Ulam猜数游戏

任之洲

清华大学 交叉信息研究院

2017 年 2 月 4 日

概述

- 围绕Ulam猜数游戏讨论一类容错的检索问题。

概述

- 围绕Ulam猜数游戏讨论一类容错的检索问题。
- 在以前的OI比赛中，容错类问题多以“噪音”的形式出现。
  - 如“以一个小概率p随机出现错误”。

概述

- 围绕Ulam猜数游戏讨论一类容错的检索问题。
- 在以前的OI比赛中，容错类问题多以“噪音”的形式出现。
  - 如“以一个小概率p随机出现错误”。
- 今天讨论的问题形式为“错误的出现次数不超过限制e”。
  - 其重要的特征为“发生的错误数量不随其它量的增长影响”。
  - 即“噪点的数量不会随信息总长的增加而增多”。

概述

- 围绕Ulam猜数游戏讨论一类容错的检索问题。
- 在以前的OI比赛中，容错类问题多以“噪音”的形式出现。
  - 如“以一个小概率p随机出现错误”。
- 今天讨论的问题形式为“错误的出现次数不超过限制e”。
  - 其重要的特征为“发生的错误数量不随其它量的增长影响”。
  - 即“噪点的数量不会随信息总长的增加而增多”。
- 讨论这个问题有什么意义呢？

概述

- 围绕Ulam猜数游戏讨论一类容错的检索问题。
- 在以前的OI比赛中，容错类问题多以“噪音”的形式出现。
  - 如“以一个小概率p随机出现错误”。
- 今天讨论的问题形式为“错误的出现次数不超过限制e”。
  - 其重要的特征为“发生的错误数量不随其它量的增长影响”。
  - 即“噪点的数量不会随信息总长的增加而增多”。
- 讨论这个问题有什么意义呢？
  - Ulam游戏实际上是一个通信类问题。
  - ~~槽点满满的~~Arbiter都已经支持评测交互题了！

概述

- 围绕Ulam猜数游戏讨论一类容错的检索问题。
- 在以前的OI比赛中，容错类问题多以“噪音”的形式出现。
  - 如“以一个小概率p随机出现错误”。
- 今天讨论的问题形式为“错误的出现次数不超过限制e”。
  - 其重要的特征为“发生的错误数量不随其它量的增长影响”。
  - 即“噪点的数量不会随信息总长的增加而增多”。
- 讨论这个问题有什么意义呢？
  - Ulam游戏实际上是一个通信类问题。
  - ~~槽点满满的~~Arbiter都已经支持评测交互题了！
- 今后是否有可能在OI比赛中出现呢？

概述

- 围绕Ulam猜数游戏讨论一类容错的检索问题。
- 在以前的OI比赛中，容错类问题多以“噪音”的形式出现。
  - 如“以一个小概率p随机出现错误”。
- 今天讨论的问题形式为“错误的出现次数不超过限制e”。
  - 其重要的特征为“发生的错误数量不随其它量的增长影响”。
  - 即“噪点的数量不会随信息总长的增加而增多”。
- 讨论这个问题有什么意义呢？
  - Ulam游戏实际上是一个通信类问题。
  - ~~槽点满满的~~Arbiter都已经支持评测交互题了！
- 今后是否有可能在OI比赛中出现呢？
  - 请随意想象。

Ulam猜数游戏

交互式模型

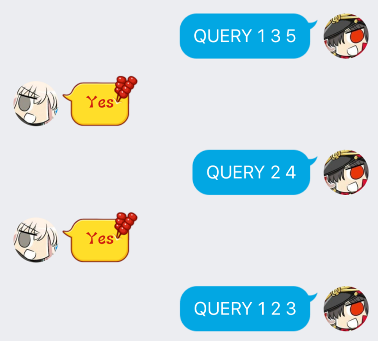
- 游戏由提问者和回答者双方交互完成。
- 在游戏开始时，回答者选择一个$[1, N]$ 范围内的整数X。
- 提问者每次可以选择一个整数集S，询问X是否在集合S内。
- 提问者希望用尽量少的询问次数来确定X的值。
- 回答者可以在游戏进行的过程中变动X的值。
  - 必须保证最后有合法解。
- 回答者可以撒谎不超过K次。

