2022 河南萌新联赛第（二）场：河南理工大学

河南理工大学算法协会

July 17, 2022

Contents

1 前言 **1**
1.1 难度 1

2 妙手 **1**
2.1 Solution 1
2.2 Code 1

3 斩龙 **2**
3.1 Solution 2
3.2 Code 2

4 宝石 **3**
4.1 Solution 3
4.2 Code 3

5 数对 **4**
5.1 Solution 4
5.2 Code 4

6 双星 **5**
6.1 Solution 5
6.2 Code 6

7 手办 **7**
7.1 Solution 7
7.2 Code 7

8 无限 **8**
8.1 Solution 8
8.1.1 解法 1 8
8.1.2 解法 2 8
8.1.3 解法 3 8
8.2 Code 8

9 0 和 1 **9**
9.1 Solution 9
9.2 Code 9

10 22 数 **10**
10.1 Solution 10
10.2 Code 10

11 签到 **11**
11.1 Solution 11
11.2 Code 11

12 大米 **12**
12.1 Solution 12
12.2 Code 12
12.3 补充 12

13 HPU **13**
13.1 Solution 13
13.2 Code 13

14 Thanks **14**
14.1 Problem Makers 14
14.2 Problem Testers 14
14.3 Support 14

1 前言

1.1 难度

本次的预估难度分为四个等级：签到、简单、中等、困难。

一共出题 17 道，正式比赛剩下 12 道题。并非是比赛题集中的 12 题质量最好，只是在考虑综合难度和比赛体验方面上来考虑，算是最优。

致歉：本次比赛 A 题和 H 题出现数据错误，给所有选手带来了不好的比赛体验，向各位选手致歉，我代表出题人和验题人给大家磕头了 Orz。

2 妙手

2.1 Solution

预估难度：简单

可以用 BFS 简单跑一下 100×100 的表，观察下表的数据，即可得出答案。

时间复杂度：$O(1)$

2.2 Code

出题人代码 :https://paste.ubuntu.com/p/wCh98N2xwS/

3 斩龙

3.1 Solution

预估难度：简单

模拟战斗过程：先将 $Dragon$ 全部击败才能对 $Virm$ 进行攻击；

在 $Virm$ 加入战斗后，当我们伤害能一下击败一只 $Dragon$ 时，该只 $Dragon$ 不能造成伤害，但 $Virm$ 能造成一次伤害；

最终 $Franxx$ 剩余血量应大于 0，才能胜利。（血量只有大于 0，才能发动最后一击）

时间复杂度：$O(1)$

3.2 Code

出题人代码 :https://paste.ubuntu.com/p/KymjZcZtWf/

验题人代码 :https://paste.ubuntu.com/p/v8mqpFxsbh/

4 宝石

4.1 Solution

预估难度：中等

对于三个数的乘积组合，可以先处理两个数的乘积，对于第 i 个宝石后面任意两个宝石的乘积标记一下，然后对于第三个要用到的宝石可以枚举去找到，当第 i 个宝石在后面能找到一个宝石整除且被标记过，那么答案加一，因为要多次查询乘积是否存在，可以使用 C++ 自带的 STL 中的 map 和 unordered_map 来实现，对于 unordered_map 的查询操作我们一般看作 $O(1)$，在本题中时间效率也是要高出 map。

时间复杂度：$O(n^2)$

4.2 Code

出题人代码 :https://paste.ubuntu.com/p/w9M3wgyGCf/

验题人代码 :https://paste.ubuntu.com/p/5HPvxkbBPM/

5 数对

5.1 Solution

预估难度：中等

题目要求我们找出满足 $a_l+a_{l+1}+...+a_{r-1}+a_r \leq x+y\times(r-l+1)$ 数对 (l,r) 的个数，首先我们可以把公式变形为 $(a_l-y)+(a_{l+1}-y)+...+(a_{r-1}-y)+(a_r-y) \leq x$，令 $b_i = a_i - y$，在对数组 b 进行预处理得到前缀和数组 sum，上面的公式又可以变成 $sum_r - sum_{l-1} <= x$，进而得到 $sum_r - x <= sum_{l-1}$，这样我们就可以在值域上维护一个树状数组，维护每个前面的前缀和数值的个数的和，枚举每个右端点 r，每次将答案累加上树状数组中大于等于 sum_r-x 的总个数，也就是所有满足上述条件的左边界，然后再给 sum_r 这个位置的个数加上 1。

