第二次作业答案

4.1 跟踪 A* 搜索算法用直线距离启发式求解从Lugoj到Bucharest问题的过程。按顺序列出算法扩展的节点和每个节点的 f, g, h 值。

L[0+244=244]
M[70+241=311], T[111+329=440]
L[140+244=384], D[145+242=387], T[111+329=440]
D[145+242=387], T[111+329=440], M[210+241=451], T[251+329=580]
C[265+160=425], T[111+329=440], M[210+241=451], M[220+241=461], T[251+329=580]
T[111+329=440], M[210+241=451], M[220+241=461], P[403+100=503], T[251+329=580], R[411+193=604], D[385+242=627]
M[210+241=451], M[220+241=461], L[222+244=466], P[403+100=503], T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627]
M[220+241=461], L[222+244=466], P[403+100=503], L[280+244=524], D[285+242=527], T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627]
L[222+244=466], P[403+100=503], L[280+244=524], D[285+242=527], L[290+244=534], D[295+242=537], T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627]
P[403+100=503], L[280+244=524], D[285+242=527], M[292+241=533], L[290+244=534], D[295+242=537], T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627], T[333+329=662]
B[504+0=504], L[280+244=524], D[285+242=527], M[292+241=533], L[290+244=534], D[295+242=537], T[251+329=580], A[229+366=595], R[411+193=604], D[385+242=627], T[333+329=662], R[500+193=693], C[541+160=701]

4.2 启发式路径算法是一个最佳优先搜索，它的目标函数是 $f(n)=(2-w)g(n)+wh(n)$。算法中w取什么值能保证算法是最优的? 当$w=0$时，这个算法是什么搜索?$w=1$呢? $w=2$呢?

$f(n)=(2-w)[g(n)+\frac{w}{2-w}h(n)]$
令$\frac{w}{2-w}h(n)<h(n)$，则Astar算法启发式函数可采纳，算法最优
得到 $0<w<1$

- $w=0$时，$f(n)=2g(n)$：一致代价搜索
- $w=1$时，$f(n)=g(n)+h(n)$：Astar搜索
- $w=2$时，$f(n)=2h(n)$：贪婪最佳搜索

4.6 设计一个启发函数，使它在八数码游戏中有时会估计过高，并说明它在什么样的特殊问题下会导致次最优解。(可以借助计算机的帮助。)证明：如果h被高估的部分从来不超过 c，A*算法返回的解的耗散比最优解的耗散多出的部分也不超过 c。

- 启发式函数：$h=h_1+h_2$，h_1是错位的数量，h_2是曼哈顿距离

假设 $h(n)\le h^*(n)+c$ 并且令 G_2 为超过最优路径 c 的次优目标点，即 $g(G_2)>C^*+c$
令节点 n 为最优路径上的任意节点，则

$$\begin{aligned} f(n) &= g(n)+h(n) \\ &\le g(n)+h^*(n)+c \\ &\le C^*+c \\ &< g(G_2) \end{aligned}$$

所以 G_2 节点不会被扩展

4.7 证明如果一个启发式是一致的，它肯定是可采纳的。构造一个非一致的可采纳启发式。

n是任意一个节点，n' 是节点n的后继节点
如果h是一致的，则 $h(n)\le c(n,a,n')+h(n')$
可以使用数学归纳法来证明：
k是从节点n到目标节点最优路径上的节点数

- 当$k=1$时，n'是目标节点，则 $h(n)\le c(n,a,n')$，成立
- 假设n'是到目标节点最优路径为k步的节点并且$h(n')$是可采纳的
- 则：

$$h(n)\le c(n,a,n')+h(n')\le c(n,a,n')+h^*(n')=h^*(n)$$

故到目标节点最优路径为k+1步的n节点也是可采纳的

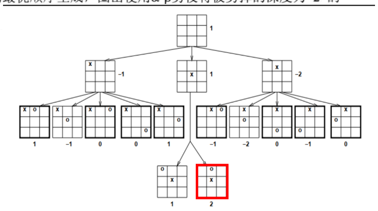
第5章习题

MiniMax，α-β剪枝（每年必考题）

5.9 本题以井字棋（圈与十字游戏）为例练习博弈中的基本概念。定义 X_n 为恰好有 n 个 X 而没有 O 的行、列或者对角线的数目。同样 O_n 为正好有 n 个 O 的行、列或者对角线的数目。效用函数给 $X_3=1$ 的棋局+1，给 $O_3=1$ 的棋局−1。所有其他终止状态效用值为 0。对于非终止状态，使用线性的评估函数定义为 $Eval(s)=3X_2(s)+X_1(s)-(3O_2(s)+O_1(s))$。

a. 估算可能的井字棋局数。
b. 考虑对称性，给出从空棋盘开始的深度为 2 的完整博弈树（即，在棋盘上一个 X 一个 O 的棋局）。
c. 标出深度为 2 的棋局的评估函数值。
d. 使用极小极大算法标出深度为 1 和 0 的棋局的倒推值，并根据这些值选出最佳的起
e. 假设结点按对α-β剪枝的最优顺序生成，圈出使用α-β剪枝将被剪掉的深度为 2 的结点。

a) 9!
b)
c)
d) 中心位置
e)



