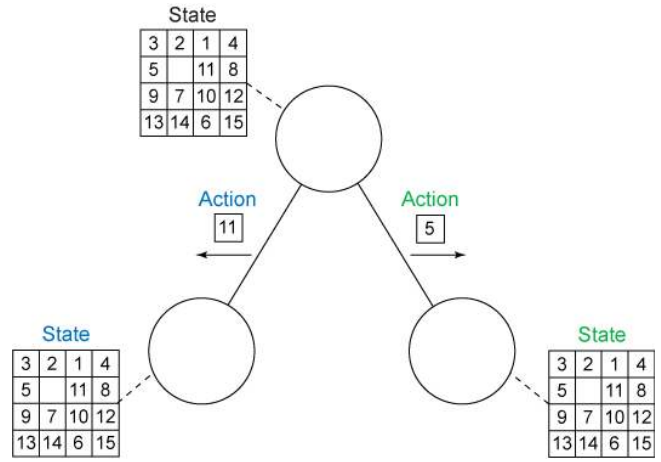
搜索过程三大要素——①搜索对象

- 搜索对象：在什么之上进行搜索
- **状态**（state）：对问题求解时某时刻进展情况的数学描述，也可以说是一个可能的解的表示。
- 一般来讲：状态是为描述某些不同事物间的差别而引入的一组**最少变量**的有序集合：$Q=（q_0,q_1,q_2,\cdots,q_n）$其中，每个元素$q_i$称为**状态变量**。**给定每个分量的一组值，就得到一个具体的状态**。



搜索过程三大要素——①搜索对象

- **状态空间：**问题的状态空间（state space）是一个表示该问题**全部可能状态及其关系**的集合。有离散和连续两种，由于真正的连续空间问题难以在计算机中表示，因此一般讨论的状态空间为**离散状态空间**。



- 它包含三种类型的集合：

① 该问题所有可能的**初始状态集合S**

② **操作符集合F**

③ **目标状态集合G**

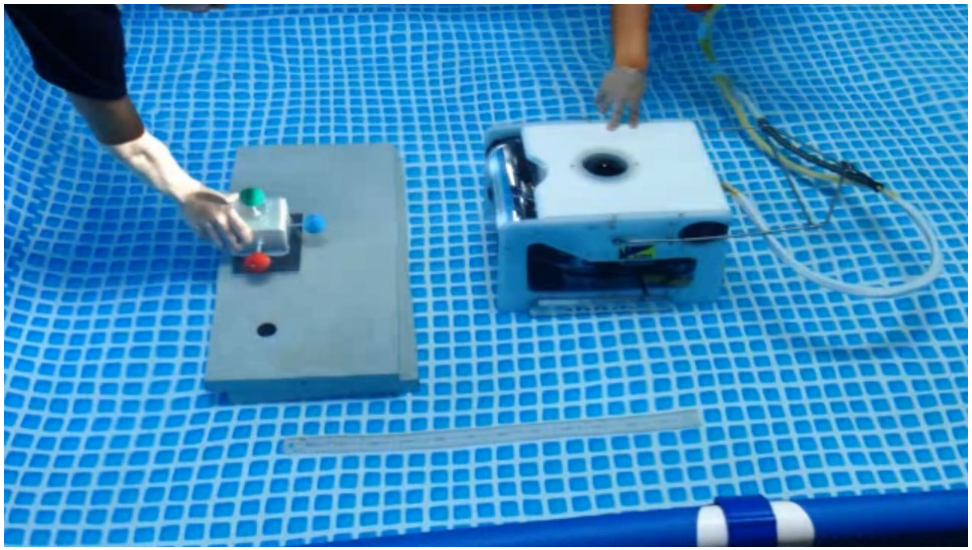
因此，可把状态空间记为**三元组（S，F，G）**。

- 状态空间通常以**图**的形式出现，图上的**节点**代表对应问题的**状态**，节点之间的**边**对应的是状态转移的**可能性**，边上的**权值**代表转移所需的**代价**。

