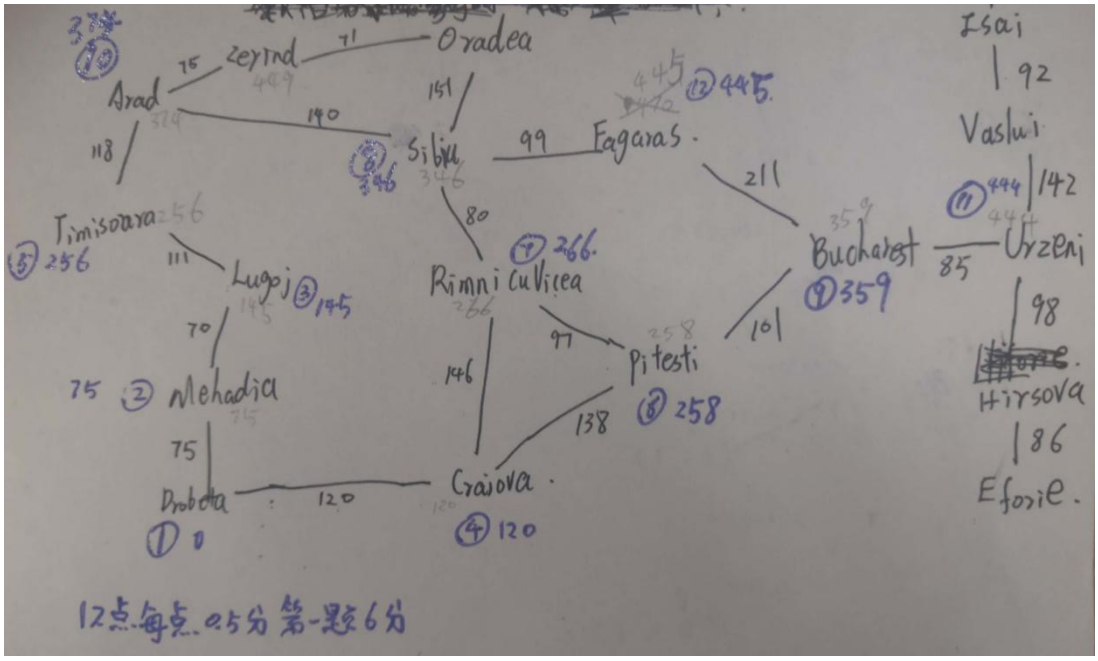