5.8 考虑图 5.17 中描述的两人游戏。

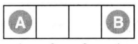


图 5.17 一个简单游戏的初始棋局

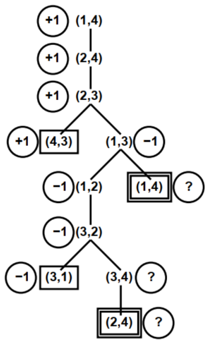
选手 A 先走。两个选手轮流走棋，每个人必须把自己的棋子移动到任一方向上的相邻空位中。如果对方的棋子占据着相邻的位置，你可以跳过对方的棋子到下一个空位。（例如，A 在位置 3，B 在位置 2，那么 A 可以移回 1。）当一方的棋子移动到对方的端点时游戏结束。如果 A 先到达位置 4，A 的值为+1；如果 B 先到位置 1，A 的值为−1。

a. 根据如下约定画出完整博弈树：
   - 每个状态用(s_A, s_B)表示，其中 s_A 和 s_B 表示棋子的位置。
   - 每个终止状态用方框画出，用圆圈写出它的博弈值。
   - 把循环状态（在到根结点的路径上已经出现过的状态）画上双层方框。由于不清楚他们的值，在圆圈里标记一个“？”。

b. 给出每个结点倒推的极小极大值（也标记在圆圈里）。解释怎样处理“？”值和为什么这么处理。

c. 解释标准的极小极大算法为什么在这棵博弈树中会失败，简要说明你将如何修正它，在（b）的图上画出你的答案。你修正后的算法对于所有包含循环的游戏都能给出最优决策吗？

d. 这个 4-方格游戏可以推广到 n 个方格，其中 $n > 2$。证明如果 n 是偶数 A 一定能赢，而 n 是奇数则 A 一定会输。



B. min(-1, ?) = -1, min(1, ?) = 1

对于 "?" 值的处理，我们采取极小极大算法的一种常见策略：对于极大节点，将 "?" 值看作负无穷大，即认为这个节点的值无限大；对于极小节点，将 "?" 值看作正无穷大，即认为这个节点的值可以无限小。这样处理可以确保在博弈树搜索中，如果一个节点的值是 "?"，那么其父节点的选择将不会受到该节点的影响，从而确保算法的正确性。

C. 标准的极小极大算法是深度优先的，会在遇到循环状态时陷入无限循环，导致死循环。

为了修正这个问题，可以使用α-β剪枝算法。α-β剪枝算法在搜索过程中可以剪掉一些不必要的分支，从而减少搜索空间。对于包含循环的游戏，α-β剪枝算法仍然能够给出最优决策，只是可能需要更多的计算和优化。

D. n=3, A输，n=4， A赢,

f(5) = f(3),
f(6) = f(4)…

- 循环检测：在执行剪枝之前，进行循环检测以避免陷入无限循环。当在搜索过程中遇到一个状态时，可以检查该状态是否已经在当前路径中出现过。如果检测到循环状态的出现，可以中断当前路径的搜索，并返回一个适当的值，以避免进入无限循环。
- 剪枝条件：在进行剪枝时，可以根据当前状态的信息和已经计算得到的值，判断是否可以提前终止该分支的搜索。在处理循环状态时，可以根据循环检测的结果来判断是否进行剪枝。
- 如果循环检测结果表明当前状态已经在当前路径中出现过，可以认为进一步搜索该分支是无意义的，因为会导致重复计算和循环状态的进入。在这种情况下，可以直接进行剪枝，即不再深入搜索该分支，并返回一个适当的值。
- 如果循环检测结果表明当前状态未出现在当前路径中，说明该状态是首次遇到的，可以继续进行搜索并使用剪枝技术。在此过程中，可以根据已经计算得到的值和评估函数的结果，确定是否可以进行剪枝。
- 综上所述，剪枝技术可以通过结合循环检测和剪枝条件来处理循环状态。循环检测可以防止进入无限循环，而剪枝条件可以根据循环检测结果判断是否进行剪枝，从而减少搜索空间和重复计算。这样可以提高算法的效率，并避免在处理循环状态时陷入无限循环的困境。

a.