搜索过程三大要素——②拓展规则

- 拓展规则：由**控制策略**和**生成系统**两部分组成。
  - 控制策略：包括**节点扩展顺序选择**，**算子选择**，**数据维护**，**搜索回路判定**，**目标测试**等。
  - 生成系统：由**约束条件**和**算子**组成。

- 几乎所有搜索算法的改进都是通过**修改或优化控制结构**来实现的，其中遗传算法中对算子的改进比较特别，而从遗传算法中有衍生处很多算法。



搜索过程三大要素——②拓展规则

- 算子：使问题从**一种状态变化为另一种状态的手段**称为**操作符**或**算子（operator）**。算子可能是某种动作，过程，规划，数学算子，运算符等。

◆设计合理的扩展节点策略：从宏观角度看，正确选择搜索次序是搜索的技巧，也是智能的体现。**好的策略比一般的方法扩展的节点少，设计更合理的策略可以提高搜索速度**。

◆避免搜索回路：搜索的对象是图，如果这个图中存在环，而且没有很强的扩展方法避免环的话，就必须有一个手段**避免搜索进入死循环**。

◆维护数据结构：在扩展节点，判断重要性等时，**数据结构的好坏决定了这些操作的效率**。在搜索中很有必要牺牲一些相对较少的时间对这些数据进行维护，以便**更快的获取数据**。

基本思想

- 通过搜索求解问题的**前提是凭借人类自身智能可以解决**，在搜索之前应对问题有充分的认识后再考虑选用合适的搜索算法。

- 搜索求解问题的**基本思想**：

① 将问题中的**已知条件**看成状态空间中**初始状态**；将问题中**要求的目标**看成状态空间中**目标状态**；将问题中**其它可能的情况**看成状态空间的**任一状态**

② 设法在状态空间寻找一条**路径**，**由初始状态出发，能够沿着这条路径达到目标状态**

基本步骤

- 搜索求解问题的**基本步骤**：

① 根据问题，**定义出相应的状态空间**，**确定出状态的一般表示**，它含有相关对象的各种可能的排列。这里仅仅是定义这个空间的状态，而**不必枚举该状态空间的所有状态**，但由此可以得出问题的**初始状态**、**目标状态**，并能够给出所有**其它状态的一般表示**。

② 规定一组**算子**，能够使状态**从一个状态变为另一个状态**。

③ 决定一种**搜索策略**，使得能够**从初始状态出发，沿某个路径达到目标状态**。

例题：水壶问题

- 给定两个水壶，一个可装**4升水**，一个能装**3升水**。
- 水壶上**没有任何度量标记**。
- 可以使用水龙头向壶中灌水。

- 问：怎样在能装**4升的水壶里恰好只装2升水**？

例题：水壶问题

- 求解过程

① 定义状态空间：

- 可将问题进行抽象，用**数偶(x, y)**表示状态空间的任一状态。
- x—表示4升水壶中所装的水量，x=0，1，2，3或4；
  y—表示3升水壶中所装的水量，y=0，1，2或3；

