

# 阿里巴巴面试题精讲

---

七月算法 曹鹏  
2015年7月30日

# 提纲

---

- 关于面试
- 面（笔）试特点
- 一些例题
  - 概率（例1，2）
  - 智力题（例3）
  - 数据结构与算法（例4，5，6）
  - 实际系统（例7）
- 总结



# 关于面试

---

## ☐ 各个公司有没有自己的题库

### ■ 题库里的题目来源(共享)

☐ 员工

☐ 网络

## ☐ 笔试和面试

### ■ 笔试

☐ 无交流, 写思路, 题型丰富

### ■ 面试

☐ 注重交流, 写完整代码



# 面(笔)试题特点

---

## ☐ 注重基础知识——题型多

- 操作系统：缓存、进程、线程、死锁
- 网络：TCP 协议、握手
- 数据结构与算法：排序、二分、树、实际系统
- 智力题（数学）
  - ☐ 概率
  - ☐ 组合数学



# 例1强强对话的概率

---

□ 例1 有8只球队，采用抽签的方式随机配对，组成4场比赛。假设其中有3只强队，那么出现强强对话(任意两只强队相遇)的概率为多少(概率)

□ 分析：

■ 所有队伍比赛的可能是多少？

□ 第一支队伍有7种对手选择，第二支队伍有5种对手选择，第三支队伍有3种对手选择，最后一支队伍有1种对手选择。 $7 * 5 * 3 * 1 = 105$



# 例1 续

---

- 没有强强联合的配对方法数是多少？
  - 3个强队选择5个弱队作为对手，最后剩下两个弱队自然匹配。  $P(5,3) * 1 = 60$
- 强强联合的配对方法数是
  - $105 - 60 = 45$
- 强强联合的概率
  - $45 / 105 = 3 / 7$
- 概率题目一般是利用组合数学公式计算方案数，再作除法。



## 例2 红球和蓝球

---

□ 例2 有两个包，甲包有8个红球和2个蓝球，乙包有2个红球和8个蓝球，先抛硬币决定从甲包还是乙包取球。确定后，一共取了11次球，每次取一个球并且放回，11次的结果是7次红球，4次蓝球，问选中甲包取球的概率？（概率）

■ 分析：这是一个后验概率的问题，即在已知结果的情况下，问某个事件发生的概率。解决方案主要是用贝叶斯公式



## 例2 续

---

- 设事件A为选中甲包，事件B为选中乙包，事件C为11次取球正好是7红4蓝。
- 则  $P(A) = P(B) = 0.5$
- 甲包发生事件C的概率  $P(C|A) = C(11,4) * 0.8^7 * 0.2^4$
- 乙包发生事件C的概率  $P(C|B) = C(11,4) * 0.8^4 * 0.2^7$
- 所求概率是  $P(A|C) = P(C|A) * P(A) / P(C)$   
 $= P(C|A) * P(A) / (P(A) * P(C|A) + P(B) * P(C|B))$



## 例3 数字游戏

---

□ 例3 在黑板上写下50个数字：1至50。在接下来的49轮操作中，每次做如下操作：选取两个黑板上的数字 $a$ 和 $b$ ，擦去，在黑板上写 $|b-a|$ 。请问最后一次动作之后剩下的数字可能是什么？为什么？（智力题）

■ 分析：寻找不变量 + 构造

□ 什么不变？

■ 剩余奇数个数的奇偶性不变



## 例3 续

---

- 如果选中是两个奇数，则结果是偶数，奇数个数减少2
- 如果选中是两个偶数，则结果是偶数，奇数个数不变
- 如果选中是一个奇数，一个偶数，则结果是奇数，奇数个数不变
- 因此，奇数个数的奇偶性没有变化。1-50中有25个奇数，所以最后剩余的那个数一定是奇数！
- 我们只证明了剩余的是奇数，显然剩余数的范围在1-50之间，那么能否证明所有的奇数都可以出现？



## 例3 续

---

- 能得到所有的奇数么？构造！
  - 想得到1，拿出(1,2)剩余(3,4) (5,6)...(49,50)配对，24对答案是1，自动消掉。
  - 想得到3，拿出(1,4)剩余(2,3),(5,6)(7,8)...(49,50)配对
  - 想得到5，拿出(1,6)剩余(2,3)(4,5),(7,8)...(49,50)配对
  - ...
  - 想得到49，拿出(1,50)剩余(2,3)(4,5)...(47,48)配对
- 所以答案是可以得到1-49的全部奇数。



## 例4 元素最大间距

---

□ 例4 有无序的实数列 $V[N]$ , 求里面大小**相邻**的实数的差的最大值, 要求线性空间和线性时间。(数据结构与算法)

■ 分析:

- Leetcode 164是本题的整数版本
- 排序是一个思路, 有没有别的办法?
- 假设最大的是 $\max$ , 最小的是 $\min$ , 并且假设 $\max \neq \min$
- 桶排序思想, 我们把区间 $[\max, \min]$ 分成 $(n+1)$ 个桶(子区间)



## 例4 续

---

- 每个桶的宽度是  $(\max - \min) / (n + 1)$
- 每个桶是左闭右开区间, 最后一个桶是双闭区间
- 最小值在第一个桶里, 最大值在最后一个桶里
- $n$ 个数放入  $(n+1)$ 个桶, 至少有一个空桶
- 同一个桶内的数差距不超过桶的宽度
- 每个桶只有最大最小值有意义