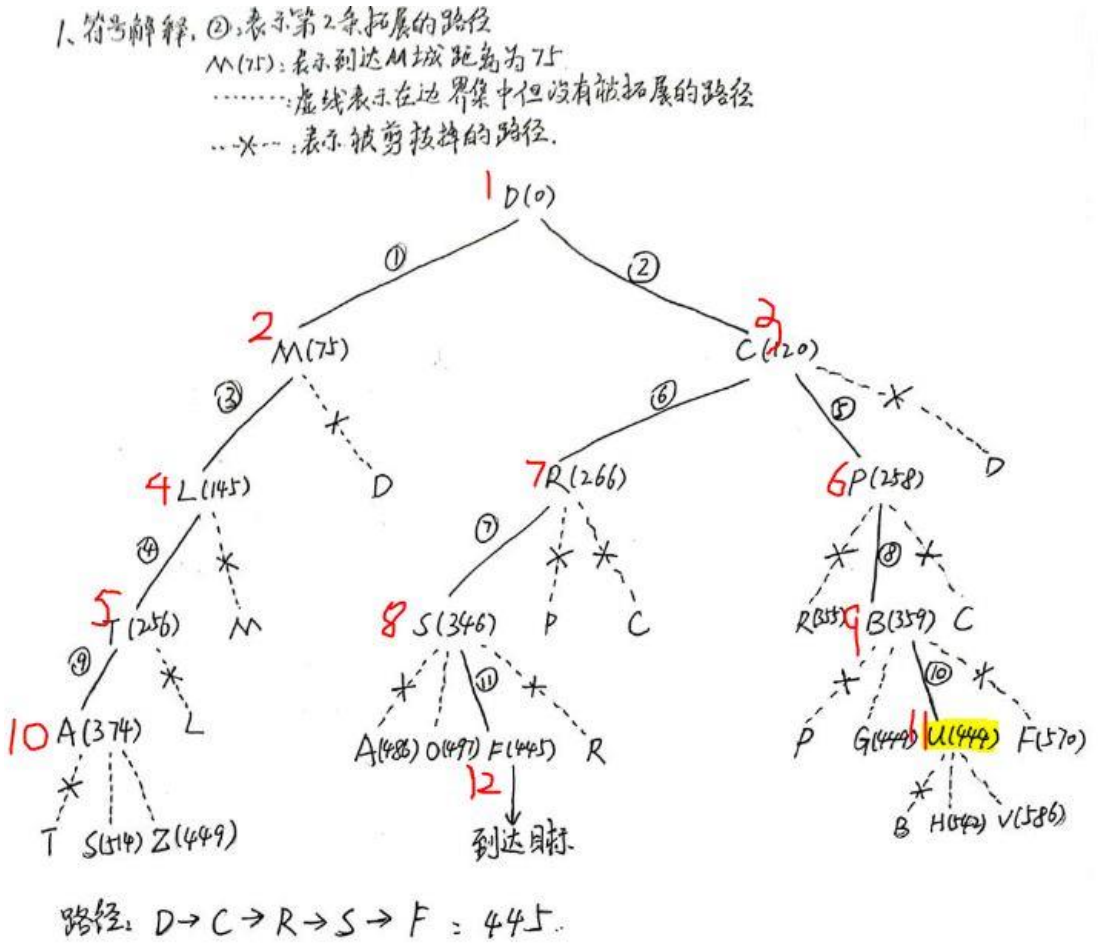
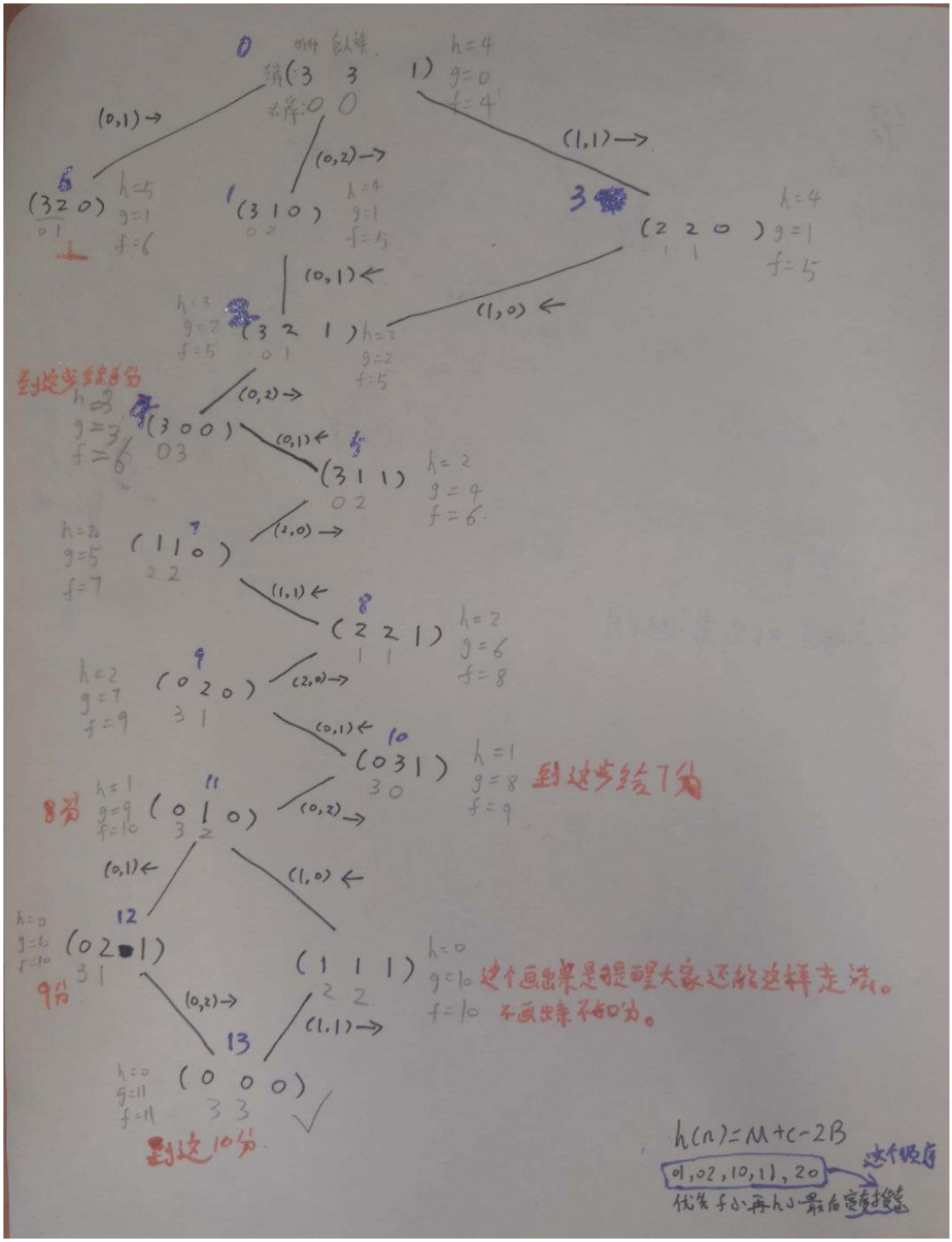
