最优决策2 解题报告

宁波市镇海中学 邹逍遥

1 试题大意

A和B玩一个游戏。

首先每个人获得一个$1 \sim n$且保证不同的随机整数，然后A在$1 \sim m$中选出一个数报出，B在“2”和“A报出的数”这两个选项中选择一个数报出。

假如报出的两个数相同（设都为x），那么随机整数大的一方获得x的收益，另一方获得$-x$的收益。

假如报出的两个数不同（设为x和y，$x < y$），那么选y的一方获得x的收益，另一方获得$-x$的收益。

现在你需要为A确定一个策略表（即拿到某个数字时有多少的概率选某个选项），最大化最坏情况下A的期望收益。

只需要输出策略表中的非0项即可。

2 数据规模

对于所有数据，数据组数不超过10，$2 \le n \le 30000, 2 \le m \le 30000$。

共有三类数据：

- $n \le 3, m \le 30000$
- $m \le 4, n \le 30000$
- $n \le 10000, m \le 10000$

时间限制0.2秒。

3 算法介绍

3.1 算法1：当$n = 2$时

最优解一定是拿到1时选1，拿到2时选除了1之外的任意一个数。

可以拿到10分。

3.2 算法2：当$m = 2$时

对于每个数字直接比较两个选项的收益即可。

直接实现一个这个算法可以拿到10分，结合之前算法可以拿到15分。

3.3 算法3：当$n = 3$且$m = 3$时

经过思考发现：当拿到数字3时一定不会选1或2，当拿到数字2时一定不会选1或3，当拿到数字1时一定不会选2。

那么只有拿到数字1时需要进行选择。由于B拿到1或3时的决策不会被你的决策影响，所以需要在B拿到数字2时获得尽可能大的利益，也就是说需要让B拿到数字2 时“ 难以作出抉择”，即两个选项收益相同。

利用这个条件就可以解出拿到数字1时的策略分布。

不过由于$n = 3, m > 3$时的策略与$n = 3, m = 3$时完全相同，所以假如没有把那部分数据特判掉的话能拿到$n = 3$的5分。

直接实现一个这个算法可以拿到5~10分，结合之前算法可以拿到20~25分。

3.4 算法4：当n较小时

容易看出即使可选的方案数非常多，但是方案一旦超过了某个值就不优了。也就是说存在一个k使得$(n = 3, m = k)$的答案和$(n = 3, m > k)$的答案完全相同。

解出了m较小的情况后，一旦发现某个m下最大值没有被使用，那么更大的m也不会被使用了。经过打表，发现$(n = 3, m = 3)$和$(n = 3, m = 4)$的结果是一样的。那么直接输出$(n = 3, m = 3)$的方案就可以通过所有$(n = 3, m \geq 3)$的数据。

同理$(n = 4, m = 5)$和$(n = 4, m = 6)$的结果是一样的。那么直接输出$(n = 4, m = 5)$的方案就可以通过所有$(n = 4, m \geq 5)$的数据。

直接实现一个这个算法可以拿到18分，结合之前算法可以拿到33分。

3.5 算法5：当$m = 3$时

以下使用“策略i”来表示A报出i这个数。

当$m = 3$时B只有在A选择策略3时才需要作出决策。

观察B的最优策略，容易发现不存在一对$i, j(i < j)$使得拿到j时放弃拿到i时进攻，否则交换这两对数字的决策结果不会变差。那么一定存在一个k使得B在拿到不超过k的数字时选择放弃，否则选择进攻。

由于进攻收益是单调不减的，可以看出B的策略是最优的等价于拿到k时放弃收益不小于进攻收益并且拿到$k + 1$时放弃收益不大于进攻收益。

枚举k，用$m \times n$个变量a_{ij}表示A拿到数字i时选择策略j的概率，那么有：

$$\Sigma_j a_{ij} = 1$$

$$a_{ij} \geq 0$$

$$5\Sigma_{i=1}^{k-1} a_{i,3} - \Sigma_{i=k+1}^{n} a_{i,3} \leq 0$$

$$5\Sigma_{i=1}^{k} a_{i,3} - \Sigma_{i=k+2}^{n} a_{i,3} \geq 0$$

由于B的策略确定，那么A选每对(i, j)的收益也可以算出来（枚举B的数字计算贡献即可），记为b_{ij}。那么要最大化的就是

$$\Sigma a_{ij} b_{ij}$$

解这个线性规划，即可得到答案。大约能跑过n为几百的数据。

直接实现一个这个算法可以拿到25分，结合之前算法可以拿到53分。

3.6 算法6：当$m > 3$时

B需要做出决策的状态变多了，但是仍然可以$O(n^{m-2})$枚举B的每一种策略。

设B在面对决策i时的分界点是$k_i(3 \leq i \leq m)$，那么有：

$$\Sigma_j a_{ij} = 1$$

$$a_{ij} \geq 0$$

$$(j+2)\Sigma_{i=1}^{k_j-1} a_{i,3} - (j-2)\Sigma_{i=k_j+1}^{n} a_{i,j} \leq 0$$

$$(j+2)\Sigma_{i=1}^{k_j} a_{i,j} - (j-2)\Sigma_{i=k_j+2}^{n} a_{i,j} \geq 0$$

最大化的仍然是

$$\Sigma a_{ij} b_{ij}$$

复杂度约为$O(n^{m-2} * \text{Simplex})$

由于复杂度较高，$1s$内只能跑过$n, m = 9$的数据，当然加上打表能获得更多分数。

直接实现一个这个算法可以拿到49~54分，结合之前算法可以拿到65~70分。

3.7 算法7：一个猜想

设$N = \lfloor \frac{n+1}{2} \rfloor$。

利用算法6打出n, m为个位数的表之后可以观察到有这样一个性质：所有最优解都可以在假设B的分界点都为N时取到（即$k_i = N$）。假如这个性质成立的话那么就可以只做一次Simplex解出答案了。

