**五子棋的核心算法**

时间:2010-03-26 20:50来源:网络 作者:佚名 点击: 3115次

介绍了五子棋程序的数据结构、评分规则、胜负判断方法和搜索算法过程。

|  |
| --- |
| 五子棋是一种受大众广泛喜爱的游戏，其规则简单，变化多端，非常富有趣味性和消遣性。这里设计和实现了一个人机对下的五子棋程序，采用了博弈树的方法，应用了剪枝和最大最小树原理进行搜索发现最好的下子位置。介绍五子棋程序的数据结构、评分规则、胜负判断方法和搜索算法过程。  **一、相关的数据结构**  关于盘面情况的表示，以链表形式表示当前盘面的情况，目的是可以允许用户进行悔棋、回退等操作。   1. CList StepList; 2. //其中Step结构的表示为： 3. **struct** Step 4. { 5. **int** m;//m,n表示两个坐标值 6. **int** n; 7. **char** side;//side表示下子方 8. }; 9. //以数组形式保存当前盘面的情况， 10. //目的是为了在显示当前盘面情况时使用： 11. **char** FiveArea[FIVE\_MAX\_LINE][FIVE\_MAX\_LINE]; 12. //其中FIVE\_MAX\_LINE表示盘面最大的行数。 13. //同时由于需要在递归搜索的过程中考虑时间和空间有效性， //只找出就当前情况来说相对比较好的几个盘面，而不是对所有的可下子的位置都进行搜索， //这里用变量CountList来表示当前搜索中可以选择的所有新的盘面情况对象的集合： 14. CList CountList; 15. //其中类CBoardSituiton为: 16. **class** CBoardSituation 17. { 18. CList StepList; //每一步的列表 19. **char** FiveArea[FIVE\_MAX\_LINE][FIVE\_MAX\_LINE]; 20. **struct** Step machineStep;//机器所下的那一步 21. **double** value;//该种盘面状态所得到的分数 22. }   **二、评分规则**  对于下子的重要性评分，需要从六个位置来考虑当前棋局的情况，分别为：-,|,/,\,//,\\  实际上需要考虑在这六个位置上某一方所形成的子的布局的情况，对于在还没有子的地方落子以后的当前局面的评分，主要是为了说明在这个地方下子的重要性程度，设定了一个简单的规则来表示当前棋面对机器方的分数。  基本的规则如下：   1. 判断是否能成5, 如果是机器方的话给予100000分，如果是人方的话给予－100000 分； 2. 判断是否能成活4或者是双死4或者是死4活3，如果是机器方的话给予10000分，如果是人方的话给予－10000分； 3. 判断是否已成双活3，如果是机器方的话给予5000分，如果是人方的话给予－5000 分； 4. 判断是否成死3活3，如果是机器方的话给予1000分，如果是人方的话给予－1000 分； 5. 判断是否能成死4，如果是机器方的话给予500分，如果是人方的话给予－500分； 6. 判断是否能成单活3，如果是机器方的话给予200分，如果是人方的话给予－200分； 7. 判断是否已成双活2，如果是机器方的话给予100分，如果是人方的话给予－100分； 8. 判断是否能成死3，如果是机器方的话给予50分，如果是人方的话给予－50分； 9. 判断是否能成双活2，如果是机器方的话给予10分，如果是人方的话给予－10分； 10. 判断是否能成活2，如果是机器方的话给予5分，如果是人方的话给予－5分； 11. 判断是否能成死2，如果是机器方的话给予3分，如果是人方的话给予－3分。   实际上对当前的局面按照上面的规则的顺序进行比较，如果满足某一条规则的话，就给该局面打分并保存，然后退出规则的匹配。注意这里的规则是根据一般的下棋规律的一个总结，在实际运行的时候，用户可以添加规则和对评分机制加以修正。  **三、胜负判断**  实际上，是根据当前最后一个落子的情况来判断胜负的。 实际上需要从四个位置判断，以该子为出发点的水平，竖直和两条分别为 45度角和135度角的线，目的是看在这四个方向是否最后落子的一方构成连续五个的棋子，如果是的话，就表示该盘棋局已经分出胜负。具体见下面的图示：  **四、搜索算法实现描述**  注意下面的核心的算法中的变量currentBoardSituation，表示当前机器最新的盘面情况, CountList表示第一层子节点可以选择的较好的盘面的集合。  核心的算法如下：   1. **void** MainDealFunction() 2. { 3. value=-MAXINT; //对初始根节点的value赋值 4. CalSeveralGoodPlace(currentBoardSituation,CountList); 5. //该函数是根据当前的盘面情况来比较得到比较好的可以考虑的几个盘面的情况， 6. //可以根据实际的得分情况选取分数比较高的几个盘面， 7. //也就是说在第一层节点选择的时候采用贪婪算法， 8. //直接找出相对分数比较高的几个形成第一层节点， 9. //目的是为了提高搜索速度和防止堆栈溢出。 