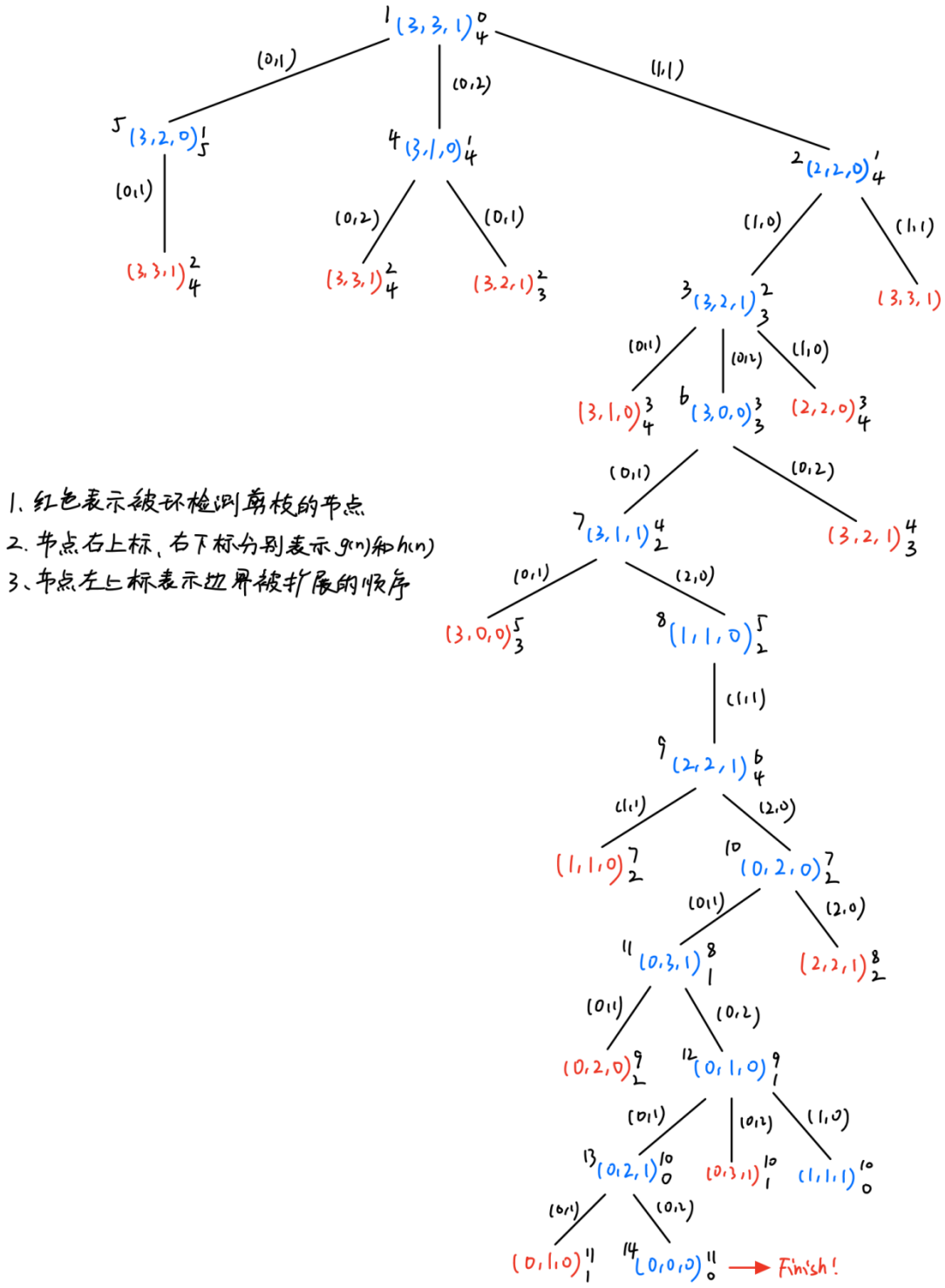


