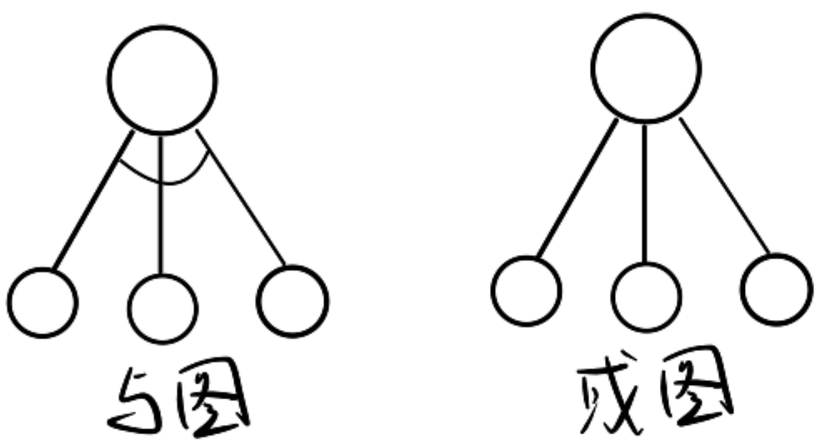
定义：将复杂问题拆分为子问题，并表达子问题间"并且/或者"逻辑的有向图

与节点：其下多个子节点必须全部有解才有解

或节点：只要一个子节点有解，本节点才有解



普通状态图等同于一个全是或节点的与或图.

用与或图求解问题，即从原问题出发，不断对问题进行分解（子问题，与节点）和变换（等价问题，或节点），而得到一个与或图.

搜索

与或图搜索需要一边扩展节点一边判断逻辑，以确定初始节点是否可解。一旦可以确定初始节点可解性搜索便停止。
在与或树中用标记法对结点自下而上进行标记。能导致根节点可解的结点组成的子树称为解树.

与或树中的叶子节点称为端节点，可被分为两类：

1. 终止节点（天然可解，称为本原问题，如数学公理）
2. 非终止节点（不可直接求解，如迷宫死胡同、无法完成的操作）

与或树中可解节点用●表示，不可解结点用×表示.

一个节点是可解节点，则必是以下情况之一

① 终止节点是可解节点

② 与节点可解当且仅当其子节点全部可解.

③或节点可解当且仅当其子节点至少一个可解.

解树的代价 F(T):

即根节点 S_0 代价。与或图中代价是从下往上逐层计算的，与树生长方向相反，计算方式如下：

①若X是终止节点，则代价 $g(X)=0$

②若x是或节点，则 $g(x)=\min\{c(x,y_i)+g(y_i)\}$ $(1\le i\le n)$

y_i 是X的子节点，$c(x,y_i)$ 为 x到 y_i 的代价.

③若X是与节点，根据问题不同有两种计算方法.

$g(x)=\sum[c(x,y_i)+g(y_i)]$ 和代价法则

$g(x)=\max[c(x,y_i)+g(y_i)]$ 最大代价法则

④若X是非终止节点，$g(x)=\infty$

希望解树T0

由于代价计算与树生长方向相反，为了在生长阶段便打展出最优解树，在每次选择扩展节点时优先选择有希望成为最优解树的节点进行扩展。这些"有希望成为最优解树"的解树称为"希望树"

定义：

①根节点 S_0 必在希望解树T0中

②若n在T0中，则必有：

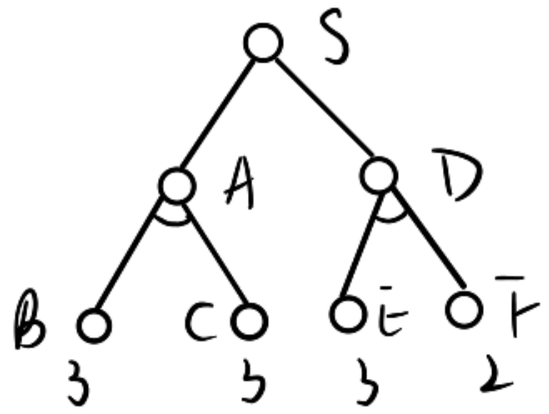
(a)若n是具有子结点集 $\{n_i \mid i=1,2,\cdots,k\}$ 的或节点，则满足 $\min[c(n,n_i)+b(n_i)]$ 的结点 n_i 必在T0中

(b)若n是与结点，则其子结点全在T0中.