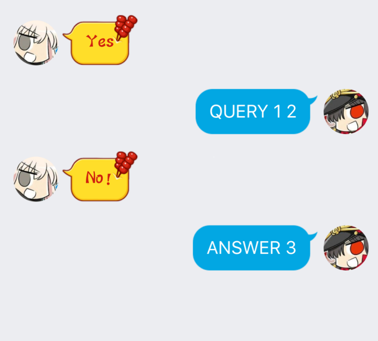
非交互式模型

- 规则与交互式模型类似，但不能实时得到反馈。
- 即一开始就要将所有询问一起给出。
- 得到的反馈中有不超过K条是错误的。

一个交互式样例

- 在该样例中$N = 5, K = 1$。





- 前两次询问的回答产生了矛盾，所以必然已经使用了撒谎机会。
- 由后两次询问容易得到答案$X = 3$。

Ulam's game

- 在游戏开始时，回答者选择一个$[1, N]$ 范围内的整数X。
- 提问者每次可以选择一个整数集S，询问X是否在集合S内。
- 提问者希望用尽量少的询问次数来确定X的值。
- 回答者可以撒谎不超过K次。
- 交互式/非交互式

- 只考虑$K = 1$的情况。
- 欢迎手玩$N = 8, K = 1$的情况。

$K = 0$的情况

- 回答者没有机会通过撒谎来进行干扰，可以采取一些简单的策略。

$K = 0$的情况

- 回答者没有机会通过撒谎来进行干扰，可以采取一些简单的策略。

- 对于交互式模型，可以每次选取一半数询问。
- 根据询问的答案可以直接筛去一半的可能解。
- 最坏次数：$\lceil \log N \rceil$

$K=0$的情况

- 回答者没有机会通过撒谎来进行干扰，可以采取一些简单的策略。

- 对于交互式模型，可以每次选取一半数询问。
- 根据询问的答案可以直接筛去一半的可能解。
- 最坏次数：$\lceil \log N \rceil$

- 对于非交互式模型，每次选择某个二进制位上是1的所有数询问。
- 这样每次询问可以确定X的一个二进制位。
- 最坏次数：$\lceil \log N \rceil$

$K = 1$的情况

- 将$K = 0$的做法进行简单扩展可以得到$K = 1$的朴素解法。

$K = 1$的情况

- 将$K = 0$的做法进行简单扩展可以得到$K = 1$的朴素解法。

- 对于交互式模型，每次询问重复两次。
- 当两次询问的回答出现不同时，再询问一次。
- 最坏询问次数：$2\lceil \log N \rceil + 1$

$K = 1$的情况

- 将$K = 0$的做法进行简单扩展可以得到$K = 1$的朴素解法。

- 对于交互式模型，每次询问重复两次。
- 当两次询问的回答出现不同时，再询问一次。
- 最坏询问次数：$2\lceil \log N \rceil + 1$

- 对于非交互式模型，由于不能得到实时反馈。
- 需要将每个二进制位询问三次。
- 询问次数：$3\lceil \log N \rceil$

问题转化

- 首先从非交互式模型入手。

粗略转化

- 需要传输一个长度为$\lceil \log N \rceil$的01串。
- 传输的形式同样也是01串，体现在该问题的询问上。
- 传输会有噪音，有不超过K位会被修改。
- 目的：最小化传输的串长L。

- 只讨论$K = 1$的情况。
- 前面的做法相当于把这个串传输三遍。

一点小改进

Algorithm I

- 首先将整个串传输两遍。
- 再添加一个奇偶校验位，表示这个串的1的个数的奇偶性。

- 假如两遍传输的串一样，那么已经成功了。
- 否则有一位出现了冲突，通过奇偶校验位可以确定这位的数值。

- 询问次数：$2\lceil \log N \rceil + 1$

进一步改进

Algorithm II

- 首先将整个串传输一遍，设传输了n位。
- 将这些位从1到n标号，设立$\lceil \log(n+1) \rceil$个奇偶校验位。
  - 第k个校验位用于检查标号第k个二进制位上是1的位置的奇偶性。
- 同时对所有位置设立一个奇偶校验位。

- 假如第二部分中有w个校验位没有匹配上。

进一步改进

Algorithm II

- 首先将整个串传输一遍，设传输了n位。
- 将这些位从1到n标号，设立$\lceil\log(n+1)\rceil$个奇偶校验位。
  - 第k个校验位用于检查标号第k个二进制位上是1的位置的奇偶性。
- 同时对所有位置设立一个奇偶校验位。