有个技巧，比如考试做题的时候可以像上图这样用铅笔画画看，能检查自己有没有画错，接下来画搜索树也快很多些。红笔序号对应上图节点序号。（其实郑同学这样标注边上也好）



上图是郑元阅同学的。Bucharest—>Urzeni 的 444<445 真的很容易会漏掉，所以考试万一考到，画图前请先在原图铅笔推下结果，然后小心谨慎排版搜索树就好。


0
(3 3 1)
h=4
g=0
f=4
(0,1)→
(0,2)→
(1,1)→
6
(3 2 0)
h=5
g=1
f=6
1
(3 1 0)
h=4
g=1
f=5
3
(2 2 0)
h=4
g=1
f=5
(0,1)←
(1,0)←
2
(3 2 1)
h=3
g=2
f=5
到这步给8分
4
(3 0 0)
h=3
g=3
f=6
(0,2)→
(0,1)←
5
(3 1 1)
h=2
g=4
f=6
7
(1 1 0)
h=2
g=5
f=7
(2,0)→
(1,1)←
8
(2 2 1)
h=2
g=6
f=8
9
(0 2 0)
h=2
g=7
f=9
(2,0)→
(0,1)←
10
(0 3 1)
h=1
g=8
f=9
到这步给7分
11
(0 1 0)
h=1
g=9
f=10
8分
(0,2)→
(0,1)←
(1,0)←
12
(0 2 1)
h=0
g=10
f=10
9分
(1 1 1)
h=0
g=10
f=10
这个画出来是提醒大家还能这样走法。
不画出来不知为。
(0,2)→
(1,1)→
13
(0 0 0)
h=0
g=11
f=11
到这10分.
h(n)=M+C-2B
01,02,10,11,20
这个顺序
优先小再小最后宽度搜索



这是是卢彦作同学作业，宽度搜索顺序虽然不是按照我之前备注的 01，02，10，11，20 画的（他先搜索的(1,1)然后是(0,2)），灵活点批改，因为太多同学这样画，这次这题也给满分（请务必留意宽度搜索顺序）。

A3（这题卢彦作同学做得很好）

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v

function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a), α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Figure 5.7 The alpha–beta search algorithm. Notice that these routines are the same as the MINIMAX functions in Figure 5.3, except for the two lines in each of MIN-VALUE and MAX-VALUE that maintain α and β (and the bookkeeping to pass these parameters along).

教材.PDF 在 P170 的算法。

伪代码 [编辑]

下面为一有限可靠性版本的Alpha-beta剪枝的伪代码[10]:

```
01 function alphabeta(node, depth, α, β, maximizingPlayer) // node = 节点, depth = 深度, maximizingPlayer = 大分玩家
02     if depth = 0 or node是终端节点
03         return 节点的启发值
04     if maximizingPlayer
05         v := -∞
06         for 每个子节点
07             v := max(v, alphabeta(child, depth - 1, α, β, FALSE)) // child = 子节点
08             α := max(α, v)
09             if β ≤ α
10                 break // β 裁剪
11         return v
12     else
13         v := ∞
14         for 每个子节点
15             v := min(v, alphabeta(child, depth - 1, α, β, TRUE))
16             β := min(β, v)
17             if β ≤ α
18                 break // α 裁剪
19         return v
```

