

# 《算法设计与分析》课程作业二

尹达恒

2020/10/27

## 1 钢条切割

### 1.1 问题描述

#### Description

给定一根长度为  $n$  ( $n \leq 10000$ ) 的钢条以及一张价格表, 请计算这根钢条能卖出的最大总收益。价格表表示为  $(l_i, p_i), 1 \leq i \leq k$ 。不在价格表中的钢条可卖出价格为 0。

#### Input

第一行输入  $m$  ( $m \leq 10$ ) 表示有  $M$  组数据。每组数据第一行输入两个 `int` 型整数  $n$  和  $k$ , 分别表示钢条长度以及价格表中不同价格数量。接下来一行输入  $k$  个价格的表示  $(l_i, p_i)$ , 均为整数,  $l_i$  可能大于  $n$ 。

#### Output

输出  $m$  行整数, 第  $i$  行表示第  $i$  组数据的最大总收益。

#### Sample Input

```
2
27 3
35 41 61 49 73 74
94 2
21 55 88 64
```

#### Sample Output

```
0
220
```

### 1.2 算法思路

令  $S_L = \{L_i | i \in [1, N]\}$  表示长  $L$  的钢条在价格表  $P = \{(l, p_l) | l \in \mathbb{N}, p \in \mathbb{R}\}$  下的最优切割方案, 其价格为  $P(S)$ , 显然最优切割方案有如下性质:

1. 最优子结构:

$$(\forall S \subseteq S_L) S = S_{\sum_{l \in S} l}$$

2. 重叠子问题:

$$P(S_L) = \max\{P(S_{L-l}) + p_l \mid (l, p_l) \in P\}$$

因此可以采用查表法, 求解长  $L$  的钢条的最优切割方案时, 使用数组存储  $P(S_i)$ , 从  $i = 0$  开始依次计算  $P(S_L) = \max\{P(S_{i-l}) + p_l \mid (l, p_l) \in P\}$  直到  $i = L$  即为要求的  $P(S_L)$ 。

### 1.3 算法伪代码

见算法 1。

---

#### Algorithm 1: 钢条切割算法伪代码

---

```

1 Function SteelCut( $L, P$ ) begin
    Input: 钢条长度  $L \in \mathbb{N}_+$ 、价格表  $P = \{(l, p_l) \mid l \in \mathbb{N}, p \in \mathbb{R}\}$ 
    Output: 最佳切割方案价格  $P(S_L)$ 
2   for  $i \in \{1, 2, 3, \dots, L\}$  do
3        $P_i = 0$ ;
4       for  $(l, p_l) \in P$  do
5           if  $l \leq i$  then
6                $P_i = \max(P_i, l + P_{i-l})$ ;
7           else
8               break;
9           end
10      end
11      记下  $P_i$ ;
12  end
13  return  $P_L$ 
14 end

```

---

## 2 最长公共子序列

### 2.1 问题描述

#### Description

给定两个字符串  $A$  和  $B$ , 请计算这两个字符串的最长公共子序列长度。

**Input**

第一行输入  $M(M \leq 10)$  表示有  $M$  组数据。每组数据输入两行字符串，字符串的长度不长于 500。

**Output**

输出  $M$  行正整数，第  $i$  行表示第  $i$  组数据的最长公共子序列长度。

**Sample Input**

```
2
abcdefg
cemg
abcdefgh
ceaaegh
```

**Sample Output**

```
3
4
```

**2.2 算法思路**

由于题干并没有要求找任意第  $k$  大的数，因此本题可以直接使用线性查找，时间复杂度为  $O(n)$ 。

**2.3 算法伪代码**

见算法 2。

## 3 最低票价

**3.1 问题描述****Description**

在一个火车旅行很受欢迎的国度，你提前一年计划了一些火车旅行。在接下来的一年里，你要旅行的日子将以一个名为 `days` 的数组给出。每一项是一个从 1 到 365 的整数。

火车票有三种不同的销售方式：

---

**Algorithm 2:** 找第二大数算法伪代码

---

```

1 Function FindNumber(S, n) begin
    Input: 数组 S、数组长度 n
    Output: 数组 S 中第二大的数
2    $R = \begin{cases} \{S_0, S_1\} (S_0 < S_1) \\ \{S_1, S_0\} (S_0 > S_1) \end{cases};$ 
3   for  $i \in \{2, 3, \dots, n\}$  do
4       if  $S_i < R_0$  then
5            $R_1 = R_0;$ 
6            $R_0 = S_i;$ 
7       end
8       else
9            $R_1 = S_i;$ 
10      end
11  end
12  return  $R_1;$ 
13 end

```

---

- 一张为期一天的通行证售价为 `costs[0]` 美元；
- 一张为期七天的通行证售价为 `costs[1]` 美元；
- 一张为期三十天的通行证售价为 `costs[2]` 美元。

