

牛客竞赛

2024牛客寒假算法基础集训营1 题解

出题人: fried-chicken

写在前面

- 出题人难度估计：
 - 大一纯萌新，刚学完课内C，觉得好玩就报了这个营，也没参加acm组织，几乎不懂acm。约1~5题
 - 大一萌新，但参加了学校的acm俱乐部，还参加了学校组织的acm寒假集训，学了些简单的acm基础，如思维、贪心、dp、STL使用等。约3~7题
 - 大二学了一年、第二次参赛 or 大一但高中有基础。约7~11题
 - 不管什么年级但总之是高手（?）。约11~13题
- 心态调整
 - 要补题要补题要补题！
 - 虽然形式是比赛，但你的新收获反而是用“补完题之后题数 - 赛时过题数”定义的，而不是赛时过题数，甚至赛事过题越少，说明新收获越多

题目难度预估

Easy	Mid-easy	Mid	Mid-hard	Hard	?
AM	BCEG	DF	HK	J	IL

- 血签是AM，其实有点担心血签和mid-easy难度差太多了，实际确实也有些gap，但看上去gap没那么大；
- D确实是黑马，因为D乱搞其实都对，所以本来没想到会这么难；
- IL难度比较有趣，真实难度可能差不多，但因为都是榜单会影响难度的题，所以实际难度得看那个题先被做出来。实际发生的事情好像也是，L碰巧榜单上先被做出来，于是做L的人就更多了。
- 题解顺序：AMEBCGLIFHKDJ

A DFS搜索

- 笑点解析：DFS搜索其实是深度优先搜索，在这里被fried-chicken解释成了寻找DFS和dfs子序列，令人忍俊不禁；
- 以dfs子序列为例
- 解法一：
 - 设置变量p表示dfs目前已经匹配到了第几个字母，遇到下一个字母就让p++，检查p是不是3，复杂度 $O(nT)$ ；
- 解法二：
 - 枚举三个位置i,j,k，判断s[i],s[j],s[k]是否分别对应dfs，复杂度 $O(n^3T)$ ；
- 没错，数据范围 n 这么小是为了连解法二都放过去，出题人太善良

M 牛客老粉才知道的秘密

- 不知道题解咋写了
- 不如直接看AC代码 🙌

```
if (n % 6 == 0) {  
    std::cout << n / 6 << "\n";  
} else {  
    std::cout << n / 6 * 2 << "\n";  
}
```

- 首先可以注意到， n 只用考虑是或不是6的倍数两种情况
- 然后手画一下两种情况，就能发现其实答案就是上面两个式子

E 本题主要考察了贪心

- 本题题目的一个目的是误导选手写贪心，其实贪心是错误的，考虑以下样例：
- 鸡A目前3分、鸡B目前2分、鸡C目前2分，B和C还要进行两场比赛，发现这种情况A一定不会是第一名；
- 验题时大家都表示这骗不到人，但看结果确实还是能骗到小白的，嘿嘿
- 事实上本题正确做法是dfs，因为每组用例的比赛局数 m 很小，所以直接用dfs枚举每一局的结果就可以做到 $O(3^m T)$ 的复杂度，可以通过本题
- 其实也顺便教了下萌新要学会看数据范围做题，当看到数据范围是dfs可过、这个dfs又不难写的时候，就别想别的了，直接dfs启动

B 关鸡

- 这题解也不知道咋写，总之就是怎么都能做，但也都不太好做
- 首先答案一定是0、1、2、3之一，这个应该好看出来，最大为3是因为可以直接把鸡周围3格给围住
- 然后就是，选一个方法来判断了
 - 比如，可以用`set<pair<int,int>>`或`map`来真的存下所有位置，然后暴力的遍历所有位置和周围距离为1的位置；
 - 或者也可以把所有点按横坐标排序，然后遍历；
 - 等等做法都是可以的
- 最后，涉及到 $(2,0)$ 的情况比较特殊，也许需要特判
- 样例从最开始4个加到了8个，每个样例都是某个验题人wa过的，所以很善良，基本大多数人还是能过样例==AC的