或者伪代码像这样。

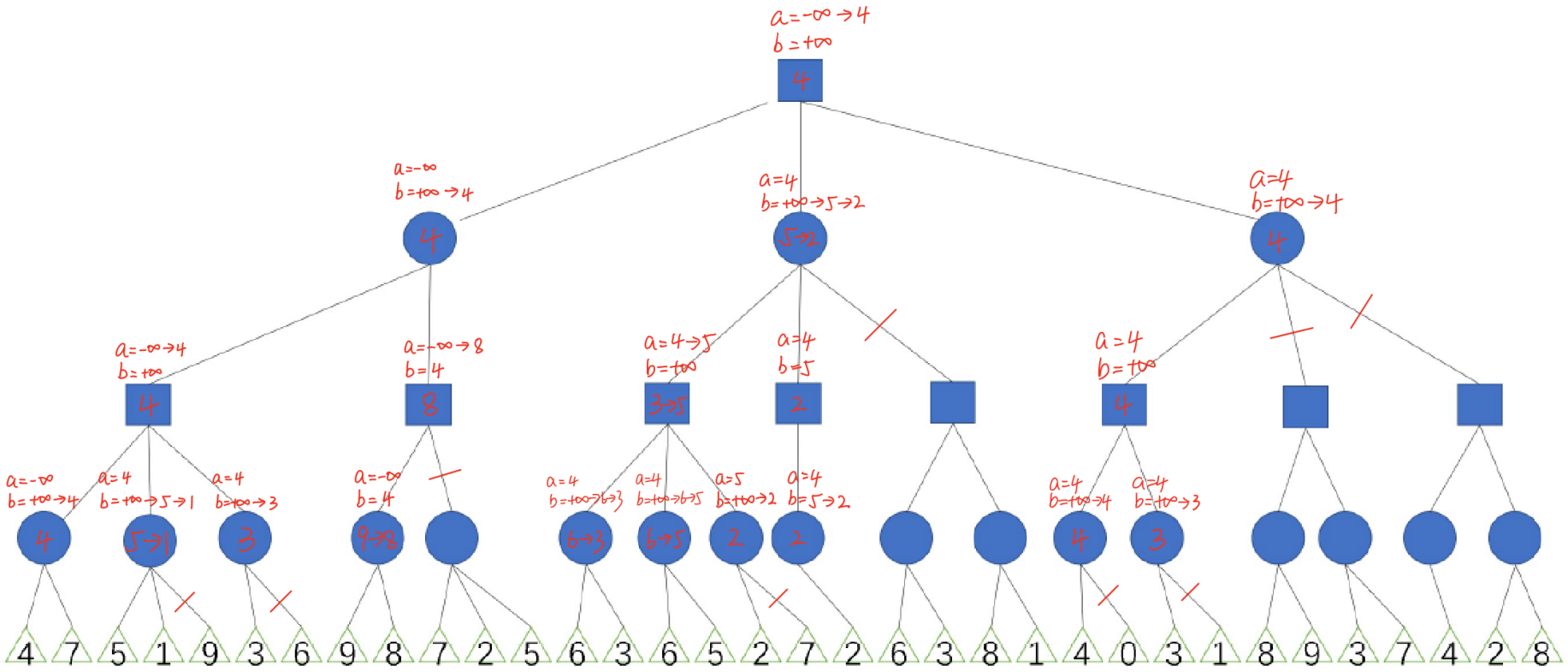
Implementing Alpha-Beta Pruning

```
AlphaBeta(n,Player,alpha,beta) //return Utility of state
If n is TERMINAL
    return V(n) //Return terminal states utility
ChildList = n.Successors(Player)
If Player == MAX
 for c in ChildList
    alpha = max(alpha, AlphaBeta(c,MIN,alpha,beta))
    If beta <= alpha
         break
 return alpha
Else //Player == MIN
  for c in ChildList
    beta = min(beta, AlphaBeta(c,MAX,alpha,beta))
    If beta <= alpha
         break
 return beta
```

