# 习题一

Hachey

## 1.1 随机折半查找

### 算法语言描述

**Algorithm 1:** RandomBinarySearch$(S, a)$

**Input:** 长度为 $n$ 的有序数组 $S = \{s_1, s_2, \cdots, s_n \mid s_i \in \mathbb{R}\}$; 待查找的元素 $a$

**Output:** $a$ 在 $S$ 中的位置 $i$

```
begin
    low ← 1; high ← n;
    while low ≤ high do
        mid ← random(low, high); ; // 随机选取一个位置
        if S[mid] = a then
            return mid;
        else if S[mid] < a then
            low ← mid + 1;
        else
            high ← mid − 1;
        end
    end
    return -1 ; // 未找到该元素
end
```

### 时间复杂度

在平均情况下，随机折半查找的时间复杂度为 $\mathcal{O}(\log n)$。证明如下：

设 $T(n)$ 为规模为 $n$ 的问题的时间复杂度。由于每次都是随机选取一个位置，在平均情况下，每次都能将问题规模缩小一半，从而有递推式：

$$T(n) = T(n/2) + \mathcal{O}(1)$$

由主定理，得随机折半查找的平均时间复杂度为 $\mathcal{O}(\log n)$。

在最坏情况下，每次迭代选择的位置都是当前搜索空间的边界，这样就不能有效地减半搜索空间。如果这种情况持续发生，那么时间复杂度将退化为 $\mathcal{O}(n)$。但是，由于随机折半查找的随机性，这种情况发生的概率非常小。所以，随机折半查找的时间复杂度的数学期望为 $\mathcal{O}(\log n)$。

## 1.2 第 K 小元素问题

**Algorithm 2:** RandomSelect$(S, k)$

**Input:** 数组 $S = \{s_1, s_2, \cdots, s_n \mid s_i \in \mathbb{R}\}$; 整数 $k$

**Output:** $S$ 中第 $k$ 小的元素 $\min(S, k)$

**begin**
    从 $S$ 中随机选取一个元素 $s$;
    $S_1 \leftarrow \{s_i \in S \mid s_i < s\}$; $S_2 \leftarrow \{s_i \in S \mid s_i > s\}$;
    **if** $|S_1| = k - 1$ **then**
        **return** $s$;
    **else if** $|S_1| \geq k$ **then**
        **return** RandomSelect$(S_1, k)$;
    **else**
        **return** RandomSelect$(S_2, k - |S_1|)$;
    **end**
**end**

**(1)** 该算法属于随机选择算法。

**(2)** 证明：设 $\text{Rank}(S, t)$ 表示集合 $S$ 中小于 $t$ 的元素的个数。由于每次 $s$ 是随机选择的，所以 $\text{Rank}(S, s)$ 的期望 $E(\text{Rank}(S, s)) = |S|/2$。则 $S_1$ 和 $S_2$ 的大小的期望分别为

$$E(|S_1|) = E(\text{Rank}(S, s)) = |S|/2$$

$$E(|S_2|) = |S| - 1 - E(\text{Rank}(S, s)) = |S|/2 - 1$$

所以，存在常数 $b = 1/2 < 1$，使得算法递归过程中所考虑集合的大小的数学期望为 $bn$。

**(3)** 证明：设算法的时间复杂度为 $T(n)$，其中 $n$ 为集合 $S$ 的大小。由于每次递归考虑的集合大小的数学期望为 $bn$，$b < 1$，且每次计算得到 $S_1$ 和 $S_2$ 的时间复杂度为 $\mathcal{O}(n)$，所以有递推式：

$$T(n) = T(bn) + \mathcal{O}(n)$$

由主定理，得算法的时间复杂度为 $\mathcal{O}(n)$。

## 1.3 多项式乘法验证问题

### 算法语言描述

**Algorithm 3:** PolynomialProdEq($p, q, r$)

**Input:** 多项式 $p(x)$, $q(x)$, $r(x)$ 及其阶数 $m, n, l$

**Output:** $p(x) \cdot q(x) = r(x)$ 是否成立

```
begin
    k ← max(m + n, l) ;
    for i ← 1 to k do
        x ← random(ℝ);
        if p(x) · q(x) ≠ r(x) then
            return false;
        end
    end
    return true;
end
```

### 时间复杂度

显然该算法的时间复杂度为 $\mathcal{O}(\max(m+n, l))$。

### 获得正确解的概率

若 $p(x) \cdot q(x) = r(x)$，则算法返回 true 的概率为 1。

若 $p(x) \cdot q(x) \neq r(x)$，设算法返回 false 的概率为 $p$。根据 Schwarz-Zippel 引理，如果两多项式在某个域 $\mathbb{F}$ 上不相等，那么它们在随机选择的点上也不相等的概率至少为$1 - d/|\mathbb{F}|$，其中 $d$ 为两多项式的最高次项的次数，$|\mathbb{F}|$ 为域 $F$ 的大小。设 $k = \max(m+n, l)$，由于进行了 $k$ 次试验，有

$$p \geq 1 - \left(\frac{k}{|\mathbb{F}|}\right)^k$$

### 随机算法的类别

由于算法可能返回错误的结果，所以该随机算法属于蒙特卡罗算法。

## 1.4 矩阵乘法验证问题

### 算法语言描述

**Algorithm 4:** MatrixProdEq$(A, B, C)$

**Input:** 矩阵 $A_{p\times q}$, $B_{q\times r}$, $C_{p\times r}$

**Output:** $AB = C$ 是否成立

**begin**

  选取 $r \times 1$ 的随机向量 $v$，使得 $v_i \in \mathbb{R}$ 且 $v_i \neq 0$;

