随机算法作业

马恒钊 陈昆仪 韩帅

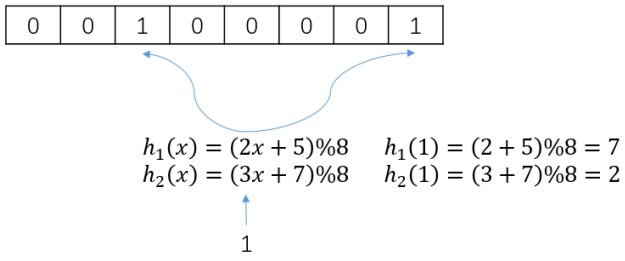
1. 证明：取三个随机正实数x, y, z，能够以x, y, z为边长围成一个钝角三角形的概率为$\frac{\pi-2}{4}$。因此也可以用这种方法来进行π的随机数值计算。

   (提示:不失一般性，设$x \leq y \leq z$，则x, y, z能够围成钝角三角形的充要条件为：$x + y \geq z, x^2 + y^2 < z^2$。然后转化为二维平面上面积的计算。)

2. Bloom Filter是计算机界广泛使用的一种数据结构，以它的简洁、高效而闻名。它的用途是来检测某个元素是否在一个集合中。构造一个Bloom Filter需要一个长为m的比特数组，以及k个哈希函数。它支持的操作只有插入和搜索两种，以下是伪代码描述，以及一个简单的例子。

Algorithm 1: Bloom Filter的基本操作

```
def Insert(p):
    if Search(p)返回"NotExist" then
        for i ∈ [1, k] do
            将比特数组中h_i(p)对应的位置1;
    else
        返回 "Fail";
def Search(p):
    if h_1(p) = 1 ∧ ··· ∧ h_k(p) = 1 then
        返回 "Exist";
    else
        返回 "NotExist";
```



Bloom Filter有一个众所周知的弱点，即假阳性。现在设已经向Bloom Filter中成功插入了n个元素，求对一个不存在于其中的元素x执行Search操作的假阳性概率，即发现对所有$1 \leq i \leq k$，$h_i(x)$都已被置1，。

(提示：假设在插入一个元素时，每个哈希函数以等概率将比特串每一位置1；假设各个哈希函数之间是相互独立的；假设各个插入到Filter中的元素之间是相互独立的。这种情况可以通过k-wise independent hashing实现，感兴趣的同学可以了解一下。)

3. 观察下面的伪代码所述的随机算法，回答问题。

Algorithm 2: RandomSelect(S,k)

Input: $S = \{s_1, \cdots, s_n\}$
Output: $min(S, k)$即S中第k小的元素
从S中随机选择一个元素s;
$S_1 = \{s_i \in S \mid s_i < s\}, S_2 = \{s_i \in S \mid s_i > s\}$;
if $|S_1| = k - 1$ **then**
    返回s;
else if $|S_1| > k - 1$ **then**
    返回$RandomSelect(S_1, k)$;
else
    返回$RandomSelect(S_2, k - |S_1|)$;

(a) 该算法属于哪一类随机算法？

(b) 设某一次调用$RandomSelect$时输入集合大小为n，证明存在常数$b < 1$，使得算法在下次递归调用时所考虑的集合大小的期望为bn。

(c) 证明：算法的期望时间复杂度为$O(n)$。

4. Alice和Bob要玩一个“心有灵犀”的游戏，请Charles来做证人。游戏的玩法是这样的：

(a) Alice和Bob各自(独立地)写下一个长为d的随机向量，分别记作a和b。

(b) Charles选定常数r，并按照以下方法生成一个长为d的随机向量：

取一个空集C;
for $i \in [1, d]$ **do**
    以$\frac{1}{2r}$的概率将i加入C中;
取一个长为d的全0向量Y;
foreach $i \in C$ **do**
    以$\frac{1}{2}$的概率将Y的第i位置1;

(c) Charles将Y向量分别出示给Alice和Bob，两人各自计算$Y \cdot a$和$Y \cdot b$。

(d) Alice和Bob交换他们所得到的点积值，若相等，则判定两人“心有灵犀”；若不等，则判定两人“心没灵犀”。

指出上述四步所描述的游戏过程是哪一类随机算法，并证明，当Alice和Bob写出的向量有h位不同时，游戏最终判定两人“心有灵犀”的概率为：$1 - \frac{1}{2}\left(1 - (1 - \frac{1}{2r})^h\right)$

5. 设有一个平面上的点集S，包含n个点。现在不断地对S进行如下操作：(1)加入一个随机点。(2)随机地删除S中一个点。假设以上操作过程中S的大小不超过$2n$，也不小于n。以下的算法可以维护一个

能够包含S中所有点的圆。

Algorithm 3: 维护圆

def `Init`**:**

随机选取S中一个点p作为圆心;
将S中其他点按照与p的距离插入优先队列H中，;
将圆的半径设置为H中的点与p的最大距离;

def `OnInsert`(p)**:**

将p插入优先队列H中;
将圆的半径设置为H中的点与p的最大距离;

def `OnDelete`(p)**:**

若p是圆心，删除p，调用`Init`函数;
否则直接将p从H中删除;

指出算法是哪一类随机算法，并分析插入(OnInsert)和删除(OnDelete)操作的期望时间复杂度。