- 假如第二部分中有w个校验位没有匹配上。
- 当$w \geq 2$时，必然是第一部分中的对应位出现的错误。

进一步改进

Algorithm II

- 首先将整个串传输一遍，设传输了n位。
- 将这些位从1到n标号，设立$\lceil \log(n+1) \rceil$个奇偶校验位。
  - 第k个校验位用于检查标号第k个二进制位上是1的位置的奇偶性。
- 同时对所有位置设立一个奇偶校验位。

- 假如第二部分中有w个校验位没有匹配上。
- 当$w \geq 2$时，必然是第一部分中的对应位出现的错误。
- 当$w = 1$时，即标号为2^k的位置发生了错误。
  - 无法分辨是传输串时发生了错误，还是校验位发生了错误。
  - 依靠所有位置的校验位来确定。

进一步改进

Algorithm II

- 首先将整个串传输一遍，设传输了n位。
- 将这些位从1到n标号，设立$\lceil \log(n+1) \rceil$个奇偶校验位。
  - 第k个校验位用于检查标号第k个二进制位上是1的位置的奇偶性。
- 同时对所有位置设立一个奇偶校验位。

- 假如第二部分中有w个校验位没有匹配上。
- 当$w \geq 2$时，必然是第一部分中的对应位出现的错误。
- 当$w = 1$时，即标号为2^k的位置发生了错误。
  - 无法分辨是传输串时发生了错误，还是校验位发生了错误。
  - 依靠所有位置的校验位来确定。

- 询问次数：$\lceil \log N \rceil + \lceil \log(n+1) \rceil + 1$
  - $n = \lceil \log N \rceil$

Hamming Code

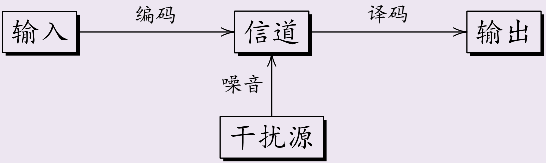
- 首先整个串传输一遍，设传输了n位。
- 将这些位从1到m标号，设立$\lceil \log(m+1) \rceil$个奇偶校验位。
  - 跳过所有形如2^k的标号，最后标到m。
  - 第k个校验位用于检查标号第k个二进制位上是1的位置的奇偶性。

- 在前一个算法中，最后一个校验位只为解决少数位置引发的问题。
- 标号时直接跳过那些位置显得更加经济实惠。

- 询问次数：$\lceil \log N \rceil + \lceil \log(m+1) \rceil$
  - $m \approx \lceil \log(N + \log N) \rceil$

细化模型

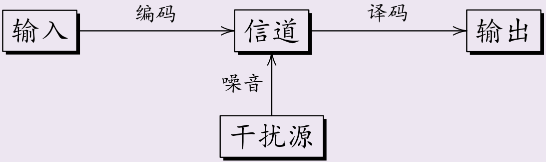
信道编码



- 设需要传输的是一个长度为n的二进制串。
- 编码后得到一个长度为$m \geq n$的二进制串。

细化模型

信道编码

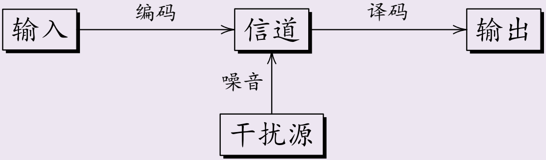


- 设需要传输的是一个长度为n的二进制串。
- 编码后得到一个长度为$m \geq n$的二进制串。

- 编码和译码可以看作两个函数。
  - 编码：n位二进制串$\rightarrow$ m位二进制串
  - 译码：m位二进制串$\rightarrow$ n位二进制串

细化模型

信道编码

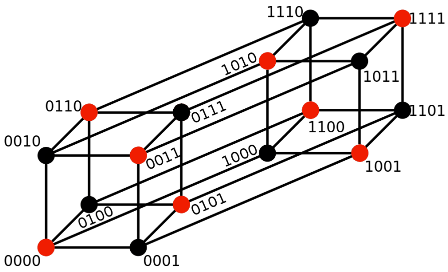


- 设需要传输的是一个长度为n的二进制串。
- 编码后得到一个长度为$m \geq n$的二进制串。

- 编码和译码可以看作两个函数。
  - 编码：n位二进制串$\rightarrow$ m位二进制串
  - 译码：m位二进制串$\rightarrow$ n位二进制串
- 这个问题中，需要确定一个编码方式。
  - 受到噪声影响后，仍能唯一确定输入的串。

