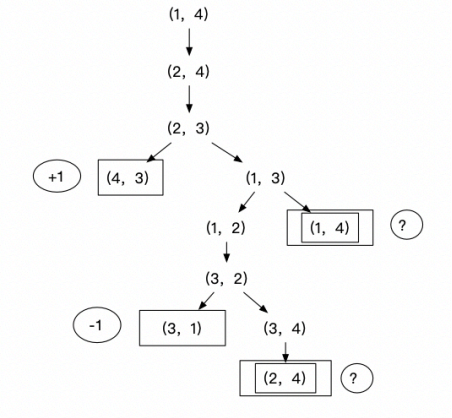
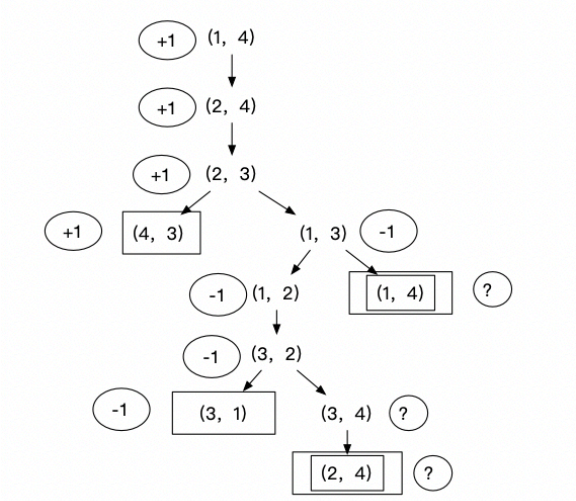
解答：

a) 根据题目要求，画出的 game tree 如下所示：




b) 加上了 backed-up minimax value 之后的 game tree 如下图所示：




处理‘？’的方式是，min(-1, ?) = -1; max(+1, ?) = +1。如果子节点是‘？’那么 backed-up value 也是‘？’

c) 问题的解答如下：

有的时候标准的 minimax 算法会失败，主要是因为其用的是深度优先算法，然后会陷入无限循环（infinite loop）。

要修复这个问题，首先我们建立一个状态栈，每产生一个新状态即入状态栈，如果我们的当前状态已经存在在状态栈中，我们需要返回“？”，然后按照（2）中所说的方法传递“？”，即可避免这种情况。

然而，虽然这种情况能够改变上述所说的无限循环的缺陷，这种改进的算法也不是对于所有的 loop 都能给出最好的结果，在有的场景中，我们可能存在无法比较“？”和其他节点的情况，在这种情况下，我们需要对于“？”定义一个值，并且对于 min、max 函数需要对应的修改。（我们书中定义的 minmax 函数是会给每一个 state 初始情况都赋一个不确定的值的，所以我们的“？”和这些值比较可能会出现错误）

d) 我们用归纳法证明：

- 首先，基本情况是 n = 3 和 n=4，n=3 是 A 输，n=4 是 A 赢。
- 对于任何 n>4 的情况，我们考虑一个子游戏在[2,3…n-1]上进行。显然如果在 n-2 上是 A 赢的话，那么 A 将会先到达 n-1，然后 B 才会到达 2，然后 A 会先到达 n，这

之后 B 才会到达 1，所以这个游戏是 A 赢。同理，如果 n-2 上是 B 赢的话，这个游戏也会是 B 赢，即 A 输掉游戏。即 n 和 n-2k 的输赢性质一样。

- 然而我们有 n=3 有 A 输、n=4 有 A 赢，所以综上，即为奇数的时候 A 输，为偶数的时候 A 赢。