由于值域很大很大，树状数组开不下，所以我们要先将前缀和数组离散化。

时间复杂度：$O(nlog(n))$

5.2 Code

出题人代码 :https://paste.ubuntu.com/p/gcgP9KTMCf/

验题人代码 :https://paste.ubuntu.com/p/nHb72MMrJR/

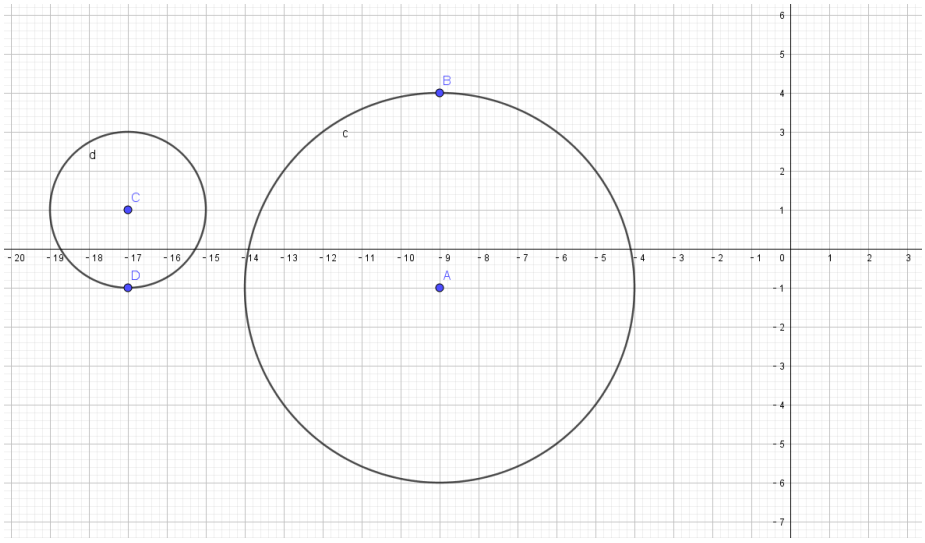
6 双星

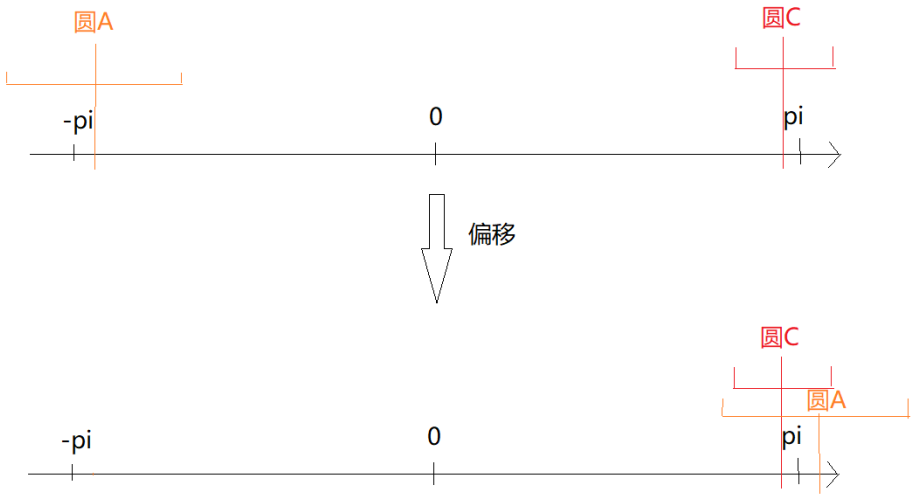
6.1 Solution

预估难度：困难

一个容易想到的思路是：我们计算出每个圆对应的极角序范围，然后判断其中一个圆的极角序是否能被另一个圆的极角序完全覆盖，被完全覆盖则不能看见，否则能看见。需要注意的是我们要调整两个圆的顺序，可以让离原点更近的圆作为 c_1 ，判断 c_2 是否被 c_1 覆盖。

但是上思路有边界问题，因为 $atan2(y,x)$ 返回的弧度值域为 $(-\pi,\pi]$ ，如果两个圆按下图放置，则会出现在原点视角是有交集但是在值域上没有交集的情况，因此需要额外做两次长度为周期的偏移，然后判断。容易知道这三次偏移最多只有一次极角序会有交集，因此若三次都没有交集即为都可以看到；否则，判断有交集的那一次 c_2 是否能被 c_1 完全覆盖，能则只能看到一个，否则能看到两个。





