一类猜数问题的研究

长沙市雅礼中学　龙凡

【摘要】

猜数问题是信息学竞赛中一种常见的类似博弈的问题。　其中部分的问题是给出猜的数与被猜数之间的大小关系作为回答信息。　作为信息学竞赛中一类经常出现的问题，很有研究的意义与价值。本文重点对这类问题的不同形式的解法进行讨论与研究。本文先提出了一个看似复杂、　棘手的问题，　然后慢慢通过分析剖析其本质，提出一个行之有效的解法，并在此基础上对其他相关的问题展开讨论。

【关键字 】

二分法、递推法、数学模型、动态规划

【引言 】

信息学竞赛中常常看到诸如此类的问题：现在有一个数　X，是 1~N 之内的。你每次可以猜一个数　Y，然后会根据你提供的　Y 与 X 的大小关系，给出　X<Y，X=Y，X>Y 三种回答。你要用尽量少的询问次数把　X 猜出来，也就是得到　X=Y 的回答。那么在最坏情况下至少需要多少次呢？

有很多问题都是在这个基础上进行了扩充和加强，从而变成了棘手的问题，但是它们的本质是相同的。　所以对这类问题的研究，　并进行一定的整理和归纳还是很有必要的。

【正文】

一、　问题的提出

引言中已经对本文要讨论的问题类型作了基本定义，　下面就来看一道如上文所说被“扩充和加强了的”猜数问题。

模型王子：

TL 家有无数个高达机器人的模型，这已经不是一个秘密。终于有一天，你忍不住了，问　TL：你家到底有多少个模型呢？

TL：就是不告诉你　。

You：说吧说吧，别保密了咯，我不会让老师知道的。

TL：这样吧，我让你猜，你每次猜一个数，我会告诉你比这个数大还是小。你快点猜出来，我马上就要去奶奶家吃饭去了。

你以你的打字速度每　1s 可以提一个问题，但是由于该遭天杀的网络延迟。TL 的回答要在　1s 之后才会传到你这里来。也就是说，只有当你提出了下一个问题之后，这个问题的答案才会告诉你。

同时，由于　TL 心高气傲。如果你低估他拥有的模型数量，他就会生气。如果你低估了他　K（$1<=K<=100$）次，他就不再陪你玩了。你最开始已经知道，　TL 的模型数量至少有　1 个，至多不超过　N（$1<=N<=10^5$）个。下面是一个　N=6时的可能的询问过程：

事件	时间
You：你有 3 个模型吧？	1s
TL：网络延迟，没反应……	1s
You：你有 5 个吧？	2s
TL：你怕我像你一样，只有 3 个模型！!!（生气）	2s
You：哦哦哦哦哦，那你有 6 个？	3s
TL：显然没有 5 个那么多。哪个像你那么有钱 。	3s
You：你有 4 个模型吧。	4s
TL：都说了没有 5 个，你还问有没有 6 个。	4s
You：你肯定是有 4 个模型咯。	5s
TL：对对对，我有 4 个模型。傻瓜，猜这么久。	5s

你现在已经知道 TL 的模型数量不少于 1 个，不超过 N个。并且知道 TL 生气 K 次之后就不会再陪你玩了，请你算算，最优询问策略在最坏情况下，至少需要多少秒才能问出具体的模型数量。

把这道题目抽象一下， 会发现就是在引言中所讲的猜数问题原型上加上了几个限制条件。本题是说： 有一个被猜数 X，是 1 到 N的范围内的整数，你每次可以给出一个整数 Y。你会在你问下一个问题之后得到你这个问题的回答，即 X 与 Y 的大小关系。并且如果你得到了 K 次 X<Y的回答，游戏就结束，你要避免这种情况的发生。

经过抽象，本题事实上就是本文要讨论的猜数问题原型加上了粗体字部分的限制条件。 但是由于限制条件的出现， 实在这道题看起来非常难以下手。 我们以往做这类题时， 依靠找规律或经验来猜测结论， 然后再加以验证， 这种办法在绝大部分情况下是适用，但是现在，我们需要系统来分析这道题目，以求得解答，那么让我们先从最简单的猜数问题看起。

