

1: 方便起见, 令下标从 0 开始, 对任意下标 i 有

$$\begin{aligned}
 y_{1,i} &= \sum_{j=0}^{n-1} e^{\frac{2\pi i}{n} j} x_{1,j} \\
 y_{1,j} \cdot y_{2,i-j \bmod n} &= \sum_{k_1=0}^{n-1} e^{\frac{2\pi i}{n} k_1} x_{1,k_1} \sum_{k_2=0}^{n-1} e^{\frac{2\pi i(i-j)}{n} k_2} x_{2,k_2} \\
 &= \sum_{k_1=0}^{n-1} \sum_{k_2=0}^{n-1} e^{\frac{2\pi i}{n} j k_1 + (i-j) k_2} x_{1,k_1} x_{2,k_2} \\
 (y_1 * y_2)_i &= \sum_{j=0}^{n-1} y_{1,j} \cdot y_{2,i-j \bmod n} \\
 &= \sum_{j=0}^{n-1} \sum_{k_1=0}^{n-1} \sum_{k_2=0}^{n-1} e^{\frac{2\pi i}{n} j k_1 + (i-j) k_2} x_{1,k_1} x_{2,k_2} \\
 &= \sum_{k_1=0}^{n-1} \sum_{k_2=0}^{n-1} x_{1,k_1} x_{2,k_2} e^{\frac{2\pi i}{n} i k_2} \sum_{j=0}^{n-1} e^{\frac{2\pi i}{n} (k_1 - k_2) j}
 \end{aligned}$$

注意到

$$\sum_{j=0}^{n-1} e^{\frac{2\pi i}{n} (k_1 - k_2) j} = \begin{cases} \frac{1 - e^{2\pi i (k_1 - k_2)}}{1 - e^{\frac{2\pi i}{n} (k_1 - k_2)}} = 0 & , k_1 \neq k_2 \\ n & , k_1 = k_2 \end{cases}$$

(部分同学对这个求和没有具体说明, 直接就给出了系数 n , 或者不加说明地就用 $\delta_{i,j}$ 代替) 所以

$$\begin{aligned}
 (Vx_1 * Vx_2)_i &= n \sum_{j=0}^{n-1} e^{\frac{2\pi i}{n} j} x_{1,j} x_{2,j} \\
 &= nV(x_1 \cdot x_2)_i
 \end{aligned}$$

从而 $V(x_1 \cdot x_2) = \frac{1}{n}(Vx_1 * Vx_2)$ 。

另, 记 V^H 为 V 的共轭转置, 则 V^H 也是一个 DFT 矩阵, 同时有 $VV^H = nI$ 。令 $z_1 = V^H y_1$, $z_2 = V^H y_2$, 有

$$\begin{aligned}
 V(z_1 \cdot z_2) &= V(V^H y_1 \cdot V^H y_2) \\
 &= V(V^H (y_1 * y_2)) \\
 &= n(y_1 * y_2)
 \end{aligned}$$

(其中第二步为 PPT 上的结论) 同时 $V(V^H y_1 \cdot V^H y_2) = V(V^H Vx_1 \cdot V^H Vx_2) = n^2 V(x_1 \cdot x_2)$, 则 $V(x_1 \cdot x_2) = \frac{1}{n}(Vx_1) * (Vx_2)$ 。(使用这个方法需要明确给出 V, V^H, V^{-1} 之间的关系, 同时说明 V^H 也满足 DFT 矩阵的性质)

2.1: 记 $v' \in \{0, 1\}^{n-1}$, $v_0 = (0, v')$, $v_1 = (1, v')$, x_0 是 x 前 2^{n-1} 个位置构成的序列, x_1 是 x 后 2^{n-1}

个位置构成的序列，有

$$\begin{aligned}
 \hat{x}(v_0) &= \sum_{u \in \{0,1\}^n} (-1)^{\langle v_0, u \rangle} x(u) \\
 &= \sum_{u' \in \{0,1\}^{n-1}} (-1)^{\langle v', u' \rangle} x_0(u') + (-1)^{\langle v', u' \rangle} x_1(u') \\
 &= \sum_{u' \in \{0,1\}^{n-1}} (-1)^{\langle v', u' \rangle} x_0(u') + \sum_{u' \in \{0,1\}^{n-1}} (-1)^{\langle v', u' \rangle} x_1(u') \\
 \hat{x}(v_1) &= \sum_{u \in \{0,1\}^n} (-1)^{\langle v_1, u \rangle} x(u) \\
 &= \sum_{u' \in \{0,1\}^{n-1}} (-1)^{\langle v', u' \rangle} x_0(u') - (-1)^{\langle v', u' \rangle} x_1(u') \\
 &= \sum_{u' \in \{0,1\}^{n-1}} (-1)^{\langle v', u' \rangle} x_0(u') - \sum_{u' \in \{0,1\}^{n-1}} (-1)^{\langle v', u' \rangle} x_1(u')
 \end{aligned}$$