6.2 Code

出题人代码 :https://paste.ubuntu.com/p/QzbvhqTf7D/

7 手办

7.1 Solution

预估难度：简单

如果把 n 做质因数分解

$$n = p_1^{\alpha_1} * p_2^{\alpha_2} * p_3^{\alpha_3} \cdots p_n^{\alpha_n}$$

对于 n 的所有约数一定是

$$\prod_{i=1}^{n} p_i^x (x \leq \alpha_i)$$

又因为所求 x 必须为有理数所以对于任何一个 p_i 其指数 x 必须在乘以 $\frac{2}{3}$ 后仍为整数，所以对于任何一个 p_i 其指数 x, 必须为 3 的倍数。

出题人代码时间复杂度：约 $O(n^{\frac{1}{3}})$

验题人代码时间复杂度：$O(\sqrt{n})$

7.2 Code

出题人代码 :https://paste.ubuntu.com/p/QndfMJmZPF/

验题人代码 :https://paste.ubuntu.com/p/cz5q2TS7nb/

8 无限

8.1 Solution

预估难度：简单/中等

8.1.1 解法 1

由题意可知 x 在集合中，那么 $x\times 4$ 和 $x\times 2+1$ 也在集合中，首先将数转化为二进制，$\times 4$ 操作与 $\times 2+1$ 操作等价于在二进制数 x 的末尾加 00 与 1，即使 x 的长度 $+1$ 或 $+2$。题目要求求出集合中小于 2^p 的数，即二进制长度小于等于 p 的数。本题解法使用的是 dp，dp 方程为 $f[i]$，属性为集合中二进制长度小于等于 i 的数的个数，因为长度为 i 的数可由长度为 $i-1$ 的数末尾加 1 以及长度为 $i-2$ 的数末尾加 00 得到，所以状态转移方程为 $f[i]=f[i-1]+f[i-2]$，又初始情况下集合中只有一个 1，因此 dp 的初始状态为 $f[1]=1$。答案是集合中二进制长度小于等于 p 的个数，即 $\sum_{i=1}^{p} f[i]$。

本题之所以标了简单，是大家可以手动模拟出前几项长度为 i 的数量，因为斐波那契数列是比较好看出来的，所以属于简单/中等难度。

时间复杂度：$O(n)$

8.1.2 解法 2

因为本题的 dp 方程是线性递推式子，故可以手动求出前几项，然后用杜教 BM 算法模板把前几项放入进行求解。

8.1.3 解法 3

验题人给出了和出题人不一样的 dp 方式, 见8.2。

8.2 Code

出题人代码 1 :https://paste.ubuntu.com/p/VpMPNKdZ3d/

出题人代码 2 :https://paste.ubuntu.com/p/n4tbMJmSy7/

验题人代码 :https://paste.ubuntu.com/p/PswNw4nxBz/

9 0 和 1

9.1 Solution

预估难度：中等

两种操作可以看成是对 01 字符串进行反转。

答案只有全 0 或者全 1。我们分别对其进行讨论即可。

假设我们正在讨论的是最终全 0 的情况，那么我们要去思考对于 1 要怎么处理。首先，我们把 01 串中连续的 1 分段，观察两个连续 1 串中间的 0 串。

(1). 如果 0 串的长度为 1 ，那么只有两边两个 1 串中的 1 的数量为 2 时，最优的操作是反转两边的 1 ，这样只消耗 2 个体力。当两边 1 串的 1 的数量大于 2 ，我们可以选择反转中间的这个 0 ，然后把新合成的这个 1 串反转，花费为 3 ，比直接反转两个一串更优。

(2). 如果 0 串的长度大于 1 ，无论是两边两个 1 串的 1 的数量是 2 还是大于 2 ，我们直接去反转两个 1 串是最优的。

最终全 1 的情况也是同理。

时间复杂度：$O(n)$

9.2 Code

验题人 1 代码 :https://paste.ubuntu.com/p/R2NnmbTdJv/

10 22 数

10.1 Solution

预估难度：中等

数位 dp 枚举范围内数的每一种可能合法状态，若达到边界条件则退出并判断当前状态是否满足要求，若没有前导 0 且当前位不受限制，当前状态已记录则直接返回，up 为当前位可取的最大值，若当前位受限制则 up 取限制条件 (即给出的 n) 下当前位上的数，否则可取到 9，然后枚举当前位置上可以填充的数，搜索下一位并把答案累加。

验题大佬用两种姿势过了这道题%%%，另外一种解法是 dfs 暴力。

时间复杂度：大概是 O($n^2 \times m$),n 是给出数的位数，m 是给出数的各个数位的数字和。

10.2 Code

出题人代码 :https://paste.ubuntu.com/p/3jkjhhCKCY/

验题人代码 :https://paste.ubuntu.com/p/tvBZMZmN2f/

11 签到

11.1 Solution

预估难度：签到

再次感谢 [14.1] 群友给的 idea。

求序列中是否存在 a、b、c、d 满足 a+b+c=d，且一个数可以重复选取, 基于以上条件可以直接对式子进行等价变形为 a+b=d-c，所以直接求所有可能的 $a[i]+a[j]$ 和 $a[i]-a[j]$ 中是否存在一对相等的值即可.