二、 从最简单的问题说起

基本猜数问题：

被猜数 X 是 1 到 N 范围内的整数， 每次你可以询问一个整数 Y 和 X 的大小关系。给出 N， 请问在最坏情况下至少需要几次才能保证猜出 X。

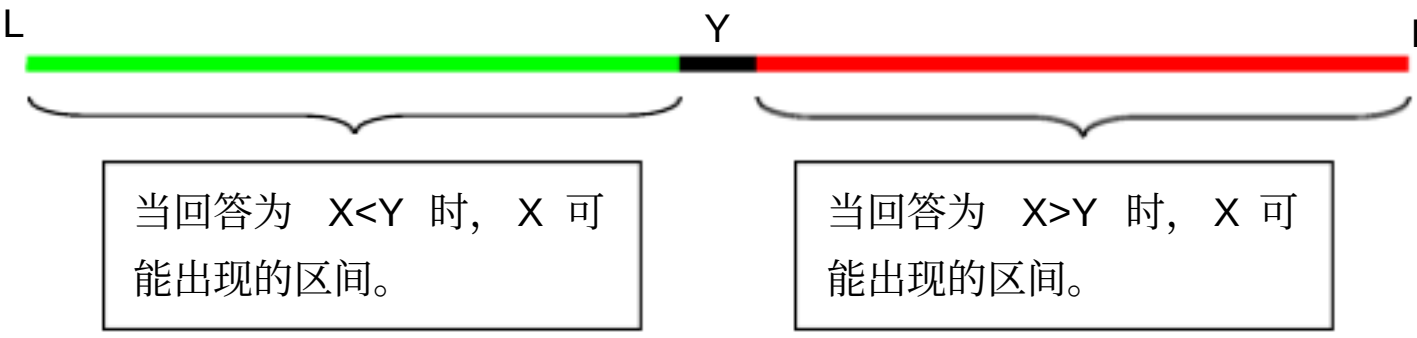
通常能够想到的方法就是基于二分的询问， 这是依据经验得来的。 具体上说就是：

如果已知 **X** 在**[L,R]** 之内，那么令 $Y = Mid = \left\lfloor \frac{L+R}{2} \right\rfloor$，如果 **Y<X** 则可以确定 **X** 在**[Mid+1,R]** 之内， **Y>X** 则可以确定 **X** 在**[L,Mid-1]** 之内， 若 **Y=X** 则表示已经猜出来了。

