概率相关面试题精讲

七月算法　曹鹏

2015年5月7日

提纲

- 简介
- 题外话
- 面试题总体分析
- 一些例题
  - 例1 关于独立的理解
  - 例2 构造随机数发生器
  - 例3 不均匀随机数发生器构造均匀
  - 例4 随机变量的和
  - 例5 水库采样
  - 例6 随机排列产生——random_shuffle
  - 例7 带权采样问题
- 总结

简介

- 概率
  - 对“独立”事件的理解
  - 古典概率 （计数、除法）
  - 条件概率
  - 期望
  - 随机数产生和利用 （采样）*

题外话

- 随机数
  - 随机数生成并不容易
    - “随机性”和“不可预测性”
      - 固定m，自然数n％m是“均匀”的，具有一定随机性，但密码学不采用它
  - 一般假设已有一个均匀的随机数生成器
- 期望的计算
  - 一般转化为方程组
    - E(A) = E(A1) * p1 + E(A2) * P2 +…+1

面试题总体分析

- 概率（简单）
  - 概率、期望的计算： 笔试
  - 随机数
    - 产生：笔试、面试
    - 利用：采样
  - 相关算法（快排） 面试

例1 关于独立的理解

- 例1 X_1,X_2,都是二元随机变量，取值0和1的概率各一半，则X_3＝ X_1 xor X_2,它与X_1、X_2独立。
- 分析：枚举，{000,011,101,110},可见X_1＝0，1时各有一半情况X_3=0,1。反直觉？
- 关于独立：用定义P(A∩B）＝ P(A) * P(B)

例2 构造随机数发生器

- 例2 假设一个随机数发生器rand7均匀产生1到7之间的随机整数，如何构造rand10，均匀产生1-10之间的随机整数？
  - 分析：关键在于，不想要的数可以扔，要保证“等概率”。
  - 方法1（笨方法）1-7之间有4个奇数，3个偶数，我们扔掉一个奇数,比如7，这样剩余3个奇数，3个偶数产生的概率相同——我们构造了一个0-1整数的均匀产生器，用它产生4个bit，对应表示整数0..15, 保留1..10就可以了。

例2 续

- 代码

```
int genBit() {
int x;
        while ((x = rand7()) == 7)
        ;
        return x & 1; // note : (x & 1) == (x % 2)
}

int rand10() {
int x;
        do {
                x = 0;
                for (int i = 0; i < 4; ++i) {
                        x = (x << 1) | genBit();
                }
        } while ((x < 1) || (x > 10));
        return x;
}
```

例2 续2

- 方法2（聪明一点）
  - 使用“七进制”：我们把1-7减去1，变为0-6。产生一个两位的七进制数，对应0-48，我们把40-48扔掉（因为这只有9个数），其余按照个位数字分类，0-9对应我们要的1-10。

```
int rand10() {
int x;
        while ((x = (rand7() - 1) * 7 + rand7() - 1) >= 40)
        ;
        return x % 10 + 1;
}
```

例2 续3

- 关键问题
  - 保证均匀，才能扔掉。
    - rand2() + rand2() – 1并不是均匀的1-3
      - 1和3的概率是1/4, 2的概率是1/2
- 分析：一个实验成功的概率是p,则不断实验直到一次成功的期望次数是1/p
  - p * 1 + (1 – p) * (x + 1) = x
- 请计算方法1和2的期望循环次数
  - 112/15和49/20

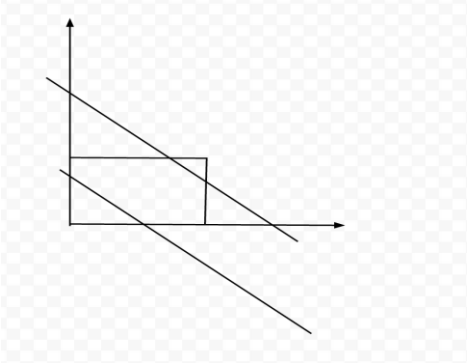
例3 不均匀随机数发生器构造均匀

- 例3 一个随机数发生器，不均匀，以概率p产生0，以(1-p)产生1，(0<p<1),构造一个均匀的随机数发生器（算法导论）
  - 分析：产生两次，(0,1)的概率与(1,0)的概率相同都是p * (1 – p)。

```
int gen() {
int x,y;
        while ((x = rand()) == (y = rand()))
        ;
        return x;
}
```

