报名号：#1989

2008 年 第一届“数学中国杯”

数学建模网络挑战赛

承 诺 书

我们仔细阅读了首届“数学中国杯”数学建模网络挑战赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的, 如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们允许数学中国网站(www.madio.net)公布论文，以供网友之间学习交流，数学中国网站以非商业目的的论文交流不需要提前取得我们的同意。

我们的参赛报名号为：

参赛队员 (签名) ：

队员 1：钮历宇

队员 2：徐晓峰

队员 3：曹珂

参赛队教练员 (签名)： 许立纬

参赛队伍组别：大学组

报名号：#1989

2008 年 第一届“数学中国杯”

数学建模网络挑战赛

编 号 专 用 页

参赛队伍的参赛号码：（请各个参赛队提前填写好）：

竞赛统一编号（由竞赛组委会送至评委团前编号）：

竞赛评阅编号（由竞赛评委团评阅前进行编号）：

报名号：#1989

2008 年 第一届“数学中国杯”
数学建模网络挑战赛

题　　目　飞机对战游戏

关 键 词　权值 搜索算法 反馈信息 非合作性完全信息动态博弈

摘　　要：

问题一：设计 C++程序，计算机*生成飞机全部 48 种摆放情况*，并对每种情况给机头所在位置赋 10，机身所在位置赋 1，其余位置赋 0，存入二维矩阵 plant[][]中。在每种摆放情况等可能出现的假设下，每次攻击之前*将 48 张图的权值累加，并攻击权值累加矩阵权值最高的点（即机头最有可能出现的点）*。在下一次攻击之时，利用之前所有反馈信息，运用 JUDGEON(),JUDGEOFF()函数进行判断，把所有不可能出现的图权值全置 0，再次累加并攻击机头最有可能出现的点。如此累加，算出 *8 步以内可打中机头，打中机头步数平均值为 4.58333。*

问题二：沿用第一题中的打击思想，计算机*生成 2352 种飞机摆放情况*，并针对机头最有可能出现的点进行打击。相比第一题，*增加一个 JUDGEOK()函数，在打中第一架飞机机头时候调用。*最后进行随机抽样，算出 *29 步以内可打中机头，打中机头步数期望为 12.4893。*

问题三：此题为非合作性完全信息动态博弈，在所有玩家采取同样的打击算法的前提下，*策略的关键在于如何有效利用自己的先手权，并结合当前所得反馈信息，选取下一步攻击的目标*，并争取获得更多的分数。

参赛队号　1989

所选题目　D 题

参赛密码 _____
（由组委会填写）

目录

一、问题重述

D 题　飞机对战游戏

（适合大学组）

有一种在学生中间比较流行的双方对战游戏。在游戏前双方各准备一张坐标纸，在上面分别制作 7×7 的方格，如图 1 所示。在自己的方格中画一架飞机，飞机呈“士”字形，其中上面的一长横占 5 个格子，下面的短横占 3 个格子，一竖占 4 个格子，最上面突出的一个格子代表机头。所画飞机的位置以及机头的指向由游戏者自己决定，游戏结束前双方不能互看对方的坐标纸。游戏时双方交替用“炮弹”打击对方，攻击的一方报告“炮弹”打击的位置，被攻击的一方报告是否命中飞机。例如：被攻击方的飞机画法如图 1 所示，攻击者报告“炮弹”的打击位置是（4，3），从图中可知，“炮弹”恰好落在飞机所在的红色格子上面，被攻击方报告飞机被击中，接下来刚才的被攻击方变成攻击方进行上面的攻击步骤，双方交替攻击对方，如果某一方被命中机头，游戏结束，被命中机头的一方失败。游戏双方都在通过打击后对方的反馈信息来猜测对方飞机的位置。

游戏比赛采用 19 局 10 胜制。

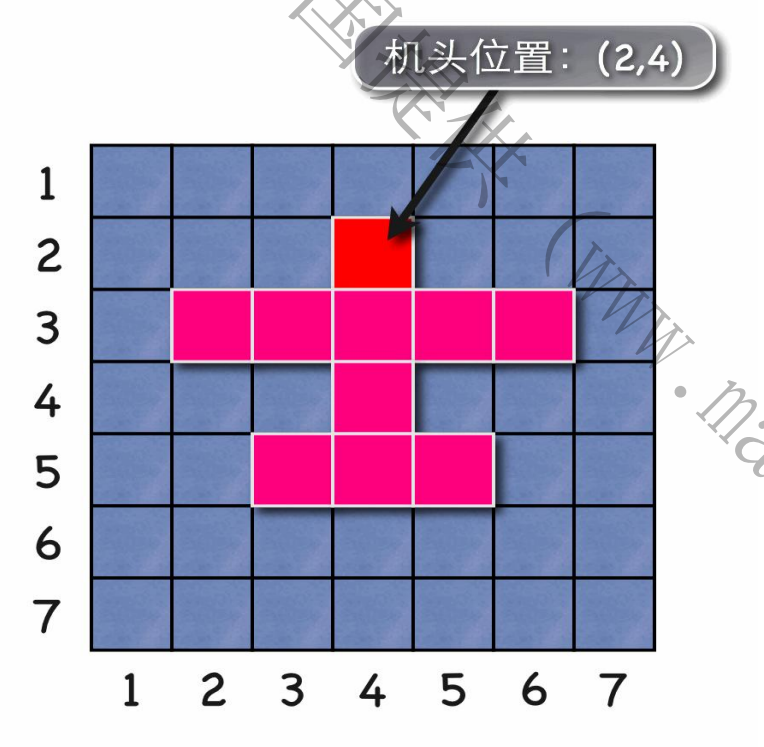


图 1. 飞机示意图

问题一：设计一个人机对战的“飞机对战”游戏。要求先由计算机进行攻击，以计算机取胜为目标，给出进行游戏的策略。

问题二：考虑在 9×9 坐标纸上画两架飞机的游戏方式，两架飞机所占的格子不能重合，游戏方法同上。其中一架飞机被命中机头时要报告有一架飞机被击落。当某方的两架飞机都被击落时游戏结束，被击落方失败。分析这种游戏方式与只画一架飞机的游戏方式在策略上的不同点。

问题三：如果将问题二中的游戏方式设计为一个网络游戏，由三个真人对战，三人轮流作为攻击方，攻击方可以选择另外两方之一作为攻击对象，每次只能发射一发“炮弹”。如果某一方的两架飞机均被击中，他将退出这一局游戏，另外两方仍将继续游戏直到二

者决出胜负。打中机头可以得 1 分，打中机头并把被攻击方踢出局可以得 3 分，打中飞机其它部位或者未击中不得分，比赛采用 18 局，最后总得分（累加每局得到的分数）最高的一方获胜，如果出现平分，加赛一局决定胜负。考虑每局都首先由你作为攻击方，设计一套游戏策略，使你能在比赛中取胜。

注：假定游戏中，不存在任意两方联合的可能。

二、问题的基本假设

问题一和问题二的假设：

1 问题一中飞机全部 48 种摆放情况等概率出现；问题二中飞机 2352 种摆放情况等概率出现，两架飞机无差别，无先后次序。

2 每局计算机攻击策略不变，即尽可能快地打中游戏人的机头，以本局取胜为目标。各局之间胜负不影响。

3 仅给出计算机的攻击策略，不考虑计算机被攻击的情况。

4 假设与计算机对战的玩家采取一种低劣的打击算法，计算计算机在 19 局 10 胜制的比赛中获胜的概率，对模型进行评价。

问题三的假设：

1 所有玩家采取同样的算法进行攻击。

2 其中一方机身及机头被击中的信息公开。

3 所有玩家仅以自己尽可能多得分为目标，不存在任意两方联合的可能。

4 所有玩家每局均采取同样的策略，与之前得分情况无关。

5 若出现三人同分，加赛在三人之间进行；

若出现两人平分，加赛仅在两人之间进行（避免出现第三人“起死回生”的情况）。

6 此游戏为非合作性完全信息动态博弈。

三、名词解释

1. 绝对优势：开局时未受任何攻击
2. 绝对劣势：开局时被两人同时攻击
3. 反馈信息：打击后或得的击中与否的信息

四、符号说明

Gi：问题三游戏当前得分。

五、模型分析

问题一：

依题目要求，游戏以打中对方机头为取胜目标。因此在每一轮打击之前，仅综合之前所有反馈信息考虑对方机头最有可能出现的点，并不考虑自身飞机情况。

在第 K+1 次打击之前，应综合前 K 次打击反馈信息。若点 P_N 反馈得知是机身，则将 P_N 点不为机身的情况剔除不作考虑。反之，若点 P_N 反馈得知非机身非机头，则将 P_N 点为机身或机头的情况剔除不作考虑。总结反馈信息，找出所有机头有可能分布，并每次打击概率最高的点，以达到获胜的目的。

将所有有可能出现的飞机摆放情况用二阶矩阵表示，并为机头赋值 10，机身赋值 1，其余赋值 0。由以下累加思想：

	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0
3	0	0	1	0	1	0	0
4	0	0	1	1	1	10	0
5	0	0	1	0	1	0	0
6	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0

+

	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	0	0	10	1	0	0
3	0	1	1	1	1	1	0
4	0	0	1	1	1	0	0
5	0	0	1	1	1	0	0
6	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0

=

	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	0	0	10	1	0	0
3	0	1	2	1	2	1	0
4	0	0	1	1	1	10	0
5	0	0	2	1	2	0	0
6	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0

可以将 48 张图全部累加，得到最终累加权值图。

由于机头权值远大于机身，总累加权值图中权值最高点必为机头最有可能出现的点，计算机就针对此点进行攻击。

问题二：

与问题一相比，策略的不同点主要在于飞机数量的不同。这就直接导致飞机可能摆放情况大大增多。再者由于两架飞机无差别，在程序遍历生成飞机摆放情况时有可能会造成重复。

因此如何罗列所有飞机摆放，是本题最大的难点。

问题三：

在游戏过程中，A,B,C 选择权一样，三者不存在优劣势。

本游戏中，反馈信息量的多少来做出下步打击判断的，某玩家被击中次数越多，其他玩家从他那里得分的概率就越大，因此在情况允许的条件下，打击提供反馈信息量最大的玩家是最优选择。