上面算法的最优性是显然的， 因为每次都将区间尽量均分， 使得最坏情况下，下一次需要处理的区间的大小尽可能的小。具体证明这里就略过。

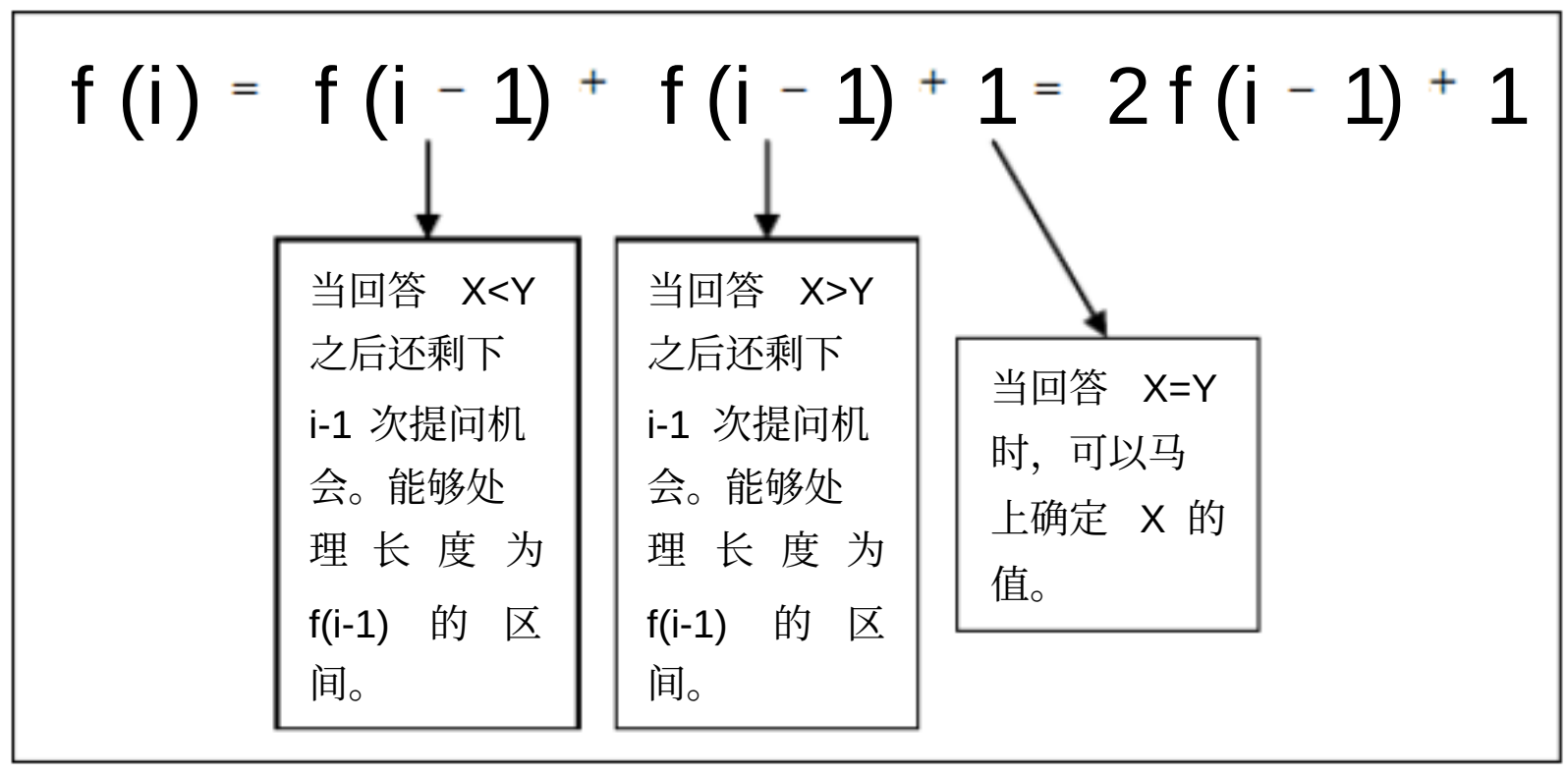
需要注意的是，问题要求的是次数。那么需要询问的次数是不是简单的 $\lceil \log_2(R-L+1) \rceil$，显然不是的。因为对于每次询问的回答有三种可能，即还存在 X=Y 的关系。

用数学的语言描述，设 f(i) 表示询问 i 次询问能够处理长度为 f(i) 的区间，下图分析了 f(i) 应该满足的关系。



当进行完了一次询问之后，就只剩下 i-1 次询问了。绿色区间的长度和红色区间的长度不能超过 i-1 次询问能够处理的范围。也就是说绿色区间和红色区间的长度最大也就是 f(i-1) 。

这样，我们就得到了相应的递推式：



可以从递推式中看到，由于有了 X=Y 的情况，$f(i) \neq 2^{i-1}$。不过我们仍然可以利用递推的方法来得到我们的算法： 已知 $f(1)=1$，根据 $f(i)=2f(i-1)+1$ 依次计算 **f(2),f(3)**…，直到某个 f(ans) 满足 $f(ans) \geq N$，则答案为 **Ans**。

这个算法的时间复杂度 O(Ans) 的，由于 f 函数的增长比指数函数略快。所以可以知道 $O(Ans)=O(\log_2 N)$。

也许有些人会问了，这个算法有什么实际价值呢？对于这个简单的问题，用最开始的二分方法，统计从开始询问直到问出 X 值，一共询问了多少次就可以

了。时间复杂度同样为 $O(\log_2 N)$。似乎递推算法完全没有必要。

对于本题也许是如此， 但是递推法相比简单的二分方法的最大优势就是它是通过数学具体分析， 而不是依据经验或猜想得出的算法。 所以递推法的可推广性更强，当问题变得复杂一些的时候，仍能适用。