通行证允许数天无限制的旅行。例如，如果我们在第 2 天获得一张为期 7 天的通行证，那么我们可以连着旅行 7 天：第 2 天、第 3 天、第 4 天、第 5 天、第 6 天、第 7 天和第 8 天。

返回你想要完成在给定的列表 `days` 中列出的每一天的旅行所需要的最低消费。

**Input**

第一行输入 `nums` 表示有 `nums` 组测试

对每组测试用例

第一行输入 `m`

第二行输入具有 `m` 个元素的 `days` 数组, `days[i]` 表示你将在 `days[i]` 这天旅行

第三行输入具有 3 个元素的 `costs` 数组，具体释义见 Description

**Output**

对每组测试数据，输出你想要完成在给定的 `days` 数组中列出的每一天的旅行所需要的最低消费。

**Sample Input**

```
2
6
1 4 6 7 8 20
2 7 15
12
1 2 3 4 5 6 7 8 9 10 30 31
2 7 15
```

**Sample Output**

```
11
17
```

**提示**

```
1 ≤ days.length ≤ 365
1 ≤ days[i] ≤ 365
days 按顺序严格递增
costs.length ≡ 3
1 ≤ costs[i] ≤ 1000
```

**3.2 算法思路**

设要查找的两个正序数组分别为  $\mathbb{M}$  和  $\mathbb{N}$ ，其组合成的正序数组为  $\mathbb{U}$ 。  
满足：

$$\begin{aligned}
 &(\forall M_i \in \mathbb{M})(M_{i+1} \geq M_i) \wedge (\forall N_i \in \mathbb{N})(N_{i+1} \geq N_i) \wedge \\
 &\mathbb{U} = \{U_i | U_i \in \mathbb{M} \cup \mathbb{N}\} \wedge (\forall U_i \in \mathbb{U})(U_{i+1} \geq U_i) \wedge \\
 &k < |\mathbb{M}| \wedge k < |\mathbb{N}|
 \end{aligned}$$

则有如下定理：

$$M_k < N_k \Rightarrow M_k < U_{2k}$$

证明：反证法。

$$\begin{aligned}
& M_k < N_k \wedge M_k \geq U_{2k} \\
& \Leftrightarrow U_{2k} \leq M_k < N_k \\
& \Rightarrow U_{2k} \in \{M_i \in \mathbb{M} | 1 \leq i \leq k\} \cup \{N_i \in \mathbb{N} | 1 \leq i \leq k-1\} \\
& \Rightarrow |\{U_i \in \mathbb{U} | 1 \leq i \leq 2k\}| < |\{M_i \in \mathbb{M} | 1 \leq i \leq k\} \cup \{N_i \in \mathbb{N} | 1 \leq i \leq k-1\}| \\
& \Rightarrow 2k \leq 2k-1 \Rightarrow \text{false}
\end{aligned}$$

$\therefore M_k < N_k \rightarrow M_k < U_{2k}$  为真命题。

查找两个正序数组  $\mathbb{M}$  和  $\mathbb{N}$  的中位数的算法可以等价为一个查找其组合成的正序数组  $\mathbb{U}$  中第  $2k$  大数  $U_{2k}$  的算法：

1. 若两数组长度和为偶数，则  $2k = (|\mathbb{M}| + |\mathbb{N}|)/2$ ，查找  $U_{2k}$  和  $U_{2k+1}$  取平均值；
2. 若两数组长度和为奇数，则  $2k = (|\mathbb{M}| + |\mathbb{N}| + 1)/2$ ，查找  $U_{2k}$ 。

而根据前述定理  $M_k < N_k \Rightarrow M_k < U_{2k}$ ，若  $M_k < N_k$ ，则  $U_{2k}$  必然不在  $\{M_i \in \mathbb{M} | 1 \leq i \leq k\}$  中，因此可以直接舍弃该部分，在  $\{M_i \in \mathbb{M} | k+1 \leq i \leq |\mathbb{M}|\}$  和  $\mathbb{N}$  中查找第  $k$  大的元素即可。显然，对于  $M_k > N_k$  时类似的情况也成立。此过程可以递归进行，直到  $M_k = N_k$  或某一轮的  $\mathbb{M}$  或  $\mathbb{N}$  为空。算法的时间复杂度为  $O(\log|\mathbb{M}| + \log|\mathbb{N}|)$ 。

### 3.3 算法伪代码

见算法 3 和算法 4。

## 4 鸡蛋掉落

### 4.1 问题描述

#### Description

你将获得  $K$  个鸡蛋，并可以使用一栋从 1 到  $N$  共有  $N$  层楼的建筑。  
每个蛋的功能都是一样的，如果一个蛋碎了，你就不能再把它掉下去。  
你知道存在楼层  $F$ ，满足  $0 \leq F \leq N$  任何从高于  $F$  的楼层落下的鸡蛋都会碎，从  $F$  楼层或比它低的楼层落下的鸡蛋都不会破。

