



历年 CSP 题目解析

仅为参考练习所用

作者: Ionlyn

组织: Shanxi University Algorithm Group

时间: December 30, 2021

版本: 1.0

特别声明

该书仅供内部学习使用，如果有侵权请联系作者。

信息学竞赛的发展，吸引越来越多的人加入了“卷”的行列。CCF CSP 的历年题解在网上也是随处可见，但题解质量参差不齐。很多题解只有标准答案，缺少题目分析；更有甚者无法通过答案，充满了分号大小写问题等错误。

本书的目的是为了实现以下几点：

- 提供规范的代码程序。这里的规范，既要具有程序的可读性，也要具备考场的简易性。
- 提供多样的解题思路。有些时候，网上的大佬往往一语道破问题求解的思路，但怎么想到的却往往不提。这里力求从部分分开始，逐渐深入，汇集众人智慧，逐步解决难题。
- 提供筛选的额外补充。做一道题的目的不是只做一道题，而是可以做到举一反三，但我们常常忽略这一点。

感谢 [Elegant^{La}T_EX](#) 提供如此精美的模板，希望这本书能够给大家带来帮助。

lonlyn

December 30, 2021

目录

第 1 章 CCF CSP 认证总览

待补充。

第 2 章 第 1 次认证（2014 年 03 月）

2.1 题目及设计知识点

2.2 201403-1

2.3 201403-2

2.4 201403-3

2.5 201403-4

2.6 201403-5

第 3 章 第 2 次认证（2014 年 09 月）

3.1 题目及设计知识点

3.2 201409-1

3.3 201409-2

3.4 201409-3

3.5 201409-4

3.6 201409-5

第 4 章 第 3 次认证（2014 年 12 月）

4.1 题目及设计知识点

4.2 201412-1

4.3 201412-2

4.4 201412-3

4.5 201412-4

4.6 201412-5

第 5 章 第 4 次认证（2015 年 03 月）

5.1 题目及设计知识点

5.2 201503-1

5.3 201503-2

5.4 201503-3

5.5 201503-4

5.6 201503-5

第 6 章 第 5 次认证（2015 年 09 月）

6.1 题目及设计知识点

6.2 201509-1

6.3 201509-2

6.4 201509-3

6.5 201509-4

6.6 201509-5

第 7 章 第 6 次认证（2015 年 12 月）

7.1 题目及设计知识点

7.2 201512-1

7.3 201512-2

7.4 201512-3

7.5 201512-4

7.6 201512-5

第 8 章 第 7 次认证（2016 年 04 月）

8.1 题目及设计知识点

8.2 201604-1

8.3 201604-2

8.4 201604-3

8.5 201604-4

8.6 201604-5

第 9 章 第 8 次认证（2016 年 09 月）

9.1 题目及设计知识点

9.2 201609-1

9.3 201609-2

9.4 201609-3

9.5 201609-4

9.6 201609-5

第 10 章 第 9 次认证（2016 年 12 月）

10.1 题目及设计知识点

10.2 201612-1

10.3 201612-2

10.4 201612-3

10.5 201612-4

10.6 201612-5

第 11 章 第 10 次认证（2017 年 03 月）

11.1 题目及设计知识点

11.2 201703-1

11.3 201703-2

11.4 201703-3

11.5 201703-4

11.6 201703-5

第 12 章 第 11 次认证（2017 年 09 月）

12.1 题目及设计知识点

12.2 201709-1

12.3 201709-2

12.4 201709-3

12.5 201709-4

12.6 201709-5

第 13 章 第 12 次认证（2017 年 12 月）

13.1 题目及设计知识点

13.2 201712-1

13.3 201712-2

13.4 201712-3

13.5 201712-4

13.6 201712-5

第 14 章 第 13 次认证（2018 年 03 月）

14.1 题目及设计知识点

14.2 201803-1

14.3 201803-2

14.4 201803-3