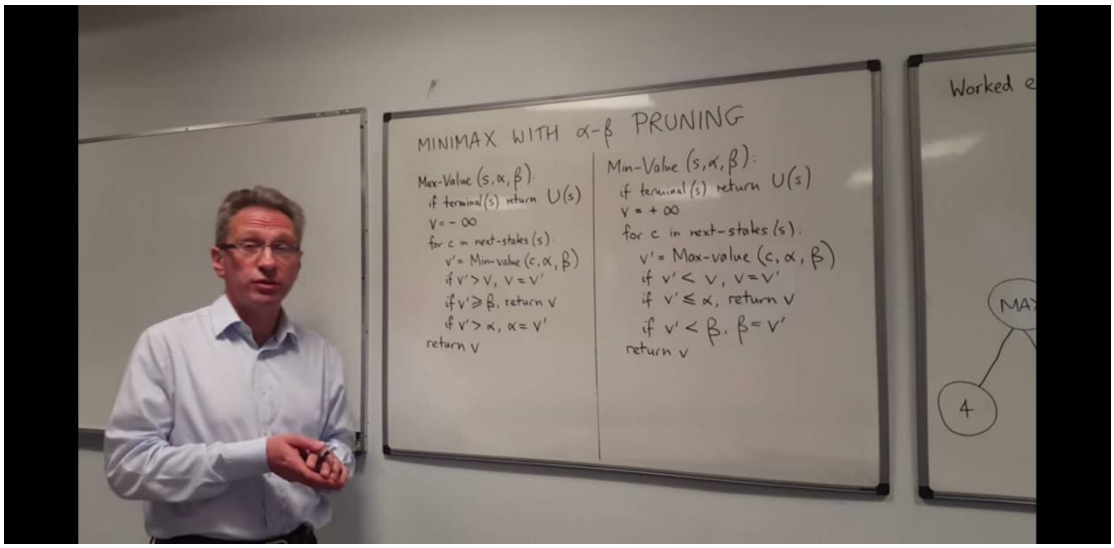
When AlphaBeta(n,Player,alpha,beta) is called, alpha is the maximum alpha value of n's ancestor Max nodes, and beta is the minimum beta value of n's ancestor Min nodes

Initial call: AlphaBeta(START-NODE,Player,-infinity,+infinity)

注意 PPT 中的 return 是 beta alpha。


a=-∞→4
b=+∞
4
a=-∞
b=+∞→4
4
a=4
b=+∞→5→2
5→2
a=4
b=+∞→4
4
a=-∞→4
b=+∞
4
a=-∞→8
b=4
8
a=4→5
b=+∞
3→5
a=4
b=5
2
a=4
b=+∞
4
a=-∞
b=+∞→4
4
a=4
b=+∞→5→1
5→1
a=4
b=+∞→3
3
a=-∞
b=4
9→8
a=4
b=+∞→6→3
6→3
a=4
b=+∞→6→5
6→5
a=5
b=+∞→2
2
a=4
b=5→2
2
a=4
b=+∞→4
4
a=4
b=+∞→3
3
4 7 5 1 9 3 6 9 8 7 2 5 6 3 6 5 2 7 2 6 3 8 1 4 0 3 1 8 9 3 7 4 2 8

上图是卢彦作同学的作业，他是自己严格根据伪代码算法一步步画的，结果是正确的，这样做很正确。
我还找到油管上有 https://youtu.be/zp3VMe0Jpf8 是介绍严格根据算法推的视频，如下图。