编码

- 可以视为将n维二进制空间中的2^n个点映射到m维二进制空间中。



- 上图为一个$n = 3 \to m = 4$的例子。
  - 红色节点为被映射到的点。
  - 发生一位错误相当于移动到一个相邻节点。

译码

- 在m维二进制空间中有2^n个特殊点。
  - 译码的过程就是找最近的特殊点。
- 设曼哈顿距离最近的两个点的距离为d_{min}。
  - 称为最小汉明距离。

译码

- 在m维二进制空间中有2^n个特殊点。
  - 译码的过程就是找最近的特殊点。
- 设曼哈顿距离最近的两个点的距离为d_{min}。
  - 称为最小汉明距离。

- 能完成t位的检错，必须满足$d_{min} \geq t + 1$。
- 能完成t位的纠错，必须满足$d_{min} \geq 2t + 1$。

译码

- 在m维二进制空间中有2^n个特殊点。
  - 译码的过程就是找最近的特殊点。
- 设曼哈顿距离最近的两个点的距离为d_{min}。
  - 称为最小汉明距离。
- 能完成t位的检错，必须满足$d_{min} \geq t+1$。
- 能完成t位的纠错，必须满足$d_{min} \geq 2t+1$。
- 即使不出成通信题，也可以用SPJ判断编码函数是否合格。

- 假设已经构造完成了$2^n \to 2^m$的映射。
- 那么第k次询问选择映射后第k位是1的数集。
- 当$N = 2^n$时，这两个问题完全等价。

- 当$K = 1$时，前面使用的汉明码就是一种优秀的一位纠错码(single-error-correcting code)。
- 当$K > 1$时，可以选用更高阶的纠错码。

- 将非交互的算法直接移植过来当然也是可行的。
- 但能够实时得到反馈可以使得策略更加灵活。

- 在参考文献[2]中指出：

性质0

- 设$f(N)$和$g(N)$分别为交互式与非交互式算法的最优询问次数。
- 当$K=1$时

$$f(N) \le g(N) \le f(N)+1$$

- 在$N=21$的时候，两种算法的询问次数首次出现不同。

回答方应该采取怎样的策略?

- 接下来介绍交互式模型$K \geq 1$时的解法。
- 首先考虑一个问题:在交互过程中,回答方应该采取怎样的策略?
- 对于面前的一次询问,回答会将局面引向两种状态。
- 如何判断哪个状态对提问方更不利?
- 比较哪个状态的熵更大。
  - 即比较状态的不确定性。
- 如何描述一个状态?如何量化不确定性?

如何表示状态？

- 在若干次询问之后，需要估计每个数在后续操作中的重要程度。

如何表示状态？

- 在若干次询问之后，需要估计每个数在后续操作中的重要程度。
- 一个重要的评判依据为“假如这个数是X，那么回答者现在已经撒谎几次了”，这个量很容易计算。
  - 对于一个数p。
  - 有k_1次询问集合包含p的回答为No，有k_2次不包含p的回答为Yes。
  - 那么假如p是答案，回答者就已经撒谎$k_1 + k_2$次了。

如何表示状态？

- 在若干次询问之后，需要估计每个数在后续操作中的重要程度。
- 一个重要的评判依据为“假如这个数是X，那么回答者现在已经撒谎几次了”，这个量很容易计算。
  - 对于一个数p。
  - 有k_1次询问集合包含p的回答为No，有k_2次不包含p的回答为Yes。
  - 那么假如p是答案，回答者就已经撒谎k_1+k_2次了。
- 为了方便之后的计算，改写为“假如这个数是X，那么回答者现在还可以撒谎几次”。
  - 继续使用的上面的例子。
  - 总的撒谎次数限制为K次。
  - 回答者还可以撒谎的次数为$K-(k_1+k_2)$次。

