《程序设计方法与艺术》课程实验报告

实验名称	3.3						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

3【题面描述】题目要求比较两副扑克牌的大小,其中每副扑克牌由 5 张牌组成。每行输人包含 10 张牌即两副牌,输出哪副牌更大,或者两副牌一样大。
牌的大小定义为:
(1)Highcard:根据牌从大到小的顺序依次比较。
(2)Pair:有一个对子和 3 张其他牌,先比较对子的大小,再比较其他牌。
(3)Two pairs:有两个对子和 1 张其他牌,从大到小比较对子的大小,再比较其他牌。
(4)Three of a kind:有 3 张面值相同的牌,比较这个值。
(5)Straight:5 张牌连续,比较最大的牌。
(6)Flush:5 张牌花色相同,根据牌从小到大的顺序依次比较。
(7)Full house:有 3 张面值相同的牌和一个对子,比较 3 个值相同的牌的大小。
(8)Four ofakind;有 4 张牌面值相同,比较这个值。
(9)Straightflush:5 张牌花色相同且面值连续,比较面值最大的牌，如果两副牌种类不同,则按照以上描述,编号大的那副牌更大;如果种类相同,则按照定义判断大小。

牌的花色有 C(clubs)、D(diamonds)、H(Hearts)、S(spades)4 种,面值有 2、3、4、5、67、8、9、T、J、Q、KA 共 13 种。每张牌用面值和花色来表示

输入说明：
第一行是一个整数 n，表示需要比较的牌序列组数。之后 n 行，每行 10 个牌面描述，第一个字符表示面值，第二个表示花色。黑方在前，白方在后。

输出说明：
针对每组牌，输出白方还是黑方牌面大，白方牌面大输出“White wins.”，黑方牌面大输出“Black wins.”。

二、解题思路和复杂度分析

我们需要为每手牌计算一个“分数”，包含：

主要牌型等级（9~1）

比较用的关键牌值（可能有多个，用于 tie-break）

步骤：

1. 解析输入，将牌转换为内部表示（面值用 2~14 表示）
2. 对每手牌按面值排序
3. 判断牌型（从最高到最低判断）
4. 为每手牌生成一个 (rank, tie_break_values) 的元组
5. 比较两手牌的元组（先 rank，再 tie_break）

牌型判断逻辑

Straight flush：同花且连续

Four of a kind：有 4 张相同面值

Full house：3 张相同 + 2 张相同

Flush：同花

Straight：连续

Three of a kind：3 张相同
Two pairs：两个对子
Pair：一个对子
High card：其他

三、解题过程（流程图/伪代码）

```
function cardValue(c):
    if c in '2'..'9': return int(c)
    if c == 'T': return 10
    if c == 'J': return 11
    if c == 'Q': return 12
    if c == 'K': return 13
    if c == 'A': return 14

function evaluateHand(hand):
    sort hand by value
    check flush, straight
    handle A-5 straight special case

    count frequency of each value
    sort by frequency then value

    if straight flush: return [9, high_card] + kickers
    if four of a kind: return [8, quad_value, kicker]
    if full house: return [7, triple_value, pair_value]
    if flush: return [6] + sorted values desc
    if straight: return [5, high_card]
    if three of a kind: return [4, triple_value] + kickers desc
    if two pairs: return [3, high_pair, low_pair, kicker]
    if one pair: return [2, pair_value] + kickers desc
    return [1] + sorted values desc

main:
    read n
    for each group:
        read black[5], white[5]
        blackScore = evaluateHand(black)
        whiteScore = evaluateHand(white)
        compare blackScore and whiteScore lexicographically
        output result
```

四、实验结果及相关测试样例

《程序设计方法与艺术》课程实验报告

实验名称	4.1						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

1.【题面描述】给出两个整数 A 和 B，可以重新排列 A 得到新的数字 C （不能有前导 0）。求在小于等于 B 的情况下，C 的最大值是多少。如果不存在输出-1。

