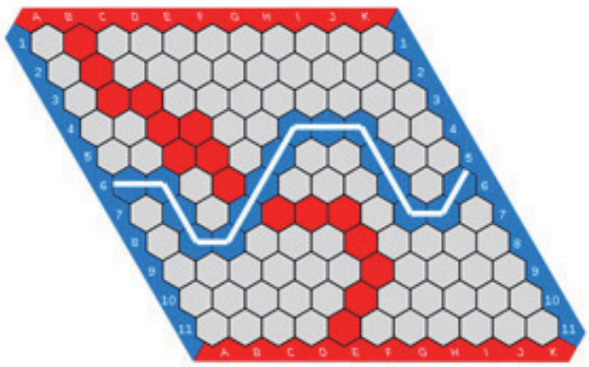
扬长避短 *最小极大及 α–β 算法*

万精油

上期的题目是讲 Hex 游戏的。我们先来介绍一下这个游戏。

下围棋累了就连五子，打桥牌困了就敲三先，爱玩的人常常是逮什么玩什么。20 多年前一个围棋棋友教我一种新棋，英文名叫 Hex。此棋大约是人类发明的棋类游戏中规则最简单的了。但看似简单，个中却奥妙无穷。像围棋一样，它也有定式、手筋、引征等等等等。喜欢数学及围棋的朋友几乎都会对它感兴趣，我自然也很快就被它吸引住了。

Hex 规则很简单：一个菱形被分为 N 乘 N 个小六边形（见下图）。通常 $N = 11$，也可以到 15。黑白双方轮流下子占据这些小六边形。谁先使自己所下的棋子连通对边谁赢。黑方连上下边，白方连左右边。下图是红蓝色而不是黑白，红方连上下边，蓝方连左右边。



这个游戏是 1942 年由丹麦人皮亚特 • 海恩（Piet Hein）发明的。大多数人认为，有趣而又规则简单的游戏早已被人发明完了（比如围棋、象棋之类的），新游戏不是规则麻烦就是没意思。但 Hex 却是既新颖，又简单，而且还有趣。发明以后很快在世面上流行起来。尤其是在数学家中间。Piet Hein 是一个很神奇的人。他是一个著名诗人，同时又是一个政治活动家。而他本行却是理论物理学家，Hex 就是他在玻尔理论物理研究所发明的。据说他当时在思考四色定理。

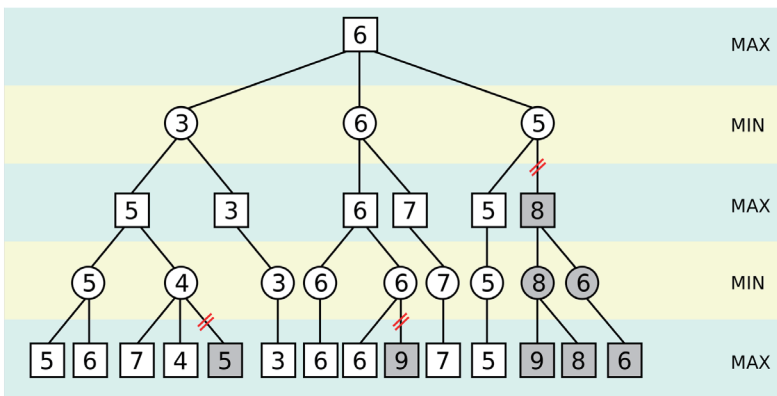
这个游戏后来又在普林斯顿被纳什重新独立发明。所以 Hex 在普林斯顿与麻省理工也很流行，并被称之为 Nash。这个游戏后来在普林斯顿以外流行要归功于数学科普作家加德纳（《数学文化》杂志以前有文章专门介绍他）。加德纳在《科学美国人》上介绍这个游戏后，它开始在更大范围内流行起来。