Algorithm 1

Require: 正整数 n , 序列 $x[0, \dots, 2^n - 1]$

```

procedure HADAMARD( $n, x$ )
    if  $n = 0$  then
        return  $x$ 
    end if
     $x_0 \leftarrow$  HADAMARD( $n - 1, x[0, \dots, 2^{n-1} - 1]$ )
     $x_1 \leftarrow$  HADAMARD( $n - 1, x[2^{n-1}, \dots, 2^n - 1]$ )
     $x' \leftarrow [0] \times 2^n$ 
    for  $i \leftarrow 0$  to  $2^{n-1} - 1$  do
         $x'[i] \leftarrow x_1[i] + x_2[i]$ 
         $x'[2^{n-1} + i] \leftarrow x_1[i] - x_2[i]$ 
    end for
    return  $x'$ 
end procedure

```

令 $T(n)$ 为输入为 n 的运行时间, 有递推关系 $T(n) = 2T(n - 1) + 2^n$, $T(1) = 2$ 。则

$$\begin{aligned}
 T(n) - (n - 1)2^n &= 2T(n - 1) - (n - 2)2^n \\
 T(n) - (n - 1)2^n &= 2(T(n - 1) - (n - 2)2^{n-1}) \\
 \frac{T(n) - (n - 1)2^n}{T(n - 1) - (n - 2)2^{n-1}} &= 2 \\
 \frac{T(n) - (n - 1)2^n}{T(1)} &= 2^{n-1} \\
 T(n) &= 2^{n-1}T(1) + (n - 1)2^n = n2^n
 \end{aligned}$$

所以 $T(n) = O(n2^n)$ 。

(部分同学没有给出算法正确性的证明，少部分同学的证明和算法之间不对应，如果讨论最高位对应将序列分成前半段和后半段，如果讨论最低位则对应将序列按下标奇偶分开)

2.2: 记 Hadamard 变换为 $\hat{x} = H[x]$ ，对于任意的 $v \in \{0, 1\}^n$, $\lambda \in \mathbb{R}$, 序列 x, y , 有

$$\begin{aligned} H[\lambda x](v) &= \sum_{u \in \{0, 1\}^n} (-1)^{\langle v, u \rangle} \lambda x(u) \\ &= \lambda \sum_{u \in \{0, 1\}^n} (-1)^{\langle v, u \rangle} x(u) \\ &= \lambda H[x](v) \\ H[x + y](v) &= \sum_{u \in \{0, 1\}^n} (-1)^{\langle v, u \rangle} (x(u) + y(u)) \\ &= \sum_{u \in \{0, 1\}^n} (-1)^{\langle v, u \rangle} x(u) + \sum_{u \in \{0, 1\}^n} (-1)^{\langle v, u \rangle} y(u) \\ &= H[x](v) + H[y](v) \end{aligned}$$

则 Hadamard 变换是一个线性变换。(需要使用线性变换的定义证明，而不是因为可以写成矩阵，所以是线性变换) 设 H_n 为 $2^n \times 2^n$ 的 Hadamard 矩阵，有

$$H_n x = \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

从而 H_n 满足以下形式 (直接写成 $H_{i,j} = (-1)^{\langle i, j \rangle}$ 也没错，但这样前面 FHT 的分析就没有意义了)

$$H_n = \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix}, n \geq 2, H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

2.3: 给定两个长为 2^n 的序列 x_1, x_2 , 类似于 FFT, 有 $H[x_1] \cdot H[x_2] = H[x_1 * x_2]$, 其中 $*$ 是待定义的卷积。考察 $H[x_1] \cdot H[x_2]$, 对于任意的 $v \in \{0, 1\}^n$, 有

$$\begin{aligned} (H[x_1] \cdot H[x_2])(v) &= \left(\sum_{u_1 \in \{0, 1\}^n} (-1)^{\langle v, u_1 \rangle} x_1(u_1) \right) \left(\sum_{u_2 \in \{0, 1\}^n} (-1)^{\langle v, u_2 \rangle} x_2(u_2) \right) \\ &= \sum_{u_1, u_2 \in \{0, 1\}^n} (-1)^{\langle v, u_1 + u_2 \rangle} x_1(u_1) x_2(u_2) \end{aligned}$$