输入说明：

第一行包含两个整数 A 和 B。1 <= A, B < 10^9。

输出说明：

输出符合条件情况下 C 的最大值。

输入样例 1：

1234 3456

输出样例 1：

3421

二、解题思路和复杂度分析

将 A 的数字排序（降序），得到最大可能数字

如果这个最大数字 ≤B，直接输出

否则，我们需要构造一个 ≤B 的最大数字

构造方法：

1. 从最高位开始，逐位确定
2. 对于当前位，尝试从大到小选择可用数字
3. 如果选择的数字 =B 的当前位，则后续位可能受限
4. 如果选择的数字 <B 的当前位，则后续位可以任意排列（选最大可能）

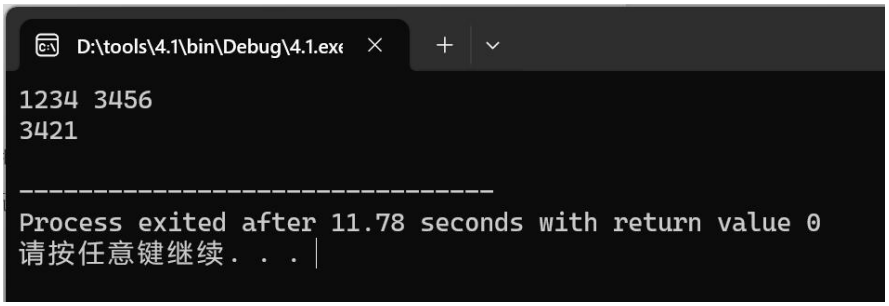
最坏情况：O(n!)，但 n ≤ 10（因为 A,B < 10^9，最多 9 位数字）实际通过剪枝很快

三、解题过程（流程图/伪代码）

```
read A, B
if length(A) > length(B):
    print -1
    return

sort A digits ascending
max_num = -1

do:
    if A[0] != '0':
```

```
            num = int(A)
            if num <= B and num > max_num:
                max_num = num
    while next_permutation(A)

    print max_num
```

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	4.2						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

2.【题面描述】小明今天的心情特别好，准备去买一些蛋糕犒劳一下自己，蛋糕店的蛋糕是排列成一行的，每个蛋糕有不同的高度，小明不想太过于麻烦，所以买的蛋糕一定要相邻，但是为了不突兀，这些蛋糕中的最高的与第二高的差值应该尽可能小。

输入说明：

正整数 n, k, 表示蛋糕店蛋糕的数目与大翔子想要买的蛋糕数(1≤k≤n≤1 000 000)接下来是 n 个正整数, 表示第 i 个蛋糕的高度，高度为大于 0 小于 10 000 000 的正整数。

输出说明：输出一个正整数，表示最小的这些蛋糕中的最高的与第二高的差值。

输入样例：

6 3

1 2 3 5 8 5

输出样例：

1

二、解题思路和复杂度分析

1. 对蛋糕按高度排序
2. 对于排序后的每个相邻对 (a[i], a[i+1])，找到包含这两个蛋糕的长度为 k 的窗口
3. 该窗口的最大值 ≥ a[i+1]，第二大值 ≤ a[i]
4. 所以差值至少是 a[i+1] - a[i]
5. 因此答案就是排序后相邻蛋糕高度差的最小值

排序：O(n log n)

计算相邻差值：O(n)

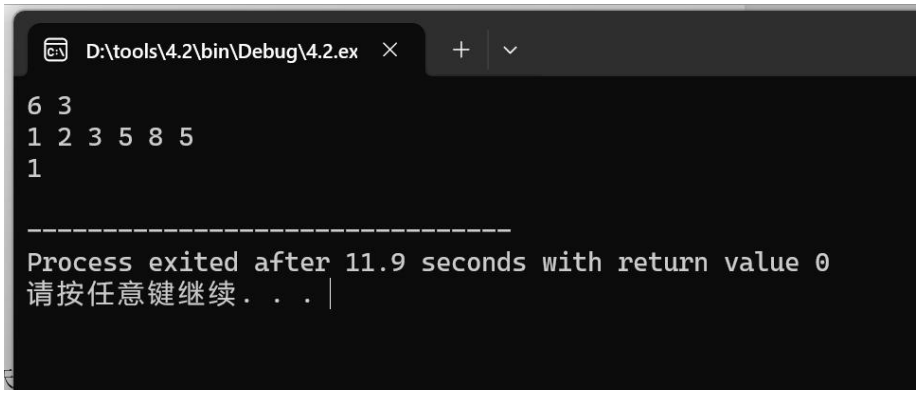
总复杂度：O(n log n)