- 初始状态为(0, 0)，目标状态为(2, ?)
- ？表示水量不限，因为问题中未规定在3升水壶里装多少水。

例题：水壶问题

② 确定一组算子：**水龙头灌水/将水倒掉/两壶互相倒水**
可以用以下8条规则来描述

r_1：（X，Y|X<4）→（4，Y） 当4升壶不满时，将其灌满

r_2：（X，Y|Y<3）→（X，3） 当3升壶不满时，将其灌满

r_3：（X，Y|X>0）→（0，Y） 将4升壶中水全部倒掉

r_4：（X，Y|Y>0）→（X，0） 将3升壶中水全部倒掉

例题：水壶问题

② 确定一组算子：**水龙头灌水/将水倒掉/两壶互相倒水**

r_5：（X，Y|X+Y≥4∧Y>0）→（4，Y−(4−X)）

将3升壶中水倒向4升壶，直到4升壶水满

r_6：（X，Y|X+Y≥3∧X>0）→（X−(3−Y)，3）

将4升壶中水倒向3升壶，直到3升壶水满

r_7：（X，Y| X+Y≤4∧Y>0）→（X+Y，0）

将3升壶中水全部倒入4升壶

r_8：（X，Y| X+Y≤3∧X>0 ）→（0，X+Y）

将4升壶中水全部倒入3升壶

例题：水壶问题

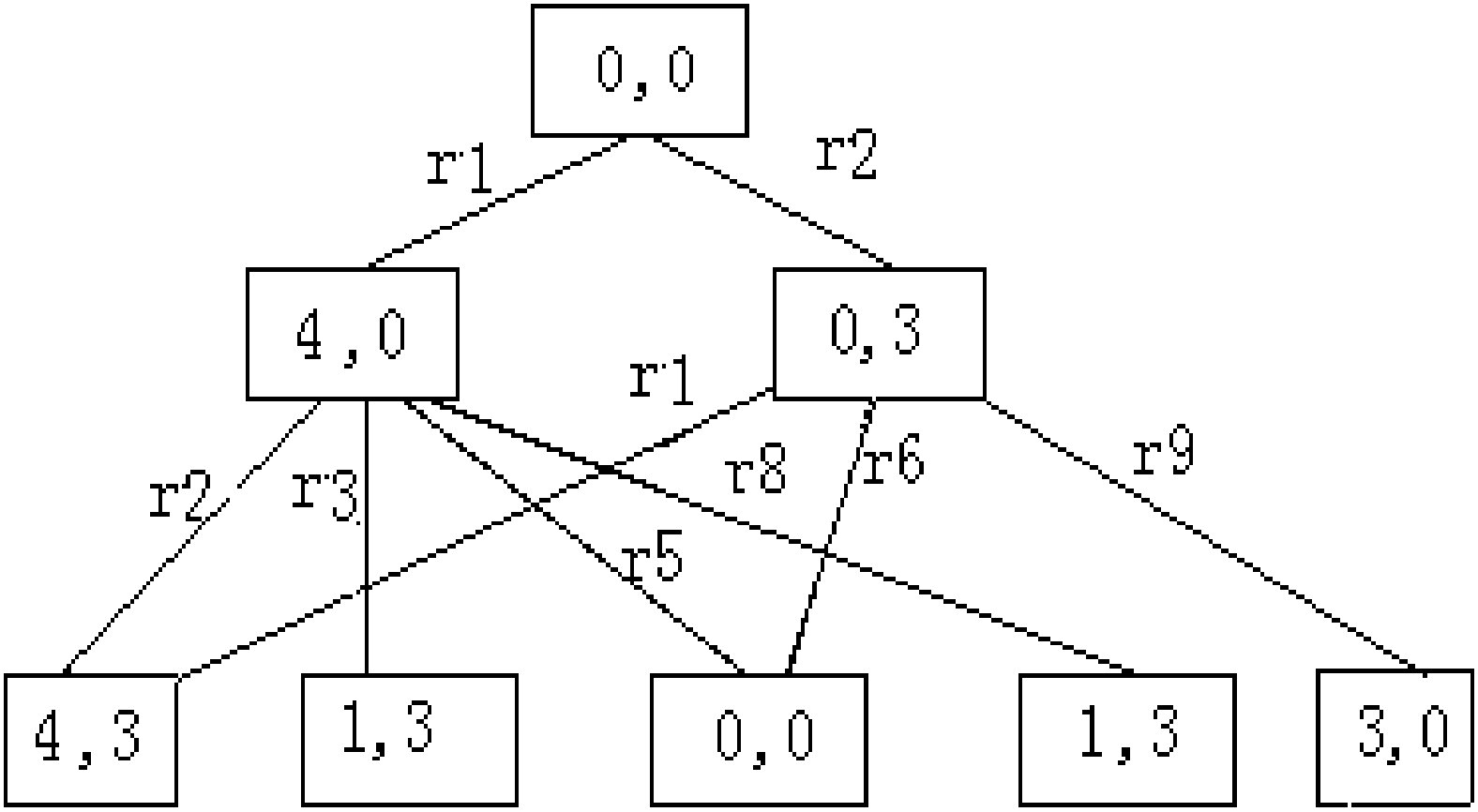
③ 选择搜索策略：

该策略为一个简单的**循环控制结构**：选择其**左部匹配当前状态**的某条规则，并按照该规则**右部的行为对此状态作适当改变**，然后**检查改变后的状态**是否为某一目标状态，若不是，则继续该循环。