如何表示状态？

- 设有w_i个数满足“假如这个数是答案，那么回答者还可以撒谎i次”，考虑将状态用如下序列表示。

$$W = \{w_K, w_{K-1}, \ldots, w_1, w_0\}$$

- 设W_i表示对应的w_i 个数构成的集合，即$|W_i| = w_i$。

如何表示状态？

- 设有w_i个数满足“假如这个数是答案，那么回答者还可以撒谎i次”，考虑将状态用如下序列表示。

$$W = \{w_K, w_{K-1}, \ldots, w_1, w_0\}$$

- 设W_i表示对应的w_i 个数构成的集合，即$|W_i| = w_i$。
- 容易得到，游戏开始时的状态为

$$W_{init} = \{N, 0, 0, \ldots, 0, 0\}$$

如何表示状态？

- 设有w_i个数满足“假如这个数是答案，那么回答者还可以撒谎i次”，考虑将状态用如下序列表示。

$$W = \{w_K, w_{K-1}, \ldots, w_1, w_0\}$$

- 设W_i表示对应的w_i 个数构成的集合，即$|W_i| = w_i$。
- 容易得到，游戏开始时的状态为

$$W_{init} = \{N, 0, 0, \ldots, 0, 0\}$$

- 可以直接推导出答案的状态满足

$$\sum_{i=0}^{K} w_i = 1$$

状态如何转移？

- 考虑回答者视角，对于一次询问，类似表示为两个不相交集合。
  - A表示被询问到的数构成的集合。
  - B表示没有被询问到的数构成的集合。
  - 类比W按撒谎次数分类表示如下

$$A = \{a_K, a_{K-1}, \ldots, a_1, a_0\} \quad B = \{b_K, b_{K-1}, \ldots, b_1, b_0\}$$

  - 同时满足$a_i + b_i = w_i$，$A_i \cap B_i = W_i$。

状态如何转移？

- 考虑回答者视角，对于一次询问，类似表示为两个不相交集合。
  - A表示被询问到的数构成的集合。
  - B表示没有被询问到的数构成的集合。
  - 类比W按撒谎次数分类表示如下

$$A = \{a_K, a_{K-1}, \ldots, a_1, a_0\} \quad B = \{b_K, b_{K-1}, \ldots, b_1, b_0\}$$

  - 同时满足$a_i + b_i = w_i$，$A_i \cap B_i = W_i$。
- 假如回答是Yes，那么状态转移到

$$W_{Yes} = \{a_K, a_{K-1} + b_K, \ldots, a_1 + b_2, a_0 + b_1\}$$

- 假如回答是No，那么状态转移到

$$W_{No} = \{b_K, b_{K-1} + a_K, \ldots, b_1 + a_2, b_0 + a_1\}$$

如何量化不确定性？

- 先不加证明地给出一个估价公式或体积(volume)函数。
- 该公式用于判断提问方是否可能在q次询问中得到答案。
- 设

$$V_q(W) = \sum_{i=0}^{K} w_i \sum_{j=0}^{i} \binom{q}{j}$$

如何量化不确定性？

- 先不加证明地给出一个估价公式或体积(volume)函数。
- 该公式用于判断提问方是否可能在q次询问中得到答案。
- 设

$$V_q(W) = \sum_{i=0}^{K} w_i \sum_{j=0}^{i} \binom{q}{j}$$

- 这个式子取对数后其实就是熵，先讨论直观意义。
- 假定还有q次询问机会，e次撒谎机会，那么可能的情况数为

$$\sum_{j=0}^{e} \binom{q}{j}$$

如何量化不确定性？

性质1

$$V_q(W) = V_{q-1}(W_{Yes}) + V_{q-1}(W_{No})$$

- 可以搁在搜索树上想象一下。
- 也可以看一看定义

$$V_q(W) = \sum_{i=0}^{K} w_i \sum_{j=0}^{i} \binom{q}{j} = \sum_{i=0}^{K} (a_i + b_i) \sum_{j=0}^{i} \binom{q}{j}$$

$$V_{q-1}(W_{Yes}) = \sum_{i=0}^{K} a_i \sum_{j=0}^{i} \binom{q-1}{j} + \sum_{i=1}^{K} b_i \sum_{j=1}^{i} \binom{q-1}{j-1}$$

$$V_{q-1}(W_{No}) = \sum_{i=0}^{K} b_i \sum_{j=0}^{i} \binom{q-1}{j} + \sum_{i=1}^{K} a_i \sum_{j=1}^{i} \binom{q-1}{j-1}$$