A 相对于 B,C 的优势在于享有先攻权。每局开局前，A 要根据当前比分和 B,C 心理来确定攻击目标。B,C 要根据当前比分和 A 的决策来做决定。

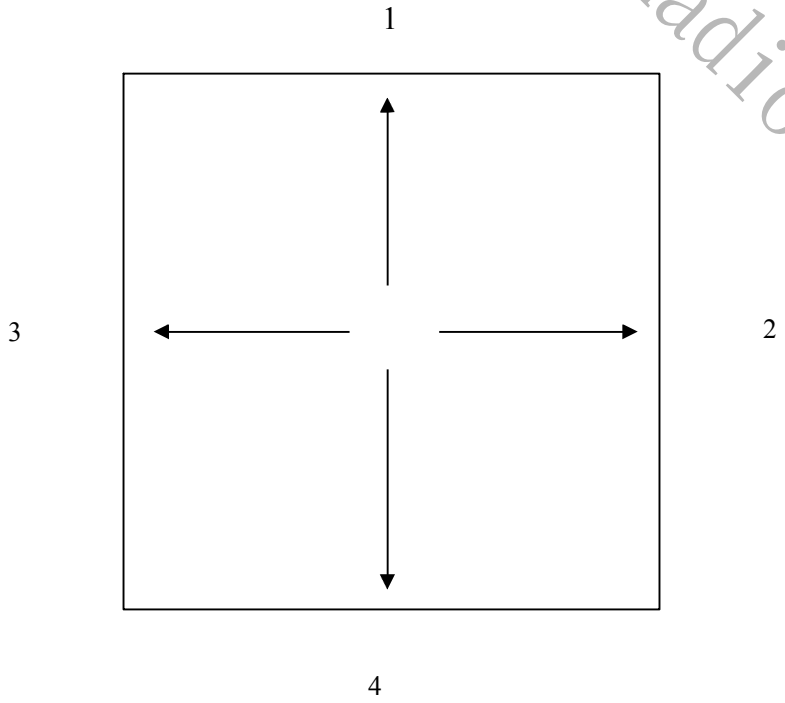
有上述分析可知，制胜的关键在于如何通过行使先攻权来引导 B,C 做出对自己有利的决策。

六、模型求解

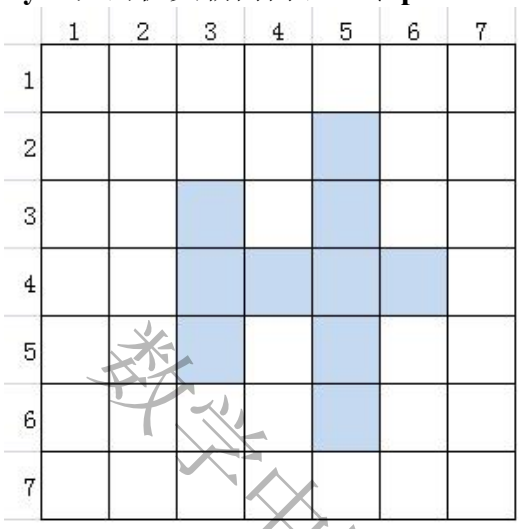
问题一：

1 飞机摆放位置的表示

为了表示飞机的其中一种摆放方式，定义一个名为 Node 的结构体，其含有 x 和 y 两个元素，用以表示飞机机头坐标。再用变量 point 表示飞机朝向（其朝向表示如下图）。



例如，由下图所示，飞机机头在(6,4)上,则将此值传给 Node 的一个对象，其中 **t.x=6，t.y=4**；而机头朝向向右，令 **point＝2**。这就决定了这架飞机的摆放位置。



2 所有飞机情况的生成

通过 **make()**函数，我们可以将所有飞机摆放情况罗列出来，并且通过机头位置和飞机朝向判断飞机机身所在位置，为下一步的权值赋予做准备。

定义二维数组 plant[7][7]，用以存放权值图。

分别用数组表示横向飞机模型 int part1[3][5]={1,1,1,1,1,0,0,1,0,0,0,1,1,1,0},

以及纵向飞机模型 int part2[5][3]={1,0,0,1,0,1,1,1,1,1,0,1,1,0,0};

并在 plant[7][7]的 7×7 格子里遍历，再调用 over()函数进行判断，若越出 7×7 的界限，则全图赋权值为 0；否则机头赋 10，机身赋 1，其余地方赋 0，最后传递给 plant[7][7]。

这样，我们便得到 196 张权值二维矩阵，而其中只有 48 张为有效矩阵（矩阵权值不全为 0）。

3 初始打击位置的确定

对 48 种飞机摆放情况，我们做了以下罗列

		1	1	1		
		1	1	1		
1	1	2	3	2	1	1
1	1	3	4	3	1	1
1	1	2	3	2	1	1
		1	1	1		
		1	1	1		

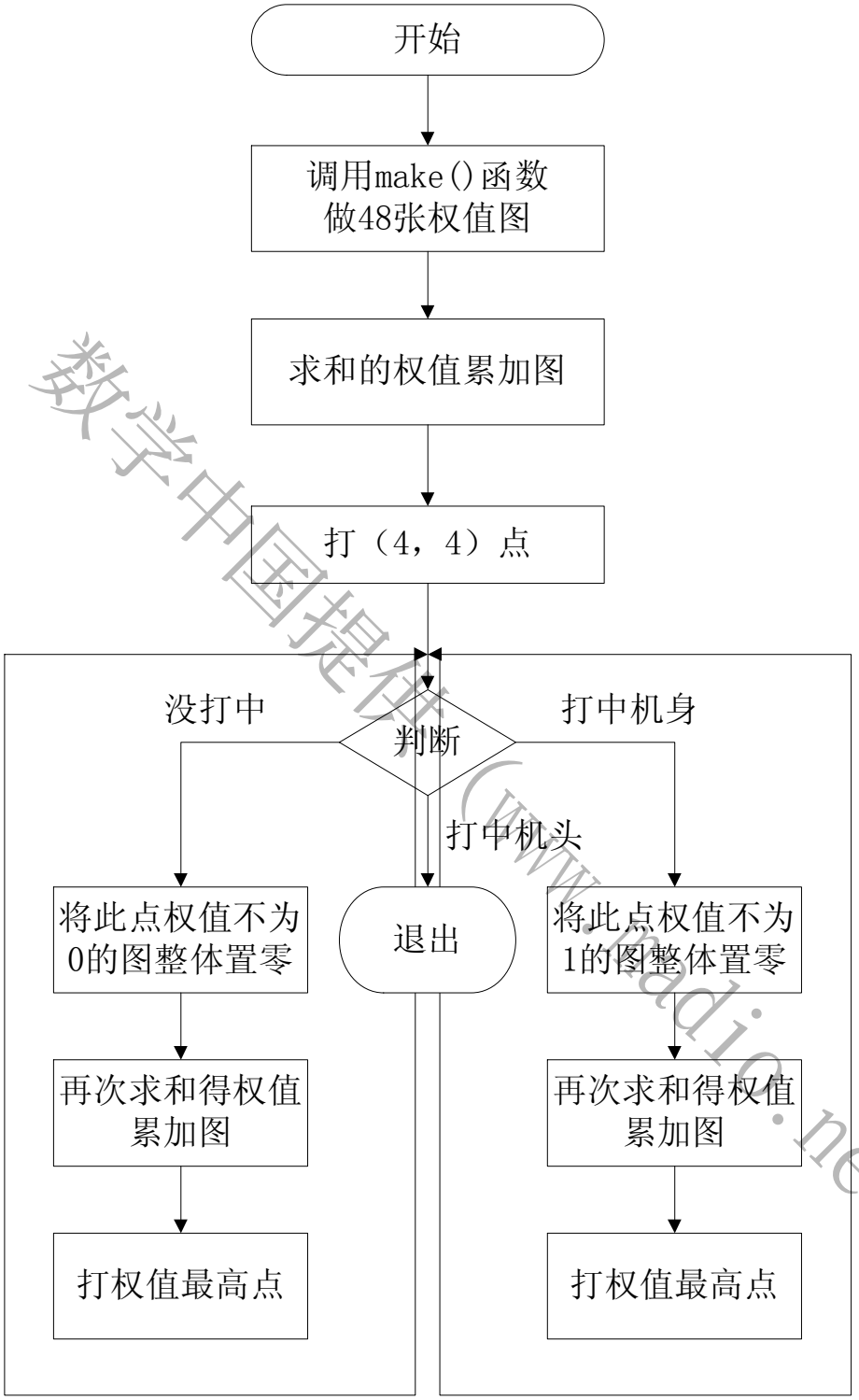
其中，表格中的数值表示机头在这个点可能出现的次数。由此可见，中间的(4,4)点出现机头的概率最高，为 $\frac{4}{48}$。本着每步尽可能打中机头的原则，我们在每一局第一步都首先击打(4,4)点。

4 根据反馈信息调用判断函数确定击打目标

若第 K 次打击点(X_K, Y_K)，若得到反馈信息为打中机身，调用 **JUDGEON()**函数，将点(X_K, Y_K)权值不为 1 的权值矩阵整体赋零，并且将这 196 个矩阵权值累加，传递给 **temp.plant[][]**中。然后用 **Max()**函数遍历 **temp.plant[][]**，找出权值最高的点（权值相等仅输出一个点），定为第 K+1 次打击的对象。

若第 K 次打击点(X_K, Y_K)，若得到反馈信息为既没有打中机身，也没有打中机头，调用 **JUDGEOFF()**函数，将点(X_K, Y_K)权值不为 0 的权值矩阵整体赋零，并且将这 196 个矩阵权值累加，传递给 **temp.plant[][]**中。然后用 **Max()**函数遍历 **temp.plant[][]**，找出权值最高的点（权值相等仅输出一个点），定为第 K+1 次打击的对象。

