斗地主AI算法——第六章の牌型判断

原文地址：https://blog.csdn.net/sm9sun/article/details/70821453

本章实现了上一章提到的检查当前是否只是一手牌函数ins\_SurCardsType

/\*

检查剩余的牌是否只是一手牌

是：  返回手牌类型数据

不是：返回错误类型（cgERROR）

\*/

CardGroupData ins\_SurCardsType(**int** arr[]);

输入很简单，就是一个状态数组。输出是手牌类型结构

**[cpp]** [view plain](https://blog.csdn.net/sm9sun/article/details/70821453) [copy](https://blog.csdn.net/sm9sun/article/details/70821453)

//牌型组合数据结构

**struct** CardGroupData

{

    //枚举类型

    CardGroupType cgType=cgERROR;

    //该牌的价值

**int**  nValue=0;

    //含牌的个数

**int**  nCount=0;

    //牌中决定大小的牌值，用于对比

**int** nMaxCard=0;

};

其中  cgType通过枚举获取，nValue计算规则参考第四章权值定义，nCount可以通过引入数组算出，nMaxCard是用于比大小的那个牌值。

首先我们要计算出剩余手牌个数，因为这样便于快速筛选分支。

**[cpp]** [view plain](https://blog.csdn.net/sm9sun/article/details/70821453) [copy](https://blog.csdn.net/sm9sun/article/details/70821453)

**int** nCount = 0;

**for** (**int** i = 3; i < 18; i++)

{

    nCount += arr[i];

}

CardGroupData retCardGroupData;

retCardGroupData.nCount = nCount;

以单牌为例，若该牌型满足单牌，则nCount==1，然后我们再找出那张牌。

**[cpp]** [view plain](https://blog.csdn.net/sm9sun/article/details/70821453) [copy](https://blog.csdn.net/sm9sun/article/details/70821453)

//单牌类型

**if** (nCount == 1)

{

    //用于验证的变量

**int** prov = 0;

**int** SumValue = 0;

**for** (**int** i = 3; i < 18; i++)

    {

**if** (arr[i] == 1)

        {

            SumValue = i - 10;

            prov++;

            retCardGroupData.nMaxCard = i;

        }

    }

**if** (prov == 1)

    {

        retCardGroupData.cgType = cgSINGLE;

        retCardGroupData.nValue= SumValue;

**return** retCardGroupData;

    }

}

对牌，三牌，炸弹同理。

三带一的话需要设置两个验证变量，例如三带一单

**[cpp]** [view plain](https://blog.csdn.net/sm9sun/article/details/70821453) [copy](https://blog.csdn.net/sm9sun/article/details/70821453)

**if** (nCount == 4)

    {

        //用于验证的变量

**int** prov1 = 0;

**int** prov2 = 0;

**int** SumValue = 0;

**for** (**int** i = 3; i < 18; i++)

        {

**if** (arr[i] == 3)

            {

                SumValue = i - 10;

                prov1++;

                retCardGroupData.nMaxCard = i;

            }

**if** (arr[i] == 1)

            {

                prov2++;

            }

        }

**if** (prov1 == 1 && prov2 == 1)

        {

            retCardGroupData.cgType = cgTHREE\_TAKE\_ONE;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

三带一对

**[cpp]** [view plain](https://blog.csdn.net/sm9sun/article/details/70821453) [copy](https://blog.csdn.net/sm9sun/article/details/70821453)

**if** (nCount == 5)

{

    //用于验证的变量

**int** prov1 = 0;

**int** prov2 = 0;

**int** SumValue = 0;

**for** (**int** i = 3; i < 16; i++)

    {

**if** (arr[i] == 3)

        {

            SumValue = i - 10;

            prov1++;

            retCardGroupData.nMaxCard = i;

        }

**if** (arr[i] == 2)

        {

            prov2++;

        }

    }

这里我们看，循环改为3~15，因为三牌、对牌是不包括王的。

四带二同理，不过四带二要考虑到带出去的那两张牌型是不是相同

**[cpp]** [view plain](https://blog.csdn.net/sm9sun/article/details/70821453) [copy](https://blog.csdn.net/sm9sun/article/details/70821453)

**if** (nCount == 6)

{

    //用于验证的变量

**int** prov1 = 0;

**int** prov2 = 0;

**int** SumValue = 0;

**for** (**int** i = 3; i < 18; i++)

    {

**if** (arr[i] == 4)

        {

            SumValue = (i - 3) / 2;

            prov1++;

            retCardGroupData.nMaxCard = i;

        }

**if** (arr[i] == 1|| arr[i] == 2)

        {

            prov2+= arr[i];

        }

    }

**if** (prov1 == 1 && prov2 == 2)

    {

        retCardGroupData.cgType = cgFOUR\_TAKE\_ONE;

        retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

    }

}

判断顺子的话用一个变量记录长度，若当前i值等于0并且之前存在i大于0的情况下，即这个长度就是顺子的长度

例如单连：

**[cpp]** [view plain](https://blog.csdn.net/sm9sun/article/details/70821453) [copy](https://blog.csdn.net/sm9sun/article/details/70821453)