- 利用组合数递推公式$\binom{q}{j} = \binom{q-1}{j} + \binom{q-1}{j-1}$证明。

如何量化不确定性？

性质2 (计算理论下界)

双方最佳策略下，提问方能在q次询问后得到答案的必要条件为

$$V_q(W) \le 2^q$$

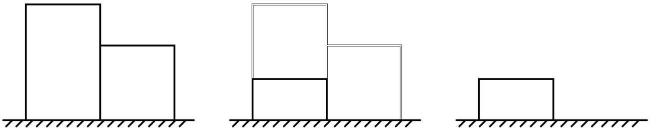
- 直接当熵来看的话很自然，另外也可以归纳证明。
- 当$q = 0$时，显然成立。
  - 当没有询问机会的时候，只能存在一个数。
- 当$q > 1$时，假如$V_q(W) > 2^q$。
  - 根据性质1，$Max\{V_{q-1}(W_{Yes}), V_{q-1}(W_{No})\} > 2^{q-1}$。
  - 回答方只需要选择超过2^{q-1}的那个分支。
  - 归纳得到结论成立。

回答者视角

- 希望将提问者引向熵更大（估价更大）的局面。
- 利用性质2可以计算出W_{Yes}和W_{No}哪个状态对提问者更不利。

提问者视角

- 希望转移到熵更小（估价更小）的局面。
- 由于性质1的约束，每次最多只能将估价减半。
- 每个元素在W_{Yes}和W_{No}中都有相应贡献。
  - 只关心贡献量的差值。



- 可以转化为集合均分问题。

提问者视角

集合带权均分

有N个数A_i，将这些数分为两部分，使得两部分的和的差值尽量小。

提问者视角

集合带权均分

有N个数A_i，将这些数分为两部分，使得两部分的和的差值尽量小。

- 选用一些简单的贪心就可以达到比较好的效果。
  - 将N个数排成降序，依次决策。
  - 每次把当前数放入较小的那一堆。

提问者视角

集合带权均分

有N个数A_i，将这些数分为两部分，使得两部分的和的差值尽量小。

- 选用一些简单的贪心就可以达到比较好的效果。
  - 将N个数排成降序，依次决策。
  - 每次把当前数放入较小的那一堆。
- 其他方法
  - 参考文献[3]，实测效果比前一种略好一点。
  - 参考文献[1]，$N = 10^{20}, K \leq 4$时的构造解。

Binary search with a fixed number of errors

容错的二叉搜索

- 提问者每次可以把元素划分成两个集合。
- 回答者指出答案在这两个集合的某一个中。
- 撒谎有一个固定的次数限制e（不随其它量增长）。

- 枚举询问次数q，将错误信息量化。
- 计算搜索空间的体积，可以用熵来理解。

$$V_q(W) = \sum_{i=0}^{e} w_i \sum_{j=0}^{i} \binom{q}{j}$$

- 转化为划分问题(partition problem)。

容错的k叉搜索

- 提问者每次可以把元素划分成k个集合。
- 回答者指出答案在这k个集合的某一个中。
- 撒谎有一个固定的次数限制e（不随其它量增长）。

- 二叉搜索问题的一般化。

- 同样枚举询问次数q，量化错误信息。
- 同样计算搜索空间的体积，不过计算方式略有不同。
- 最后转化为k划分问题(k-partition problem)。

k叉搜索的搜索空间

性质2 (改)

双方最佳策略下，提问方能在q次询问后得到答案的必要条件为

$$V_{k,q}(W) = \sum_{i=0}^{e} w_i \sum_{j=0}^{i} (k-1)^j \binom{q}{j} \leq k^q$$

- $(k-1)^j$：除了需要计算出错的位置，还要考虑出错后的状态。
- k^q：回答者的答案有k种情况，即每次选择k个分支中的一个，用熵理解就是得到$\log_2 k$ bit的信息。

k叉搜索的搜索空间

性质2 (改)

双方最佳策略下，提问方能在q次询问后得到答案的必要条件为

$$V_{k,q}(W)=\sum_{i=0}^{e} w_i \sum_{j=0}^{i} (k-1)^j \binom{q}{j} \leq k^q$$

- $(k-1)^j$：除了需要计算出错的位置，还要考虑出错后的状态。
- k^q：回答者的答案有k种情况，即每次选择k个分支中的一个，用熵理解就是得到$\log_2 k$ bit的信息。