- 这样循环搜索下去，直到出现(2, ?)的状态为止
- 从（0，0）到（2，？）的路径上所用的操作序列就是所求的解
- 有多种算子序列都是水壶问题的解

例题：水壶问题

- 水壶问题搜索图（部分）



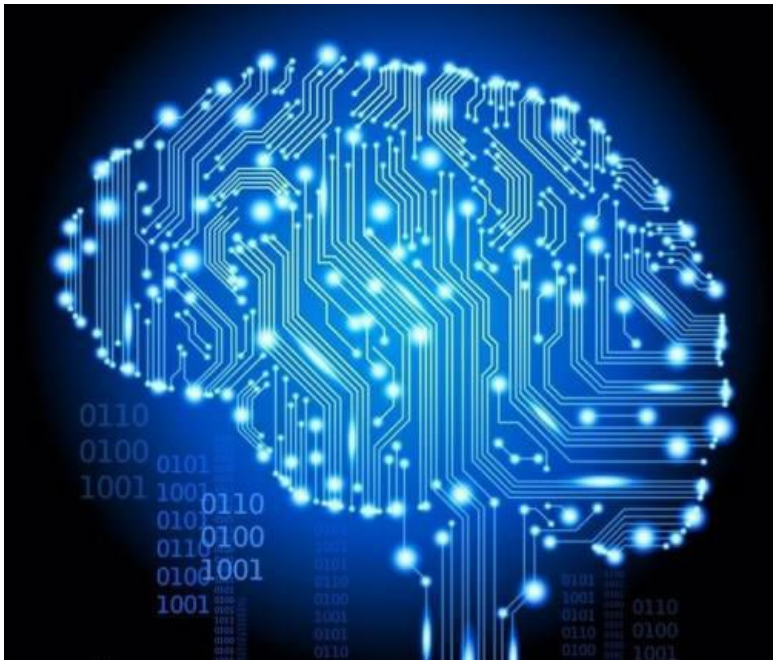
例题：水壶问题

- 水壶问题其中一条搜索路径

4升壶里的水	3升壶里的水	应用规则
0	0	初始状态
0	3	r_2
3	0	r_7
3	3	r_2
4	2	r_5
0	2	r_3
2	0	r_7

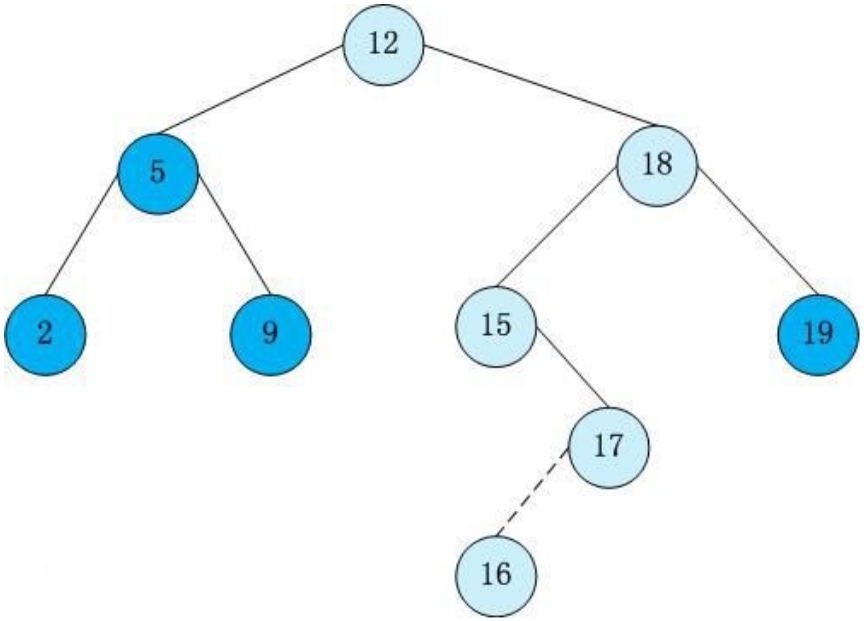
盲目搜索方法

- **盲目搜索**方法又叫**非启发式搜索**，是一种**无信息搜索**（uninformed search），一般只**适用于求解比较简单的问题**。下面我们要讨论的几个搜索方法，它们均属于盲目搜索方法。



宽度优先搜索

- 在一个**搜索树**中，如果搜索是以**同层邻近节点依次扩展**节点的，那么这种搜索就叫**宽度优先搜索**（breath-first search），这种搜索是**逐层**进行的，在对下一层的任一节点进行搜索之前，**必须搜索完本层的所有节点**