因为不是矩形，Hex 的棋盘制作比较麻烦，所以很多人听说此棋也没机会下。我听说这个游戏后写了一个在 DOS 下供人下此棋的程序（当时还没有视窗系统），不需棋盘，只需键盘操作就可以下了。开始只是想写个程序用来供两人互下。后来决定加一些人工智能，使它可以与人对下。当然，我只是业余搞一搞，那个程序水平不是很高，稍微下得好一点的人就可以赢它。虽然如此，当时计算机程序还不是很普及，搞这个的人也不多，那个程序在当时还算是比较领先的，后来搞 Hex 研究的计算机专家们写这方面的论文还偶尔在参考文献里提到它。我那个程序虽然不厉害，但里面用到的基本思想是几乎所有

二人对抗游戏程序需要用到的，现在我们就来介绍一下这个基本思想——最小极大算法。

二人对抗游戏，计算机程序那一方在考虑一步棋的走法时，需要在许多可行走法中挑最好的一步。一个很自然的问题是，什么叫最好？几乎所有有趣的游戏，一步走完不能马上判断其好坏，除非是象棋把对方将死了，或者围棋吃别人一大块。大多数情况是要好几步甚至好几十步以后才能做一个相对有效的判断。所以，判断一步棋的好坏要看下一步。下一步该对方走。在对方下一步可以走的所有走法中，对我方最不利的是什么，这就是上一步带来的最坏结果。聪明的读者可能已经意识到，绝大多数情况，只考虑下一步也是不够的，还必须继续往深处走。专业棋手有时要考虑十几步甚至几十步。计算机程序要成高手，也必须要考虑许多步。考虑过程中，自己选择的时候挑最好的一步，对方下的时候挑对自己最坏的一步。如果给每一步赋值，那就是在自己的选择中取最大值，在对方的选择中取最小值。这个算法的名字就是由此而来，minimax，最小极大值。中文里的现成成语就是扬长避短。

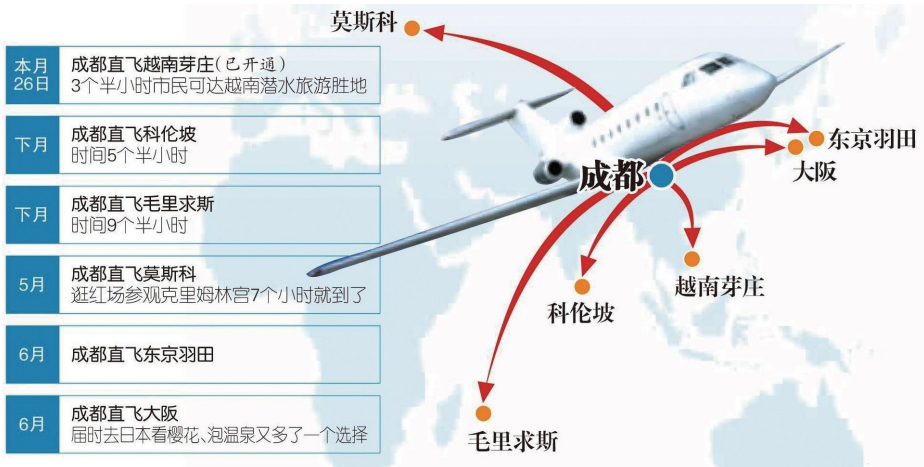
一步一步地考虑下去，走到底就会有结果，就可以判断好坏。但是，每一步的选择不少，如果每一步都走到底，运算要求太大，对很多游戏来说几乎是不可能的任务。那么，不走到底我们如何判别好坏呢？前面一直在说赋值，这个值怎么赋？对不同的游戏有不同的方法。比如象棋，我们可以数一数双方棋盘上的棋子数。当然，这个棋子数是加权过的，比如一个车等于 7 个卒之类的，也就是说判断一下棋盘上双方子力。探索到 N 步以后，给双方子力赋一个值，这就是选择那步棋的值。对围棋来说，数子是不行的，需要点目，走到一定步数以后我们可以大致点目。对每一个不同的游戏，需要采取不同的赋值法，使得其搜索能够成立。我们用下面这个图对这个最小极大算法给一个总结。