5 程序流程图



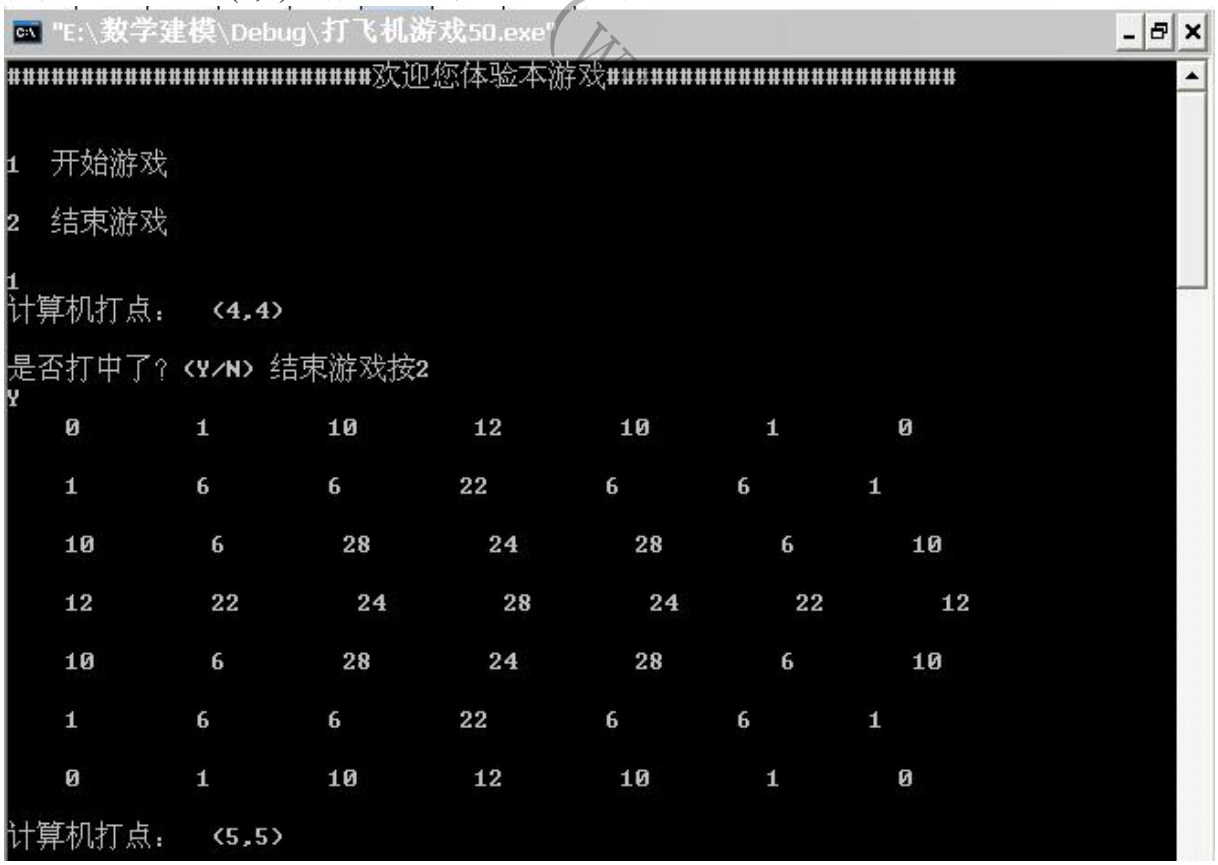
6 程序实现实例

下面把飞机隐藏在方向朝上，机头位置为(4,2)的地方，调用程序进行分析。

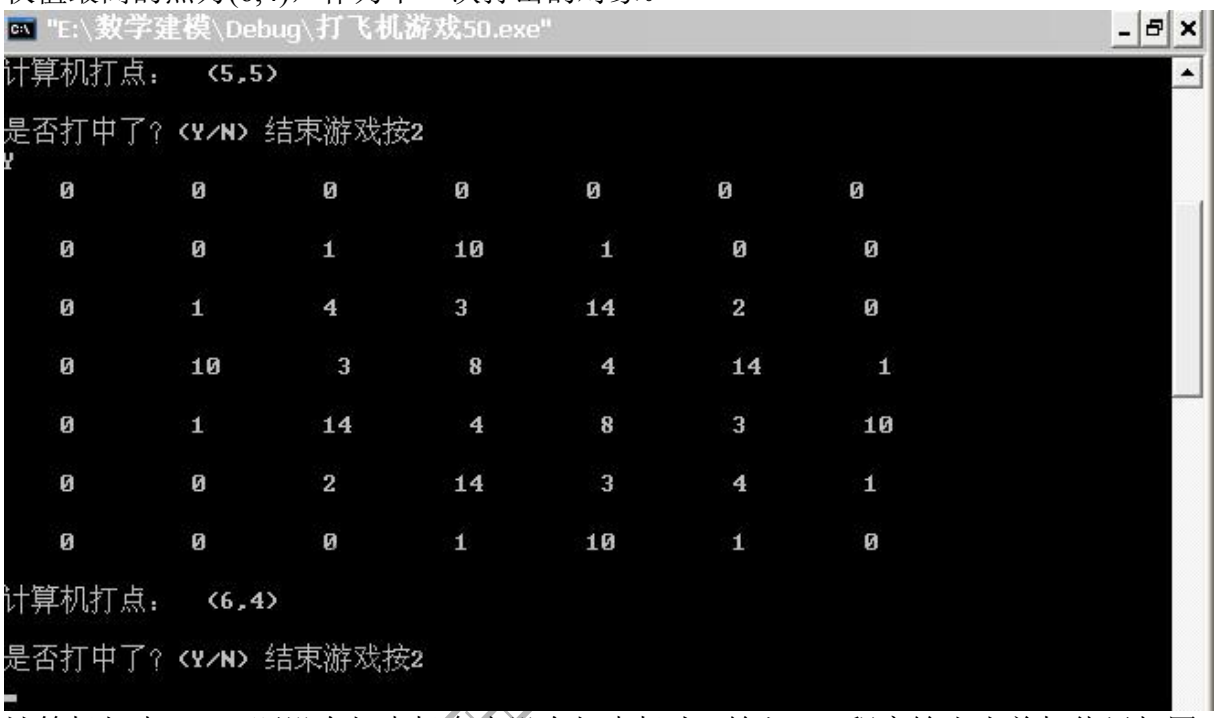
	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	0	0	10	0	0	0
3	0	1	1	1	1	1	0
4	0	0	0	1	0	0	0
5	0	0	1	1	1	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0

运行程序，按 1 开始游戏。

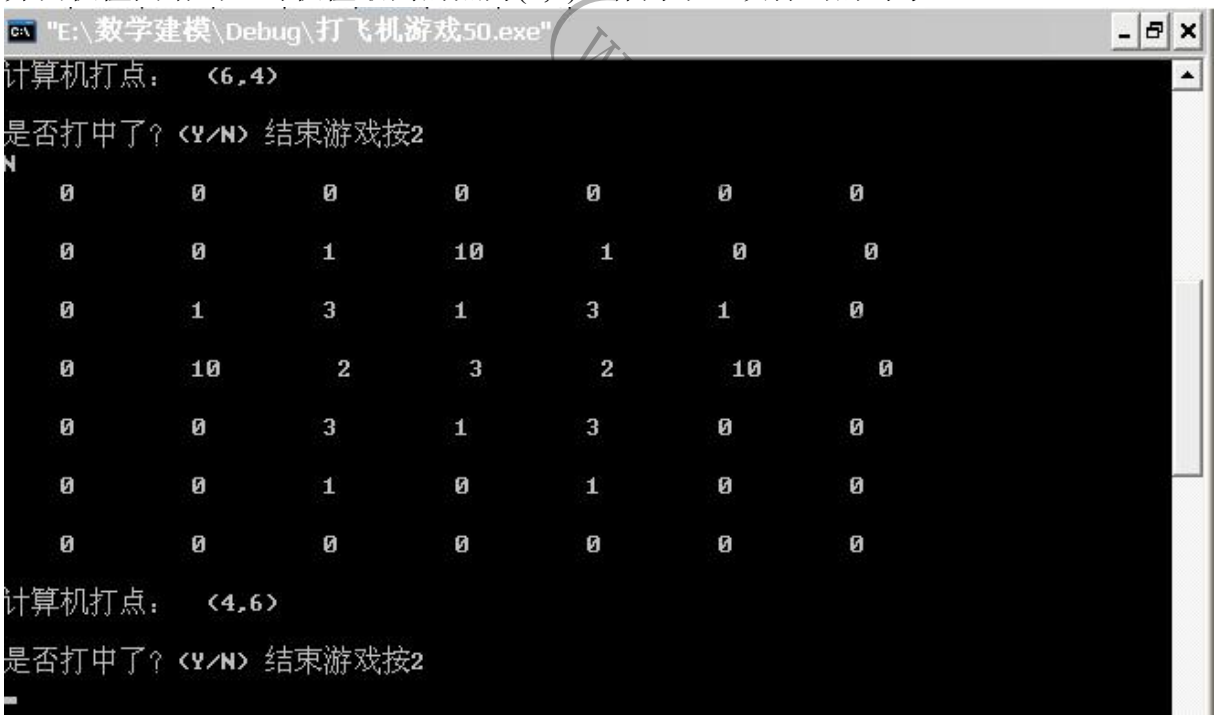
计算机首先打点(4,4)，打中机身，输入 Y，程序输出权值累加图，并由权值图给出此时权值最高的点为(5,5)，作为下一次打击的对象。



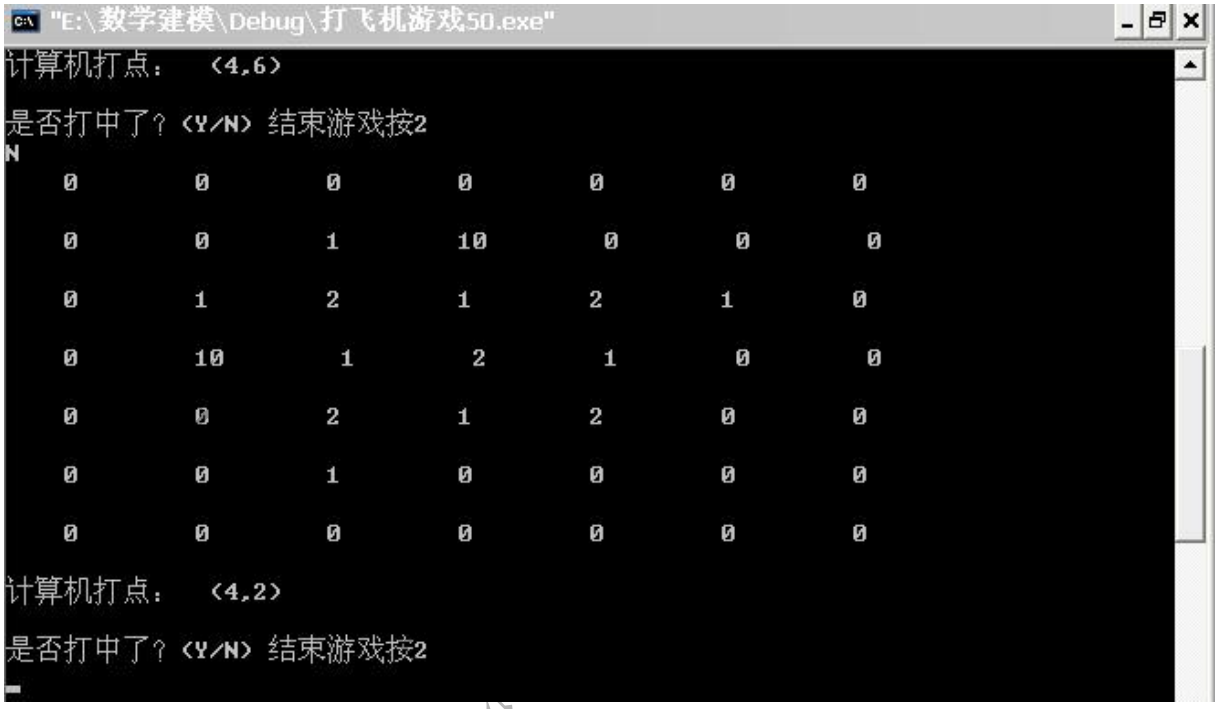
计算机打点(5,5)，打中机身，输入 Y，程序输出当前权值累加图，并由权值图给出此时权值最高的点为(6,4)，作为下一次打击的对象。



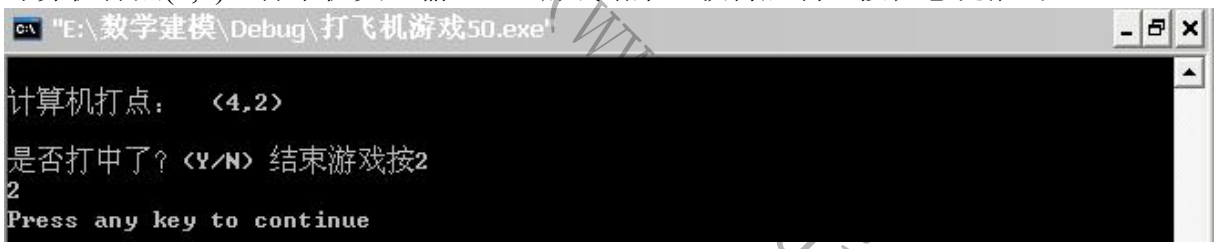
计算机打点(6,4)，既没有打中机身也没有打中机头，输入 N，程序输出当前权值累加图，并由权值图给出此时权值最高的点为(4,6)，作为下一次打击的对象。