三、解题过程（流程图/伪代码）

```
read n, k
read heights[0..n-1]

window = empty multiset
for i from 0 to k-1:
    window.insert(heights[i])

min_diff = INF
for i from k to n:
```

```
        max1 = last element of window
        max2 = second last element of window
        diff = max1 - max2
        min_diff = min(min_diff, diff)

        if i < n:
            remove heights[i-k] from window
            insert heights[i] into window

    print min_diff
```

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	4.3						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

3.【题面描述】给定一个正整数数列和正整数 p，设这个数列中的最大值是 M，最小值是 m，如果 M<=m×p,则称这个数列是“完美数列”。现在给定参数 p 和一些正整数，请编程从中选择尽可能多的数已构成一个“完美序列”。

输入说明：输入第一行给出两个正整数 N 和 p，其中 N（N<10^5）是输入正整数的个数，p(p<10^9)是给定的参数。第二行给出 N 个正整数，每个数不超过 10^9。

输出说明：在一行中输出最多可以选择的数，以便用他们组成一个“完美序列”。

输入样例：

10 8

2 3 20 4 5 1 6 7 8 9

输出样例：

8

二、解题思路和复杂度分析

令 i 和 j 分别指向序列的第一个元素和最后一个元素，而后根据 a[i]+a[j]与 M 的大小进行下列三种选择，使 i 不断向右移动、j 不断向左移动，知道 i>=j。

(1)若 a[i]+a[j]==M,则 i 右移一位,j 左移一位;

(2)若 a[i]+a[j]>M，则 j 左移一位；

(3)若 a[i]+a[j]　<M,则 i 右移一位；

时间复杂度：O(n log n)（主要由排序决定）

空间复杂度：O(n)

三、解题过程（流程图/伪代码）

开始

输入 n, p

初始化数组 a[0..n-1]

对于 i 从 0 到 n-1:

　　输入 a[i]

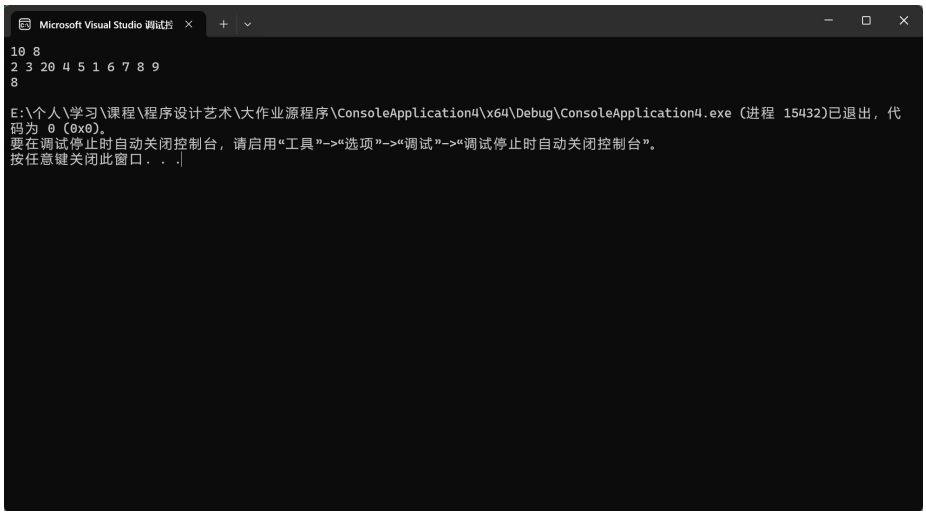
对数组 a 进行升序排序

count = 0

i = 0, j = 0

```
    当 i < n 且 j < n:
        当 j < n 且 a[j] <= a[i] * p:
            count = max(count, j - i + 1)
            j = j + 1
        i = i + 1

    输出 count
结束
```