事实上这个性质的确是成立的。使用这种做法就可以跑出m, n为几百的数据了。下面来证明这个性质的正确性。

引理1：将一个k_i减一答案不会变差当且仅当存在一个最优策略的a_{i,k_i}为0。将一个k_i加一答案不会变差当且仅当存在一个最优策略的a_{i,k_i+1}为0。

引理2：将任何一个大于N的k_i减一之后解线性规划答案都不会变差。

证明：由于B的策略是大于k_i即跟，那么A用小于等于k_i的数字是显然不可能在B跟的时候获胜的。也就是说在小于k_i的数中A用哪个跟效果都是一样的，所以A 一定会使用较小的那些数字来选择这个策略而使用较大的数字来选择策略2，不可能出现一个较小数字选择了策略2而较大数字选择了策略3 ~ m中的某一个的情况。由于每个策略下A 的决策都满足小于k_i部分的概率总和不超过大于k_i部分的概率总和，所以小于等于N的数字中至少有一个会选择策略2。也就

是说满足$N \le t \le k_i$ 的数字t 中不可能有选择策略3 ~ m的。于是一定存在一个最优策略，它的a_{i,k_i}为0。

引理3：将任何一个小于N的k_i加一之后解线性规划答案都不会变差。

证明：考虑A拿到N这个数时的策略，假如选择策略2，那么收益是$\frac{2(2N-n-1)}{n-1}$，而选择策略$p(p \ge 3)$的收益不会超过$\frac{p(2N-n-1)}{n-1}$。由于$\frac{p(2N-n-1)}{n-1} \le \frac{2(2N-n-1)}{n-1}$，所以在拿到$N$这个数且$N > k_i$时是不会选择策略$i$的。同理可以推出永远都不会在拿到的数$\le N$且$> k_i$的时候选择策略$i$。因为$k_i < N$，也就是说$a_{i,k_i+1}$一定为0。

综合引理2,3即可得证。

直接实现一个这个算法可以拿到64~71分，结合之前算法可以拿到82~87分。

3.8 算法8：使用贪心代替线性规划

在算法7的基础上，由于分界线k_i已经固定了，那么可以直接计算出每个数字选每个策略的收益，那么只要在保证满足分界线不动的情况下贪心选取最大的选项即可。

首先将所有数的策略都置为策略1和策略2中较优的那个。由于之前证明了小于分界线的数中选择3 ~ m的策略一定是最小的那些数，而那些选择策略1的数不管选择什么策略收益都是等价的，所以可以直接把它们看成一类数，不妨称之为“炮灰”。

由于必须保证第i条分界线上下选取概率和的比例为$\frac{i-2}{i+2}$，所以不妨给每个比分界线大的数字的策略都分配一个固定比例的炮灰，即每个数字j选策略i 的收益都需要减去$\frac{i-2}{i+2} \times ((-1) - \frac{2(N-1)-i(n-N)}{n-1})$。由于已经满足了所有不等式的限制，所以每个大于$N$的数字的策略一定是100%选取2 ~ m中的收益最大的那个策略。然后根据每个策略被选择的次数使用一些炮灰来平衡之后输出答案即可。

注意到炮灰数量有可能不够，所以在炮灰用完之后需要使用那些选择策略2的数字，需要重新计算贡献。不过由于本题的前三小的数字选项固定为了1,2,3所以可以证明这种情况不会出现（也就是说假如选项不是1 ~ m而是输入给定的m个数就有可能出现这种情况需要特判）。

计算出所有的收益之后枚举每个数字选取哪个策略，时间复杂度$O(mn)$。

直接实现一个这个算法可以拿到84~95分，结合之前算法可以拿到90~95分。

3.9 算法9：决策单调性

在算法8的基础上，由于每个决策受到炮灰带来的收益减少是固定的，而数字越大把策略从p调成$p+1$获得的收益增幅也是更大的（显然）。所以对于大于N的某个数字i的策略一定不会大于$i+1$的策略。

那么利用单调性从大到小扫一遍的时间复杂度就是$O(m+n)$了。

代码实现非常简单，std仅800B。

只需要实现这一个算法就可以拿到100分。

4 得分估计

预计所有选手拿到至少15分，有一半选手拿到20~25分。

暴搜或者乱搞或者随机调整打出表之后能获得n,m比较小的分数，做出$m=3$可以获得20分，写出线性规划打表也能获得50+分，预计有1~3名选手拿到30~60分。

由于观察出正解的性质之后只要10分钟就能写完std，预计有0~2名选手能拿到60-100的分数。

5 总结

这道题的代码实现难度只有NOIP普及组难度，主要的难点在于如何看出最后的结论。

依我看最后的结论并不明显，在没有小范围的表进行辅助思考的情况下直接想出正解是非常困难的。而打出小范围的表就需要涉及到一些线性规划与纳什均衡的知识和比较大的代码量。而如果使用随机调整之类的乱搞来做小范围数据的话虽然代码复杂度降低了但是能打出的表也小了，所以想出正解的难度也高了。也就是说选手需要取舍写代码的时间和思考的时间。

所以说本题的考察点既有线性规划也有转化博弈问题的水平，由选手自己的取舍决定。不过总的来说还是思考多于码代码，希望以后能在OI比赛中多见到一些更考察思考能力的题目。