宽度优先搜索

- 宽度优先搜索算法如下：

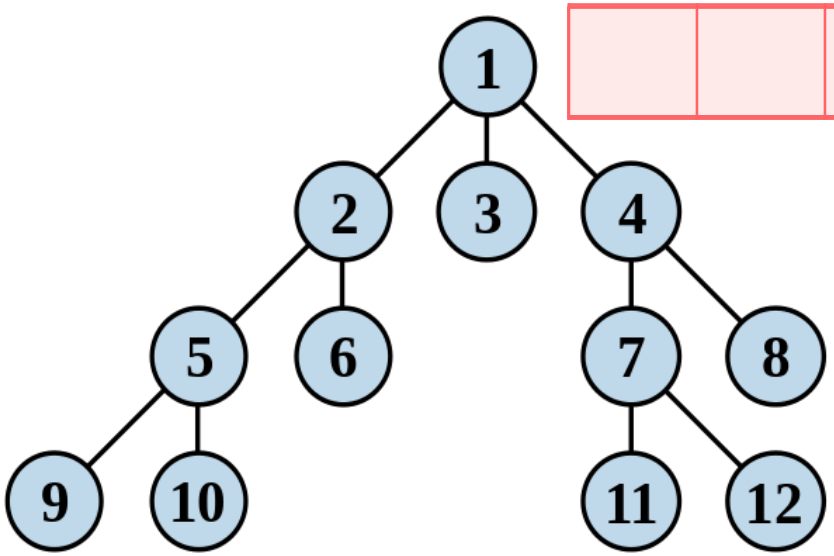
① 令N为一个由**初始状态**构成的**表**

② 若N为**空退出**，标志**失败**

③ 令n为N中**第一个点**，将n从N中**删除**

④ 若n**是目标**，则退出，标志**成功**

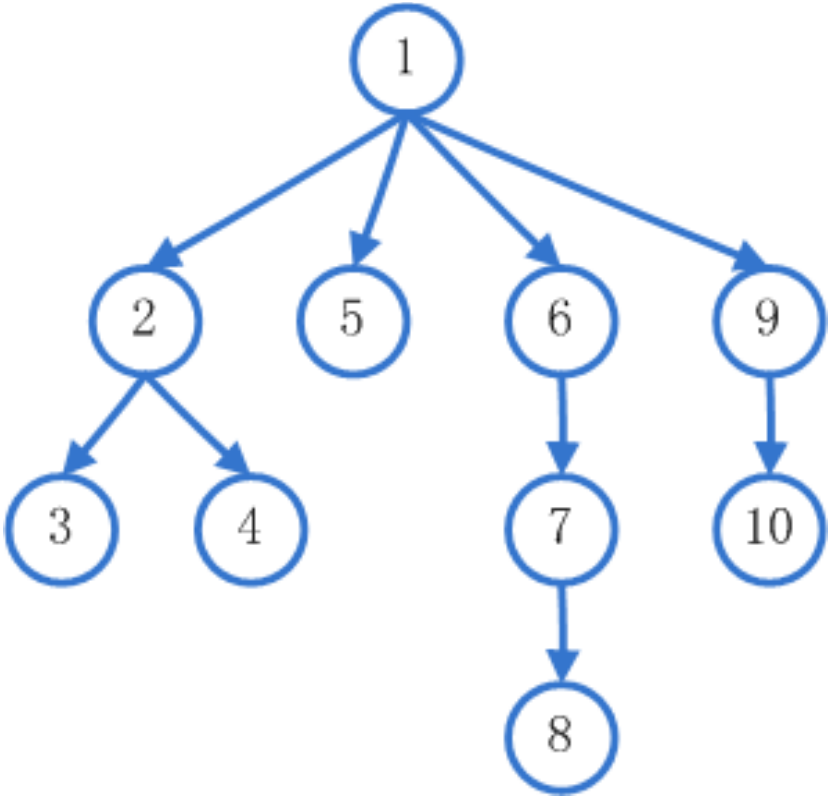
⑤ 若n**不是目标**，将n的**后继点**加入到N表的**尾部**，转②



- **优点**：若问题**有解**，则**可找出最优解**
- **缺点**：**效率低**，组合爆炸问题难以解决

深度优先搜索

- 与宽度优先搜索对应的一种盲目搜索方法叫做**深度优先搜索**（depth-first search）。在深度优先搜索中，我们**首先扩展最新产生的(即最深的)节点。深度相等的节点可以任意排列**。