14.5 201803-4

14.6 201803-5

第 15 章 第 14 次认证（2018 年 09 月）

15.1 题目及设计知识点

15.2 201809-1

15.3 201809-2

15.4 201809-3

15.5 201809-4

15.6 201809-5

第 16 章 第 15 次认证（2018 年 12 月）

16.1 题目及设计知识点

16.2 201812-1

16.3 201812-2

16.4 201812-3

16.5 201812-4

16.6 201812-5

第 17 章 第 16 次认证（2019 年 03 月）

17.1 题目及设计知识点

17.2 201903-1

17.3 201903-2

17.4 201903-3

17.5 201903-4

17.6 201903-5

第 18 章 第 17 次认证（2019 年 09 月）

18.1 题目及设计知识点

18.2 201909-1

18.3 201909-2

18.4 201909-3

18.5 201909-4

18.6 201909-5

第 19 章 第 18 次认证（2019 年 12 月）

19.1 题目及设计知识点

19.2 201912-1

19.3 201912-2

19.4 201912-3

19.5 201912-4

19.6 201912-5

第 20 章 第 19 次认证（2020 年 06 月）

20.1 题目及设计知识点

20.2 202006-1

20.3 202006-2

20.4 202006-3

20.5 202006-4

20.6 202006-5

第 21 章 第 20 次认证（2020 年 09 月）

21.1 题目及设计知识点

21.2 202009-1

21.3 202009-2

21.4 202009-3

21.5 202009-4

21.6 202009-5

第 22 章 第 21 次认证（2020 年 12 月）

22.1 题目及设计知识点

22.2 202012-1

22.3 202012-2

22.4 202012-3

22.5 202012-4

22.6 202012-5

第 23 章 第 22 次认证（2021 年 04 月）

23.1 题目及设计知识点

23.2 202104-1

23.3 202104-2

23.4 202104-3

23.5 202104-4

23.6 202104-5

第 24 章 第 23 次认证（2021 年 09 月）

24.1 题目及设计知识点

24.2 202109-1

24.3 202109-2

24.4 202109-3

24.5 202109-4

24.6 202109-5

第 25 章 第 24 次认证（2021 年 12 月）

25.1 题目及涉及知识点

题目编号	题目名称	知识点
1	序列查询	数学
2	序列查询新解	数学
3	登机牌条码	模拟，多项式除法
4	磁盘文件操作	线段树
5	极差路径	树分治

25.2 202112-1 序列查询

题目背景

西西艾弗岛的购物中心里店铺林立，商品琳琅满目。为了帮助游客根据自己的预算快速选择心仪的商品，IT 部门决定研发一套商品检索系统，支持对任意给定的预算 x ，查询在该预算范围内 ($\leq x$) 价格最高的商品。如果没有商品符合该预算要求，便向游客推荐可以免费领取的西西艾弗岛定制纪念品。

假设购物中心里有 n 件商品，价格从低到高依次为 A_1, A_2, \dots, A_n ，则根据预算 x 检索商品的过程可以抽象为如下序列查询问题。

题目描述

$A = [A_0, A_1, A_2, \dots, A_n]$ 是一个由 $n+1$ 个 $[0, N)$ 范围内整数组成的序列，满足 $0 = A_0 < A_1 < A_2 < \dots < A_n < N$ 。（这个定义中蕴含了 n 一定小于 N 。）

基于序列 A ，对于 $[0, N)$ 范围内任意的整数 x ，查询 $f(x)$ 定义为：序列 A 中小于等于 x 的整数里最大的数的下标。具体来说有以下两种情况：

1. 存在下标 $0 \leq i < n$ 满足 $A_i \leq x < A_{i+1}$ ，此时序列 A 中从 A_0 到 A_i 均小于等于 x ，其中最大的数为 A_i ，其下标为 i ，故 $f(x) = i$ 。
2. $A_n \leq x$ ，此时序列 A 中左右的数都小于等于 x ，其中最大的数是 A_n ，故 $f(x) = n$ 。