每次移动，你可以取一个鸡蛋（如果你有完整的鸡蛋）并把它从任一楼层  $X$  扔下（满足  $1 \leq X \leq N$ ）。

你的目标是确切地知道  $F$  的值是多少。

无论  $F$  的初始值如何，你确定  $F$  的值的的最小移动次数是多少？

---

**Algorithm 3:** 在两个正序数组中找第  $k$  大数
 

---

```

1 Function  $FindK(\mathbb{M}, \mathbb{N}, k)$  begin
   Input: 数组  $\mathbb{M}$  和  $\mathbb{N}$ 、整数  $k$ 
   Output: 数组  $\mathbb{M}$  和  $\mathbb{N}$  中第  $k$  大的数
2   if  $\mathbb{M} = \emptyset$  then return  $N_k$ ;
3   if  $\mathbb{N} = \emptyset$  then return  $M_k$ ;
4   if  $k = 1$  then return  $\min(M_1, N_1)$ ;
5    $d = \lfloor \frac{k}{2} \rfloor$ ;
6   if  $d > |\mathbb{M}|$  then
7     if  $M_{|\mathbb{M}|} \leq N_d$  then
8       return  $FindK(\emptyset, \mathbb{N}, k - |\mathbb{M}|)$ ;
9     else
10      return  $FindK(\mathbb{M}, \{N_i \in \mathbb{N} | i > d\}, k - d)$ ;
11    end
12  end
13  if  $d > |\mathbb{N}|$  then
14    if  $N_{|\mathbb{N}|} \leq M_d$  then
15      return  $FindK(\mathbb{M}, \emptyset, k - |\mathbb{N}|)$ ;
16    else
17      return  $FindK(\{M_i \in \mathbb{M} | i > d\}, \mathbb{N}, k - d)$ ;
18    end
19  end
20  if  $M_d = N_d$  then
21    if  $k$  为偶数 then return  $M_d$  (或  $N_d$ );
22    if  $k$  为奇数 then return  $\min(M_{d+1}, N_{d+1})$ ;
23  end
24  if  $M_d < N_d$  then
25    return  $FindK(\{M_i \in \mathbb{M} | i > d\}, \mathbb{N}, k - d)$ ;
26  else
27    return  $FindK(\mathbb{M}, \{N_i \in \mathbb{N} | i > d\}, k - d)$ ;
28  end
29 end

```

---



**Algorithm 4:** 找两个正序数组的中位数

---

```

1 Function FindMedium( $\mathbb{M}$ ,  $\mathbb{N}$ ) begin
    Input: 正序数组  $\mathbb{M}$ 、 $\mathbb{N}$ 
    Output: 数组  $\mathbb{M}$  和  $\mathbb{N}$  的中位数
2 if  $|\mathbb{M}| + |\mathbb{N}|$  为奇数 then
3     if  $\mathbb{M} = \emptyset$  then return  $N_{\frac{|\mathbb{M}|+|\mathbb{N}|+1}{2}}$ ;
4     if  $\mathbb{N} = \emptyset$  then return  $M_{\frac{|\mathbb{M}|+|\mathbb{N}|+1}{2}}$ ;
5     return FindK( $\mathbb{M}$ ,  $\mathbb{N}$ ,  $\frac{|\mathbb{M}|+|\mathbb{N}|+1}{2}$ );
6 else
7     if  $\mathbb{M} = \emptyset$  then return  $(N_{\frac{|\mathbb{M}|+|\mathbb{N}|}{2}} + N_{\frac{|\mathbb{M}|+|\mathbb{N}|}{2}+1})/2$ ;
8     if  $\mathbb{N} = \emptyset$  then return  $(M_{\frac{|\mathbb{M}|+|\mathbb{N}|}{2}} + M_{\frac{|\mathbb{M}|+|\mathbb{N}|}{2}+1})/2$ ;
9     return
         $(\text{FindK}(\mathbb{M}, \mathbb{N}, \frac{|\mathbb{M}|+|\mathbb{N}|}{2}) + \text{FindK}(\mathbb{M}, \mathbb{N}, \frac{|\mathbb{M}|+|\mathbb{N}|}{2} + 1))/2$ ;
10 end
11 end