例4 随机变量的和

- 例4（笔试题）实数随机变量x和y分别在[0,a]与[0,b]之间均匀分布(a和b是给定的实数)，再给一个实数z,问x + y <= z的概率？
  - 分析x和y分布是一个矩形，求直线x + y = z下边在矩形内的面积与矩形本身的面积比。



例5 水库(Reservoir)采样

- 例5 流入若干个对象(整数)，事先不知道个数。如何随机取出k个（k小于总数）？
  - 算法：用一个数组a保存k个数 a[0..k -1]
  - 对于第i个元素(i = 1,2,…)
    - 如果i <= k： 则a[i -1]存放这个元素
    - 否则：产生随机数x = rand() % i
      - 若x < k,则用a[x]存放这个元素（扔掉之前的元素）

例5 续

- 算法优点
  - 不需要预先知道元素个数（可以一个一个流入）
- 证明，假设目前已经流入n > k个元素，
- 第i(i <= k)个元素被选中的可能性
  - 1 * k / (k + 1) * (k + 1) / (k + 2) *…*(n -1) / n = k / n
- 第i (i > k)个元素被选中的可能性
  - k / i * i / (i + 1) * (i + 1) / (i + 2) *…* (n – 1) / n = k / n

例5 续

- 思考与扩展
  - k == 1的特殊性
  - 一个若干行的大文件，随机选择一行
  - 一个不知道长度的链表，随机选择一个或者多个元素
  - 带权采样——如果每个元素权重不同，如何办？
    - 见例7

例6 随机排列产生——random_shuffle

- 例6 用数组a[0..n -1]随机产生一个全排列
  - 方法1—— 一般不符合要求
    - 产生一个[1,n!]的随机数，然后求出一个排列
  - 方法2 常规方法 请思考证明
    - 初值
      - a[i]和a[i..n – 1]交换

```
for (int i = 0; i < n; ++i) {
        a[i] = i;
}

for (int i = 0; i < n; ++i) {
        swap(a[i], a[rand() % (n - i) + i]);
}
```

例7 带权采样问题

- 例7 给定n种元素，再给定n个权值，按权值比例随机抽样一个元素。为了方便我们可以假设权值全是整数。
  - 方法1 复制若干份，每个元素复制权值那么多份，用例5的方法水库采样。
    - 例： 3个a，2个b，6个c
      - 变为aaa, bb, cccccc
  - 优点： 可以使用已有的方法
  - 缺点： 需要自己复制

例7 续

- 方法2 每个元素按照权值对应一个区间
  - 例如 3个a，2个b，6个c
  - a对应[0..2]，b对应[3..4], c对应[5..10]
  - 随机产生一个[0..10]的随机数，二分查找最后对应的元素是哪一个
  - 优点：省空间
  - 缺点： 需要二分查找
- 方法3 假设有m种元素
  - （1）先按1/m的概率随机选择一种元素
  - （2）*再产生随机数根据权值决定能否选择这种元素，如果能则选取它并结束，否则返回（1）

例7 续2

- 详细分析
  - 第（2）步的概率多大？
    - $P_{i1} = W_i / W_{tot}$ 或$P_{i2} = W_i / W_{max}$ 无关紧要（正比于W_i）
  - 实验一次成功的概率？
    - $P_{suc} = 1/m * sigma(P_i)$ 注意P_i取不同值的差别
    - 失败(谁也没选中) 的概率 $P_{lose} = 1 – P_{suc}$
    - 最终选择第i个的概率
      - $1/m * P_i + P_{lose} * 1/m * P_i + P_{lose}^2 * (1/m * P_i) +….$
      - 无穷递缩等比数列， 显然正比于P_i

例7 续3

- 关于步骤(2)
  - $P_i = a / b$
    - 老办法
      - 产生随机数 % b, 看是否小于a
      - 或者产生[1..b]的随机数看是否小于等于a
      - 或者产生[0..b – 1]的随机数看是否小于a
- 期望次数
  - $1/P_{succ} = m / sigma(P_i)$ (注意分母不一定是1)
    - $P_{i1} = W_i / W_{tot}$ 期望恰好是m
    - $P_{i2} = W_i / W_{max}$ 期望是$m * W_{max} / sigma(W)$，比m小一些
- 应用
  - 按照分数给用户推荐歌曲、产品等

总结

- 采样
- 概率算法
  - 快速排序 pivot的选择——避免最差情况
  - 在线雇佣问题（算法导论）
    - 不假设输入分布情况
    - Hash函数解决碰撞
      - 一致性hash
    - 多次尝试
      - 如一个算法有一半的可能性得到正确（最优）解——尝试30次，几乎能得到正确（最优）解

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