令 $sum(A)$ 表示 $f(0)$ 到 $f(N-1)$ 的总和，即：

$$sum(A) = \sum_{i=0}^{N-1} f(i) = f(0) + f(1) + f(2) + \dots + f(N-1)$$

对于给定的序列 A ，试计算 $sum(A)$ 。

输入格式

从标准输入读入数据。

输入的第一行包含空格分隔的两个正整数 n 和 N 。

输入的第二行包含 n 个用空格分隔的整数 A_1, A_2, \dots, A_n 。

注意 A_0 固定为 0，因此输入数据中不包括 A_0 。

输出格式

输出到标准输出。

仅输出一个整数，表示 $sum(A)$ 的值。

样例

输入格式 #1:

```
3 10
2 5 8
```

输出格式 #1:

```
15
```


解释 #1:

$A = [0, 2, 5, 8]$

i	0	1	2	3	4	5	6	7	8	9
$f(i)$	0	0	1	1	1	2	2	2	3	3

如上表所示, $\text{sum}(A) = f(0) + f(1) + \dots + f(9) = 15$ 。

考虑到 $f(0) = f(1)$ 、 $f(2) = f(3) = f(4)$ 、 $f(5) = f(6) = f(7)$ 以及 $f(8) = f(9)$, 亦可通过如下算式计算 $\text{sum}(A)$:

$$\text{sum}(A) = f(0) \times 2 + f(2) \times 3 + f(5) \times 3 + f(8) \times 2$$

输入格式 #2:

```
9 10
1 2 3 4 5 6 7 8 9
```

输出格式 #2:

```
45
```

子任务

50 % 的测试数据满足 $1 \leq n \leq 200$ 且 $n \leq N \leq 1000$;

全部的测试数据满足 $1 \leq n \leq 200$ 且 $n \leq N \leq 10^7$ 。

提示

若存在区间 $[i, j]$ 满足 $f(i) = f(i+1) = \dots = f(j-1)$, 使用乘法运算 $f(i) \times (j-i)$ 代替将 $f(i)$ 到 $f(j-1)$ 逐个相加, 或可大幅提高算法效率。

25.2.1 50% 数据——模拟

25.2.1.1 思路

模拟一下这个过程, 计算出每一个 $f(i)$ 后加起来即可。

考虑针对确定的 x , 如何求解 $f(x)$ 。我们可以从小到大枚举 A 中的数, 枚举到第一个大于等于 x 的数即可。注意末尾的判断。

枚举 x 时间复杂度 $\mathbf{O}(N)$, 计算 $f(x)$ 时间复杂度 $\mathbf{O}(n)$, 整体时间复杂度 $\mathbf{O}(nN)$ 。

25.2.1.2 C++ 实现

待补充。

25.2.2 100% 数据——利用 $f(x)$ 单调性

25.2.2.1 思路

为了方便, 设 $f(n+1) = \infty$ 。

通过模拟, 可以得到一个显然的结论:

定理 25.1 ($f(x)$ 的单调性)

对于 $x, y \in [0, N]$, 若 $x \leq y$, 则 $f(x) \leq f(y)$ 。



那么, 我们可以从小到大枚举 x , 同时记录目前 $f(x)$ 的值, 设为 y , 那么 A_{y+1} 是第一个大于 x 的数。当需要计算 $f(x+1)$ 的时候, 我们从小到大依次判断 A_{y+1}, A_{y+2}, \dots 是否满足条件, 直到遇到第一个大于 $f(x+1)$ 的数 A_z , 那么 $f(x+1) = z - 1$ 。之后, 在 $f(x+1)$ 的基础上以同样的步骤求 $f(x+2)$, 直到求完所有的值。