C 按闹分配

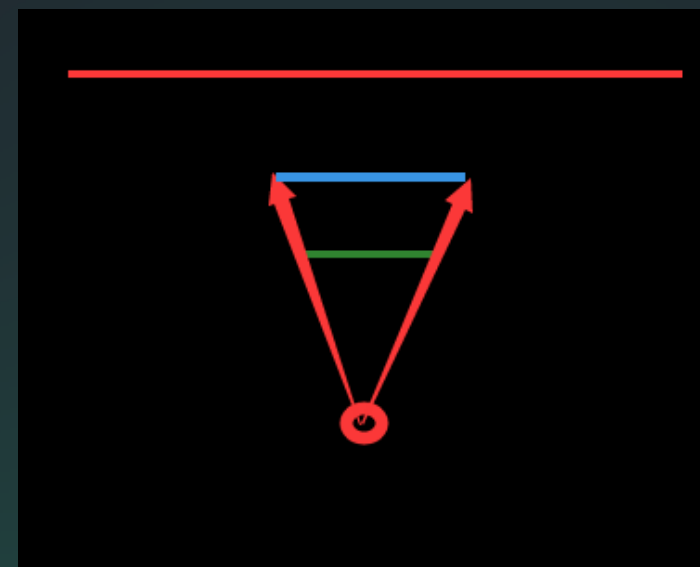
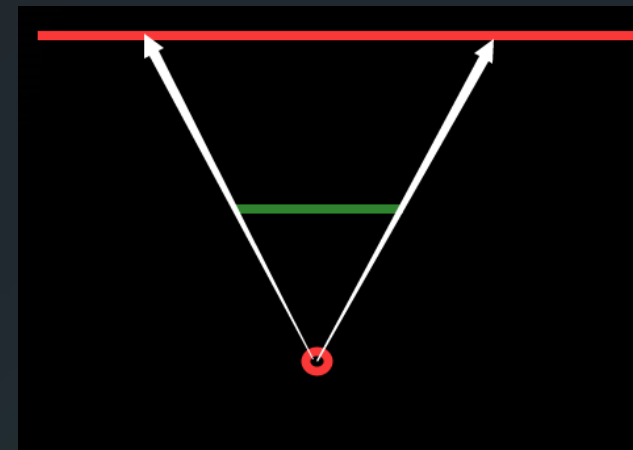
- 首先要注意到，打断一个人和插到他前面两种情况的 S_c 一样：
 - 比如三个人4 5 6，鸡是7，现在插第二个人的队，那么计算可以发现
 - 4 7 5 6 和 4 2 7 3 6的情况（下划线表示5的那个人） S_c 一样
- 然后就是，算下插到某人前面会让不满意度增大多少；
- 发现就是让之后的每个人完成任务时间都推迟 t_c ；
- 所以总不满意度增加就是 $t_c * \text{排在鸡后面的人数}$
- 现在是给定了最大不满意度 M ，所以我们可以求出排在鸡后面的人数最多是 $\lfloor \frac{M}{t_c} \rfloor$ ，之中方括号表示下取整；
- 然后再看排在鸡前面的人的时间加起来是多少就行了，这个可以用前缀和快速查询；

G why买外卖

- 首先我们将输入的所有满减券按照 a_i 由小到大排序;
- 这样的好处是, 任意一个方案用到的满减券都一定是数组的一个前缀; (比如我买了100元的, 那排序后数组满足 $a_i \leq 100$ 的前缀就是可以用的满减券)
- 接下来我们假想用前 i 张满减券, 那就要求商品原价是 a_i , 减去满减优惠 $\sum_{j \leq i} b_j$, 实际要支付 $a_i - \sum_{j \leq i} b_j$ 元;
- 因此, 若 $a_i - \sum_{j \leq i} b_j \leq m$ 则可以使用这张券, 此时令 $ans = \max(ans, m + \sum_{j \leq i} b_j)$ 即可;
(含义为, 您购买商品的原价等于手里的钱+最多能获得的满减优惠)