- 容易证明也有类似的性质1成立。
- 不难理解最后会转化为k划分问题。

一个经典问题

称球问题

- 有n个球，其中$n-1$个球重量一样，剩下一个稍重一些。
- 有一架天平，可以通过称重来找出那个偏重的球。
- 最小化称重次数。

- 将n个球标号后，可能的答案有n种。
- 天平返回的情况有三种，可以利用熵计算出称重次数的下界为$\lceil \log_3 n \rceil$。

一个经典问题

称球问题

- 有n个球，其中$n-1$个球重量一样，剩下一个稍重一些。
- 有一架天平，可以通过称重来找出那个偏重的球。
- 最小化称重次数。

- 将n个球标号后，可能的答案有n种。
- 天平返回的情况有三种，可以利用熵计算出称重次数的下界为$\lceil \log_3 n \rceil$。

- 一个广为流传的版本中，剩下的一个球可能偏重也可能偏轻。
- 由于还要考虑偏轻和偏重两种情况，故下界为$\lceil \log_3 2n \rceil$。
- 为了减少繁琐的讨论，只考虑上面的简化版本。

称球问题 (改)

- 有n个球，其中$n-1$个球重量一样，剩下一个稍重一些。
- 你的朋友有一架天平，但他不肯每次都老实地帮你称重。
- 称重请求得到的反馈中，会错误不超过e次。
- 最小化称重请求次数。

- 设天平上的两堆球是A、B，剩下那堆为C。
- 天平的三种结果事实上将答案分别指向这三堆球之一。

- 可以看出这是一个三叉搜索问题。
- 有一个额外限制：$|A| = |B|$。

- 参考文献[4]给出了$e = 2$时的策略。

- 掉线的同学可以尝试重连了。

- 接下来介绍两个与Ulam游戏类似但不相关的问题。
  - Selecting the largest element with errors
  - Sorting with with errors

Selecting the largest element with errors

容错的最大值问题

- 有n个权值互不相同物品，需要在其中找出最大的那一个。
- 每个物品的权值都是未知的，只能通过一个比较器来进行比较。
- 比较器每次可以比较两个不同的物品，返回值只会是“大于”或“小于”中的一个。
- 这个比较器总共有e次机会返回错误的答案。
- 求比较次数尽量少的操作策略。

Selecting the largest element with errors

一个重要子问题

- 比较两个物品A、B。

- 不断地使用比较器，直到某一种答案出现$e+1$次时停止。
- 设t_i为返回的错误信息次数，即比较次数为$e+1+t_i$。

Selecting the largest element with errors

一个重要子问题

- 比较两个物品A、B。

- 不断地使用比较器，直到某一种答案出现$e+1$次时停止。
- 设t_i为返回的错误信息次数，即比较次数为$e+1+t_i$。

- 在原问题中，将上面的算法重复$n-1$次就能找到最大值。
- 由于错误次数限制为$e \geq \sum t_i$，故需要的询问次数最多为

$$(e+1)(n-1)+e$$

一个重要子问题

- 比较两个物品A、B。

- 不断地使用比较器，直到某一种答案出现$e+1$次时停止。
- 设t_i为返回的错误信息次数，即比较次数为$e+1+t_i$。

- 在原问题中，将上面的算法重复$n-1$次就能找到最大值。
- 由于错误次数限制为$e \geq \sum t_i$，故需要的询问次数最多为

$$(e+1)(n-1)+e$$

- 接下来证明这个算法是最优的。
- 存在应答策略使得询问次数至少为$(e+1)(n-1)+e$。

第一阶段

- 该应答策略分为两个阶段。
- 第一阶段包含前$(e+1)(n-1)-1$次询问，剩下都归于第二阶段。
- 首先确立一个假定的大小顺序

$$x_1 < x_2 < \cdots < x_{n-1} < x_n$$

- 对于第一阶段的所有询问，都按这个顺序如实处理。

第二阶段

- 定义一个称为负票(negative vote)的概念。
- 假如一次比较的结果为$a < b$，那么称a收到了一张负票。
- 如果一个物品收到了$e + 1$张负票，那么它一定不可能成为最大值。

第二阶段

- 定义一个称为负票(negative vote)的概念。
- 假如一次比较的结果为$a < b$，那么称a收到了一张负票。
- 如果一个物品收到了$e + 1$张负票，那么它一定不可能成为最大值。