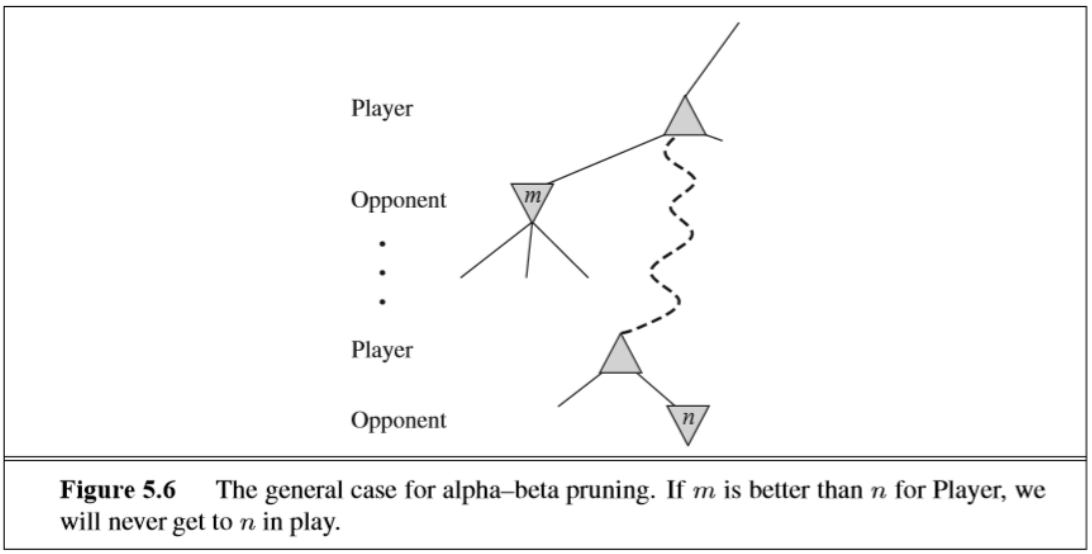


Figure 5.6 The general case for alpha–beta pruning. If m is better than n for Player, we will never get to n in play.

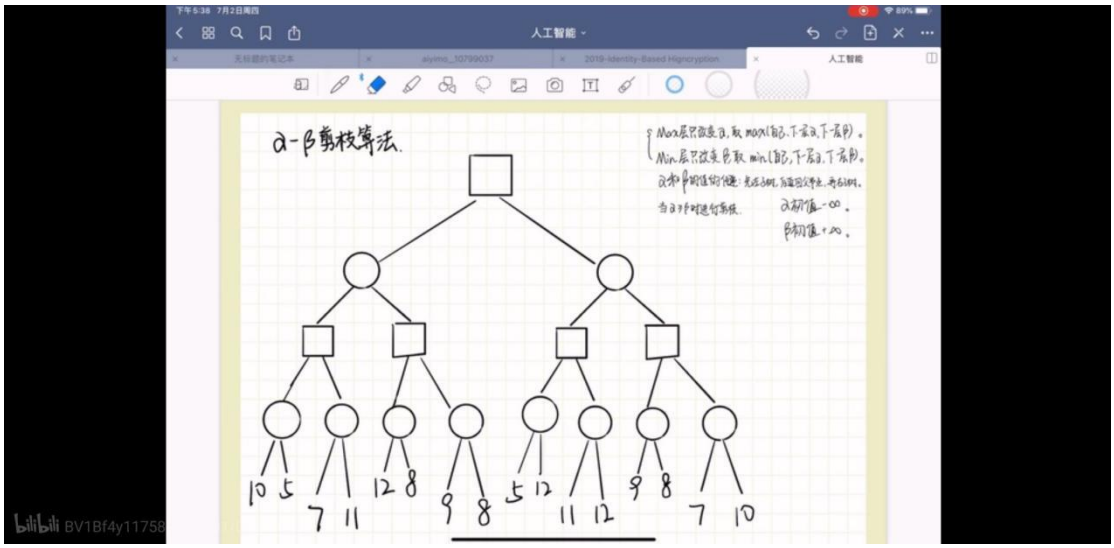
$\alpha =$ the value of the best (i.e., highest-value) choice we have found so far at any choice point along the path for MAX.

$\beta =$ the value of the best (i.e., lowest-value) choice we have found so far at any choice point along the path for MIN.

教材上这样说，alpha 是目前为止路径搜索过程中发现的 max 最佳值；beta 是目前为止路径搜索过程中发现的 min 最佳值)。
但是这里需要注意的事情是，Alpha，Beta 在每个递归中都是局部变量，其实严格来说这句话描述的 Alpha,Beta 应该是树顶的 Alpha Beta。

之前 ta(我)分享的 https://www.bilibili.com/video/BV1Bf4y11758 参考视频，最开始是这里有人转了油管手写教程 https://www.bilibili.com/video/BV1a7411K7g1 ，然后那个视频用苹果