10. pos=CountList.GetHeadPosition(); 11. CBoardSituation\* pBoard; 12. **for**(i=0;ivalue=Search(pBoard,min,value,0) 13. { 14. Value=Select(value,pBoard－>value,max); 15. //取value和pBoard－>value中大的赋给根节点 16. } 17. **for**(i=0;ivalue) 18. //找出那一个得到最高分的盘面 19. { 20. currentBoardSituation=pBoard; 21. PlayerMode=min; //当前下子方改为人 22. Break; 23. } 24. } 25. //其中对于Search函数的表示如下： 26. //实际上核心的算法是一个剪枝过程，其中在这个搜索过程中相关的四个参数为： 27. //（1）当前棋局情况；（2）当前的下子方，可以是机器(max)或者是人(min)； 28. //（3）父节点的值oldValue；（4）当前的搜索深度depth。 29. **double** Search(CBoardSituation& board,**int** mode,**double** oldvalue,**int** depth) 30. { 31. CList m\_DeepList; 32. **if**(deptholdvalue==TRUE) 33. { 34. { 35. **if**(mode==max) 36. value=select(value,search(successorBoard,min,value,depth＋1),max); 37. **else** 38. value=select(value,search(successorBoard,max,value,depth＋1),min); 39. } 40. **return** value; 41. } 42. **else** 43. { 44. **if** (goal(board)!=0) 45. //这里goal(board)<>0表示已经可以分出胜负 46. **return** goal(board); 47. **else** 48. **return** evlation(board); 49. } 50. } 51. //注意这里的goal(board)函数是用来判断当前盘面是否可以分出胜负， 52. //而evlation(board)是对当前的盘面从机器的角度进行打分。 53. //下面是Select函数的介绍，这个函数的主要目的是根据 PlayerMode情况， 54. //即是机器还是用户来返回节点的应有的值。 55. **double** Select(**double** a,**double** b,**int** mode) 56. { 57. **if**((a>b && mode==max)||(a< b && mode==min)) 58. **return** a; 59. **else** 60. **return** b; 61. }   **五、小结**  在Windows操作系统下，用VC＋＋实现了这个人机对战的五子棋程序。 和国内许多只是采用规则或者只是采用简单递归而没有剪枝的那些程序相比，在智力上和时间有效性上都要好于这些程序。同时所讨论的方法和设计过程为用户设计其他的游戏（如象棋和围棋等）提供了一个参考。 |

|  |
| --- |
|  |
| **五子棋算法探讨**  作者：[青青](http://www.wuzi8.com/ShowAuthor.asp?ChannelID=1&AuthorName=7P7P) 文章来源：[成都金点](http://www.wuzi8.com/ShowCopyFrom.asp?ChannelID=1&SourceName=%239&,-%60%25S) 点击数： 5254 更新时间：2006-5-25 |
|  |
| |  | | --- | | 近来随着计算机的快速发展，各种棋类游戏被纷纷请进了电脑，使得那些喜爱下棋，又常常苦于没有对手的棋迷们能随时过足棋瘾。而且这类软件个个水平颇高，大有与人脑分庭抗礼之势。其中战胜过国际象棋世界冠军-卡斯帕罗夫的“深蓝”便是最具说服力的代表；其它像围棋的“手淡”、象棋的“将族”等也以其优秀的人工智能深受棋迷喜爱；而我们今天将向大家介绍的是五子棋的算法。  　　当我们与电脑对战时，您知道这些软件是怎样象人脑一样进行思考的吗？前不久我曾编写过一个五子棋的游戏，在这里就以此为例和大家一起探讨探讨。  　　总的来说（我们假定您熟悉五子棋的基本规则），要让电脑知道该在哪一点下子，就要根据盘面的形势，为每一可能落子的点计算其重要程度，也就是当这子落下后会形成什么棋型（如：“冲四”、“活三”等），然后通览全盘选出最重要的一点，这便是最基本的算法。当然，仅\*当前盘面进行判断是远远不够的，这样下棋很容易掉进玩家设下的陷阱，因为它没有考虑以后的变化。所以在此基础上我们加入递归调用，即：在电脑中预测出今后几步的各种走法，以便作出最佳选择，这也是我们下棋时常说的“想了几步”。如此一来您的程序便具有一定的水平了。