考虑该算法的时间复杂度, 枚举 x 的复杂度是 $O(N)$, 而 A 数组中每个数对多被枚举一次, 枚举所有 x 的整体复杂度 $O(n)$, 可以得到整体复杂度 $O(N+n)$ 。

25.2.2.2 C++ 实现

```
#include <algorithm>
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;
#define ll long long
#define il inline
const int maxn = 210;
int n, N;
int a[maxn];
ll ans = 0;
int main() {
    scanf("%d%d", &n, &N);
    for (int i = 1; i <= n; ++i) {
        scanf("%d", &a[i]);
    }
    int cur = 0;
    for (int i = 0; i < N; ++i) {
        while (cur < n && a[cur + 1] <= i)
            ++cur;
        ans += cur;
    }
    printf("%lld\n", ans);
    return 0;
}
```

25.2.3 100% 数据——阶段求和**25.2.3.1 思路**

在提示中, 指出了可以将 $f(x)$ 相同的值一起计算。现在需要解决的问题是如何快速确定 $f(x)$ 值相等的区间。

通过观察和模拟可以发现, 随着 x 增大, $f(x)$ 只会在等于某个 A 数组的值时发生变化。更具体的说, 对于某个属于 A 数组的值 A_i 来说, $[A_i, A_{i+1} - 1]$ 间的 $f(x)$ 值是相同的, 这样的数共有 $A_{i+1} - A_i$ 个。

也可以以另一种方式理解：对于一个值 y ，考虑有多少 x 满足 $f(x) = y$ 。当 $x < A_y$ 时， $f(x) < y$ ，当 $x \geq A_{y+1}$ 时， $f(x) > y$ 。只有 $x \in [A_y, A_{y+1}]$ 时才能得到 $f(x) = y$ 。

得到范围后，我们就可以根据 A 数组来进行求和计算。

考虑 $f(x) = n$ 的处理：我们可以得知满足 $f(x) = n$ 的 x 共有 $N - A_n$ 个，根据上文推算，我们可以将 A_{n+1} 设置为 $A_n + (N - A_n) = N$ 即可等效替代。

时间复杂度 $O(n)$ 。

25.2.3.2 C++ 实现

```
#include <algorithm>
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;
#define ll long long
#define il inline
const int maxn = 210;
int n, N;
int a[maxn];
ll ans = 0;
int main() {
    scanf("%d%d", &n, &N);
    for (int i = 1; i <= n; ++i) {
        scanf("%d", &a[i]);
    }
    a[n + 1] = N;
    for (int i = 1; i <= n + 1; ++i) {
        // 处理区间 [A(i-1), A(i)] 的 f(x) 值的和
        ans += 1ll * (a[i] - a[i - 1]) * (i - 1);
    }
    printf("%lld\n", ans);
    return 0;
}
```

25.3 202112-2 序列查询新解

题目背景

上一题“序列查询”中说道： $A = [A_0, A_1, A_2, \dots, A_n]$ 是一个由 $n+1$ 个 $[0, N)$ 范围内整数组成的序列，满足 $0 = A_0 < A_1 < A_2 < \dots < A_n < N$ 。基于序列 A ，对于 $[0, N)$ 范围内任意的整数 x ，查询 $f(x)$ 定义为：序列 A 中小于等于 x 的整数里最大的数的下标。

对于给定的序列 A 和整数 x ，查询 $f(x)$ 是一个很经典的问题，可以使用二分搜索在 $O(\log n)$ 的时间复杂度内轻松解决。但在 IT 部门讨论如何实现这一功能时，小 P 同学提出了些新的想法。

题目描述

小 P 同学认为，如果事先知道了序列 A 中整数的分布情况，就能直接估计出其中小于等于 x 的最大整数的大致位置。接着从这一估计位置开始线性查找，锁定 $f(x)$ 。如果估计得足够准确，线性查找的时间开销可能比二分查找算法更小。