有同学反应卡常，经检查数据均在题目要求范围内，为了不让一些奇怪的暴力过去，我们加强了数据，但是可能最主要的原因是牛客评测问题，我们一开始的 1e9 复杂度的代码牛客是可以过的，为了卡掉这些错解，我们出了一个小时的数据。被卡常个人感觉是牛客问题。

时间复杂度:O(n^2logn)

11.2 Code

出题人代码 :https://paste.ubuntu.com/p/tzrVDmySwQ/

验题人代码 :https://paste.ubuntu.com/p/4jpC7vkqQv/

12 大米

12.1 Solution

预估难度：困难

w_i 初始都是奇数，所有的 w_i 在施法的过程中都只会在 w_i（奇数）和 $2w_i$（偶数）之间来回转换，因此每次操作就相当于对区间 $[l,r]$ 内的整数做一次 w_i 到 $2w_i$（或 $2w_i$ 到 w_i ）的翻转。

有一道经典的问题：初始给你一个长度为 n 仅由 '0'和 '1'组成的 01 序列，然后有 m 次询问，每次询问对区间 $[l,r]$ 的 01 进行翻转，然后询问你序列里 1 的数量。我们可以用线段树，每个节点维护区间和 sum（显然 sum 就是区间内 1 的个数），父节点 p 的信息可以由左儿子 ls，右儿子 rs 得出，$sum_p = sum_{ls} + sum_{rs}$，作修改时，节点区间内 1 的数量变成了 0 的数量，$sum_p = r - l + 1 - sum_p$ 。

对于这道题，我们将奇数和偶数分开，分别维护奇数和 sum_{odd} ，偶数和 sum_{eve}，父节点的信息维护同上。修改时，区间内的奇数都乘 2，然后变成偶数，偶数都除 2，然后变成奇数，相当于 sum_{odd} 乘 2，sum_{eve} 除 2，然后再把二者交换，$sum_{odd} = \frac{sum_{eve}}{2}, sum_{eve} = 2sum_{odd}$。

用分块或者其他能够维护区间和的数据结构应该也没问题。

时间复杂度：$O(nlog(n))$

12.2 Code

出题人代码 :https://paste.ubuntu.com/p/Qcy6p2cXT5/

验题人代码 :https://paste.ubuntu.com/p/BC9TBs8TBN/

12.3 补充

这道题还有一个 hard 版本：《大米 2》，唯一的区别是：hard 版本中不保证 w_i 初始时是奇数。做法于《大米》类似。对于一些 $w_i = 2^p \times k$ ，例如 $24 = 2^3 \times 3$，这些数只有操作 $p-1$ 次后变成 $w_i = 2^0 \times k$ 或 $w_i = 2^1 \times k$ 这种 $p <= 1$ 的形式才会进入我们想要的翻转循环，而我们注意到 p 的大小是 log 级别的，我们可以维护一个标记 f := ** 区间内的所有数的 p 是否小于等于 1，也就是二进制形式的后两位不全为 0**。我们在修改的时候，对于 f 为 1 的区间直接像《大米》那样去维护，对于 f 为 0 的区间接着暴力往下去递归，直到区间长度为 1，进行修改。

每个数被暴力递归的次数不超过 log 次，因此时间复杂度为 $O(nlog^2(n))$

大米 2 题目链接：http://499d347a05.qicp.vip/problem/442/

大米 2 代码：https://paste.ubuntu.com/p/5j47q9ZwdJ/

13 HPU

13.1 Solution

预估难度：签到

降低难度后加的签到题，直接遍历找到有多少个 HPU 即可。

请注意，string 中的 s.size() 是无符号整数，应该转为 int 类型。

时间复杂度：$O(n)$

13.2 Code

出题人代码 :https://paste.ubuntu.com/p/tjFnVbnJmP/

14 Thanks

14.1 Problem Makers

河南理工大学算法协会 20 级成员、河南理工大学 19 级汪子涵、2022ccpc 河南省赛交流群“昔日”、“helloworld”两位网友。

14.2 Problem Testers

河南理工大学算法协会 20 级成员、河南工业大学孔维飒、河南工业大学李志豪、河南理工大学 20 级贾亚硕、河南理工大学 19 级牛付壮 (先后仅为验题顺序)。

14.3 Support

组织河南萌新联赛的各位教练、牛客平台。