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	5.1						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

1 【题面描述】厨师准备给小朋友们制作 m 道菜，每道菜均使用 k 克原材料。为此，厨师购入了 n 种原材料，原材料从 1 到 n 编号，第 i 种原材料的质量为 di 克。n 种原材料的**质量之和恰好为 m×km×k 克**，其中 di 与 k 都是**正整数**。

制作菜品时，一种原材料可以被用于多道菜，但为了让菜品的味道更纯粹，厨师打算每道菜**至多使用 2 种**原材料。现在请你判断是否存在一种满足要求的制作方案。更具体地，方案应满足下列要求：

- 共做出 m 道菜。
- 每道菜至多使用 2 种原材料。
- 每道菜恰好使用 k 克原材料。
- 每道菜使用的每种原材料的质量都为正整数克。
- n 种原材料都被恰好用完。

若存在满足要求的制作方案，你还应该给出一种具体的制作方案。

输入格式

本题单个测试点包含多组测试数据。

第一行一个整数 T 表示数据组数。对于每组数据：

- 第一行三个正整数 n,m,k 分别表示原材料种数、需要制作的菜品道数、每道菜品需使用的原材料的质量。
- 第二行 n 个整数，第 i 个整数表示第 i 种原材料的质量 di。

输出格式

对于每组测试数据：

- **若不存在满足要求的制作方案，则输出一行一个整数 −1**;
- 否则你需要输出 m 行，每行表示一道菜品的制作方案，根据使用的原材料种数，格式为下列两种之一：
  - 依次输出一行两个整数 i 和 x，表示该道菜使用 x 克第 i 种原材料制作。你应保证 1≤i≤n，x=k。
  - 依次输出一行四个整数 i、x、j 和 y，表示该道菜使用 x 克第 i 种原材料与 y 克第 j 种原材料制作。你应保证 1≤i,j≤n1≤i,j≤n，i≠ji =j，x+y=kx+y=k，x,y>0x,y>0。

二、解题思路和复杂度分析

这道题要求将 n 种原材料恰好用完，制作 m 道菜，每道菜使用 k 克原材料且至多使用两种原材料。解题的核心思路是：首先通过贪心策略处理 m≥n 的情况，不断用最大的原材料单独制作菜品；当 m=n-1 时采用最小最大配对法，用最小的原材料搭配最大的原材料组成菜品；对于 m<n-1 的情况，则运用 bitset 动态规划，将原材料按 d_i-k 的值分成两组，使两组和相等，再在组内进行贪心配对。这样通过分阶段处理，逐步构造出满足要求的菜品分配方案。

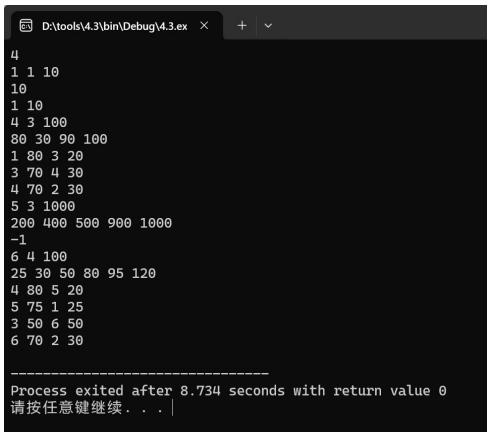
三、解题过程（流程图/伪代码）

```
T = 输入组数
for 每组数据:
    n, m, k = 输入
    d = 输入数组
    # 阶段 1: m >= n 时，用最大原材料单独做菜
    while m >= n:
        idx = d 中最大值的下标
        输出 (idx, k)
        d[idx] -= k
        m -= 1
     If m == n-1:
        # 阶段 2: 最小配最大
        while m > 0:
            mn = d 中最小值的下标
            mx = d 中最大值的下标
            x = d[mn], y = k - x
            输出 (mn, x, mx, y)
            d[mx] -= y
            d[mn] = 0
            m -= 1
    elif m == n-2:
        # 阶段 3: DP 找分组
        dp = bitset[偏移量]
        dp[偏移量] = 1
        for i=1 to n:
            shift = d[i] - k
            if shift > 0: dp |= dp << shift
            else: dp |= dp >> (-shift)
        if !dp[偏移量-k]:
            输出 -1
            continue
        # 回溯分组
        g1 = []; g2 = []
        p = 偏移量 - k
        for i=n downto 1:
            shift = d[i] - k
            if dp_prev[p]: g2.append(i)
            else: g1.append(i); p -= shift
        # 分别在 g1 和 g2 内部用阶段 2 方法配对
        for g in [g1, g2]:
            while len(g) > 1:
                mn = g 中最小值下标
                mx = g 中最大值下标
```