深度优先搜索

- 深度优先搜索算法如下：

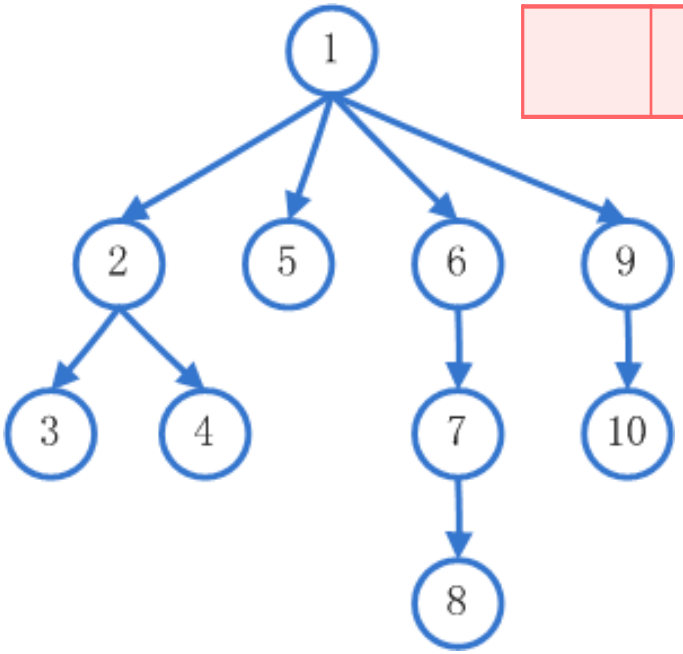
① 令N为一个由**初始状态**构成的**表**

② 若N为**空退出**，标志**失败**

③ 令n为N中**第一个点**，将n从N中**删除**

④ 若n**是目标**，则退出，标态**成功**

⑤ 若n**不是目标**，将n的**后继点**加入到N表的**首部**，转②

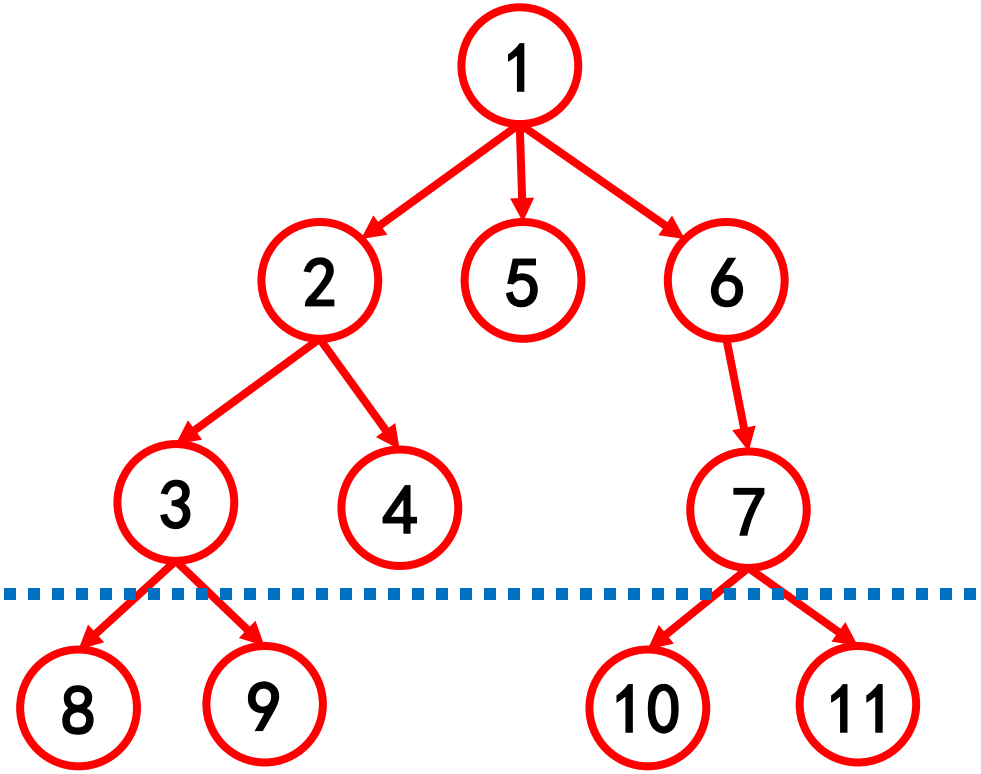


				8			9		

- **优点：节省大量时间和空间**
- **缺点：不一定能找到解，**
  在无限搜索树的情况下，最坏的情况可能是不停机。

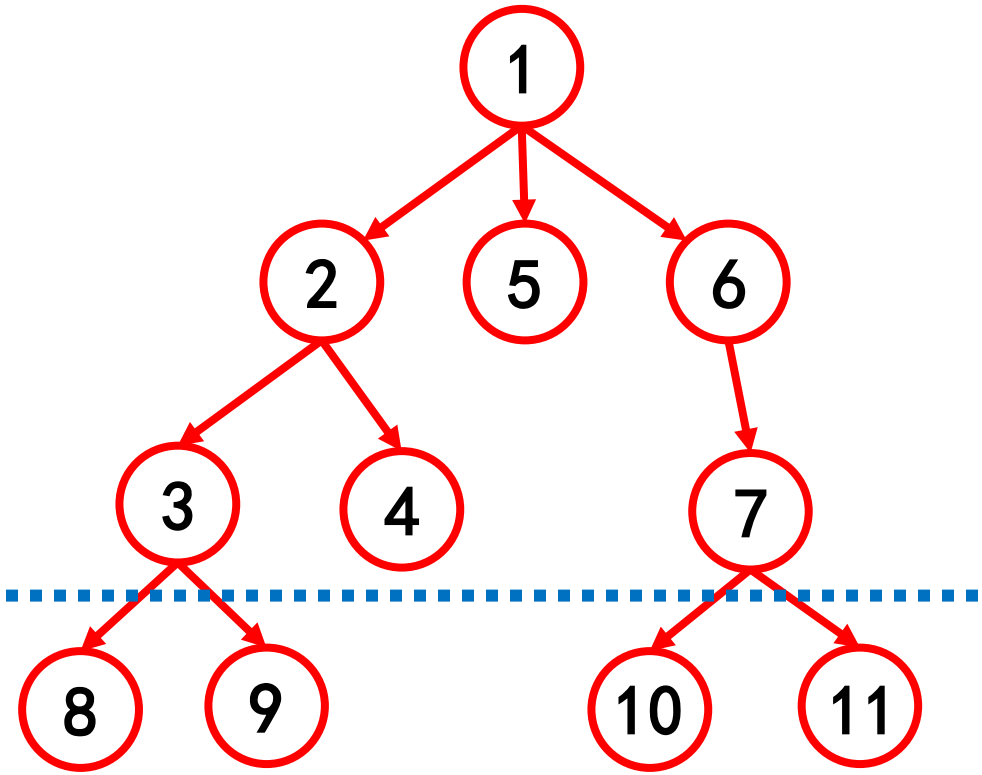
分支有界搜索

- **分枝有界搜索**（branch-and-bound）也是一种**深度优先搜索**，但**每个分支都规定了一个统一的搜索深度**，搜索到这个深度后，**如果没有找到目标便自动退回到上一层**，继续搜索。



迭代加深搜索