三、 基本猜数问题的两个加强

现在要直接解决原题仍然比较困难， 我们不妨先来看看和原问题相关的两个基本猜数问题的加强：

加强 **1**：

被猜数 X 是 1 到 N 范围内的整数， 每次你可以询问一个整数 Y 和 X 的大小关系。同时若你得到了 K 次以上（含 K 次）X>Y的回答，那么你就失败了，你要避免这种情况的发生。给出 N、K，问在最坏情况下需要几次询问才能保证猜出被猜数 X。

由于引入了限制条件： 同时若你得到了 K次以上（含 K 次）X>Y的回答，那么你就失败了。 所以没有办法再用简单的二分来解决， 那么就来尝试用递推的思路解决这道题目，由于多了一个变量 k，所以需要二维递推。

设 $f(i, j)$ 表示用 i 次询问，在得到了 j 次以上 X>Y 回答后游戏就会结束的情况下，最大能够处理的区间长度。类似的，我们可以作出如下分析：

当回答为 X<Y 时，X 可能出现的区间。这时询问次数减少了一次，所以这段区间的长度最长是 $f(i-1, j)$。

当回答为 X>Y 时， X 可能出现的区间。 这时不仅询问次数减少一次，而且还得到了一次 X>Y 的回答，所以这段区间的长度最长为 $f(i-1, j-1)$。

由上面的图例可以看出 ，$Y = L + f(i-1, j)$

所以可以得到 $f(i, j)$ 的递推式为：

$$f(i, j) = f(i-1, j) + f(i-1, j-1) + 1$$

类似的可以得到算法： 已知 $f(0, j) = 0$， $f(i, 0) = 0$，依次按照递推式，递推求解 **f**。直到某个 $f(ans, k)$，满足 $f(ans, k) \geq n$，则答案为 **Ans**。这个算法的时间复杂度为 $O(Ans \times K)$。由于 $f(i, j)$ 的增长速度比组合函数 C_i^j 更快，所以算法

的时间复杂度是很低的。

上面的例子中， 在求递推式的时候， 我们是通过确定三种情况分别需要处理的区间长度，从而确定 f (i, j) 的整个区间长度。在这里需要强调一点的是，当我们把所有 f 函数的值都求出来之后， 对于每个状态 f (i, j) 我们所需要采取的决策，也就是询问的数 Y 也已经确定了！ 对于本题，上面的图例显示 **Y** 左边至多只能有 f (i −1, j) 个数， 右边至多有 f (i −1, j −1) 个数， 由于 $R - L + 1 \le f(i, j)$ 那我们不妨就令 $Y = L + f(i-1, j)$，这样就能在任何情况下均满足条件。 可以看出这里 $Y \ne \left\lfloor \frac{L+R}{2} \right\rfloor$，也就是说询问的时候并不一定将整个区间均分， 二分在这里并不适用。