上图中有 MAX 的行就是对下面的各种值中取最大值。上图中有 MIN 的行就是对下面的各种值中取最小值。计算机考虑的第一步有 3 个选择，就是上图中标有 3, 6, 5 的圆圈。三个数的最大值是 6，所以最上面是一个 6。当然，那三个数 3, 6, 5 不是本身就有的，这些值都是深挖到几步以后一步一步反馈上来的。我们先考虑最左面那一个选择。假设做了这个选择，那么对方有两种选择，也就是标有 5, 3 的方框。同样，我们又先从左面第一个选择开始，到下一步，如此下去，直到我们要考虑的最后一步。上图是只深挖到四步的情况。也就是说到第四步后，我们需要其赋值。假设赋值是 5 与 6。这两种结果是对方的选择（方框表示的都是对方的选择），我们取其中最小值 5，填入上面的圆圈，这就是第三步选择最左面那一步后对方能给我方的最大破坏。我们至少可以得到赋值为 5 的结果。再接着考虑第三步的另一种选择，那一步走下去后，对方可以有三个回应，其

赋值为 7, 4, 5。最小值是 4，所以我们把 4 填入那个圆圈。现在，在第三步的两个选择中，一个结果是 5，另一个是 4。因为这是计算机一方，所以我们选择最大值，也就是我们能得到的最好结果，5。把这个 5 填入到上面的方框。以此类似，我们一步一步的往下走，然后把那些值按极小最大的原则一步一步地反馈上去。最后我们得到第一步三种选择的三个赋值，3, 6, 5。我们当然选最大值，6。也就是说，这个极小最大算法得到的结果是，在三个选择中，中间那个可以得到最好结果。所以，计算机程序选择中间那一步。

这个搜索方法中，我们总是沿着一条路走到搜索的最底层再返回去搜下一步。这种搜法有一个专有名词叫深度优先（英文叫 Depth-First Search)。这在对弈程序中很普遍，因为不搜到一定程度不好赋值，从而不知道好坏，没办法选择。但在另一些搜索中，如果每一步的赋值都有了，那么我们可以采用先横向搜索，而不是纵向搜索。比如，我们如果要搜索从成都到波士顿最少需要转几次飞机，就不能用深度优先，因为很多城市之间都有航班，可能形成循环。即使不循环，深度优先也不利于搜最少转机线路。这个时候，我们就要用广度优先（英文叫 Breadth-First Search)，横向搜索。先搜成都能一步飞到的所有城市。再搜从那些城市能一步飞到的所有城市，如此搜下去，最先找到波士顿的就是转机最少的线路。如果有多个转机最少的线路并列，再在其中选最短的。



如前所说，在对弈程序中还是深度优先最适用，我们还是回来谈深度优先。

前面说我们每一步都要在许多可行的步法中选择。这个“许多”不是所有可行的步法，只能在相对比较有价值的步法中选择。像围棋这样的游戏，搜索所有可行步法是不现实的。第一步有 361 种可选择的走法，第二步有 360 种，搜索空间成指数增长，几步下来就超出计算机的运算能力。所以，我们必须先做一些淘汰（这个工作说起来就一句话，实际执行起来不是一件简单的事）。把每一步的搜索限制在一个固定范围内，比如说 k 种走法。假设每一步考虑 k 种走法，我们搜索的深度是 d, 那么搜索空间就是 k 的 d 次方。在计算机运算中，这种指数形式的增长是很可怕的。但是，要让我们的程序变得很强，我们就必须要增加 k, 增加 d，这个 k 的 d 次方就变成一个可怕的障碍。

于是，大家开始想办法缩减这个 k 的 d 次方。我们副标题中的“α-β 算法”就是办法之一。我们现在就来对它做一个简单介绍。

我们注意到这极小最大算法中，自己一方各种选择中取最大值。那么，当已经知道其中一个选择的值的时候，我们知道这一步至少不会比那个值小。用数学语言来写就是：