- **迭代加深搜索**（iterative deepening）：在**分支有界搜索**的基础上，对**界dmax进行迭代**，保证了对宽度节点的搜索，**如果没有找到解，再加深深度**。



通用或图搜索算法

- 图搜索算法：**只记录**状态空间那些**被搜索过的状态**，它们组成一个**搜索图G**。

- G由**两张表内的节点**组成：

① **Open表**：用于存放已经生成，且已用启发式函数作过估计或评价，但尚未产生它们的后继节点的那些结点，也称**未考察结点**。

② **Closed表**：用于存放已经生成，且**已考察过的结点**。

- 还有一个**辅助结构Tree**：节点为**G的一个子集**。Tree用来**存放当前已生成的搜索树**，该树由**G的反向边(反向指针)组成**

通用或图搜索算法

- 或图通用搜索算法实现过程：

设S_0为初态，S_g为目标状态

① 产生一**仅由S_0组成的Open表**

② 产生一**空的Closed表**

③ 如果open为空，失败退出

④ **在open表上**按某一原则选出第一个**优先结点，称为n，放n到closed表中**，并**从open表中去掉n**

⑤ 若**n∈Sg**，则**成功**退出。此时解为**在Tree中**沿指针从**n到s0的路径，或n本身**。

（如八皇后问题给出n即可，八数码问题要给出路径）

通用或图搜索算法

⑥ 产生n的**一切后继**，将后继中**不是n的先辈点**(前驱点)的所有点构成一个集合**M**，**将M装入G作为n的后继**，这就除掉了既是n的先辈又是n的后继的结点就**避免了回路**，节点之间有**偏序关系**存在

⑦对**M中**的元素P，分别作两类处理：

⑦-1 **若P∉G**，即P不在open表中也不在closed表中，则**P加入open表**，**同时加入搜索图G**中，对P进行估计**放入Tree中**

⑦-2 **若P∈G**，则决定是否更改Tree中P到n的指针

⑧ 转③

通用或图搜索算法

- 注：在步骤⑦-1中，若产生的后继节点放在：
  - Open表的**表尾**，算法相当于**宽度优先**
  - Open表的**表头**，算法相当于**深度优先**
  - 根据**启发式函数的值选出最佳者**后，再放在Open表的**表头**，算法相当于**最佳优先**搜索