计算机打点(4,6)，既没有打中机身也没有打中机头，输入 N，程序输出当前权值累加图，并由权值图给出此时权值最高的点为(4,2)，作为下一次打击的对象。



计算机打点(4,2)，打中机头，输入 2，游戏结束，获得胜利，按任意键推出。



问题二:

在问题二中，依然仿照问题一，采取每次攻击机头最有可能出现位置的策略。但由于增添了一架飞机，飞机权值图的生成函数将进行全面修改，且增添一个反馈信息判断函数 JUDGEOK()，在打中第一架飞机机头时候调用。

1 飞机生成函数 create　　首先生成以下四架飞机的 5×5 矩阵。

		100		
1	**1**	**1**	**1**	**1**
		1		
	1	**1**	**1**	

	1	1	1	
		1		
1	1	1	1	1
		100		

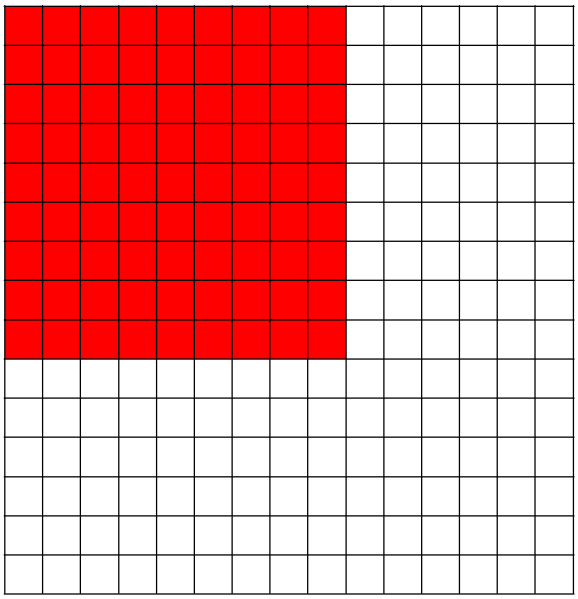
	1			
	1		**1**	
100	**1**	**1**	**1**	
	1		**1**	
	1			

		1		
1		**1**		
1	**1**	**1**	**100**	
1		**1**		
		1		

然后让其中两个矩阵在一个 15×15 矩阵中遍历摆放，并进行以下判断：

判断一：15×15 矩阵中不存在权值为 2，101，200 的点（即不存在两飞机重叠）。

判断二：15×15 矩阵的前 9×9 方格权值之和为 218。



若同时符合以上两个条件，则说明此飞机图为有效图。

同时我们还注意到，若两机头同向，每种情况均重复了两遍，因此我们在生成飞机权值图时分成两类思考：

情况一：两机头同向，将权值整体减半。

情况二：两机头不同向，不作更改。

这样，在最后权值累加的时候，将不会出现重复的情况。

注：之所以给机头赋值高达100，是为了避免某位置出现机身可能过多，影响机头最有可能位置的判断。

2 选择第一轮攻击位置

由累加权值图：

0	99	6676	5294	4580	5294	6676	99	0
99	248	5108	3461	2673	3461	5108	248	99
6676	5108	7860	12695	11306	12695	7860	5108	6676
5294	3461	12695	15526	13686	15526	12695	3461	5294
4580	2673	11306	13686	11828	13686	11306	2673	4580
5294	3461	12695	15526	13686	15526	12695	3461	5294
6676	5108	7860	12695	11306	12695	7860	5108	6676
99	248	5108	3461	2673	3461	5108	248	99
0	99	6676	5294	4580	5294	6676	99	0

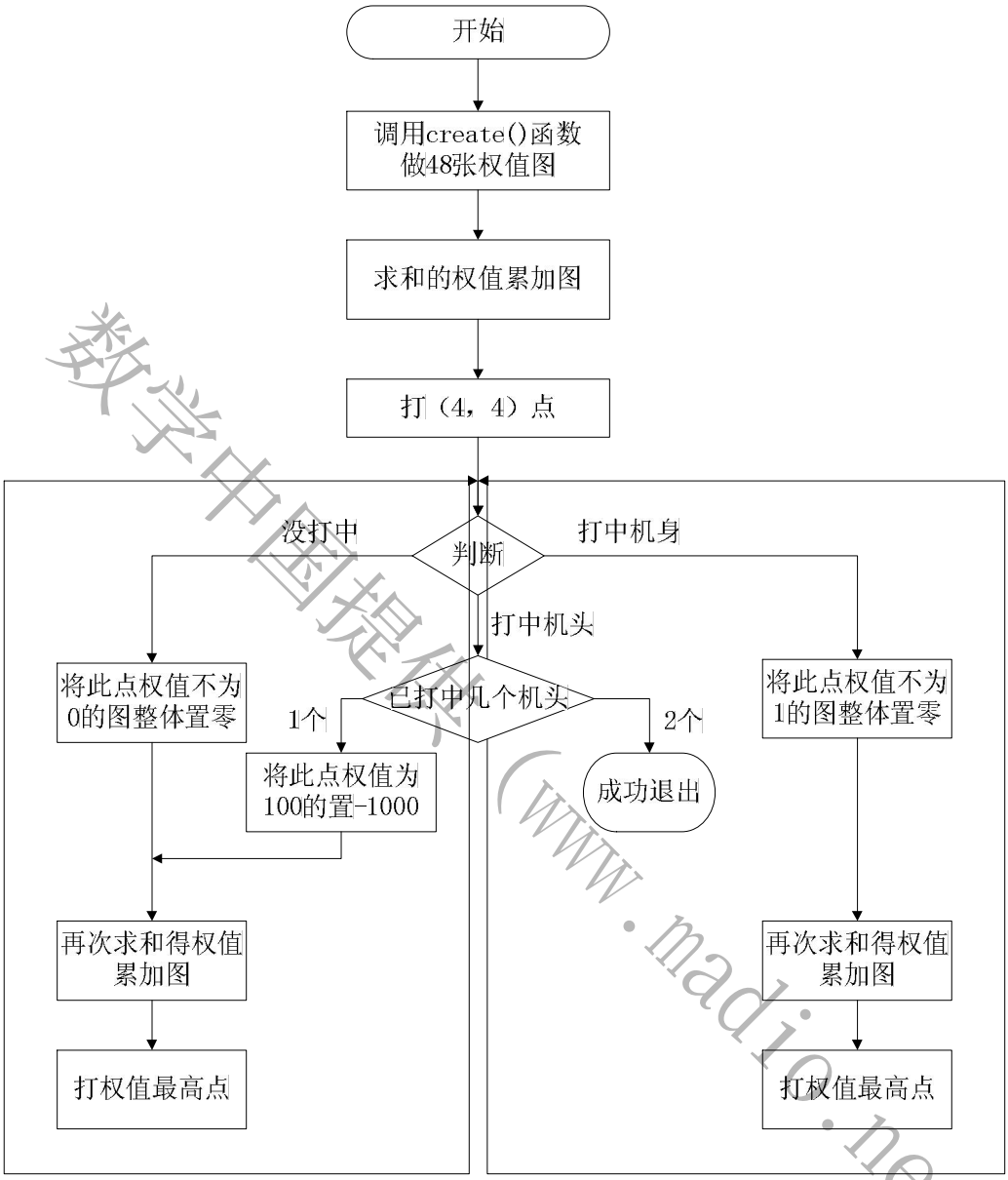
可知(4,4)、(4,6)、(6,4)、(6,6)权值最高，选择(4,4)为第一轮打击目标。

3 新增判断函数 JUDGEOK()

若第 K 次打击点(X_K, Y_K)，得到反馈信息为打中第一架飞机机头，调用 **JUDGEOK()** 函数，将点(X_K, Y_K)权值不为 100 的权值矩阵机身赋 0，机头赋－1000，并且将这 2352 个矩阵权值累加，找出权值最高的点（权值相等仅输出一个点），定为第 K+1 次打击的对象。