什么？不信！过来试试吧！  　　总体思路弄清之后，下面进行具体讨论：   一：数据结构  　　先来看看数据结构，我们需要哪些变量？  　　首先得为整个棋盘建立一张表格用以记录棋子信息，我们使用一个15\*15的二维数组 Table[15][15] (15\*15是五子棋棋盘的大小)，数组的每一个元素对应棋盘上的一个交\*点，用‘0’表示空位、‘1’代表己方的子、‘2’代表对方的子；这张表也是今后分析的基础。  　　在此之后还要为电脑和玩家双方各建立一张棋型表Computer[15][15][4]和Player[15][15][4]，用来存放棋型数据，就是刚才所说的重要程度，比如用‘20’代表“冲四”的点，用‘15’代表“活三”的点，那么在计算重要性时，就可以根据20&gt;15得出前者比后者重要，下子时电脑便会自动选择“冲四”的点。那为什么棋型表要使用三维数组呢？因为棋盘上的每一个点都可以与横、竖、左斜、右斜四个方向的棋子构成不同的棋型，所以一个点总共有4个记录；这样做的另一个好处是可以轻易判断出复合棋型，例如：如果同一点上有2个‘15’就是双三、有一个‘15’和一个‘20’就是四三。  　　怎么样！3个数组构成了程序的基本数据骨架，今后只要再加入一些辅助变量便可以应付自如了。应该不会太难吧？OK！有了这么多有用的数据，我们就可以深入到程序的流程中去了。   二：程序流程  　　我们主要讨论五子棋的核心算法，即：人工智能部分，而其他像图形显示、键盘鼠标控制等，因较为简单，所以就不作过多介绍了。  　　首先，请仔细阅读图1：  　　我们看到本程序由六个基本功能模块构成，各模块的详细分析如下：  　　（1）初始化：首先，建立盘面数组Table[15][15]、对战双方的棋型表Computer[15][15][4]和Player[15][15][4]并将它们清零以备使用；然后初始化显示器、键盘、鼠等输入输出设备并在屏幕上画出棋盘。  　　（2）主循环控制模块：控制下棋顺序，当轮到某方下子时，负责将程序转到相应的模块中去，主要担当一个调度者的角色。  　　（3）玩家下子：当轮到玩家下时，您通过键盘或鼠标在棋盘上落子，程序会根据该点的位置，在Table[15][15]数组的相应地方记录‘2’，以表明该子是玩家下的。  　　（4）盘面分析填写棋型表：本程序核心模块之一，人工智能算法的根本依据！其具体实现方法如下：您在下五子棋时，一定会先根据棋盘上的情况，找出当前最重要的一些点位，如“活三”、“冲四”等；然后再在其中选择落子点。但是，电脑不会像人一样分析问题，要让它知道哪是“活三”、哪是“冲四”，就得在棋盘上逐点计算，一步一步的教它。  　　先来分析己方的棋型，我们从棋盘左上角出发，向右逐行搜索，当遇到一个空白点时，以它为中心向左挨个查找，如果遇到己方的子则记录然后继续，如果遇到对方的子、空白点或边界就停止查找。左边完成后再向右进行同样的操作；最后把左右两边的记录合并起来，得到的数据就是该点横向上的棋型，然后把棋型的编号填入到Computer[x][y][n]中就行了（x、y代表坐标，n=0、1、2、3分别代表横、竖、左斜、右斜四个方向）。而其他三个方向的棋型也可用同样的方法得到，当搜索完整张棋盘后，己方棋型表也就填写完毕了。然后再用同样的方法填写对方棋型表。  　　注意：所有棋型的编号都要事先定义好，越重要的号数越大！  　　OK! 怎么样？有点累了吧？不过千万别泄气！因为好戏还在后头。  　　Let‘s go！  　　（5）电脑下子：有了上面填写的两张棋型表，现在要作的就是让电脑知道在哪一点下子了。其中最简单的计算方法，就是遍历棋型表Computer[15][15][4]和Player[15][15][4]找出其中数值最大的一点，在该点下子即可。但这种算法的弱点非常明显，只顾眼前利益，不能顾全大局，这就和许多五子棋初学者一样犯了“目光短浅”的毛病。  　　要解决这个问题，我们引入‘今后几步预测法’，具体方法是这样的： 首先， 让电脑分析一个可能的点，如果在这儿下子将会形成对手不得不防守的棋型（例如：‘冲四’、‘活三’）；那么下一步对手就会照您的思路下子来防守您，如此一来便完成了第一步的预测。这时再调用模块4对预测后的棋进行盘面分析，如果出现了‘四三’、‘双三’或‘双四’等制胜点，那么己方就可以获胜了（当然对黑棋而言‘双三’、‘双四’是禁手，另当别论）；否则照同样的方法向下分析，就可预测出第二步、第三步……  　　等一等，要是盘面上没有对手必须防的棋型，哪该怎么办呢？进攻不成的话就得考虑防守了，将自己和对手调换一下位置，然后用上面的方法来预测对手的棋，这样既可以防住对手巧妙的攻击，又能侍机发动反击，何乐而不为呢！  　　但是必须告诉大家的是：预测法的运算量相当之大，据我的经验，用Pentium-100预测3步的走法平均需要15秒以上时间，所以建议预测量在5步以内。可别小瞧了这5步，有时它甚至会走出让您拍手叫绝的妙着呢！  　　（6）胜负判断：务须多言，某方形成五子连即获胜；若黑棋走出‘双三’、‘双四’或长连即以禁手判负。  　　到现在为止，整个五子棋软件就基本完成了，其水平大约在中级上下。当然，这种算法并不是最好的，但我相信它的基本思路是正确的。如果您有什么问题或好的想法，欢迎给我发E-mail: softboy@sina.com，我期待着您的见解。 | |