x = d[mn], y = k - x
输出 (mn, x, mx, y)
d[mx] -= y
d[mn] = 0
从 g 中移除 mn
if d[mx] == 0: 从 g 中移除 mx

else:
输出 -1

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	5.2						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

2【题面说明】在河上有一座独木桥，一只青蛙想沿着独木桥从河的一侧跳到另一侧。在桥上有一些石子，青蛙很讨厌踩在这些石子上。由于桥的长度和青蛙一次跳过的距离都是正整数，我们可以把独木桥上青蛙可能到达的点看成数轴上的一串整点：0，1，……，L（其中 L 是桥的长度）。坐标为 0 的点表示桥的起点，坐标为 L 的点表示桥的终点。青蛙从桥的起点开始，不停的向终点方向跳跃。一次跳跃的距离是 S 到 T 之间的任意正整数（包括 S,T）。当青蛙跳到或跳过坐标为 L 的点时，就算青蛙已经跳出了独木桥。题目给出独木桥的长度 L，青蛙跳跃的距离范围 S,T，桥上石子的位置。你的任务是确定青蛙要想过河，最少需要踩到的石子数。

输入说明：输入的第一行有一个正整数 L（1 <= L <= 10^9），表示独木桥的长度。第二行有三个正整数 S，T，M，分别表示青蛙一次跳跃的最小距离，最大距离，及桥上石子的个数，其中 1 <= S <= T <= 10，1 <= M <= 100。第三行有 M 个不同的正整数分别表示这 M 个石子在数轴上的位置（数据保证桥的起点和终点处没有石子）。所有相邻的整数之间用一个空格隔开。

输出说明：输出只包括一个整数，表示青蛙过河最少需要踩到的石子数。

二、解题思路和复杂度分析

由于桥的长度 L 很大但石子很少，且青蛙每次只能跳 S 到 T 的短距离，因此两个石子之间的大段空档可以压缩——当相邻石子间距超过 T 时，通过取模 T 并加上 T 来保持可达性不变，从而将总长度压缩到约 M*T 级别，最后在压缩后的坐标上用动态规划计算从起点到终点（及稍远处）的最少踩石子数。

排序石子：O(M log M)

动态规划：压缩后最大坐标 $\le M\times(2T) \approx 2000$，状态数 $O(M\times T)$，每个状态转移尝试 T-S+1 种跳法，所以 $\mathbf{O(M\times T^2)}$，由于 $M\le100, T\le10$，这约是 $\mathbf{10^4}$ 量级

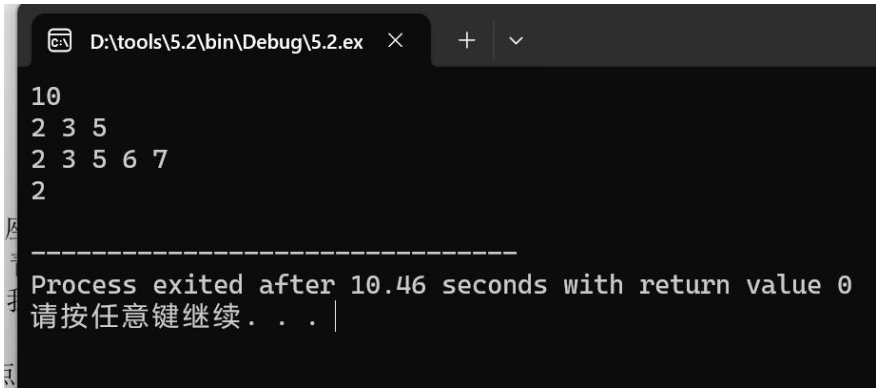
总复杂度：$\mathbf{O(M \log M + M\times T^2)}$