其中, $(u_1 + u_2)_i = (u_1)_i + (u_2)_i$, 注意到 $(-1)^{\langle v, u_1 + u_2 \rangle}$ 等价于 $(-1)^{\langle v, u_1 \oplus u_2 \rangle}$, 其中 \oplus 为按位异或, 上式可以被改写为

$$\begin{aligned} (H[x_1] \cdot H[x_2])(v) &= \sum_{u_1, u_2 \in \{0, 1\}^n} (-1)^{\langle v, u_1 \oplus u_2 \rangle} x_1(u_1) x_2(u_2) \\ &= \sum_{u \in \{0, 1\}^n} (-1)^{\langle v, u \rangle} \sum_{u_1 \oplus u_2 = u} x_1(u_1) x_2(u_2) \\ &= H[x_1 * x_2] \end{aligned}$$

所以 $(x_1 * x_2)(v) = \sum_{u_1 \oplus u_2 = v} x_1(u_1) x_2(u_2) = \sum_{u \in \{0, 1\}^n} x_1(u) x_2(u \oplus v)$ 。

(部分同学没有理解题目，只是给出了 $H[x * y] = H[x] \cdot H[y]$ 的表达式，而没有给出 $x * y$ 的具体形式。少部分同学没有说明 \oplus 的含义。少部分同学直接照搬 FFT 的卷积，给出了 $i + j = k \bmod N$ 的形式，并且在证明中使用了 $\langle v, u_1 \rangle + \langle v, u_2 \rangle = \langle v, u_1 + u_2 \rangle$ ，此时 v, u 也是两个向量，需要把 $+$ 定义为向量加法，但向量意义下的 \bmod 未定义)

3.1: 不妨假设插入元素为 $[n]$ ，以随机顺序插入后，元素 i 成为 BST 的根（即第一个插入）的概率为 $\frac{1}{n}$ ，此时左子树中一定有 $i - 1$ 个元素，右子树中一定有 $n - i$ 个元素，所以 $H_n = 1 + \max(H_{i-1}, H_{n-i})$ ， i 以均匀概率在 $[n]$ 中取值，有

$$\begin{aligned} \mathbb{E}[Y_n] &= \sum_{i=1}^{n-1} \frac{1}{n} \mathbb{E}[2^{1+\max(H_{i-1}, H_{n-i})}] \\ &\leq \sum_{i=1}^{n-1} \frac{2}{n} \mathbb{E}[2^{H_{i-1}} + 2^{H_{n-i}}] \\ &= \frac{4}{n} \sum_{i=0}^{n-1} \mathbb{E}[2^{H_i}] = \frac{4}{n} \sum_{i=0}^{n-1} \mathbb{E}[Y_i] \end{aligned}$$

(需要使用 $Y_n = 2^{H_n}$ 得到 $Y_n = 2^{1+\max(H_{i-1}, H_{n-i})}$ 才能得出需要的结论，而不是直接把 Y_n 当成是深度为 H_n 的树的大小，因为这样只能得到 $Y_n + 1 = 2(\max(Y_{i-1}, Y_{n-i}) + 1)$ 。在取期望时，需要说明 $1/n$ 的来源，即 i 是在 $[n]$ 中的均匀分布。同时，均匀随机排列生成的二叉树和均匀随机一个二叉树是不等价的，因为不同的排列可以生成相同的二叉树)

3.2: 假设在过程中存在常数 c 使得 $\mathbb{E}[Y_n] \leq n^c$ ，以下归纳证明 $\mathbb{E}[Y_{n+1}] \leq (n+1)^c$ 。

$$\begin{aligned} \mathbb{E}[Y_{n+1}] &\leq \frac{4}{n+1} \sum_{i=0}^n \mathbb{E}[Y_i] \leq \frac{4}{n+1} \sum_{i=1}^n i^c \\ &\leq \frac{4}{n+1} \int_1^{n+1} x^c dx \leq \frac{4}{n+1} \frac{(n+1)^{c+1}}{c+1} \\ &= \frac{4}{(c+1)} (n+1)^c \end{aligned}$$

注意到只要 $c \geq 3$ ，则 $\mathbb{E}[Y_{n+1}] \leq (n+1)^c$ 。所以 $\mathbb{E}[Y_n] = n^{O(1)}$ 。