**if** (nCount >= 5)

{

    //用于验证的变量

**int** prov = 0;

**int** SumValue = 0;

**int** i;

**for** (i = 3; i < 15; i++)

    {

**if** (arr[i] == 1)

        {

            prov++;

        }

**else**

        {

**if** (prov != 0)

            {

**break**;

            }

        }

    }

    SumValue = i - 10;

**if** (prov == nCount)

    {

        retCardGroupData.nMaxCard = i-1;

        retCardGroupData.cgType = cgSINGLE\_LINE;

        retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

    }

}

王炸就更好判断了，直接判断arr[17]和arr[16]就好了

下面贴出完整代码：

**[cpp]** [view plain](https://blog.csdn.net/sm9sun/article/details/70821453) [copy](https://blog.csdn.net/sm9sun/article/details/70821453)

/\*

检查剩余的牌是否只是一手牌

是：  返回手牌类型数据

不是：返回错误类型（cgERROR）

\*/

CardGroupData ins\_SurCardsType(**int** arr[])

{

**int** nCount = 0;

**for** (**int** i = 3; i < 18; i++)

    {

        nCount += arr[i];

    }

    CardGroupData retCardGroupData;

    retCardGroupData.nCount = nCount;

    //单牌类型

**if** (nCount == 1)

    {

        //用于验证的变量

**int** prov = 0;

**int** SumValue = 0;

**for** (**int** i = 3; i < 18; i++)

        {

**if** (arr[i] == 1)

            {

                SumValue = i - 10;

                prov++;

                retCardGroupData.nMaxCard = i;

            }

        }

**if** (prov == 1)

        {

            retCardGroupData.cgType = cgSINGLE;

            retCardGroupData.nValue= SumValue;

**return** retCardGroupData;

        }

    }

    //对牌类型

**if** (nCount == 2)

    {

        //用于验证的变量

**int** prov = 0;

**int** SumValue = 0;

**int** i = 0;

**for** (i = 3; i < 16; i++)

        {

**if** (arr[i] == 2)

            {

                SumValue = i - 10;

                prov++;

                retCardGroupData.nMaxCard = i;

            }

        }

**if** (prov == 1)

        {

            retCardGroupData.cgType = cgDOUBLE;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //三条类型

**if** (nCount == 3)

    {

        //用于验证的变量

**int** prov = 0;

**int** SumValue = 0;

**int** i = 0;

**for** (i = 3; i < 16; i++)

        {

**if** (arr[i] == 3)

            {

                SumValue = i - 10;

                prov++;

                retCardGroupData.nMaxCard = i;

            }

        }

**if** (prov == 1)

        {

            retCardGroupData.cgType = cgTHREE;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //三带一单

**if** (nCount == 4)

    {

        //用于验证的变量

**int** prov1 = 0;

**int** prov2 = 0;

**int** SumValue = 0;

**for** (**int** i = 3; i < 18; i++)

        {

**if** (arr[i] == 3)

            {

                SumValue = i - 10;

                prov1++;

                retCardGroupData.nMaxCard = i;

            }

**if** (arr[i] == 1)

            {

                prov2++;

            }

        }

**if** (prov1 == 1 && prov2 == 1)

        {

            retCardGroupData.cgType = cgTHREE\_TAKE\_ONE;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //三带一对

**if** (nCount == 5)

    {

        //用于验证的变量

**int** prov1 = 0;

**int** prov2 = 0;

**int** SumValue = 0;

**for** (**int** i = 3; i < 16; i++)

        {

**if** (arr[i] == 3)

            {

                SumValue = i - 10;

                prov1++;

                retCardGroupData.nMaxCard = i;

            }

**if** (arr[i] == 2)

            {

                prov2++;

            }

        }

**if** (prov1 == 1 && prov2 == 1)

        {

            retCardGroupData.cgType = cgTHREE\_TAKE\_TWO;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //四带两单

**if** (nCount == 6)

    {

        //用于验证的变量

**int** prov1 = 0;

**int** prov2 = 0;

**int** SumValue = 0;

**for** (**int** i = 3; i < 18; i++)

        {

**if** (arr[i] == 4)

            {

                SumValue = (i - 3) / 2;

                prov1++;

                retCardGroupData.nMaxCard = i;

            }

**if** (arr[i] == 1|| arr[i] == 2)

            {

                prov2+= arr[i];

            }

        }

**if** (prov1 == 1 && prov2 == 2)

        {

            retCardGroupData.cgType = cgFOUR\_TAKE\_ONE;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //四带两对

**if** (nCount == 8)

    {

        //用于验证的变量

**int** prov1 = 0;

**int** prov2 = 0;

**int** SumValue = 0;

**for** (**int** i = 3; i < 16; i++)

        {

**if** (arr[i] == 4)

            {

                SumValue = (i - 3) / 2;

                prov1++;

                retCardGroupData.nMaxCard = i;

            }

**if** (arr[i] == 2|| arr[i] == 4)

            {

                prov2+= arr[i]/2;

