历年 CSP 题目解析

仅为参考练习所用

作者：lonlyn
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 ElegantLATEX 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

1 CCF CSP 认证总览 **1**

2 第 23 次认证（2021 年 9 月） **2**

3 第 24 次认证（2021 年 12 月） **3**

3.1 题目及涉及知识点 3
3.2 202112-1 序列查询 4
3.2.1 50% 数据——模拟 5
3.2.1.1 思路 5
3.2.1.2 C++ 实现 5
3.2.2 100% 数据——利用 $f(x)$ 单调性 5
3.2.2.1 思路 5
3.2.2.2 C++ 实现 6
3.2.3 100% 数据——阶段求和 6
3.2.3.1 思路 6
3.2.3.2 C++ 实现 7
3.3 202112-2 序列查询新解 8

第 1 章 CCF CSP 认证总览

待补充。

第 2 章 第 23 次认证（2021 年 9 月）

待补充。

第 3 章 第 24 次认证（2021 年 12 月）

3.1 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

3.2 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（$\le x$）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 $A_1, A_2, \cdots, A_n$，，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

$A = [A_0, A_1, A_2, \cdots, A_n]$ 是一个由 $n+1$ 个 $[0, N)$ 范围内整数组成的序列，满足 $0 = A_0 < A_1 < A_2 < \cdots < A_n < N$ 。（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 $0 \le i < n$ 满足 $A_i \le x < A_{i+1}$，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. $A_n \le x$，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：

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

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 $A_1, A_2, \cdots, A_n$ 。

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，$sum(A)=f(0)+f(1)+\cdots+f(9)=15$。

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 $sum(A)$：

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

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 $1\le n\le 200$ 且 $n\le N\le 1000$；
全部的测试数据满足 $1\le n\le 200$ 且 $n\le N\le 10^7$。

提示

若存在区间 $[i,j)$ 满足 $f(i)=f(i+1)=\cdots=f(j-1)$，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。

3.2.1 50% 数据——模拟

3.2.1.1 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 $\mathbf{O}(nN)$。

3.2.1.2 C++ 实现

待补充。

3.2.2 100% 数据——利用 $f(x)$ 单调性

3.2.2.1 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 3.1 ($f(x)$ 的单调性)

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

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 $A_{y+1}, A_{y+2}, \cdots$ 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 $\mathbf{O}(N)$，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 $\mathbf{O}(n)$，可以得到整体复杂度 $\mathbf{O}(N + n)$。

3.2.2.2 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;
}
```

3.2.3 100% 数据——阶段求和

3.2.3.1 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 $\mathbf{O}(n)$。

3.2.3.2 C++ 实现

```cpp
#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;
}
```

3.3 202112-2 序列查询新解

题目背景

上一题“序列查询”中说道：$A = [A_0, A_1, A_2, \cdots, A_n]$ 是一个由 $n+1$ 个 $[0, N)$ 范围内整数组成的序列，满足 $0 = A_0 < A_1 < A_2 < \cdots < A_n < N$ 。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 $\mathbf{O}(\log n)$ 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 $A_1, A_2, \cdots, A_n$ 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 $\lfloor \rfloor$ 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：

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

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 $A_1, A_2, \cdots, A_n$ 。

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 $1 \le n \le 200$ 且 $n \le N \le 1000$；

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

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。