L 要有光

- 省流：答案即为 $3cw$
- 证明：
- 当光源高度低于 $2h$ 时，可以发现在俯视图中，绿墙对应的线段总是阴影三角形的中位线，如右图所示，此时阴影部分即图中等腰矩形面积计算可得 $3cw$ ；
- 当光源高度高于 $2h$ 时，俯视图如右图所示，此时阴影没有打到墙上，阴影部分是图中蓝色和绿色所组成的等腰梯形，由于蓝色线段小于 $2 \times$ 绿色线段长，所以此时等腰梯形面积一定小于 $3cw$ ；
- 因此，总把光源贴地放做到 $3cw$ 就是最优的



I It's bertrand paradox. Again!

- 其实做法很简单，核心思路就是
 - 1、找到两种方法会使得哪个统计量有显著区别，尝试区分这个统计量的均值；
 - 2、如果这个统计量不好直接求，您可以本地真的实现以下两种生成方式，然后生成大量数据，真的随机出这个统计量的真实值；
 - 3、比如求出法1该统计量均值m1、法2该统计量均值m2，那对于输入数据，直接看输入数据的该统计量m和m1与m2谁更近就行了。
- 统计量选法也非常多，基本都可以过；
- std使用了一种比较巧妙的不需要真编程求统计量均值的方法：
- 考虑横纵坐标不超过70的区域，看有多少点落在该区域；
- bit-noob的方法由于是生成坐标均匀的，应该有 $\frac{141 \times 141}{199 \times 199}$ 概率落在该区域；而buaa-noob落在该区域的概率应该更大，因此比较实际落在该区域的点数与 $\frac{141 \times 141}{199 \times 199} \times 10^5$ 的差是否超过一个阈值即可

```
int n,x,y,r;
double E=100000*(19881.0/39601); // bit-noob
int main(){
    cin>>n;
    int sum=0;
    rep(i,1,n){
        scanf("%d%d%d",&x,&y,&r);
        if(-70<=x&&x<=70&&-70<=y&&y<=70){
            sum++;
        }
    }
    if(abs(sum-E)>2000) puts("buaa-noob");
    else puts("bit-noob");
    return 0;
}
```

F 鸡数题！

- 这题比较无聊，是这套题里唯一一道考了个具体知识点、不会就做不出来的题；
- 相关知识：第二类斯特林数
- 假设您已经学完了斯特林数，那本题的做法是：
 - 将 n 个bit每个bit看作一件物品
 - 将 m 个数字每个看作一个集合
 - 注意到这几乎就是第二类斯特林数 $S(n, m)$ 的定义： n 个有区别球放到 m 个无区别盒子
 - 还差在哪里呢？我们发现，本题的盒子（ a_i ）并不是无区别的，但因为第二个条件要求了每个盒子里的数从小到大有序，所以对于 $S(n, m)$ 中的每一种方案，对其按 a_i 大小排序就得到了本题的一种方案，且可以证明这是一种一一映射（重点是证明没有多对一）
- 因此，本题答案就是斯特林数 $S(n, m)$
- 百度一下第二类斯特林数的容斥求法，就可以在 \log 复杂度里做出这题了

H 01背包，但是bit

- 提示：建议先考虑，每个物品的重量都只含1bit的情况，可以帮助您对这题要解决什么问题有个大致了解；
- 记所选物品重量加起来是 c ，枚举 m 里是1的某个bit，强制 c 里该位为0，则该位将 m 分成了前后两部分：
 - 对于前面的那部分位（更高位），要求所选的物品这些位必须是 m 的子集（即 m 对应位是1才能选）
 - 对于后面的那部分位（更低位），没有任何限制
- 因此，枚举 m 里每一位作为这个分界，每个物品就变成了要么能选要么不能选、彼此之间也不影响，所以把能选的都选上就好