比如说，如果 A_1, A_2, \dots, A_n 均匀分布在 $(0, N)$ 的区间，那么就可以估算出：

$$f(x) \approx \frac{(n+1) \cdot x}{N}$$

为了方便计算，小 P 首先定义了比例系数 $r = \lfloor \frac{N}{n+1} \rfloor$

，其中 $\lfloor \cdot \rfloor$ 表示下取整，即 r 等于 N 除以 $n+1$ 的商。进一步地，小 P 用 $g(x) = \lfloor \frac{x}{r} \rfloor$

表示自己估算出的 $f(x)$ 的大小，这里同样使用了下取整来保证 $g(x)$ 是一个整数。

显然，对于任意的询问 $x \in [0, N)$ ， $g(x)$ 和 $f(x)$ 越接近则说明小 P 的估计越准确，后续进行线性查找的时间开销也越小。因此，小 P 用两者差的绝对值 $|g(x) - f(x)|$ 来表示处理询问 x 时的误差。

为了整体评估小 P 同学提出的方法在序列 A 上的表现，试计算：

$$error(A) = \sum_{i=0}^{N-1} |g(i) - f(i)| = |g(0) - f(0)| + \dots + |g(N-1) - f(N-1)|$$

输入格式

从标准输入读入数据。

输入的第一行包含空格分隔的两个正整数 n 和 N 。

输入的第二行包含 n 个用空格分隔的整数 A_1, A_2, \dots, A_n 。

注意 A_0 固定为 0，因此输入数据中不包括 A_0 。

输出格式

输出到标准输出。

仅输出一个整数，表示 $error(A)$ 的值。

样例

输入格式 #1:

```
3 10
2 5 8
```

输出格式 #1:

5

解释 #1:

$$A = [0, 2, 5, 8]$$

$$r = \lfloor \frac{N}{n+1} \rfloor = \lfloor \frac{10}{3+1} \rfloor = 2$$

i	0	1	2	3	4	5	6	7	8	9
$f(i)$	0	0	1	1	1	2	2	2	3	3
$g(i)$	0	0	1	1	2	2	3	3	4	4
$ g(i) - f(i) $	0	0	0	0	1	0	1	1	1	1

输入格式 #2:

```
9 10
1 2 3 4 5 6 7 8 9
```

输出格式 #2:

0

输入格式 #3:

```
2 10
1 3
```

输出格式 #3:

6

解释 #3:

$$A = [0, 1, 3]$$

$$r = \lfloor \frac{N}{n+1} \rfloor = \lfloor \frac{10}{2+1} \rfloor = 3$$

i	0	1	2	3	4	5	6	7	8	9
$f(i)$	0	1	1	2	2	2	2	2	2	2
$g(i)$	0	0	0	1	1	1	2	2	2	3
$ g(i) - f(i) $	0	1	1	1	1	1	0	0	0	1

子任务

70 % 的测试数据满足 $1 \leq n \leq 200$ 且 $n \leq N \leq 1000$;

全部的测试数据满足 $1 \leq n \leq 10^5$ 且 $n \leq N \leq 10^9$ 。

提示

需要注意，输入数据 $[A_1 \cdots A_n]$ 并不一定均匀分布在 $(0, N)$ 区间，因此总误差 $error(A)$ 可能很大。

25.4 202112-3 登机牌条码

题目背景

西西艾弗岛景色优美，游人如织。但是，由于和外界的交通只能靠渡船，交通的不便严重制约了岛上旅游业的发展。西西艾弗岛管委会经过努力，争取到了一笔投资，建设了一个通用航空机场。在三年紧锣密鼓的主体建设后，西西艾弗岛通用航空机场终于开始进行航站楼内部软硬件系统的安装和调试工程了。小 C 是机场运营公司信息部的研发工程师，最近，信息部门的一项重要任务是，研发登机牌自助打印系统。如图所示的是设计部门根据国际民航组织的行业标准设计的登机牌样张。

25.5 202112-4 磁盘文件操作

25.6 202112-5 极差路径