三、解题过程（流程图/伪代码）

```
读入 L, S, T, M
读入石子位置数组 a[1..M]
排序 a

// 路径压缩
a[0] = 0
a[M+1] = L
for i = M+1 down to 1:
    d = a[i] - a[i-1]
    if d > T:
```

```
            a[i] = a[i-1] + (d % T) + T     // 或者 a[i] = a[i-1] + T + (d % T) 更安全
        // 这样压缩后，a[M+1] 就是新终点

    L2 = a[M+1] + T   // 可能跳过终点，所以多算一些

    标记石子位置：hasStone[x] = 1 如果 x 在压缩后的 a[1..M] 中

    dp[0] = 0
    for i = 1 to L2:
        dp[i] = INF
        for j = S to T:
            if i - j >= 0:
                dp[i] = min(dp[i], dp[i-j])
        if hasStone[i]:
            dp[i] += 1

    ans = min(dp[L2], dp[L2-1], ..., dp[a[M+1]])
    输出 ans
```

四、实验结果及相关测试样例



《程序设计方法与艺术》课程实验报告

实验名称	5.3						
姓　　名	傅家琪		计算机与信息学院	班　　级	计科 24-4	学　　号	2024210858
实验日期	2025.10.23		指导教师	石雷		成　　绩	

一、实验目的和要求

3.【题面描述】虽然小欧长大了，但她还是很喜欢找点游戏自娱自乐。有一天，她在纸上写了一串数字：1,1,2,5,4。接着她擦掉了一个 1，结果发现剩下 1,2,4 都在自己所在的位置上，即 1 在第 1 位，2 在第 2 位，4 在第 4 位。她希望擦掉某些数后，剩下的数列中在自己位置上的数尽量多。她发现这个游戏很好玩，

于是开始乐此不疲地玩起来……不过她不能确定最多能有多少个数在自己的位置上，所以找到你，请你帮忙计算一下！

输入说明：

第一行为一个数 n，表示数列的长度。

接下来一行为 n 个用空格隔开的正整数，第 i 行表示数 Ai。

输出说明：

一行一个整数，表示擦掉某些数后，最后剩下的数列中最多能有多少个数在自己的位置上，即 Ai=i 最多能有多少。

二、解题思路和复杂度分析

设 dp[i][j]表示考虑原数组前 i 个数，子序列长度为 j 时，得到的最大匹配数。

转移：

不选第 i*i* 个数：dp[i][j]=dp[i−1][j]$dp[i][j]=dp[i-1][j]$

选第 i 个数作为子序列的第 j 个数：

那么要求 A[i]=j 才能形成匹配，此时

dp[i][j]=max(dp[i][j],dp[i−1][j−1]+1)

如果 A[i]≠j，则选了之后匹配数不增加：

$dp[i][j]=\max(dp[i][j],dp[i-1][j-1])$

初始：$dp[0][0]=0$，其余为 -∞。

结束：max$j$=0..n　$dp[n][j]$。

复杂度 O(n2)$O(n2)$，

三、解题过程（流程图/伪代码）

```
读入 n
读入 A[1..n]

dp[0..n][0..n] 初始化为 -∞
dp[0][0] = 0

for i = 1 to n:
    for j = 0 to n:
        # 不选 A[i]
        dp[i][j] = dp[i-1][j]
        # 选 A[i] 作为子序列的第 j 个元素
        if j >= 1:
            if A[i] == j:
                dp[i][j] = max(dp[i][j], dp[i-1][j-1] + 1)
            else:
                dp[i][j] = max(dp[i][j], dp[i-1][j-1])

ans = 0
for j = 0 to n:
    ans = max(ans, dp[n][j])

输出 ans
```

四、实验结果及相关测试样例