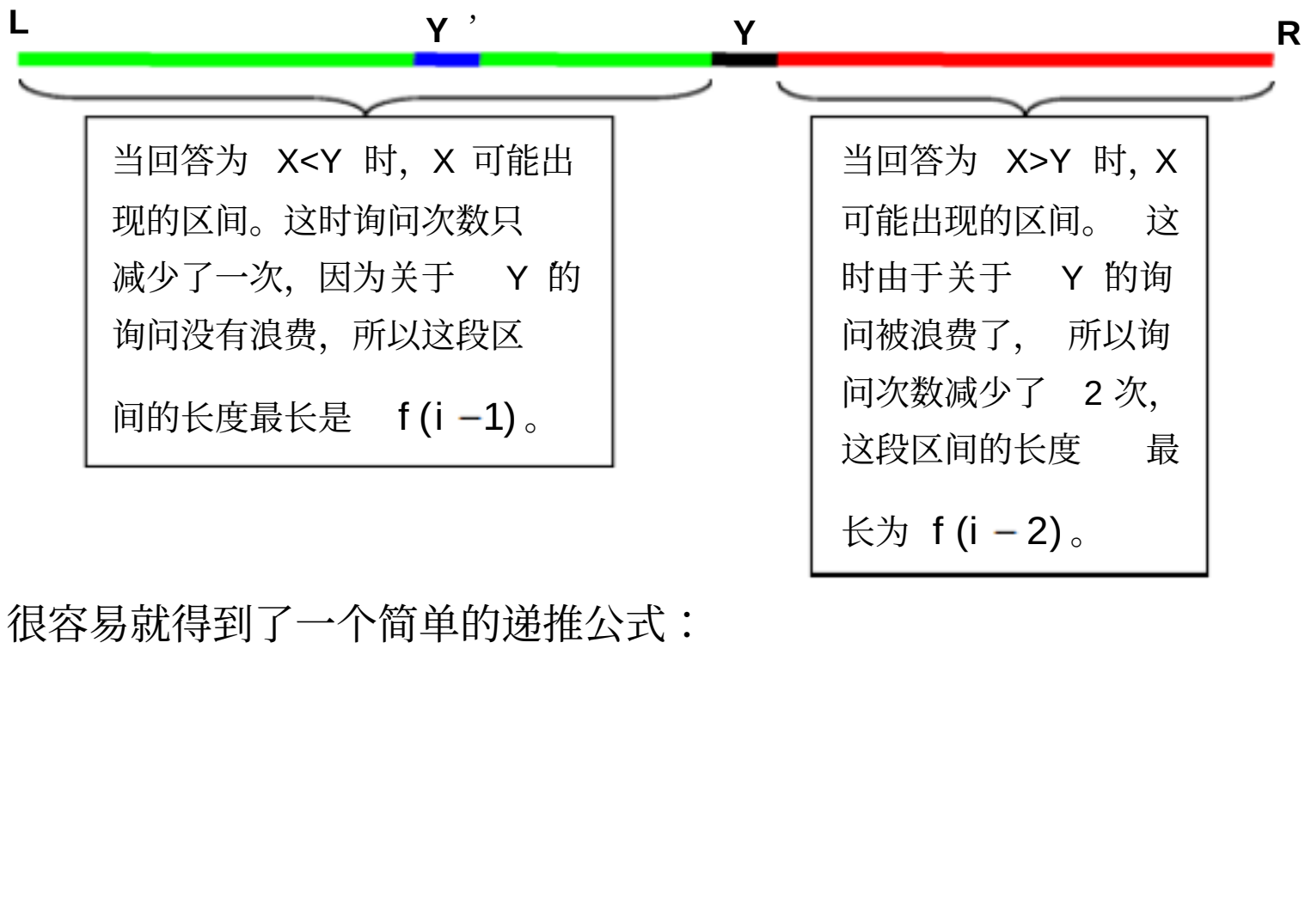
可以看出， 这道题可以看作是本文开篇提出的问题的一部分， 下面再来看基础猜数问题的另一种加强。

加强 **2**：

被猜数 X 是 1 到 N 范围内的整数， 每次你可以询问一个整数 Y 和 X 的大小关系。不同于普通猜数问题， 本次提问的回答你不会马上得到， 而是会在你下一次提出问题后才告诉你。 也就是你每问一个问题之后， 得到的回答是你上一次提问的答案。给出 N, 请问在最坏情况下至少需要几次才能保证猜出 X？

由于提问的答案不能马上获得， 所以思考起来有一定的难度。 但是仔细分析的话，还是能够用递推来解决这道题目。 如同上题一样， 我们也根据回答的三种情况来求得递推式，但是问题就是： 当我们询问了一个 **Y** 之后，不会得到关于 **Y** 的任何信息。要得到这些信息我们必须再询问一个数 **Y**’。显然，在我们什么也不知道的情况下，只能凭空询问，要么把 Y 问在小于 Y 的区间，要么问在大于 Y 的区间， 且只有当询问完之后，才知道 X 与 Y 的关系，才知道关于 Y 的询问是不是多余和浪费。 由于这里没有其他的限制条件， 我们可以认为， Y 问在小于 Y 的区域和问在大于 Y 的区域是等价的。 不妨设 **Y** 问在了小于 **Y** 的区域。

设 f (i) 表示询问 i 次询问能够处理的最长区间长度。



很容易就得到了一个简单的递推公式：

$$f(i) = f(i-1) + f(i-2) + 1$$

同时根据上图，通过类似上一道题的分析，可以知道： $Y = L + f(i-1)$，$Y' = L + f(i-2)$。而对于题目要求求出的最少询问次数，类似的可以得到算法：