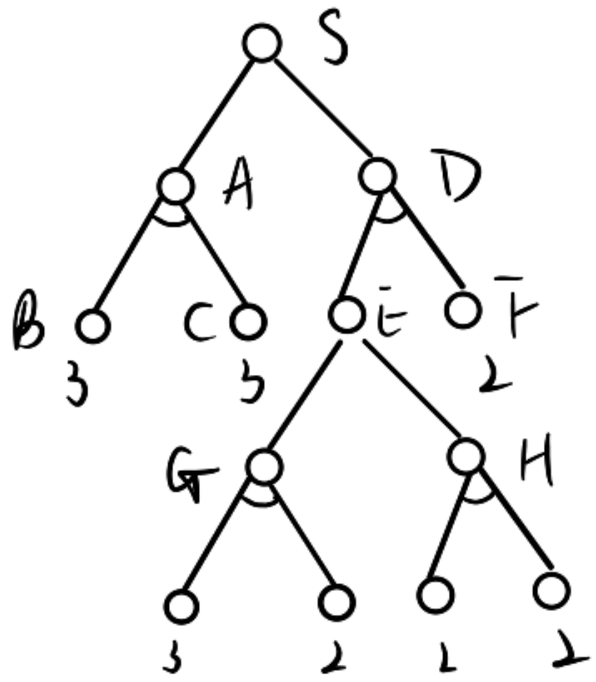
最好优先搜索

不断选择、修正希望子树，即每次选择代价最小的子树作为希望解树并扩展.

如：



右子树代价更小，选作希望解树



左子树代价更小，选作希望解树.

如此循环直到搜索结束

算法：1、将S_0放入OPEN表中

2、求出希望树T

3、依次将OPEN表中T的端节点N放入CLOSED表中.

4、若N是终止节点，则：

①标记N为可解节点

②将N的先辈节点中的可解节点标记为可解节点.

③若S_0可以被标记为可解节点，则T就是最优解树，退出

④否则，从OPEN表中删去具有可解先辈的可解节点

5、若N不是终止节点且不可扩展，则

①标示N不可解.

②将N的先辈节点中的不可解节点都标记为不可解节点

③若S_0也被标记为不可解节点，则搜索失败，退出.

④否则从OPEN表中删去有不可解先辈的所有节点.

6、若N不是终止节点，但可扩展，则：

①扩展N，产生其所有子节点

②将这些子节点放入OPEN表中，并为每个子节点分配一个指向父节点的指针

③计算这些子节点及其先辈节点的f值.

7、转2.

博弈树

双人零和、交替走步的博弈树就是一颗与或树。把我方同合称为MAX层，对方走步称为MIN层。

MAX：只要有一个子节点能赢，则当前节点必胜，即与节点

MIN：只有所有子节点都能赢，当前节点才必胜，即或节点

特点：1、博弈初始状态即初始节点

2、MAX层与MIN层交替出现

3、所有使己方获胜的终局都是本原问题，相应节点为可解节点，反之为不可解节点

我们讨论的博弈是二人零和、全信息、非偶然的，即

①双方交替行动，终局只有三种：赢/输/平

②对弈过程中双方都了解当前格局以及历史格局

③每一步都会选择对己方最有利而对对方最不利的决策

④双方利益完全对立，它们各有一个策略集D_0、D_1，每步只选择其中一个策略对弈，得失由each己的赢得函数φ_i：$D_0 \times D_1 \to R\ (i=0,1)$来决定，且有$\varphi_0 + \varphi_1 = 0$.

保险策略：立足于最坏的考虑去争取最好的结局，即

$$Max\ [\min \varphi_i(d_i, d_j)]$$

注：$d_i \in D_i$，即玩家i的一个具体策略。

$\varphi_i(d_i, d_j)$：玩家0选d_i、玩家1选d_j时，玩家i的收益。

若有 $d_0^* \in D_0$，$d_1^* \in D_1$ 使得

$$\varphi_0(d_0^*, d_1^*) = max\ [\min \varphi_0(d_0, d_1)]$$

$$\varphi_1(d_0^*, d_1^*) = max\ [\min \varphi_1(d_0, d_1)]$$

则称(d_0^*, d_1^*)为保险策略平衡解，此时双方无法单方面改变策略来更大化己方收益

极大极小分析法

又称Minimax搜索或博弈搜索，时间复杂度$O(b^m)$

- MIN结点的子结点有一个对MAX不利，则该结点对MAX不利
- MAX结点的子结点有一个对MAX有利，则该结点对MAX有利
- 认为使MAX方获胜的终局为可解端节点，使MIN方获胜的为不可解端结点。可解端结点的得分为+b，不可解为-b
- 从端结点一步步推算各结点的得分