# 例4 续2

## □ 整数版本代码

```
class Solution {
public:
    long long mul(long long x, long long y) {
        return x * y;
    }
    int maximumGap(vector<int> &num) {
        int n = num.size();
        if (n < 2) {
            return 0;
        }
        int mini = num[0], maxi = num[0];
        for (int i = 1; i < n; ++i) {
            mini = min(mini, num[i]);
            maxi = max(maxi, num[i]);
        }
        if (maxi == mini) {
            return 0;
        }
        // (n + 1) delta = (maxi - mini) / (n + 1)

        vector<bool> empty(n + 1, true);
        vector<int> pmax(n + 1);
        vector<int> pmin(n + 1);
        };
```

```
for (int i = 0; i < n; ++i) {
    int ind = mul(num[i] - mini, n + 1) / (maxi - mini);
    if (ind > n) {
        ind = n;
    }
    if (empty[ind]) {
        empty[ind] = false;
        pmax[ind] = pmin[ind] = num[i];
    }
    else {
        pmax[ind] = max(pmax[ind], num[i]);
        pmin[ind] = min(pmin[ind], num[i]);
    }
}
int last = -1, answer = 0;
for (int i = 0; i <= n; ++i) {
    if (!empty[i]) {
        if (last < 0) {
            last = pmax[i];
        }
        else {
            answer = max(answer, pmin[i] - last);
            last = pmax[i];
        }
    }
}
return answer;
```



# 例5 随机采样

---

- 例5 有一个函数`int getNum()`，每运行一次可以从一个数组`V[N]`里面取出一个数，`N`未知，当数取完的时候，函数返回`NULL`。现在要求写一个函数`int get()`，这个函数运行一次可以从`V[N]`里随机取出一个数，而这个数必须是符合 $1/N$ 平均分布的，也就是说`V[N]`里面任意一个数都有 $1/N$ 的机会被取出，要求空间复杂度为 $O(1)$ 。（数据结构与算法）
- 分析：数据是“流式”的，即我们只能通过不断`get`获取下一个元素，而不允许（或者没必要）存储下全部元素。
  - 水库（蓄水池）采样的经典应用



## 例5 续

---

- 算法：不断get对于第i个元素，以 $1/i$ 的概率选择它。
- 证明：对前n个元素，选择每个元素的概率是 $1/n$ 
  - 无论第i个元素之前如何，选择第i个元素的概率是 $1/i$
  - 后面第m个元素不被选择的概率是 $(m-1)/m$
  - 最终能保留第i个元素的概率
    - $(1/i) * (i/(i+1)) * ((i+1)/(i+2)) * \dots * ((n-1)/n) = 1/n$



## 例5 续2

---

### □ 示意代码（整数不能设置为NULL）

```
int get() {  
    int x;  
    int result;  
    for (int i = 1; (x = getNum()) != NULL; ++i) {  
        if (rand() % i == 0) result = x;  
    }  
    return result;  
}
```

### □ 可以扩展到采样k个的一般情况



## 例6 数组查找

□ 例6  $A[i]$ 是一个严格递增的**整数**数组，其中所有的数字都不相等，请设计一种算法，求出其中所有的 $A[i]=i$ 的数字并分析时间复杂度，不分析复杂度不得分。(数据结构与算法)

■ 分析：如果令 $B[i] = A[i] - i$

□ 则 $B[i + 1] - B[i] = A[i + 1] - A[i] - 1 \geq 0$

□ 说明 $B[i]$ 是单增的（不一定严格）

□ 相当于寻找 $B[i] = 0$

□ 简单二分即可，时间复杂度 $O(\log n)$



# 例7

---

- 例7 述有一大批数据，百万级别的。数据项内容是：用户ID、科目ABC各自的成绩。其中用户ID为0~1000万之间，且是连续的，可以唯一标识一条记录。科目ABC成绩均在0~100之间。有两块磁盘，空间大小均为512M，内存空间64M。
- 1). 为实现快速查询某用户ID对应的各科成绩，问磁盘文件及内存该如何组织；
- 2). 改变题目条件，ID为0~10亿之间，且不连续。问磁盘文件及内存该如何组织；
- 3). 在问题2的基础上，增加一个需求。在查询各科成绩的同时，获取该用户的排名，问磁盘文件及内存该如何组织。（实际系统）
- 分析：开放问题



## 例7 续

---

### □ 1) ID连续, 可以用数组存

- 如果不从0开始, 可以减一个偏移量, 相当于存3000000个int (3M), 需要空间大概12M, 内存够用

### □ 2) ID不连续, 需要map或者hash, 比如对某个大质数取余数。空间复杂度大概12M + hash的键的空间, 按内存要求, 至少可以有12000000个(4倍)key, 这样一共耗费60M



## 例7 续2

---

### ☐ 3) 需要对100万的数据作排序

- 取决于hash表的大小，如果内存不够，需要作外部归并排序

- ☐ 内存排小块，写入磁盘文件

- ☐ 文件不断作归并排序

- ☐ 得到rank值，还可以存入内存

### ☐ 优化思路：

- 各科成绩0-100，可以考虑压缩，不一定要存一个int (32bit)，用char就可以。
- 排序策略不一定用快排，可以计数排序



# 总结

---

□ 笔试注重基础

□ 面试注重交流

- 不要把面试当成笔试

- 给面试官积极的情绪

- 没有标准答案——开放问题

- 多提假设，简化问题

- 函数头部要自己写出

□ 无固定套路

□ 多总结、思考、归纳



# 谢谢大家

---

## ☐ 更多视频尽在：

- <http://www.julyedu.com/>

- ☐ 免费视频

- ☐ 直播课程

- ☐ 面试问答

## ☐ Contact us: 微博

- @七月算法

- @七月问答

- @曹鹏博士