已知 $f(0) = 0, f(1) = 0$，根据 $f(i) = f(i-1) + f(i-2) + 1$依次计算 f 值，直到出现某个 f 值 $f(ans) \geq n$，则 Ans 为所求的最少询问次数。

下面来分析算法的时间复杂度， 同前几题一样， 这个算法的时间复杂度也和 Ans 有关，就为： $O(Ans)$。由于 **f** 函数的增长比 **Fibonacci** 数列更快，所以可以认为 $O(Ans) = O(\log_{\partial} N) = O(\log_2 N)$。

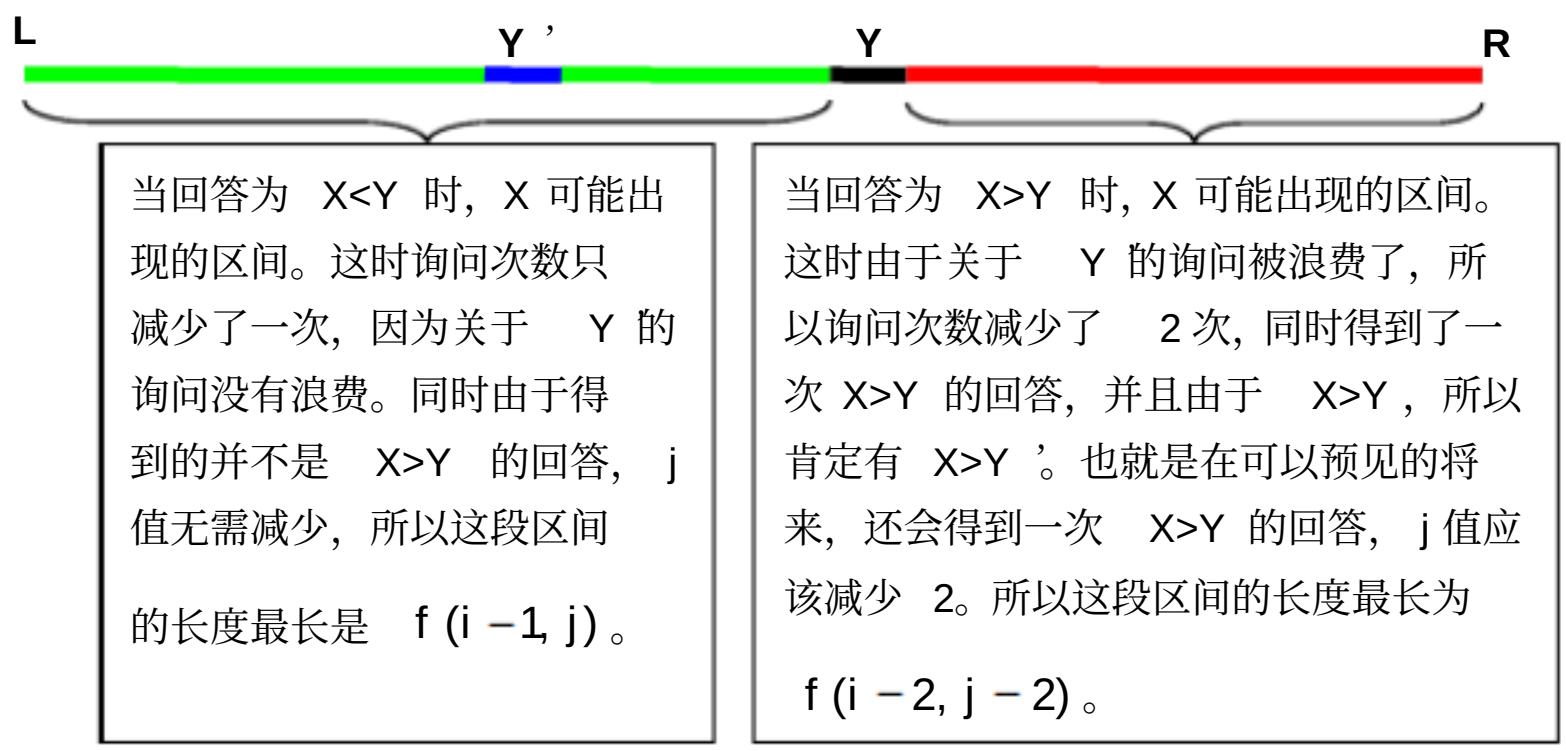
上面就是我们利用递推法， 通过系统的分析， 解决两道由基础猜数问题变化、加强而来的题目的全过程。有这些基础，我们现在可以回到原问题的研究了。