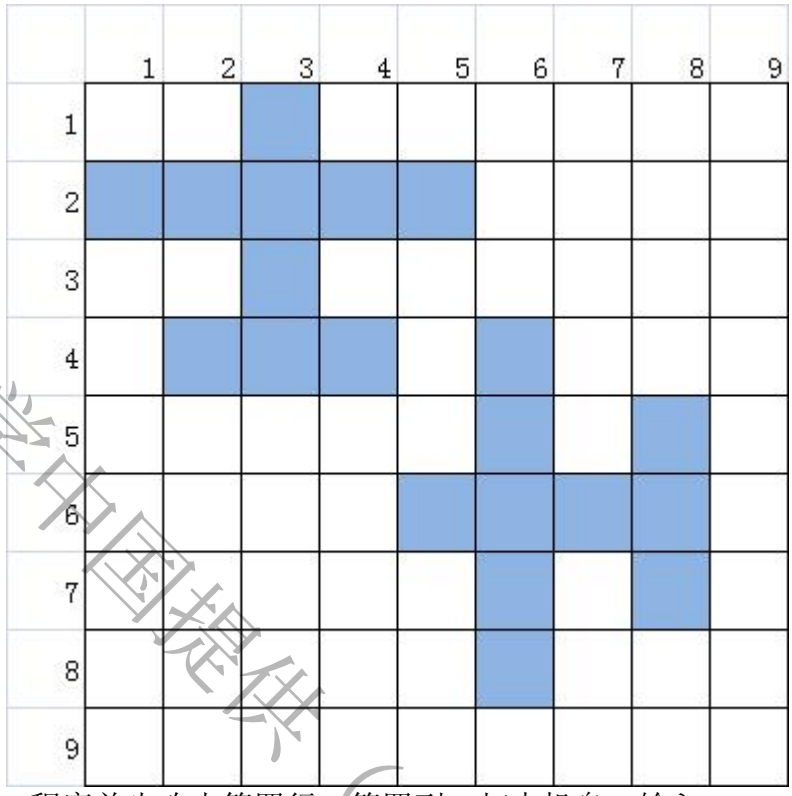
注：之所以机头置为－1000，是为了保证这一点在以后永远不会再被打击。

4 程序流程图



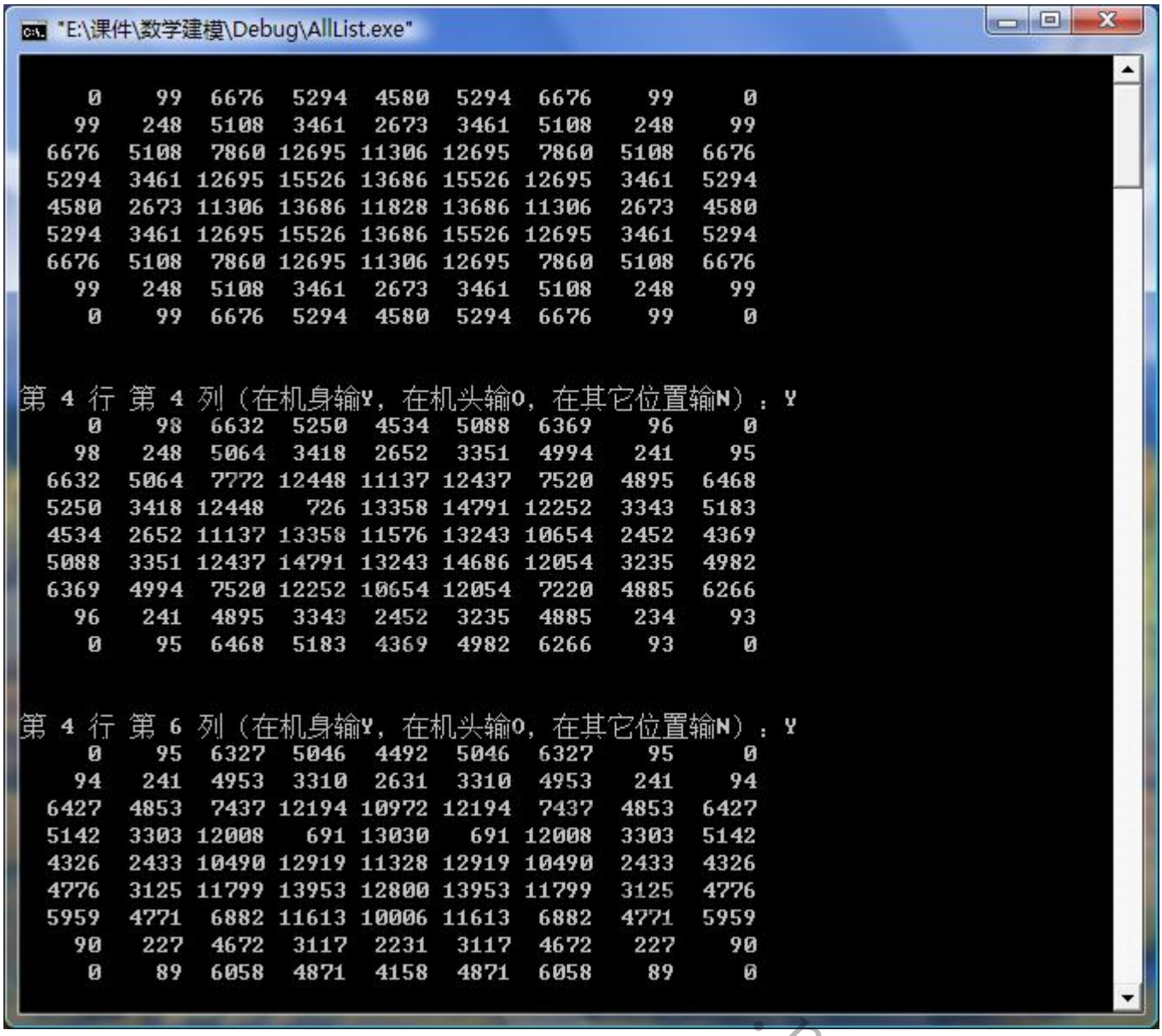
5 程序实现实例

现将两架飞机隐藏如下，执行程序。

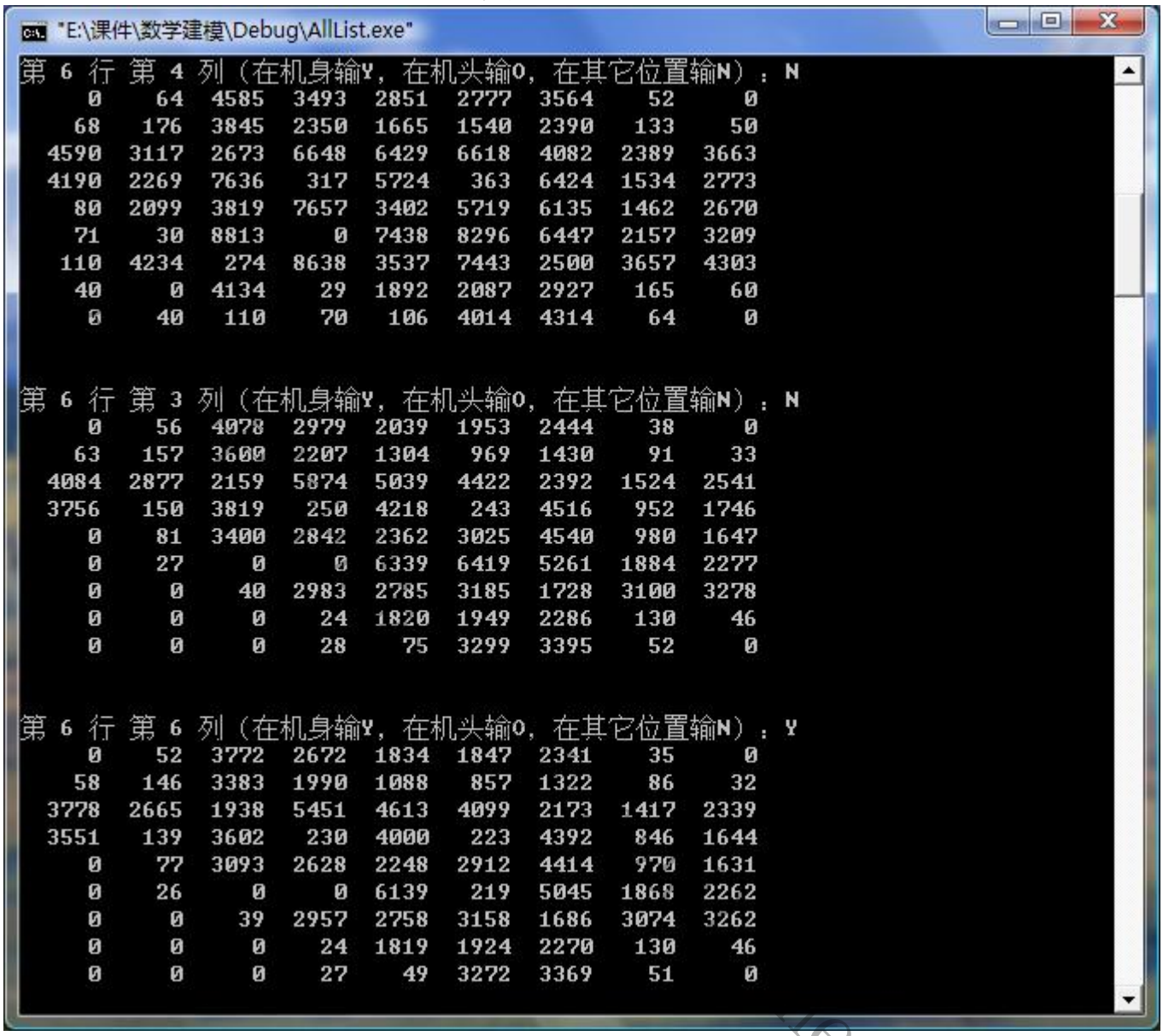


程序首先攻击第四行，第四列，打中机身，输入 Y。

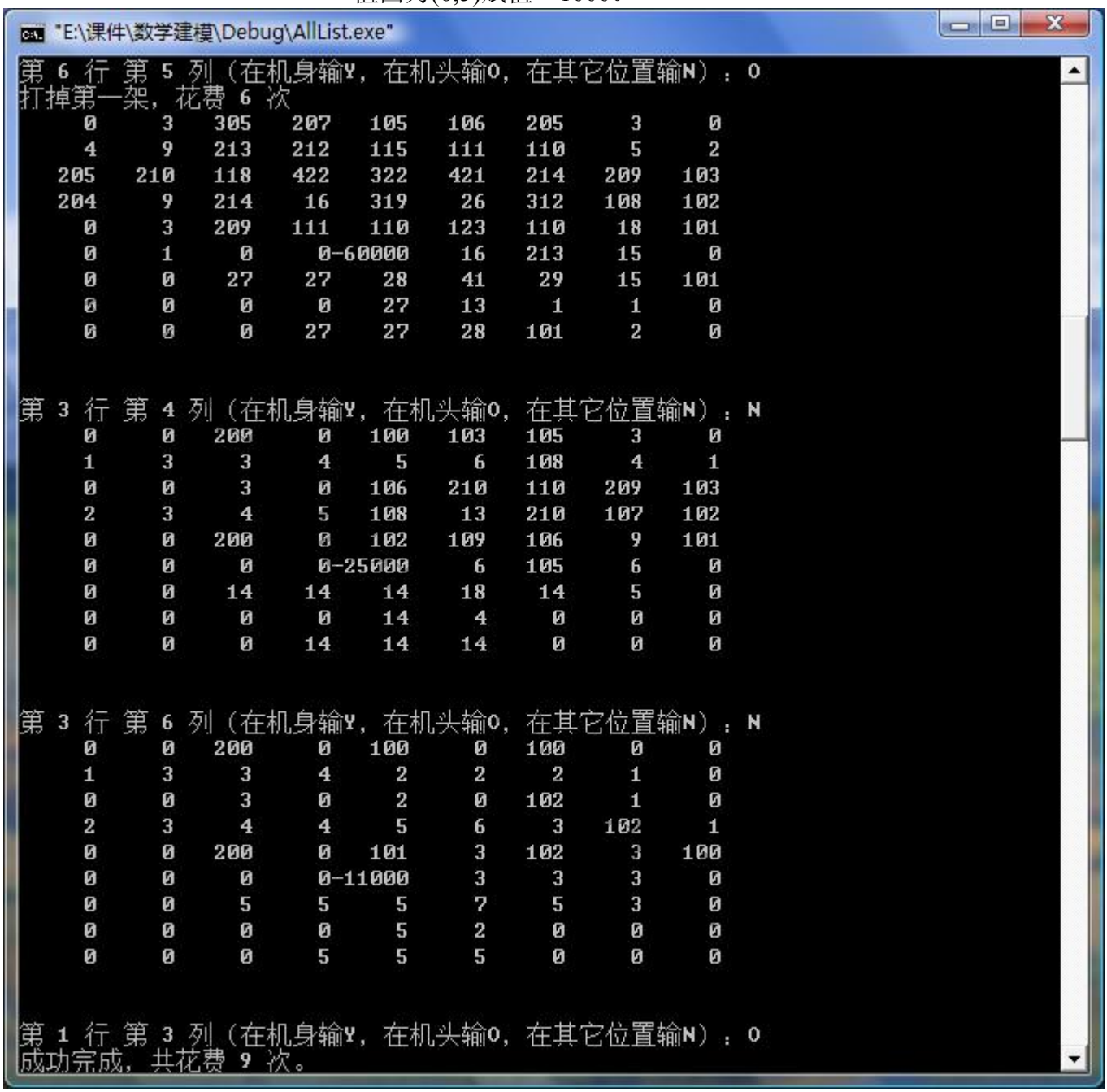
接下来程序攻击第四行，第四列，打中机身，输入 Y，筛选一部分权值图。



接下来程序攻击第六行，第三列以及第六行，第六列，分别输入 N、Y，筛选一部分权值图。



第六行第五列攻击完毕后，打中第一架飞机机头，调用 JUDGE()函数，并在每张剩余权值图为(6,5)赋值－1000。



经过九次打击以后，两架飞机均被击落，游戏获胜。

问题三：

1 以下是 11 种开局以及 A 应采取的策略：（Gi 表示当前得分）

1.Ga=Gb=Gc
A 攻击 C
B 为了获得更多信息以便尽快得分 B 会攻击 C（可以充分利用 A 攻击后的反馈信息）
C 攻击 A,B 是等价的。

A,B 对 C 有优势，C 陷入了绝对劣势
A 对 B 是否有优势取决于 C 的选择，但是这里 C 攻击 A,B 都不会使其处境改善，因此他地选择是随机的

2. Ga=Gb>Gc
A 攻击 C
B 为了获得更多信息以便尽快得分 B 会攻击 C（可以充分利用 A 攻击后的反馈信息）
C 攻击 A,B 是等价的。

A,B 对 C 有优势，C 陷入了绝对劣势
A 对 B 是否有优势取决于 C 的选择，但是这里 C 攻击 A,B 都不会使其处境改善，因此他地选择是随机的

3. Ga=Gb<Gc
A 攻击 C
B 为了获得更多信息以便尽快得分 B 会攻击 C（可以充分利用 A 攻击后的反馈信息），同时还可以降低 C 的得分机会
C 攻击 A,B 是等价的。

A,B 对 C 有优势，C 陷入了绝对劣势
A 对 B 是否有优势取决于 C 的选择，但是这里 C 攻击 A,B 都不会使其处境改善，因此他地选择是随机的

1. Ga>Gb=Gc
A 攻击 C
B 在此局中对 A,C 的优势已形成，B 为了获得更多信息以便尽快得分 B 会攻击 C（可以充分利用 A 攻击后的反馈信息）
C 攻击 A，以便降低 A 得分的机会

A 对 C 有优势，但对 B 有劣势，C 陷入了绝对劣势。但如果 A 先攻击 B,B 之后会攻击 A，C 会配合 B 攻击 A，B 取得了绝对优势，A 陷入了绝对劣势。

2. Ga>Gb>Gc
A 攻击 C
B 在此局中对 A,C 的优势已形成，B 为了获得更多信息以便尽快得分 B 会攻击 C（可以充分利用 A 攻击后的反馈信息）
C 攻击 A，以便降低 A 得分的机会

A 对 C 有优势，但对 B 有劣势。但如果 A 先攻击 B,B 随后会攻击 A，C 会配合 B 攻击 A，B 取得了绝对优势，A 陷入了绝对劣势。

3. Ga>Gc>Gb
A 攻击 C
B 在此局中对 A,C 的优势已形成，B 为了获得更多信息以便尽快得分 B 会攻击 C（可以充分利用 A 攻击后的反馈信息）
C 攻击 A，以便降低 A 得分的机会

A 对 C 有优势，但对 B 有劣势。但如果 A 先攻击 B,B 之后会攻击 A，C 会配合 B 攻击 A，B 取得了绝对优势，A 陷入了绝对劣势。

4. Gc>Ga>Gb
A 攻击 C
B 为了获得更多信息以便尽快得分 B 会攻击 C（可以充分利用 A 攻击后的反馈信息），同时还可以降低 C 的得分机会
C 攻击 A。

A 对 C 有优势，但对 B 有劣势。但如果 A 先攻击 B,B 随后会攻击 A，C 会配合 B 攻击 A，B 取得了绝对优势，A 陷入了绝对劣势。

5. Gb>Ga>Gc
A 攻击 C
B 攻击 C
C 攻击 B，为使自己最后胜出 C 会选择遏制分数最高的 B 得分

A 在此局中取得了绝对优势

9. Gb>Gc>Ga
A 攻击 B
B 攻击 C
C 攻击 B，为使自己最后胜出 C 在同等信息反馈量下选择使得分最高的 B 陷入绝对劣势

A 在此局中取得了绝对优势

10. Ga<Gb=Gc
A 攻击 C
B 为了获得更多信息以便尽快得分 B 会攻击 C（可以充分利用 A 攻击后的反馈信息）