(部分同学在归纳过程中仍然使用了渐进符号 O ，将一些常数隐藏到了 O 中，但如果这些常数随着 n 是递增的，那么可能会导致当 $n \rightarrow \infty$ 时，常数无界，不能再被忽略)

3.3: 注意到 $f(x) = \log x$ 是上凸函数，由 Jensen Inequality， $\mathbb{E}[H_n] = \mathbb{E}[\log Y_n] \leq \log \mathbb{E}[Y_n] = \log n^{O(1)} = O(1) \log n = O(\log n)$ 。

4.1: 令 $n = 2^k, k = 0, 1, \dots$ ， $G(k) = T(2^k)$ 。原递推关系等价于 $G(k) = 2G(k-1) + \frac{2^k}{k}$ 。考虑递归树展开，在第 $i = 0, 1, \dots, k-1$ 层，产生 2^i 个子问题，每个子问题的贡献为 $\frac{2^{k-i}}{k-i}$ ；第 k 层产生 2^k 个子问题，

每个子问题的贡献为 $O(1)$ 。有

$$\begin{aligned} G(k) &= O(2^k) + \sum_{i=0}^{k-1} 2^i \frac{2^{k-i}}{k-i} \\ &= O(2^k) + \sum_{i=0}^{k-1} \frac{2^k}{k-i} \\ &= O(2^k) + 2^k \sum_{i=1}^{k-1} \frac{1}{i} \\ &= O(2^k) + \Theta(2^k \ln k) \\ &= \Theta(2^k \ln k) \end{aligned}$$

其中倒数第二个等式因为 $\sum_{i=1}^{k-1} \frac{1}{i} \geq \int_1^k \frac{1}{x} dx = \ln k$, 同时 $\sum_{i=1}^{k-1} \frac{1}{i} = 1 + \sum_{i=2}^{k-1} \frac{1}{i} \leq 1 + \int_1^{k-1} \frac{1}{x} dx = 1 + \ln(k-1) - \ln 2$, 则 $\sum_{i=1}^{k-1} \frac{1}{i} = \Theta(\ln k)$ 。

所以 $T(n) = \Theta(n \ln \log n)$ 。

4.2: 由主定理可知, $T(n) = \Theta(n^{\log_2 3})$ 。

4.3: 注意到 $T(n) = \sqrt{n}T(\sqrt{n}) + n \Leftrightarrow \frac{T(n)}{n} = \frac{T(\sqrt{n})}{\sqrt{n}} + 1$ 。考虑 $n^{(1/2)^t} = c$, 其中 $c > 1$ 是一个任意常数, 则 $t = \log \frac{\log n}{\log c} = \Theta(\log \log n)$ 。

从而 $\frac{T(n)}{n} = \frac{T(c)}{c} + t = O(1) + \Theta(\log \log n) = \Theta(\log \log n)$, 所以 $T(n) = \Theta(n \log \log n)$ 。

(1 和 3 不满足主定理, 有些同学错误地使用了主定理。少数同学使用了 Akra-Bazzi 定理, 但没有叙述完整内容)

5.1: 注意到 $\sum_{i=0}^n a^i b^{n-i} = \frac{a^{n+1} - b^{n+1}}{a - b}$, 只要计算 $\frac{a^{n+1} - b^{n+1}}{a - b} \bmod p$ 。

由费马小定理, 若 p 为质数, 且 $\gcd(x, p) = 1$, 则 $x^{p-2} \equiv 1 \pmod{p}$ 。从而 x 在 $\bmod p$ 意义下有逆元 $x^{-1} \equiv x^{p-2} \pmod{p}$ 。

若 $\gcd(a - b, p) = 1$, 只要计算 $(a^{n+1} - b^{n+1})(a - b)^{p-2} \bmod p$, 其中 a^{n+1}, b^{n+1} 使用快速幂算法计算, 需要 $O(\log n)$ 。

若 $\gcd(a - b, p) \neq 1$, 由于 p 是质数, 则 $a - b$ 必须为 p 的倍数, 不妨假设 $a = k_1 p + t, b = k_2 p + t$, 其中 $k_1, k_2 \in \mathbb{Z}, t \in \{0, 1, \dots, p-1\}$ 。此时, $\sum_{i=0}^n a^i b^{n-i} \equiv \sum_{i=0}^n t^i t^{n-i} \pmod{p}$, 而 $\sum_{i=0}^n t^i t^{n-i} = nt^n$, 因此只要计算 t^n , 需要 $O(\log n)$ 。