  **if** $A(Bv) \neq Cv$ **then**

    **return** false;

  **end**

  **return** true;

**end**

### 时间复杂度

已知阶为 $p \times q$ 的矩阵 $A$ 与阶为 $q \times r$ 的矩阵 $B$ 相乘的时间复杂度为 $\mathcal{O}(pqr)$，则计算 $Bv$ 的时间复杂度为 $\mathcal{O}(qr)$，计算 $A(Bv)$ 的时间复杂度为 $\mathcal{O}(pq)$，计算 $Cv$ 的时间复杂度为 $\mathcal{O}(pr)$。所以该算法的时间复杂度为 $\mathcal{O}(pq + pr + qr)$，小于矩阵乘法的时间复杂度 $\mathcal{O}(pqr)$。

### 获得正确解的概率

若 $AB = C$，则算法返回 true 的概率为 1。

若 $AB \neq C$，设算法返回 false 的概率为 $p$，则 $p$ 为所选的 $v$ 落入 $AB - C$ 的零空间的概率。由于 $AB - C$ 的零空间的维数不大于 $r$，则 $v$ 落入 $AB - C$ 的零空间的概率至少为 $1 - 1/|\mathbb{R}|^r$。所以，获得正确解的概率至少为 $1 - 1/|\mathbb{R}|^r$。

### 随机算法的类别

由于算法可能返回错误的结果，所以该随机算法属于蒙特卡罗算法。

## 1.5 最小割问题

### 算法语言描述

**Algorithm 5:** RandomMinCut($G$)

**Input:** 一个多重无向连通图 $G = (V, E)$

**Output:** $G$ 的一个最小边割

**begin**

- 为图 G 的任意边赋予一个随机独立的正权值;
- 找出 G 的最小生成树 $T$;
- 删除 T 中权值最大的一条边得到两棵树 $T_1$ 和 $T_2$;
- 令 $T_1$ 的顶点集为 $C$，则 $T_2$ 的顶点集为 $V - C$;
- $cut \leftarrow \{uv \mid uv \in E, u \in C, v \in V - C\}$;
- **return** $cut$;

**end**

### 证明

易知，如果图的最小割只包含一条边，则此算法输出最小割的概率为 1。在最坏的情况下，图的最小割包含 $n-1$ 条边，其中 $n$ 为图的顶点数，那么算法在每一步都必须选择属于最小割的边来收缩，才能输出最小割。设 $p$ 为算法输出最小割的概率，易知

$$p \geq \frac{1}{n(n-1)}$$

所以，最小割问题的上述随机算法输出最小割的概率为 $\Omega(1/(n^2))$。

## 1.6 最大独立子集问题

### 算法语言描述

**Algorithm 6:** RandomMaxIndSet($G$)

**Input:** 一个简单连通图 $G = (V, E)$

**Output:** $I \subseteq V$ 使得 $\forall uv \in E$ 均有 $u \in I$ 和 $v \in I$ 中至多一个成立

```
begin
    为 V 中的每个顶点随机分配 {1, 2, ···, |V|} 中唯一标签，不同顶点具有不同标签;
    I ← ∅, S ← V;
    while S ≠ ∅ do
        u ← S 中标签最小的顶点;
        I ← I ∪ {v};
        从 S 中删除 u 及其所有邻接点;
    end
    return I;
end
```

**(1)** 证明：由于每次从 $S$ 中选择顶点加入 $I$ 时，都会删除 $u$ 及其所有邻接点，所以 $I$ 是 $G = (V, E)$ 的一个独立集。

**(2)** 证明：因为 $u$ 的度数为 $d_u$，所以 $u$ 有 $d_u$ 个邻接点，记为 $N(u)$。由于顶点的标签是随机分配的，所以 $u$ 在 $N(u)$ 中的标签最小的概率为 $1/d_u$。所以，$u$ 在第一轮迭代中被选中的概率为 $1/d_u$，从而 $u \in I$ 的概率为 $1/d_u$。

**(3)** 根据 Caro-Wei 定理，对于图 $G = (V, E)$，每个顶点的度数为 $d_v$，则最大独立集大小的期望 $E(\alpha(G))$ 满足

$$E(\alpha(G)) \geq \sum_{v \in V} \frac{1}{d_v + 1}$$

由 **(1)** 知给定算法一定返回一个独立集，所以算法输出的独立集大小的数学期望为

$$E(|I|) = \sum_{v \in V} \frac{1}{d_v + 1}$$

## 1.7 冒泡排序倒置数

证明：在随机排列的情况下，任意两个元素 $a_i$ 和 $a_j$ 需要交换的概率期望为 $1/2$。列表中有 $n(n-1)/2$ 个元素对，所以总的倒置对数的期望值为

$$E = \frac{1}{2} \times \frac{n(n-1)}{2} = \frac{n(n-1)}{4}$$

## 1.8 函数篡改问题

### 算法语言描述

**Algorithm 7:** RandomMod$(F, z)$

**Input:** 数组 F=$\{F(0), F(1), \cdots, F(n-1)\}$; 整数 $z$

**Output:** $F(z)$

```
begin
    x ← random(0, n − 1); y ← random(0, n − 1);
    if F(x + y) mod m = (F(x) + F(y)) mod m then
        Return F(z);
    end
    else
        return random(0, m − 1);
    end
end
```

运行算法 3 次后，应该返回 3 次运行结果中出现次数最多的值。此时算法得到正确 $F(z)$ 的概率为

$$1 - \left(1 - \frac{1}{2}\right)^3 = \frac{7}{8}$$