推算：根据具体问题定义一个估价函数估算端结点得分

MAX节点：选子节点中最大的得分作为自己得分

MIN节点：选子节点中最小的得分作为自己得分

这种估值方法叫倒推估值

另外有一种常用估值方法叫静态估值，即只看当前局面而不往后看，如看象棋盘面上的子力来估值。实际运用中，由于树空间

太大，算法只会生成合理的树的一部分，用静态估值给叶节点打分，然后用倒推估值给中间节点打分

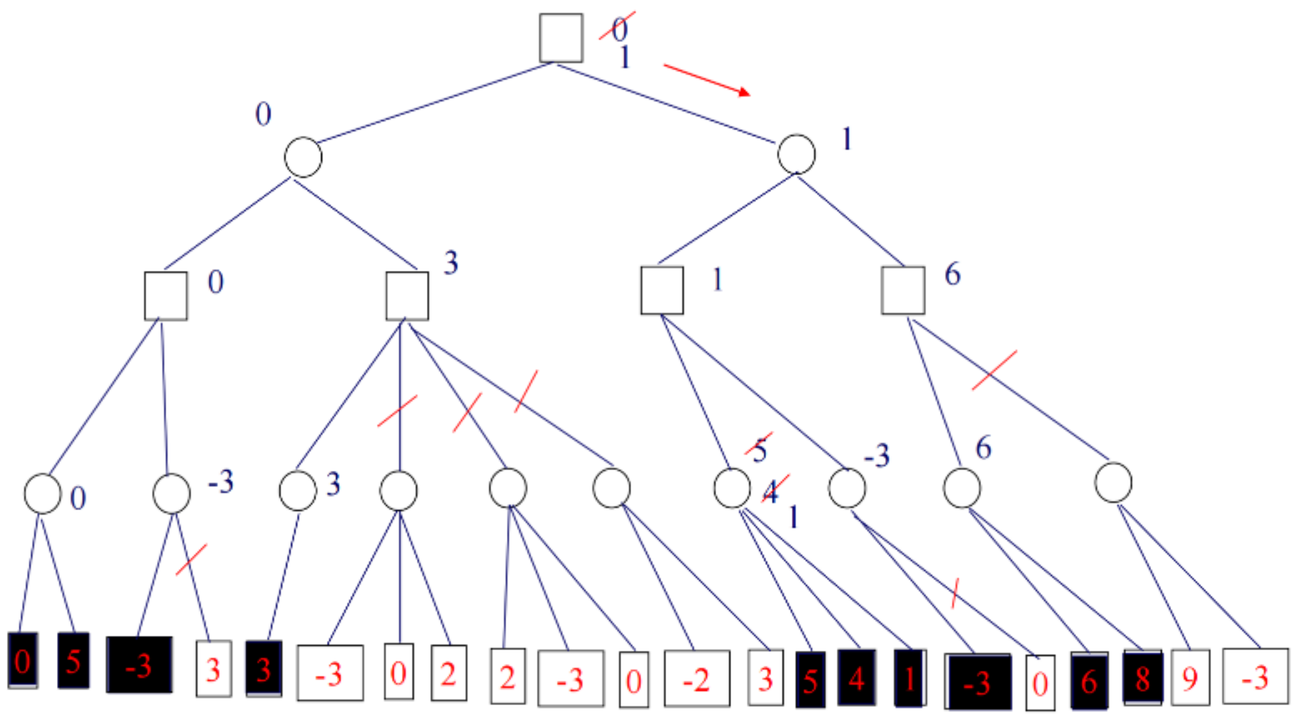
α-β剪枝

边生成树边估计各结点倒推值，并根据评估出的倒推值范围，停止扩展那些不再需要扩展的子节点

称或节点下界为α值，与节点上界为β值

α剪枝：任何与结点x的β值≤父节点α值则停止扩展x

β剪枝：任何或结点x的α值≥父节点β值则停止扩展x



最好时间复杂度：$O(b^{m/2})$

蒙特卡洛树搜索

多臂老虎机问题

有K台老虎机放在智能体面前，每次选择一台启动，每台老虎机各自有一定概率吐出一些(≥0)硬币，但智能体事先不知道这些概率。现在智能体已玩了T(T>K)次，则下一次选择哪一台能得到最多的回报？

如果用简单的贪心算法来解决这个问题，即根据几次游玩得到的硬币数量来估算一个机器得分概率，并选择概率最高的一台

进行游玩，会陷入局部最优解：如第三台老虎机概率实际上最高，但是由于前几次运气不好导致智能体估计其概率很低，转而游玩其它机器，而第三台机器不再被选择。

本质上是探索（尝试不同选择以发现潜在奖励）和利用（选择目前最优的决策以获取高奖励）之间的矛盾

UCB算法

又称上置信界算法。思想是给每个选择一个评分，评分=预估奖励概率+不确定性度量。具体来说：

$$UCB = \overline{X_j} + C \times \sqrt{\frac{2\ln n}{n_j}}$$

$\overline{X_j}$是第j个老虎机的平均奖赏值

n是选择总次数　n_j是选第j个老虎机的次数

C是平衡因子

蒙特卡洛树搜索（MCTS）

大致分为四步：

1、选择：从根节点开始向下走，直到遇到没完全扩展的节点L。
用UCB（这里称为UCT）来选择最有潜力的分支。
并记录下每个节点选择次数与得分均值。

2、扩展：若L不是终局，就随机扩展它的一个未被扩展过的后继结点M

3、模拟：从M开始，快速随机搜索（不考虑任何东西，纯随机）一直下到终局。记录模拟次数与胜利次数，估计该节点的分数

4、回溯：把模拟结果从M开始向上走，更新路径上所有节点的访问次数（+1），胜利次数（模拟赢了就加）

UCT：UCB算法在游戏树中的称呼

$$UCT(Vi,V) = \frac{Q(Vi)}{N(Vi)} + C\sqrt{\frac{\log(N(V))}{N(Vi)}}$$

N(Vi)是访问Vi次数，Q(Vi)是Vi赢的次数

V是Vi父节点