(部分同学在讨论非素数时, 仅给出了“可以使用扩展欧几里得算法得到结果”的说法, 但并没说明 exgcd 的具体实现以及如何通过 exgcd 得到结果)

Algorithm 2

Require: 正整数 n, a, b , 模数 p , p 为质数

```

procedure CAL( $n, a, b, p$ )
    if  $a - b \bmod p \neq 0$  then
         $A \leftarrow \text{FastExp}(a, n + 1, p)$ 
         $B \leftarrow \text{FastExp}(a, n + 1, p)$ 
         $C \leftarrow \text{FastExp}(a - b, p - 2, p)$ 
        return  $A \cdot B \cdot C \bmod p$ 
    else
         $t \leftarrow a \bmod p$ 
         $T \leftarrow \text{FastExp}(t, n, p)$ 
        return  $n \cdot T \bmod p$ 
    end if
end procedure

```

5.2: 注意到, 若 n 为奇数, 有 $\sum_{i=0}^n a^i b^{n-i} = (a^{\lfloor n/2 \rfloor + 1} + b^{\lfloor n/2 \rfloor + 1}) \sum_{i=0}^{\lfloor n/2 \rfloor} a^i b^{\lfloor n/2 \rfloor - i}$; 若 n 为偶数, $\sum_{i=0}^n a^i b^{n-i} = -a^{n/2} b^{n/2} + (a^{n/2} + b^{n/2}) \sum_{i=0}^{n/2} a^i b^{n/2 - i}$ 。

Algorithm 3

Require: 正整数 n, a, b , 模数 p ,

```

procedure CALC( $n, a, b, p$ )
    if  $n = 0$  then
        return 1
    else
        return  $(a + b) \bmod p$ 
    end if
    if  $n \bmod 2 = 0$  then
         $A \leftarrow \text{FastExp}(a, \lfloor n/2 \rfloor + 1, p)$ 
         $B \leftarrow \text{FastExp}(b, \lfloor n/2 \rfloor + 1, p)$ 
         $C \leftarrow \text{Calc}(\lfloor n/2 \rfloor, a, b, p)$ 
        return  $(A + B) \cdot C \bmod p$ 
    else
         $A \leftarrow \text{FastExp}(a, n/2, p)$ 
         $B \leftarrow \text{FastExp}(b, n/2, p)$ 
         $C \leftarrow \text{Calc}(n/2, a, b, p)$ 
        return  $((A + B) \cdot C - A \cdot B) \bmod p$ 
    end if
end procedure

```

有 $T(n) = T(n/2) + O(\log n)$, 可知 $T(n) = O(\log^2 n)$ 。

注意到 Alg. 3 中, 除分治外, 额外的复杂度来源于快速幂的计算, 但每次都重新计算 a^n, b^n 是没有必要的, 可以在递归的同时计算 a^n, b^n 。

Algorithm 4

Require: 正整数 n, a, b , 模数 p ,

```

procedure CALCULATION( $n, a, b, p$ )
    if  $n = 0$  then
        return  $1, 1, 1$ 
    else
        return  $a \bmod p, b \bmod p, (a + b) \bmod p$ 
    end if
    if  $n \bmod 2 = 0$  then
         $A, B, C \leftarrow \text{Calculation}(\lfloor n/2 \rfloor, a, b, p)$ 
        return  $a \cdot A^2 \bmod p, b \cdot B^2 \bmod p, (a \cdot A + b \cdot B) \cdot C \bmod p$ 
    else
         $A, B, C \leftarrow \text{Calculation}(n/2, a, b, p)$ 
        return  $A^2 \bmod p, B^2 \bmod p, ((A + B) \cdot C - A \cdot B) \bmod p$ 
    end if
end procedure

```

注意到 Alg. 4 并不关心 p 是否为质数, 所以该算法对任意输入都为 $O(\log n)$ 。

Appendix:

以下给出 $O(\log n)$ 计算 $a^n \bmod p$ 的快速幂算法。

Algorithm 5

Require: 整数 a , 正整数 n , 模数 p

```

procedure FASTEXP( $a, n, p$ )
     $Exp \leftarrow 1$ 
     $A \leftarrow a$ 
     $index \leftarrow n$ 
    while  $index > 0$  do
        if  $index \bmod 2 = 1$  then  $Exp \leftarrow Exp \cdot A \bmod p$ 
        end if
         $A \leftarrow A^2 \bmod p$ 
         $index \leftarrow \lfloor index/2 \rfloor$ 
    end while
end procedure

```