四、 原问题的研究

现在让我们回到原问题。

事实上，如果将上面两个问题的限制条件合并， 就得到了原问题。 我们已经把上面两个问题用递推法完整的解决， 下面来研究合并了之后的原问题。 综合解决上面两道题目的经验，我们仍然尝试使用递推法来求解：

设 $f(i, j)$ 表示用 i 次询问机会，在得到了 j 次以上 X>Y 回答后游戏就会结束的情况下，最大能够处理的区间长度。由于本题的回答也是有“延迟”的，所以在询问了 Y 之后，我们必须“盲目”的询问一个 Y’，以求获得 Y 的回答。类似上面的几题，可以做出如下分析：

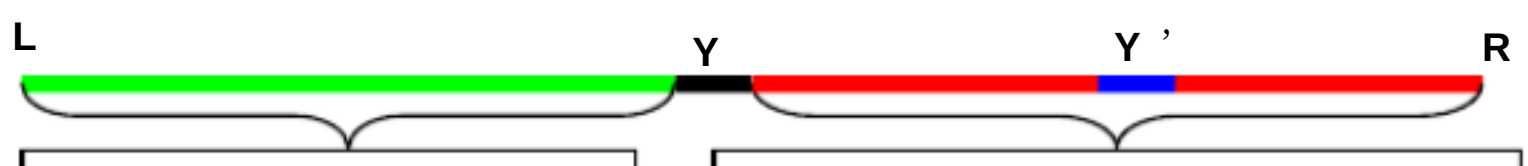


根据这个分析，我们可以很轻松的得到递推式：

$$f(i,j) = f(i-1,j) + f(i-2,j-2) + 1$$

这道题是不是就这样解决了呢？严谨的读者可能已经发现，这样是漏洞的。因为我们这个式子的得出是建立在前面的一个假设上： 不妨设 **Y** 问在了小于 **Y** 的区域。而在这道题里， 针对前面题目的这个假设并不成立。 这是因为存在一个获得 X>Y 的回答次数的限制条件。 也就是 我们必须讨论 **Y** '询问在大于 **Y** 区域里的情况。

这并不复杂，我们有：



当回答为 X<Y 时，X 可能出现的区间。由于 Y 的询问被浪费了，所以询问次数减少了 2 次。而 X<Y 所以肯定有 X<Y '，即 J 无需减少。所以这段区间的长度最长是 $f(i-2,j)$。

当回答为 X>Y 时，X 可能出现的区间。这时由于关于 Y '的询问并没有被浪费，询问次数减少 1 次。同时得到了一次 X>Y 的回答。虽然可能存在 X>Y '的回答，但是这在以后的针对 Y '的分析中还会被讨论到， 即这里只需要考虑 J 值减少 1。所以这段区间的长度最长为 $f(i-1,j-1)$。

现在我们终于发现， 我们写出来的式子已经不在是一个递推式。 而是一个需要决策的动态规划的式子， 这在一定程度上也说明递推和动态规划方法其实是相通的。由于是区间的最大可能值， 所以我们把两种情况统统考虑， 取最大值即得到：

$$f(i,j) = \max\{f(i-1,j) + f(i-2,j-2), f(i-2,j) + f(i-1,j-1)\} + 1$$

得到了这个动态规划的式子， 后面的工作就和前面的两道题很类似了。 已知 $f(0,j)=0, f(1,j)=0, f(i,0)=0$。根据动态规划式子，依次计算 **f** 值，直到存在一个 $f(ans,k) \geq N$，则答案为 **Ans**。而对于具体询问值 Y，这个算法同样能够求出来。我们根据 $f(i,j)$ 的不同决策，决定不同的 Y 值：当 $f(i,j) = f(i-1,j) + f(i-2,j-2) + 1$ 可以知道 $Y = L + f(i-1,j)$，而当 $f(i,j) = f(i-2,j) + f(i-1,j-1) + 1$ 的时候，则 $Y = L + f(i-2,j)$。

