

2-7

$$P_i = \frac{d}{T_i} (x - n_i) = \frac{d}{T_i} (x - n_i) \cdot \frac{d}{T_i + 1} (x - n_i) = P_i(x) * P_i(n_i)$$

$P_i(x)$  和  $P_i(n_i)$  为  $\frac{d}{2}$  次多项式。

$\therefore O(d \log d)$

$$\begin{cases} T(d) = 2T(\frac{d}{2}) + O(d \log d) & d > 1 \\ T(d) = O(1) & d = 1 \end{cases}$$

递推  $T_w = O(n \log n)$

2-15

```

import numpy as np
n = int(input() )
a = np.zeros((n, n))
b = np.zeros(n)

def tournament (n):
    if n == 1:
        a[1][1] = 1
        return
    if n%2 == 1:
        tournament (n+1)
        return
    tournament (n//2)
    makecopy(n)

def makecopy(n):
    if n/2>1 or n%2 == 1:
        copyodd(n)
    else:
        copy(n)

def copy(n):
    m = n//2
  
```

```

    for i in range(1,m+1):
        for j in range(1,m+1):
            a[i][j+m] = a[i][j] + m
            a[i+m][j] = a[i][j+m]
            a[i+m][j+m] = a[i][j]

def copyodd(n):
    m = n//2
    for i in range(1,m+1):
        b[i] = m + i
        b[i+m] = b[i]
    for i in range(1,m+1):
        for j in range(1,m+2):
            if a[i][j]>m:
                a[i][j] = b[i]
                a[m+i][j] = (b[i] + m) % n
            else:
                a[m+i][j] = a[i][j] + m

    for j in range(2,m+1):
        a[i][m+j] = b[i+j-1]
        a[b[i+j-1]][m+j] = i

```

2-6

```

import math
# 字典序排列的 rank
def permrank (n,strpi):
    if n == 1:
        return 1
    else:
        substr = strpi[1:n+1]
        adjusted_substr = ''.join(
            str(int(char) - 1) if int(char) > int(strpi[0]) else char
            for char in substr)

        return permrank(n-1,adjusted_substr) + (int(strpi[0])-1) *
math.factorial(n-1)
def permnext(stri):#找到字典序的下一个排列
    arr = list(map(int,stri))
    n = len(arr)
    i = n-2
    while i >= 0 and arr[i] >= arr[i+1]:
        i -= 1
    if i == -1:

```

```

        return arr
    j = n-1
    while arr[j] <= arr[i]:
        j -= 1
    arr[i], arr[j] = arr[j], arr[i]
    arr[i+1:] = arr[n-1:i:-1]
    str2 = ''.join(map(str, arr))
    return str2
filename = 'permutations.txt'
with open(filename) as f:
    #第一行是 n
    n = int(f.readline())
    #第二行是排列的第一个数
    strpi = f.readline().strip()
    #打开文件输出排列的 rank
    with open('permutations_rank.txt', 'w') as fw:
        fw.write(str(permrnk(n, strpi)-1))
        fw.write('\n')
        fw.write(str(permnxt(strpi)))

```

2-9

```

def hanoi(n, a, b, c):
    if n == 1:
        print(f" {1} {a} {c}") # 将第 n 个圆盘从 a 移动到 c
    else:
        hanoi(n-1, a, c, b) # 将前 n-1 个圆盘从 a 移动到 b, 使用 c 作为辅助
        print(f" {n} {a} {c}") # 将第 n 个圆盘从 a 移动到 c

        a_top.pop()
        c_top.append(n)

        hanoi(n-1, b, a, c) # 将 n-1 个圆盘从 b 移动到 c, 使用 a 作为辅助

```



同济大学  
TONGJI UNIVERSITY

SHANGHAI  
PEOPLE'S REPUBLIC OF CHINA

证明:

假设圆盘个数小于  $n$ .  $\text{hanoi}(n, A, B, C)$  不违反规则 (3) 且在移动过程中 B 上最低圆盘与 A 上同奇偶  
辅助塔座与 A 不同奇偶.  
当圆盘个数为  $n$  时.

1)  $\text{hanoi}(n-1, A, CB)$

2) move  $(A, B)$

3)  $\text{hanoi}(n-1, C, B, A)$

步骤 1) 不违反规则 (3) C 上最低圆盘与 A 同奇偶 B 与 A 不同奇偶 从而  
C 与 A 不同奇偶. B 与 A 同奇偶

步骤 2) 不违反 (3) 且 B 上最低圆盘与 A 同奇偶

步骤 3) 不违反规则 (3) 且在移动过程中 B 上倒数第 2 个圆盘的奇偶与 A 上  
相同奇偶性. A 上最低圆盘不同奇偶性. 从而塔座 A 上倒数第 2 个圆盘  
奇偶与 A 上最低圆盘奇偶性. 故在 A 上最低圆盘的奇偶与 A 具有相同  
奇偶性. ~~证~~

由数学归纳法和.,  $\text{hanoi}(n, A, B, C)$  不违反规则 (3).