手绘板把那个案例用中文又讲了一遍。



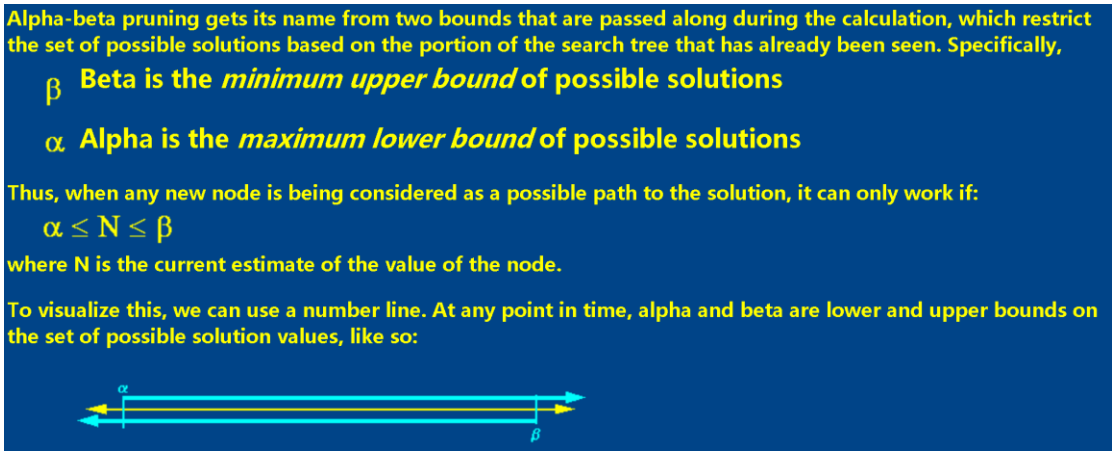
最初始的来源 https://www.youtube.com/watch?v=_i-IZcbWkps 是 student from University 0f Texas 发的。我看这个视频简单易行，频繁被转载，评论区纷纷点赞没看到指正的，就发到群里（我用它书上 PPT 上没做出反例就选择相信，其实我没有严格去证明）。

我做的过程中，视频中的方法也能做出来根节点的剪枝情况，觉得视频中的方法是一种改进或者更方便手算的方法，因为这样做 alpha/beta 更能满足书上说的：

beta 是目前为止路径搜索过程中发现的 min 最佳值（最小上界）。

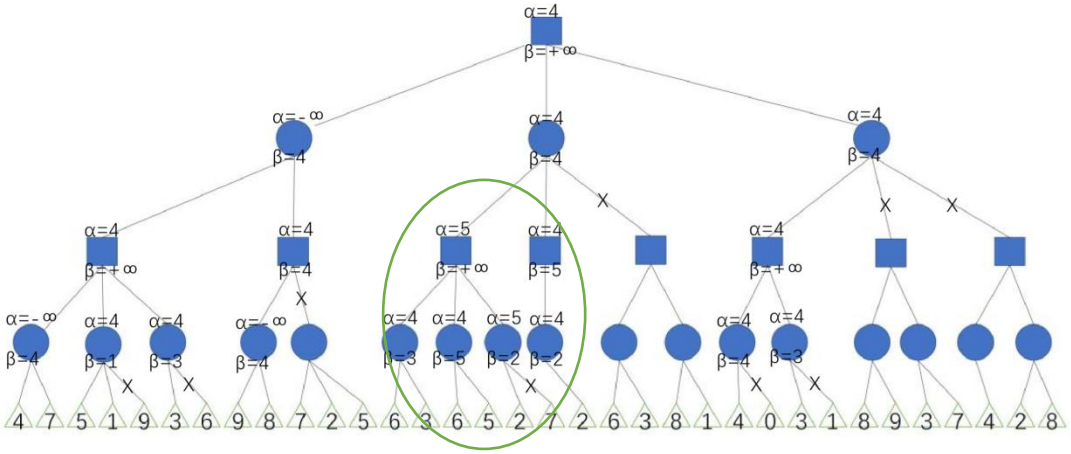
alpha 是目前为止路径搜索过程中发现的 max 最佳值（最大下界）。

看这张图：



来自：http://web.cs.ucla.edu/~rosen/161/notes/alphabeta.html

因为从左往右生成值然后探索，Alpha-Beta 左中右遍历方式能保证把左边子树中探索得到的 alpha,beta 传递过来新探索的点，又能保证(下面画圈部分子树往上传递)顶部 alpha beta 正确性。



用视频中的方法画出来的和谷正阳同学上图是一样的。

最正确的方法是严格根据代码的算法一步步画的，这样最保险的。但是不管怎样，最后如果考试考到这个，也只需要的得出正确剪枝位置。
评分标准是：少画一刀扣一分，多画一刀扣一分，完美扣对 9 刀得 10 分。