这个算法的时间复杂度为 $O(AnsK)$，由于 f(i,j) 从递推式来看仍然是一个增长非常快的函数， 所以本算法的实际运行效果非常好。 对于题目给出的范围以内的数据，均在 0.1s 内求出了答案。

至此，我们就把一道看似非常复杂的猜数问题， 通过从最初始情况开始的一步一步分析， 逐步地解决了。 在解决本题的过程中， 可以看出递推法在这类猜数

问题中是非常行之有效的一种分析方法。 它为我们提供了一种系统的、 从本质上分析问题的方法， 来代替盲目的猜测与尝试。 不仅局限于这类问题， 我们在面对其他问题的时候， 也要尽量用系统的数学式的思路来思考问题， 而避免进行停留在表面的分析。这一点对于解决所有问题都是相通的。

五、 同类问题的讨论

硬币游戏机：

某游戏厅有一个硬币游戏机， 游戏的基础就是最普通的猜数游戏。 你猜一个数 Y，然后机器告诉你猜对了或者比目标数 X大还是小。如果你猜对了，游戏直接结束，如果你猜的数比 X大，你就需要投入 a 枚硬币让游戏继续， 如果你猜的数比 X小，你就需要投入 b 枚硬币让游戏继续。 已知 X是 1-N 里面的数。N已知，请问需要多少枚硬币才能保证猜出 X？

应用递推法进行分析，设 f (i) 表示 i 枚硬币能够猜出的区间大小。根据前面题目的经验，稍做分析就可以得出如下递推式：

$$f(i) = f(i-a) + f(i-b) + 1$$

然后求出最小的 Ans 满足 f (ans) ≥ n 就可以了，这时候答案就是 Ans。值得注意的是，这里和其他题目不同，初始条件中 $f(0) = 1$， $f(k) = 0 (k < 0)$。

这个算法的时间复杂度同样为 O(Ans) 。

另类猜数：

被猜数 X是 1 到 N范围内的整数， 每次你可以询问一个整数 Y和 X的大小关系。不同于普通猜数问题， 本次提问的回答你不会马上得到， 也不是在你下一次提出问题后，而是在你再提问 c 次之后才告诉你。也就是你每问一个问题之后，得到的回答是之前第 c 次提问的答案。给出 N、 C，请问在最坏情况下至少需要几次才能保证猜出 X？

这是 SGU 题库中的一道题。

与第三节第二题不同，这里的问题是：在提出了一个问题之后，中间有 C 次问题，需要“ 盲目 ”的问。那么到底怎么分配这 C 次提问呢？如果我们考虑所有的情况显然是不可能的，因为 C 次提问，每次都可以选择问在 >Y 的区域还是<Y 的区域， 所以总共有 2^C 种不同的情况。 也就是我们如果将它写成正规的动态规划式子需要 2^C 个值里取最大值。同时只设一个 f (i) 。

根据平时的经验，最优值总是出现在极端的情况。我们这里尝试将 C 次盲目提问全部投资在 >Y 的区域（全部投资在 <Y 也是一样的）。最后我们得到了递推式：

$$f(i) = f(i-1) + f(i-c-1) + 1$$

用这个递推式计算出来的答案，和盲目搜索进行比对，发现小数据的时候完全一样。将这个递推程序提交到 SGU 上去，通过了题库提供的所有数据。至此已经可以基本确定这个递推式是正确的了。但是遗憾的是，暂时作者还没有想到什么好方法来严格证明其正确性，希望读者能够继续对这道题进行研究，写这道题的解法也是希望能够起到一个抛砖引玉的作用。

【总结】

同类的例题远不止本文提到的这些，而其中大部分都能够使用本文介绍的方法来解决。

而对于一道题目，最重要的是要仔细分析题目，学会如何抛开繁杂的题目描述，从数学模型的角度系统的来分析问题，一旦你做到了这些，你会发现很多难题都迎刃而解。而这才是本文真正的重点所在，也是作者写这篇文章的真正目的。

【参考文献】

1. 朱晨光同学 2004 年国家集训队论文。
2. 《算法艺术与信息学竞赛》 刘汝佳、黄亮
3. SGU 题库试题——http://acm.sgu.ru