```

---

**Input**

第一行输入 `nums` 表示有 `nums` 组测试

每组测试输入 `K`, `N`, 表示有 `K` 个鸡蛋, `N` 层楼

**Output**

对每组测试数据, 输出确定  $F$  的最小移动次数

**Sample Input**

```

3
1 2
2 6
3 14

```

**Sample Output**

```

2
3
4

```

提示

$$\begin{aligned} 1 &\leq K \leq 1000 \\ 1 &\leq N \leq 10000 \end{aligned}$$

## 4.2 算法思路

设目标值为  $x$ ，要查找的矩阵为  $A_{m \times n} = (a_{i,j})$ ，满足：

$$\begin{cases} a_{i,j} \leq a_{i,j+1} & (i \in [1, m], j \in [1, n-1]) \\ a_{i,j} \leq a_{i+1,j} & (i \in [1, m-1], j \in [1, n]) \end{cases}$$

显然有如下定理：

$$\begin{aligned} x > a_{0,n} &\Rightarrow (\forall a_{1,j}, j \in [1, n])(x > a_{0,j}) \\ x < a_{0,n} &\Rightarrow (\forall a_{i,n}, i \in [1, m])(x > a_{i,n}) \end{aligned}$$

即当目标值大于矩阵右上角值时，目标值必大于矩阵第一行的值；目标值小于矩阵右上角值时，目标值必小于矩阵第一列的值。因此，我们可以从矩阵右上角开始搜索，若目标值较大，则向下搜索；若目标值较小，则向左搜索，直到找到目标值或超出矩阵范围。此算法时间复杂度为  $O(m+n)$ 。

更进一步，对于向下或向左的搜索过程，我们可以使用二分查找，找出当前行不大于目标值的最大元素（目标值较小时）或当前列不小于目标值的最小元素（目标值较大时）。改进后算法的时间复杂度可以达到  $O(\log(m) + \log(n))$ 。

## 4.3 算法伪代码

见算法 5。

# 5 平台截图

**Algorithm 5:** 搜索二维矩阵算法伪代码

```

1 Function  $FindNumber(A_{m,n}, x)$  begin
    Input: 待查矩阵  $A_{m,n} = (a_{i,j})$ 、目标值  $x$ 
    Output:  $A_{m,n}$  中是否存在  $a_{i,j} = x$ 
2 if  $m = 0 \vee n = 0$  then return false;
3 if  $x = a_{1,n}$  then return true;
4 if  $x > a_{1,n}$  then
5     二分查找满足条件的  $k$ :  $k > 1 \wedge a_{k,n} \geq x \wedge a_{k-1,n} < x$ ;
6     return  $FindNumber(A'_{k,n} = (a_{i,j})(i \in [k, n], j \in [1, n]), x)$ ;
7 else
8     二分查找满足条件的  $k$ :  $k > 1 \wedge a_{1,k} \leq x \wedge a_{1,k+1} > x$ ;
9     return  $FindNumber(A'_{m,k} = (a_{i,j})(i \in [1, n], j \in [1, k]), x)$ ;
10 end
11 end

```

力扣 探索 题库 讨论 竞赛 企业 商店

新动态 下载App Plus会员 中 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

题目描述 评论 (2.1k) 题解 (1838) 提交记录

执行结果: **通过** 显示详情

执行用时: **60 ms**, 在所有 C++ 提交中击败了 **36.65%** 的用户

内存消耗: **87.3 MB**, 在所有 C++ 提交中击败了 **17.16%** 的用户

标题一下

与题解, 分享你的解题思路

提交时间 提交结果 运行时间 内存消耗 语言

提交时间	提交结果	运行时间	内存消耗	语言
几秒前	通过	60 ms	87.3 MB	C++
5 分钟前	执行出错	N/A	N/A	C++

```

1 class Solution {
2 public:
3     double findMedianSortedArrays(vector<int>& S1, vector<int>& S2) {
4         int N1 = S1.size(), N2 = S2.size();
5         int N = N1 + N2;
6         if (N % 2 == 1)
7         {
8             if (N1 <= 0) return double(S2[N / 2]);
9             if (N2 <= 0) return double(S1[N / 2]);
10        }
11    }
12 }

```

您上次编辑到这里, 代码已从浏览器本地的临时存储中恢复了 还原默认代码模板

测试用例 代码执行结果 测试用例 提交

已完成 执行用时: 0 ms

输入 [1,3] [2]

输出 2.00000

预期结果 2.00000

控制台中 如何创建一个测试用例?

执行代码 提交

图 1: LeetCode 4 Accepted 截图

[Change info](#) / [My status](#) / [Problem list](#) / [Online status](#) / [Logout](#)

4 Problems Solved!

User	Problem	Result	Time	Memory	Language	Submit time
yindaheng98	<b>1040</b>	Accepted	382MS	344KB	g++	2020-10-16 17:48:03
yindaheng98	<b>1005</b>	Accepted	0MS	0KB	g++	2020-10-14 22:07:30
yindaheng98	<b>1004</b>	Accepted	0MS	0KB	g++	2020-10-14 21:30:15
yindaheng98	<b>1002</b>	Accepted	0MS	0KB	g++	2020-10-15 17:08:58

图 2: 47.99.179.148 Accepted 截图