C 攻击 B，为使自己最后胜出 C 要遏制 B 得分

A 在此局中取得了绝对优势

11. Ga<Gb<Gc
A 攻击 C
B 为了获得更多信息以便尽快得分 B 会攻击 C（可以充分利用 A 攻击后的反馈信息），

同时可已遏制 C 得分
C 攻击 B，为使自己最后胜出 C 要遏制 B 得分

A 在此局中取得了绝对优势

2 加赛情况

若出现三人同分，加赛在三人之间进行，A 仍依照之前策略打击，获胜概率依然很大。

若出现两人平分，避免出现第三人“起死回生”的情况，加赛仅在两人之间进行。此时应用问题二中的打击方法，获胜概率也很大。

七、模型评价

问题一：

1 计算机打击次数分布及期望

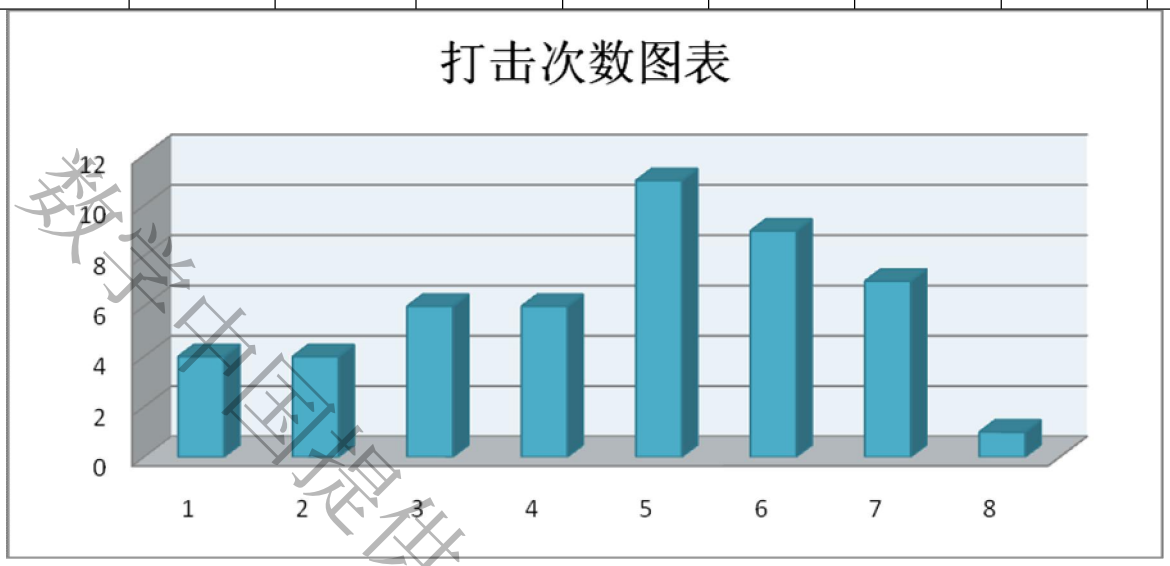
针对飞机 48 种摆放可能，用程序实现，可以列出下表：

飞机朝向	机头坐标	打击次数	飞机朝向	机头坐标	打击次数
向上	(1,3)	7	向下	(4,3)	5
	(1,4)	8		(4,4)	1
	(1,5)	7		(4,5)	4
	(2,3)	6		(5,3)	4
	(2,4)	6		(5,4)	4
	(2,5)	5		(5,5)	2
	(3,3)	3		(6,3)	5
	(3,4)	7		(6,4)	3
	(3,5)	7		(6,5)	4
	(4,3)	4		(7,3)	5
	(4,4)	1		(7,4)	7
	(4,5)	3		(7,5)	5
向左	(3,1)	6	向右	(3,4)	3
	(4,1)	7		(4,4)	1
	(5,1)	6		(5,4)	2
	(3,2)	6		(3,5)	5
	(4,2)	5		(4,5)	5
	(5,2)	6		(5,5)	2
	(3,3)	3		(3,6)	5
	(4,3)	6		(4,6)	4
	(5,3)	6		(5,6)	5
	(3,4)	3		(3,7)	6
	(4,4)	1		(4,7)	7
	(5,4)	2		(5,7)	5
总打击次数	**220**	**飞机分布情况**	**48**	**平均打击次数**	**4.5833333**

得出平均每局计算机打了 4.583333 步即可打中玩家机头。

经过统计，得到计算机打中机头次数分布，并由此算出 n 的概率密度函数：

n	1	2	3	4	5	6	7	8
times	4	4	6	6	11	9	7	1
P_n	0.0833	0.0833	0.1250	0.1250	0.2292	0.1875	0.1458	0.0208
$F_{机n1}(X \le n)$	0.0833	0.1666	0.2916	0.4166	0.6458	0.8333	0.9791	1.0000



2 计算机在与采取随机不重复策略的玩家游戏时每局获胜概率

现假设与计算机对战的玩家采取一种较为低劣的打击方法，即在 33 个有可能出现机头的位置随机不重复地选取一个进行打击。其打中机头所打击的次数分布服从均匀分布，即：

$$P_{人n2}=\frac{1}{33}=0.0303,\quad n=1,2,3\cdots\cdots,\ 33$$

其概率密度函数为：

$$F_{人n2}(X\le n)=\frac{n}{33},\quad n=1,2,3\cdots\cdots,\ 33$$

由于是由计算机先开始进行攻击，则计算机在第 n 步获胜的概率密度为：

$$P_{n3}(n1\le n2)=P_{机n}\times[1-F_{人n2}(X\le n)]$$

求得 P3 的分布为：

n	1	2	3	4	5	6	7	8
P3	0.0833	0.080776	0.117424	0.113636	0.201418	0.159091	0.119291	0.016388

求和得 0.8913,即计算机在与采取随机不重复策略的玩家游戏时，获胜的概率高达 0.8913！

3 19 局 10 胜制计算机获胜概率

考虑游戏的 19 局 10 胜制，由上假设计算机每局获胜概率达 0.8913，则计算机比赛获胜概率为：

$P = C_{19}^{10} * 0.8913 \wedge 10 * 0.1087 \wedge 9 + C_{19}^{11} * 0.8913 \wedge 11 * 0.1087 \wedge 8 + ... + C_{19}^{19} * 0.8913 \wedge 19 * 0.1087 \wedge 0$