- 在第一阶段中，x_n必定没有收到负票。
- 且必定存在至少一个x_m收到的负票数量小于$e + 1$。
  - 因为第一阶段只有$(e + 1)(n - 1) - 1$次比较。

第二阶段

- 定义一个称为负票(negative vote)的概念。
- 假如一次比较的结果为$a < b$，那么称a收到了一张负票。
- 如果一个物品收到了$e + 1$张负票，那么它一定不可能成为最大值。

- 在第一阶段中，x_n必定没有收到负票。
- 且必定存在至少一个x_m收到的负票数量小于$e + 1$。
  - 因为第一阶段只有$(e + 1)(n - 1) - 1$次比较。
- 接下来e次询问中，通过e次错误机会保证x_m不再收到更多负票。
  - 和x_m无关的比较事件如实回答。

第二阶段

- 定义一个称为负票(negative vote)的概念。
- 假如一次比较的结果为$a < b$，那么称a收到了一张负票。
- 如果一个物品收到了$e+1$张负票，那么它一定不可能成为最大值。

- 在第一阶段中，x_n必定没有收到负票。
- 且必定存在至少一个x_m收到的负票数量小于$e+1$。
  - 因为第一阶段只有$(e+1)(n-1)-1$次比较。
- 接下来e次询问中，通过e次错误机会保证x_m不再收到更多负票。
  - 和x_m无关的比较事件如实回答。

- x_n和x_m成为最大值在逻辑上都是成立的。
  - 只需将x_n或x_m各自收到的负票视为错误信息。
- 共$(e+1)(n-1)-1+e$次询问后，仍无法确定答案。

容错的排序问题

- 有n个权值互不相同物品，需要将他们排序。
- 每个物品的权值都是未知的，只能通过一个比较器来进行比较。
- 比较器每次可以比较两个不同的物品，返回值只会是“大于”或“小于”中的一个。
- 这个比较器总共有e次机会返回错误的答案。
- 求比较次数尽量少的操作策略。

- 由于常数比较繁琐，这里只考虑复杂度。
- 随便选择一个基于比较的$O(n \log n)$排序算法，改用前面的$O(e)$比较方法，就能得到一个$O(en \log n)$的排序算法。

Sorting with errors

- 考虑插入排序，假如前k个物品已经排好了，将第$k+1$个物品插入到合适的位置。
- 维护一个支持二分的数据结构。
  - 退役选手只想到平衡树。
- 插入的时候进行二分，每次插入时需要进行$O(\log n)$次比较操作。
  - 先不考虑容错。
- 通过和前后两个数$O(e)$比较来检查插入位置是否正确。
- 重新插入的次数为$O(e)$，检查的次数为$O(n+e)$。
- 复杂度$O(n \log n + en + e^2)$

总结

- Ulam猜数游戏-非交互式
  - 纠错码
- Ulam猜数游戏-交互式
  - 计算搜索空间大小
  - 扩展到k叉搜索的情况
- 其它容错类问题
  - 容错的最大值问题
  - 容错的排序问题

参考文献

[1] R. Hill, J.P. Karim, Searching with lies: the Ulam problem, Discrete Mathematics 106/107 (1992) 273-283.

[2] Andrzej Pelc, Searching games with errors - fifty years of coping with liars, Theoretical Computer Science 270 (2002) 71-109.

[3] E.L. Lawler, S. Sarkissian, An algorithm for "Ulam's Game" and its application to error correcting codes, Inform. Process. Lett. 56 (1995) 89 – 93.

[4] W. Liu, Q. Zhang, Z. Nie, Searching for a counterfeit coin with two unreliable weighings, Discrete Applied Mathematics, 2005, 150(1) 160-181.

[5] B. Ravikumar, K. Ganesan, K.B. Lakshmanan, On selecting the largest element in spite of erroneous information, Proc. 4th Annu. Symp. on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, Vol. 247, Springer, Berlin, 19 – 21 February 1987, pp. 88 – 99.

[6] P.M. Long, Sorting and searching with a faulty comparison oracle, Technical Report UCSC-CRL-92 – 15, University of California at Santa Cruz, November 1992.

[7] 曹雪虹, 张宗橙, 信息论与编码(第二版), 清华大学出版社, 2009.