K 牛镇公务员考试

- 也许需要的前置知识：基环内向树、排列的复合运算（置换环）
- 首先，每个 i 向 a_i 连一条有向边，表示 a_i 的答案被 i 限制住了；
- 这样得到的图是一个基环内向树森林，其形态特点可以百度学习下；
- 对于一个基环内向树，考虑怎么求答案：
 - 首先所有连到环的链可以直接无视，它们不影响答案，这是因为，这个链所连接的环上那个点确定答案后、链上每个点的答案可以由此反向传播依次得到（至于为何每个点答案唯一？这是因为该点走到环的路径上所有排列复合得到的仍是排列），因此，环上点确定方案、则链也随之确定唯一一种方案，所以可以忽略链；
 - 对于环，我们随机选个起点，暴力枚举这个点选ABCDE中哪个选项然后暴力模拟来check这个选项是否能让这个环满足条件即可；
- 因此，每个基环内向树的答案是0~5之间整数；
- 最后的答案是每个基环内向树答案的乘积；

D 数组成鸡

- 各种看似奇怪的暴力做法都是对的，核心是要注意到：询问的M范围不大，所以数组稍微长一点儿，就很可能溢出 10^9 的范围，所以要考虑的方案其实很少；
- 出题人做法：
 - 最终数组，绝对值非1的最多30个；
 - $n > 30$ ：枚举哪个数变成1或-1，然后把此时数组乘积算出来，在枚举前先判断一下这个数变的话能不能保证变后绝对值非1的最多30个，所以真正要枚举的数不多；
 - $n \leq 30$ ，此时要么第一个数绝对值在 $\sqrt{10^9}$ 内、要么第二个数绝对值在 $\sqrt{10^9}$ 内，否则，前两个数乘积的绝对值一定大于 $1e9$ ；因此我们 $\sqrt{10^9} * n$ 的暴力枚举就可以；
- 实际上，不需要像上面做法这么有道理，一些更混乱的枚举方法也都是其实对的
- 比如下一页这份验题人代码

D 数组成鸡

```
43 int main(void)
44 {
45     ios::sync_with_stdio(false);
46     cin.tie(0);cout.tie(0);
47     ll l,r,x;
48     int q,i;
49     cin>>n>>q;
50     for(i=1;i<=n;++i)
51         cin>>a[i];
52     sort(a+1,a+n+1);
53     s.emplace(0);
54     a[0]=INF1;
55     for(i=1;i<=n;++i)
56     {
57         if(a[i]==a[i-1])
58             continue;
59         for(l=-a[i]-1;check(l);--l);
60         for(r=-a[i]+1;check(r);++r);
61     }
62     while(q--)
63     {
64         cin>>x;
65         cout<<(s.count(x)?"Yes\n":"No\n");
66     }
67     return 0;
68 }
```

直接令 a_i 变成0，从0开始枚举附近的数，直到`check()`返回false为止

`check`返回false的条件：
数组乘积大于 $1e9$

J 又鸟之亦心

- 笑点解析：标题是“鸡之恋”，故事主人公又是小又和小鸟，令人忍俊不禁；
- 最关键的性质：考虑第 i 个任务结束时，一定有一个人在 a_i 处；（听上去是废话）
- 这提示我们，只需要记录，另一个人可能的位置，可以用 `set<int>` 来存；
- 具体做法：
- 二分一个答案 `mid`，`check` 的写法是遍历所有的任务，用一个 `set<int>` 存：此时，一个人在 a_i 处，另一个人可能的位置集合；
- 若某个时刻 `set` 为空，则 `check` 失败；否则，`check` 成功；
- 注意向 `set` 里插入 a_i 的时机需要仔细考虑，一种正确的插法是：
 - 在 a_{i+1} 处判断若 $|a_i - a_{i+1}| < mid$ 是否成立
 - 若成立，插入 a_i
- `jiangly` 代码写的就是这个思路，大家可以参考



牛客竞赛

AC.NOWCODER.COM

THANKS

AC.NOWCODER.COM