$$n_2 = max(n_3, n_{31}, \ldots, n_{3b_3})$$

$$n_1 = min(max(n_3, n_{31}, \ldots, n_{3b_3}), n_{21}, \ldots, n_{2b_2})$$

…

b.

$$n_1 = min(l_2, max(l_3, n_3, r_3), r_2)$$

… $min(l_j, n_j, r_j)$

c.

$$min(l_2, l_4, \ldots, \boxed{l_j})$$

d.

$$max(l_3, l_5, \ldots, l_k)$$

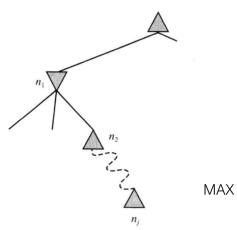


图 5.18 是否剪掉结点 n_j 时的情形

5.13 请给出α-β剪枝正确性的形式化证明。要做到这一点需考虑图 5.18。问题为是否要剪掉结点 n_j，它是一个 MAX 结点，是 n_1 的一个后代。基本的思路是当且仅当 n_1 的极小极大值可以被证明独立于 n_j 的值时，会发生剪枝。

a. n_1 的值是所有后代结点的最小值：$n_1 = \min(n_2, n_{21}, \cdots, n_{2b_2})$。请为 n_2 找到类似的表达式，以得到用 n_j 表示的 n_1 的表达式。

b. 深度为 i 的结点 n_i 的极小极大值已知，l_i 是在结点 n_i 左侧结点的极小值（或者极大值）。同样，r_i 是在 n_i 右侧的未探索过的结点的极小值（或者极大值）。用 l_i 和 r_i 的值重写 n_1 的表达式。

c. 现在重新形式化表达式，来说明为了向 n_1 施加影响，n_j 不能超出由 l_i 值得到的某特定界限。

d. 假设 n_j 是 MIN 结点的情况，请重复上面的过程。

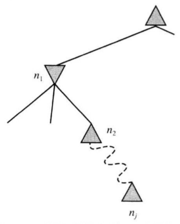


图 5.18　是否剪掉结点 n_j 时的情形

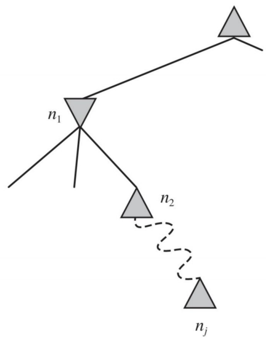


Figure 5.18 Situation when considering whether to prune node n_j.

a.

$$n_2 = max(n_3, n_{31}, \ldots, n_{3b_3})$$

$$n_1 = min(max(min(\ldots(min(n_j, n_{j1}, \ldots, n_{jb_j}), \ldots)\ldots), n_{31}, n_{3b_3}), n_{21}, \ldots, n_{2b_2})$$

依次类推，替代n3, ... 直到包含nj

b

$$n_1 = \min(l_2, \max(l_3, n_3, r_3), r_2)$$

$$n_1 = min(l_2, max(l_3, min(\ldots max(l_{j-1}, min(l_j, n_j, r_j), r_{j-1})\ldots), r_3), r_2)$$

继续扩展n3直到nj为止，最深的一层为 $\min(l_j, n_j, r_j)$

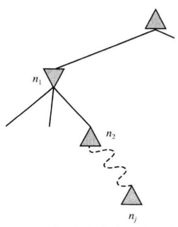


图 5.18　是否剪掉结点 n_j 时的情形

c. 如果$n_j > l_j, min(l_j, n_j, r_j)$与$n_j$无关，那么$n_1$也与$n_j$无关；

如果$n_j > l_{j-2}$，

$min(l_j, n_j, r_j) \neq n_j$时，$n_1$与$n_j$无关；

$min(l_j, n_j, r_j) = n_j$时，$max(l_{j-1}, min(l_j, n_j, r_j), r_{j-1}) \geq n_j > l_{j-2}$，

$min(l_{j-2}, max(l_{j-1}, min(l_j, n_j, r_j), r_{j-1}), r_{j-2})$与$n_j$无关

nj 只要大于任意一个下标为偶数的lj,就不会对n1造成影响，综上nj>min(l2,l4,,,lj)时对n1无影响

d

nj 只要小于所有下标为奇数的lj,就不会对n1造成影响，综上nj<max(l3,l5,,,lj)时对n1无影响

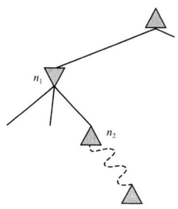


图 5.18　是否剪掉结点 n_j 时的情形