=0.9996059

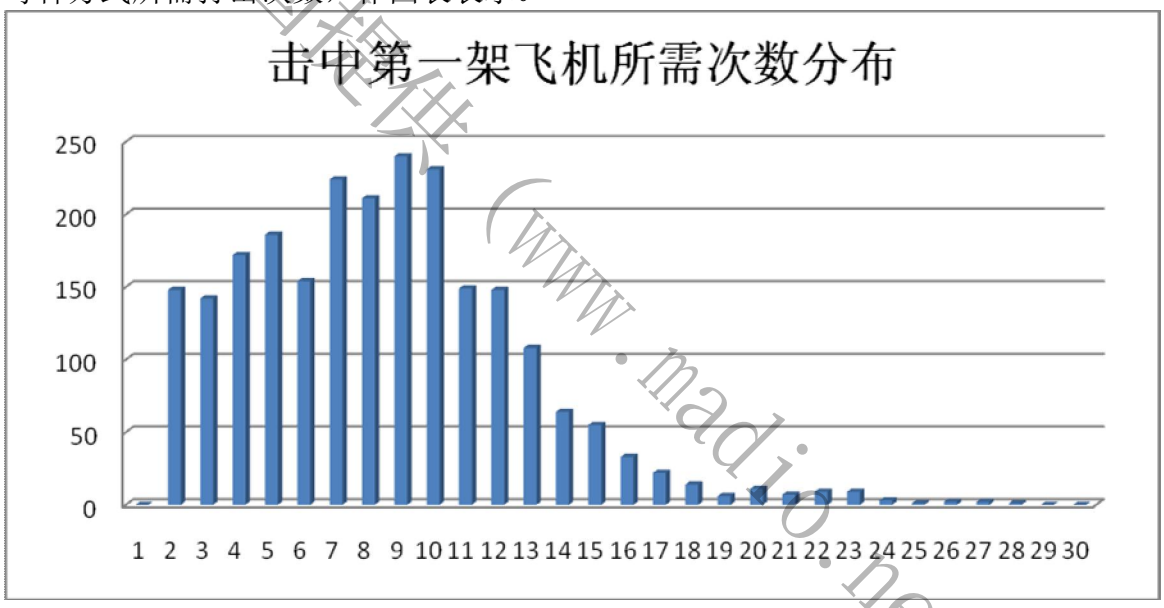
可见，由于采用策略上的绝对优势，计算机获胜的概率极高，为 0.9996059。

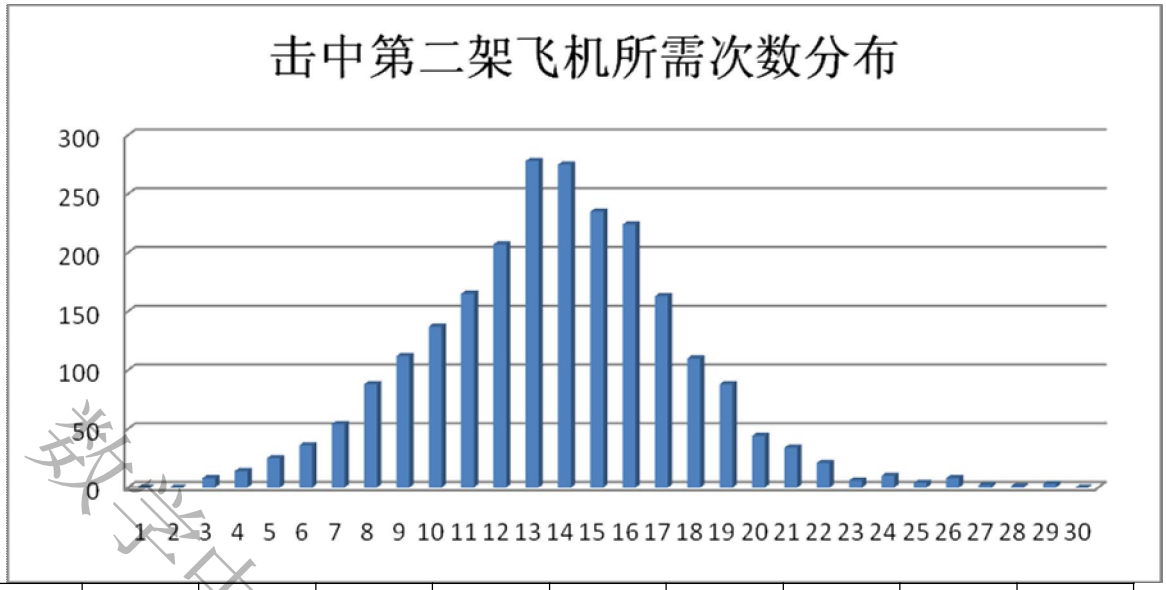
问题二：

1 计算机打击次数分布及期望

针对飞机 2352 种摆放可能，用程序实现。

修改程序，让计算机生成所有飞机，并用“机机对战”的方式遍历所有摆放方式，算出每种方式所需打击次数，作图表表示。





次数	1	2	3	4	5	6	7	8	9	10
比例	0	0	0.0034	0.0055	0.0089	0.0128	0.0200	0.0255	0.0361	0.0387
次数	11	12	13	14	15	16	17	18	19	20
比例	0.0476	0.0612	0.0838	0.0846	0.0753	0.0748	0.0548	0.0383	0.0332	0.0145
次数	21	22	23	24	25	26	27	28	29	30
比例	0.0106	0.0060	0.0021	0.0021	0.0017	0.0025	0.0009	0.0004	0.0013	0

由表格可得出 2352 种飞机摆放可能打击次数最少为 2 次，最多为 29 次，期望为 12.4893，且基本服从正态分布。

2 19 局 10 胜制计算机获胜概率

假设每局计算机获胜概率为 P1,则在 19 局 10 胜制下获胜概率为：

$$P = C_{19}^{10} * P1\wedge 10 * (1-P1)\wedge 9 + C_{19}^{11} * P1\wedge 11 * (1-P1)\wedge 8 + ... + C_{19}^{19} * P1\wedge 19 * (1-P1)\wedge 0$$

问题三：

1 依照以上打击策略，分析当前得分状况，从而确定打击目标人，由于 A 的先手优势，获胜概率将远大于 B,C。

2 首先不难发现，A 采用如上策略不会是自己陷入绝对劣势，同时还能在 8，9，10，11 四种情况中获得绝对优势。相比 B,C 而言，A 的赢面更大。

九、模型推广

问题一和问题二实际上采用的是计算机常用的搜索算法，这种算法在五子棋和围棋当中实际上已广泛运用。其核心思想在于总结反馈信息，并对不可能出现的状况一一排除，最终缩小目标范围，增大获胜机率。

问题三是一个非合作性完全信息动态博弈问题。由《百度百科》可知，博弈论的基本概念包括：参与人、行为、信息、战略、支付函数、结果、均衡。

根据参与者能否形成约束性的协议，以便集体行动，博弈可分为合作性博弈和非合作性博弈。所谓合作性博弈是指参与者从自己的利益出发与其他参与者谈判达成协议或形成联盟，其结果对联盟方均有利；而非合作性博弈是指参与者在行动选择时无法达成约束性的协议。

从知识的拥有程度来看，博弈分为完全信息博弈和不完全信息博弈。严格地讲，完全信息博弈是指参与者的策略空间及策略组合下的支付，是博弈中所有参与者的“公共知识”的博弈。对于不完全信息博弈，参与者所做的是努力使自己的期望支付或期望效用最大化。

博弈又分静态博弈和动态博弈。静态博弈指参与者同时采取行动，或者尽管参与者行动的采取有先后顺序，但后行动的人不知道先采取行动的人采取的是什么行动。动态博弈指参与者的行动有先后顺序，并且后采取行动的人可以知道先采取行动的人所采取的行动。

附录一：问题一程序代码

```
#include<iostream.h>

struct   Node{
        int    x;

        int    y;
};


class Plant{
```

```
public:

   Node t;

   int plant[7][7];

   void make(int row,int col,int point);
   int over(int x);

   Plant operator+(Plant obj);

   void   set0();

   Node   judgeoff(Node t,Plant *p);

   Node   judgeon(Node t,Plant *p);
   friend ostream &operator<<(ostream &out,Plant &obj);

   Node     Max();
};

void Plant::set0(){
   int i,j;

   for(i=0;i<7;i++)
      for(j=0;j<7;j++){
         plant[i][j]=0;
      }
}


Plant Plant::operator+(Plant obj){
   int i,j;
   Plant temp;

   for(i=0;i<7;i++)
      for(j=0;j<7;j++){
         temp.plant[i][j]=plant[i][j]+obj.plant[i][j];
      }
      return temp;
}
ostream &operator<<(ostream &out,Plant &obj){
   int i,j;
   for(i=0;i<7;i++){

      for(j=0;j<7;j++){
         out<<"      "<<obj.plant[i][j]<<"      ";
      }
      out<<endl<<endl;
```

```
        }
        return out;
}


void Plant::make(int row,int col,int point)
{
        int part1[3][5]={1,1,1,1,1,0,0,1,0,0,0,1,1,1,0},
          part2[5][3]={1,0,0,1,0,1,1,1,1,1,0,1,1,0,0};
        int n,m;

        if(point==1){
          if(over(row-4)==1&&over(col-3)==1&&over(col+1)==1)
          {
                plant[row-1][col-1]=10;
                for(n=row-2;n>=row-4;n--)
                      for(m=col-3;m<=col+1;m++)
                      {
                            plant[n][m]=part1[row-2-n][m-col+3];
                      }
          }
          else
                for(n=0;n<=6;n++)
                      for(m=0;m<=6;m++)
                            plant[n][m]=0;
        }
      if(point==2){
          if(over(row+2)==1&&over(col-3)==1&&over(col+1)==1)
          {
                plant[row-1][col-1]=10;
          for(n=row;n<=row+2;n++)
                for( m=col-3;m<=col+1;m++)
                      plant[n][m]=part1[n-row][m-col+3];
          }
                else
                      for(n=0;n<=6;n++)
                            for(m=0;m<=6;m++)
                                  plant[n][m]=0;
        }
      if(point==3){
          if(over(col+2)==1&&over(row-3)==1&&over(row+1)==1)
          {
                plant[row-1][col-1]=10;
                for( n=col;n<=col+2;n++)
                      for( m=row-3;m<=row+1;m++)
                            plant[m][n]=part2[m-row+3][n-col];
          }
          else
                for(n=0;n<=6;n++)
                            for(m=0;m<=6;m++)
```

```
                    plant[n][m]=0;
        }
        if(point==4){
          if(over(col-4)==1&&over(row-3)==1&&over(row+1)==1)
          {
              plant[row-1][col-1]=10;
          for(n=col-2;n>=col-4;n--)
              for( m=row-3;m<=row+1;m++)
                  plant[m][n]=part2[m-row+3][col-2-n];
          }
          else
              for(n=0;n<=6;n++)
                  for(m=0;m<=6;m++)
                      plant[n][m]=0;
         }
}


int Plant::over(int x)
{
        if(x>=0&&x<=6)
          return 1;
        else
          return 0;
}

Node Plant::judgeoff(Node t,Plant *p){

        int i,n,m;

        Plant temp;

        temp.set0();

        for(i=0;i<196;i++){

          if(p[i].plant[t.x-1][t.y-1]==1||p[i].plant[t.x-1][t.y-1]==10){

              for(n=0;n<7;n++)
                  for(m=0;m<7;m++)
                      p[i].plant[n][m]=0;
          }
        }

        for(n=0;n<7;n++)
          for(m=0;m<7;m++)
              for(i=0;i<196;i++)
              {
                  temp.plant[n][m]=temp.plant[n][m]+p[i].plant[n][m];
              }
```

```
            cout<<temp;

        return    temp.Max();
}

Node Plant::judgeon(Node t,Plant *p){

        int i,n,m;

        Plant temp;

        temp.set0();

        for(i=0;i<196;i++){

          if(p[i].plant[t.x-1][t.y-1]!=1){

                for(n=0;n<7;n++)
                     for(m=0;m<7;m++)
                          p[i].plant[n][m]=0;
          }
        }

        for(n=0;n<7;n++)
          for(m=0;m<7;m++)
                for(i=0;i<196;i++)
                {
                     temp.plant[n][m]=temp.plant[n][m]+p[i].plant[n][m];
                }
                cout<<temp;

        return    temp.Max();
}


Node Plant::Max(){

        int max,i,j;

        Node t;

        t.x=1;

        t.y=1;

        max=plant[0][0];

        for(i=0;i<7;i++)
          for(j=0;j<7;j++)
          {
```

```
            if(plant[i][j]>=max){
                max=plant[i][j];
                t.x=i+1;
                t.y=j+1;
            }
        }
        return t;
}


void main(){
    Plant all[196];

    int n,m,i;

    int flag1=1,
      flag2=1;

    char a,b;


    for(i=0;i<196;i++)
      all[i].set0();

    int j=0;

    for(n=1;n<8;n++)
      for(m=1;m<8;m++)
          for(i=1;i<=4;i++)
          {
              all[j].make(n,m,i);
              j++;
          }

    Plant p;

      Node   t;

      t.x=4;

      t.y=4;
```

```
        while(flag1){

cout<<"##########################»¶Ó-ÄúÌåÑé±¾ÓÎÏ·##########################"<<
endl<<endl<<endl;


        cout<<"1    ¿ªÊ¼ÓÎÏ·"<<endl<<endl;
        cout<<"2    ½áÊøÓÎÏ·"<<endl<<endl;


        cin>>a;
        switch(a)
        {
        case '1':
          while(flag2){

          cout<<"¼ÆËã»ú´òµã£º    ("<<t.x<<","<<t.y<<")"<<endl<<endl;


          cout<<"ÊÇ·ñ´òÖÐÁË£¿(Y/N) ½áÊøÓÎÏ·´2"<<endl;

          cin>>b;

          while(!(b=='Y'||b=='N'||b=='2')){

               cout<<"ÇëÕýÈ·ÌáÊ¾£º"<<endl;

                cin>>b;
          }


          switch(b){

          case 'N':

               t=p.judgeoff(t ,all);

               break;


          case    'Y':
               t=p.judgeon(t,all);

               break;

          case '2':

               flag2=0;
```

```
                break;
            }
            }
        case '2':

            flag1=0;

            break;
        }
            }
}
```