            }

        }

               //注意这里prov2==4因为四牌也是两个对

**if** (prov1 == 1 && prov2 == 4)

        {

            retCardGroupData.cgType = cgFOUR\_TAKE\_TWO;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //炸弹类型

**if** (nCount == 4)

    {

        //用于验证的变量

**int** prov = 0;

**int** SumValue = 0;

**for** (**int** i = 3; i < 16; i++)

        {

**if** (arr[i] == 4)

            {

                SumValue += i - 3 + 7;

                prov++;

                retCardGroupData.nMaxCard = i;

            }

        }

**if** (prov == 1)

        {

            retCardGroupData.cgType = cgBOMB\_CARD;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //王炸类型

**if** (nCount == 2)

    {

**int** SumValue = 0;

**if** (arr[17] > 0 && arr[16] > 0)

        {

            SumValue = 20;

            retCardGroupData.nMaxCard = 17;

            retCardGroupData.cgType = cgKING\_CARD;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //单连类型

**if** (nCount >= 5)

    {

        //用于验证的变量

**int** prov = 0;

**int** SumValue = 0;

**int** i;

**for** (i = 3; i < 15; i++)

        {

**if** (arr[i] == 1)

            {

                prov++;

            }

**else**

            {

**if** (prov != 0)

                {

**break**;

                }

            }

        }

        SumValue = i - 10;

**if** (prov == nCount)

        {

            retCardGroupData.nMaxCard = i-1;

            retCardGroupData.cgType = cgSINGLE\_LINE;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //对连类型

**if** (nCount >= 6)

    {

        //用于验证的变量

**int** prov = 0;

**int** SumValue = 0;

**int** i;

**for** (i = 3; i < 15; i++)

        {

**if** (arr[i] == 2)

            {

                prov++;

            }

**else**

            {

**if** (prov != 0)

                {

**break**;

                }

            }

        }

        SumValue = i - 10;

**if** (prov \* 2 == nCount)

        {

            retCardGroupData.nMaxCard = i - 1;

            retCardGroupData.cgType = cgDOUBLE\_LINE;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //三连类型

**if** (nCount >= 6)

    {

        //用于验证的变量

**int** prov = 0;

**int** SumValue = 0;

**int** i;

**for** (i = 3; i < 15; i++)

        {

**if** (arr[i] == 3)

            {

                prov++;

            }

**else**

            {

**if** (prov != 0)

                {

**break**;

                }

            }

        }

        SumValue = (i - 3) / 2;

**if** (prov \* 3 == nCount)

        {

            retCardGroupData.nMaxCard = i - 1;

            retCardGroupData.cgType = cgTHREE\_LINE;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //三带一连类型

**if** (nCount >= 8)

    {

        //用于验证的变量

**int** prov1 = 0;

**int** SumValue = 0;

**int** i, j;

**for** (i = 3; i < 15; i++)

        {

**if** (arr[i] >= 3)

            {

                prov1++;

            }

**else**

            {

**if** (prov1 != 0)

                {

**break**;

                }

            }

        }

        SumValue = (i - 3)/2;

**if** (prov1 \* 4 == nCount)

        {

            retCardGroupData.nMaxCard = i - 1;

            retCardGroupData.cgType = cgTHREE\_TAKE\_ONE\_LINE;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    //三带二连类型

**if** (nCount >= 10)

    {

        //用于验证的变量

**int** prov1 = 0;

**int** prov2 = 0;

**int** SumValue = 0;

**int** i, j;

**for** (i = 3; i < 15; i++)

        {

**if** (arr[i] == 3)

            {

                prov1++;

            }

**else**

            {

**if** (prov1 != 0)

                {

**break**;

                }

            }

        }

**for** (j = 3; j < 16; j++)

        {

**if** (arr[j] == 2|| arr[j] == 4)

            {

                prov2+= arr[j]/2;

            }

        }

        SumValue = (i - 3) / 2;

**if** (prov1 == prov2&&prov1 \* 5 == nCount)

        {

            retCardGroupData.nMaxCard = i - 1;

            retCardGroupData.cgType = cgTHREE\_TAKE\_TWO\_LINE;

            retCardGroupData.nValue = SumValue;

**return** retCardGroupData;

        }

    }

    retCardGroupData.cgType = cgERROR;

**return** retCardGroupData;

}

/\*

检查剩余的牌是否只是一手牌（vector重载）

是：  返回手牌类型数据

不是：返回错误类型（cgERROR）

\*/

CardGroupData ins\_SurCardsType(vector<**int**>list)

{

**int** arr[18];

    memset(arr, 0, **sizeof**(arr));

**for** (vector<**int**>::iterator iter = list.begin(); iter != list.end(); iter++)

    {

        arr[\*iter]++;

    }

**return** ins\_SurCardsType(arr);

}

怎么样，这么多牌型枚举是不是很头疼？放心吧，接下来的主动出牌算法、被动出牌算法的枚举会更头疼！



所以~从下一章开始我们就要讲出牌的策略了，首先是被动出牌。