附录二：问题三程序代码

```
#include<iostream.h>
#include<iomanip.h>
#include<fstream.h>
#include<stdlib.h>

int all[9][9]={0};
int lx=-1,ly=-1;

int allmap1[2000][9][9]={0};
int allmap1num=0;
int allmap2[2000][9][9]={0};
int allmap2num=0;
int allmap[2000][9][9]={0};
int allmapnum=0;

int plane[4][5][5]=    {
        0,0,100,0,0,
          1,1,1,1,1,
          0,0,1,0,0,
          0,1,1,1,0,
          0,0,0,0,0,

          0,0,1,0,0,
          1,0,1,0,0,
          1,1,1,100,0,
          1,0,1,0,0,
          0,0,1,0,0,

          0,1,1,1,0,
          0,0,1,0,0,
          1,1,1,1,1,
          0,0,100,0,0,
          0,0,0,0,0,
```

```
        0,1,0,0,0,
        0,1,0,1,0,
        100,1,1,1,0,
        0,1,0,1,0,
        0,1,0,0,0
};

void create(int first[][5],int second[][5],int fxlocal,int fylocal);
void getmap();
void findmax(int &x,int &y);
void judgeon(int lx,int ly);
void judgeoff(int lx,int ly);
void judgeOk(int lx,int ly);

void main()
{
    cout<<endl;
    int x=0,y=0;
    char c=' ';
    int alltimes=0,times=0;
    int flag=1,j=1;
    int err=0,last=2;
    getmap();
    while(1)
    {
      findmax(x,y);
      cout<<"µÚ "<<x+1<<" ÐÐ" <<" µÚ "<<y+1<<"
ÁÐ£¨ÔÚ»úÉíÊäY£¬ÔÚ»úÍ·ÊäO£¬ÔÚÆäËüÎ»ÖÃÊäN£©£º";
      cin>>c;
      times++;
      switch(c)
      {
      case 'Y':
          judgeon(x,y);
          break;
      case 'N':
          judgeoff(x,y);
          break;
      case 'O':
          last--;
          if(last==1)
          {
              cout<<"´òµôµÚÒ»¸Ü£¬»¨Ñ "<<times<<" ´Î"<<endl;
          }
          judgeOk(x,y);
          if(last==0)
          {
              cout<<"³É¹¦Íê³É£¬¹²»¨Ñ "<<times<<" ´Î£"<<endl;
              return;
          }
```

```
            break;
        case 'C':
            return;
        }
    }
}

void create(int first[][5],int second[][5],int fxlocal,int fylocal)
{
    int map[15][15]={0};
    int i,j,sum=0,a,b;
    int err=0;
    for(i=0;i<9;i++)
    {
        for(j=0;j<9;j++)
        {
            err=0;


            for( a=0;a<15;a++)
            {
                for( b=0;b<15;b++)
                {
                    map[a][b]=0;
                }
            }


            for(a=0;a<5;a++)
            {
                for(b=0;b<5;b++)
                {
                    map[fxlocal+a][fylocal+b]+=first[a][b];
                }
            }


            for(a=0;a<5;a++)
            {
                for(b=0;b<5;b++)
                {
                    map[i+a][j+b]+=second[a][b];
                }
            }


            sum=0;
            for(a=0;a<9;a++)
            {
                for(b=0;b<9;b++)
```

```
                {
                    sum+=map[a][b];
                    if(map[a][b]==2 || map[a][b]==101 || map[a][b]==200)
                    {
                        err=1;
                        break;
                    }
                }
            }


            if(sum!=218)
            {
                err=1;
            }

            if(err==0)
            {
                for(a=0;a<9;a++)
                {
                    for(b=0;b<9;b++)
                    {
                        allmap[allmapnum][a][b]=map[a][b];
                    }
                }
                allmapnum++;
            }
        }
    }
}


void getmap()
{
    for(int k=0;k<9;k++)
    {
      for(int l=0;l<9;l++)
      {
          create(plane[0],plane[0],k,l);
          create(plane[1],plane[1],k,l);
          create(plane[2],plane[2],k,l);
          create(plane[3],plane[3],k,l);
      }
    }

    for(int i=0;i<allmapnum;i++)
    {
      for(int j=0;j<9;j++)
      {
          for(int k=0;k<9;k++)
```

```
            {
                allmap1[i][j][k]+=allmap[i][j][k];
                allmap[i][j][k]=0;
            }
        }
    }
    allmap1num=allmapnum;
    allmapnum=0;

    for(    k=0;k<9;k++)
    {
        for(int l=0;l<9;l++)
        {
            create(plane[0],plane[1],k,l);
            create(plane[0],plane[2],k,l);
            create(plane[0],plane[3],k,l);
            create(plane[1],plane[2],k,l);
            create(plane[1],plane[3],k,l);
            create(plane[3],plane[2],k,l);
        }
    }

    for( i=0;i<allmapnum;i++)
    {
        for(int j=0;j<9;j++)
        {
            for(int k=0;k<9;k++)
            {

                allmap2[i][j][k]=allmap[i][j][k];
                allmap[i][j][k]=0;

            }
        }
    }
    allmap2num=allmapnum;
    allmapnum=0;
}

void findmax(int &x,int &y)
{
    int flag=1;
    x=y=0;

    for(int j=0;j<9;j++)
    {
        for(int k=0;k<9;k++)
        {
            all[j][k]=0;
        }
```

```
        }

        for(int i=0;i<allmap1num;i++)
        {
          for(int j=0;j<9;j++)
          {
               for(int k=0;k<9;k++)
               {
                    all[j][k]+=allmap1[i][j][k]*0.5;
               }
          }
        }
        for( i=0;i<allmap2num;i++)
        {
          for( int j=0;j<9;j++)
          {
               for(int k=0;k<9;k++)
               {
                    all[j][k]+=allmap2[i][j][k];
               }
          }
        }
        for( i=0;i<9;i++)
        {
          for(int j=0;j<9;j++)
          {
               if(all[x][y]<all[i][j])
               {
                    x=i;
                    y=j;
               }
          }
        }
        for(i=0;i<9;i++)
        {
          for(int j=0;j<9;j++)
          {
               cout<<setw(6)<<all[i][j];
          }
          cout<<endl;
        }
        cout<<endl<<endl;
}

void judgeon(int lx,int ly)
{
        for(int i=0;i<allmap1num;i++)
        {
          if(allmap1[i][lx][ly]==100    )
```

```
        {
            for(int j=0;j<9;j++)
            {
                for(int k=0;k<9;k++)
                {
                    allmap1[i][j][k]=0;
                }
            }
        }
    }

    for( i=0;i<allmap2num;i++)
    {
        if(allmap2[i][lx][ly]==100 )
        {
            for(int j=0;j<9;j++)
            {
                for(int k=0;k<9;k++)
                {
                    allmap2[i][j][k]=0;
                }
            }
        }
    }
}

void judgeoff(int lx,int ly)
{
    for(int i=0;i<allmap1num;i++)
    {
        if(allmap1[i][lx][ly]==1 || allmap1[i][lx][ly]==100)
        {
            for(int j=0;j<9;j++)
            {
                for(int k=0;k<9;k++)
                {
                    allmap1[i][j][k]=0;
                }
            }
        }
    }

    for( i=0;i<allmap2num;i++)
    {
        if(allmap2[i][lx][ly]==1||allmap2[i][lx][ly]==100)
        {
            for(int j=0;j<9;j++)
            {
                for(int k=0;k<9;k++)
                {
```

```
                    allmap2[i][j][k]=0;
                }
            }
        }
    }
}

void judgeOk(int lx,int ly)
{
    for(int i=0;i<allmap1num;i++)
    {
      if(allmap1[i][lx][ly]!=100)
      {
        for(int j=0;j<9;j++)
        {
            for(int k=0;k<9;k++)
            {
                allmap1[i][j][k]=0;
            }
        }
      }
      else
      {
        allmap1[i][lx][ly]=-1000;
      }
    }

    for( i=0;i<allmap2num;i++)
    {
      if(allmap2[i][lx][ly]!=100)
      {
        for(int j=0;j<9;j++)
        {
            for(int k=0;k<9;k++)
            {
                allmap2[i][j][k]=0;
            }
        }
      }
      else
      {
        allmap2[i][lx][ly]=-1000;
      }
    }
}
```