

北京理工大学第九届程序
设计新生赛正赛题解

A. 出题人的碎碎念

- 出题人: fried-chicken;
- 题意: 阅读一段关于ACM俱乐部和新生赛的文章, 回答五个问题, 答对四个就算通过;
- 预期难度: Easy.
- 首次提交: 1min 王凯
- 首次通过: 4min 李坚辉
- 正确率: 328 corrects/ 652 unaccepted

A. 出题人的碎碎念

- 问题一：根据原文推理，可以看出ICPC银川站发放了50,100,150的金、银、铜牌；又可以看出我校四支队伍分别为冠军、112名、126名、148名，所以答案是一金三银零铜（1 3 0）；
- 问题二：根据原文信息，本题是fried-chicken所出，且其他题目中没有出现zyw的都是fried-chicken所出的，通过阅读整个题册可以看出，还有“客观的校长”与“零时困境”两题，因此总共有三题（3）；

A. 出题人的碎碎念

- 问题三：“本次新生赛面向所有大一和大二对程序设计竞赛有兴趣的同学”可以看出只有19级和20级能参加本次新生赛，而参加银川区域赛的同学中，有3名20级同学和1名19级同学，他们都可以（并且事实上也都）参加本次新生赛（4）；
- 问题四：“bit_liyiwen”中的“_”与“chenhongta0”中的“0”都是非小写字母（2）；
- 问题五：
 - 说法一：错，ACM俱乐部有前身；
 - 说法二：错，题目难度不按顺序排；
 - 说法三：肯定错；
 - 说法四：错，是第一个区域赛冠军，但不是第一个进入World Finals；
 - （0）

A. 出题人的碎碎念

- 花絮：
- 问题一原本的奖牌发放规则其实更绕一些，现在的是简化的了；
- 为了问题二，我们特意在很多原来没zyw的题面中都加入了zyw，所以有些地方zyw出现的比较强行；
- 问题四的主要坑点“chenhongta0”表示：其实他有个小号确实就叫“chenhongtao”；
- 一开始本题要求全部答对，害怕提交过多导致评测机崩溃，特意没把这道题放到A题；在特意改成了答对4个就行之后又重新放到了A题的位置上。

A. 出题人的碎碎念



工藤草莓

工藤草莓 21:43:59
zyw大佬要不顺便把zyw范数题解写了

@王马老吉



王马老吉

工藤草莓 21:44:00
zyw大佬要不顺便把zyw范数题解写了

@工藤草莓 草，那这场不全是我的题了



zhanggengchen

zyw把题目中有zyw的题和自己出的题的题解写了就行



zhanggengchen



王马老吉

zhanggengchen 21:48:23
zyw把题目中有zyw的题和自己出的题的题解写了就行

@zhanggengchen ???

B. 页面置换

- 出题人: bit_liyiwen
- 题义: 模拟操作系统中内存中的页面置换过程, 统计三种不同方法的缺页中断次数
- 预期难度: Medium.
- 首次提交: 19min 李依凡
- 首次通过: 37min 罗博文
- 正确率: 24 corrects/ 117 unaccepted

B. 页面置换

- FIFO方法可以用队列外加一个visit数组来实现，进入队列的时候visit数组之为1，出队列的时候置为0.
- LRU方法可以暴力的模拟，使用一个数组来记录，每次扫描整个数组来确定，进入数组的新元素放在最前，如果访问的元素再数组中就把他放到最前端即可。
- OPT方法需要建立一个next数组，next[i][j]表示在i位置之后最近的j所在的位置，如果j不再出现置为-1即可。每次访问可以暴力的扫描每个数的下一个出现的位置，找到合理的即可。
- P.S.OPT的全称是optimized，即最优化的，所以OPT的答案一定是最优的（证明留作作业），所以只需要计算OPT的答案即可。

C.Rekop Med1oh Saxet

- 出题人:bit_liyiwen
- 题义: 使用类似于德州扑克的一种规则, 每人有两张私有牌, 牌桌上有三张公共牌, 问在只看到自己和桌上的牌的情况下能否锁定胜局。
- 预期难度: Medium-Hard.
- 首次提交: 46min 李宗浩
- 首次通过: NaN
- 正确率: 0 correct/ 13 unaccepted

C.Rekop Med1oh Saxet

- 枚举剩下的所有牌中的两张给对手之后比大小看似可以做，实际上应该会TLE。
 - 所以需要做一些简单的思考，例如：
 - 皇家同花顺好像可以直接赢
 - 四条好像也可以直接赢
 - 葫芦好像怎么样也赢不了
 -
- 所以这道题为了不超时只能不断的做特判，并且优化代码结构。

C.Rekop Medloh Saxet

- 下面给出题解的大概思路
- 先判断同花，这时对手只剩下两张都是同花色的牌才能赢，这时候可以枚举一下剩下两张牌，输出之前判断是皇家同花顺还是同花顺还是普通同花。
- 四条直接赢
- 葫芦一定输
- 顺子一定输
- 三条需要特殊处理，当且仅当手牌为对子的情况下有可能赢，只要对手不能凑出同花或顺子就能赢，其余情况不难看出会输。
- 剩下的排型都赢不了

C.Rekop Med1oh Saxet

- 花絮:
- 这道题是某次比赛出题人读假了的题
- 把题目倒过来看就是Texas Hold'em Poker, 即德州扑克。

D. 法力 风暴

- 出题人: bit_liyiwen
- 题义: 有很多消耗两种不同法术力的法术, 求在不能施放任何一个法术的时候的最大法力值。
- 预期难度: Medium.
- 首次提交: 19min 李依凡
- 首次通过: 26min 张浩源
- 正确率: 40 corrects/ 480 unaccepted

D. 法力 风暴

- 不妨先用两次循环来判断一下是不是INF
- 之后可以对法术进行排序，按 r 从小到大排序， r 相同按 b 从小到大排序。
- 之后遍历所有的法术，遍历到第 i 个法术时满足前 i 个法术的 r ，但是不满足这 i 个法术的 b 。每次刷新 b 的最小值，与第 $i+1$ 个法术的 r 减去1求和之后维护答案即可。（ b 的最小值为0后不再刷新答案）。
- 因为 n 很小，所以使用冒泡排序也是可以通过的。

E. 肉夹馍

- 出题人: zhanggengchen
- 题义: 输出奇怪的肉肉夹馍图案。
- 预期难度: Easy.
- 首次提交: 26min 陈林峰
- 首次通过: 30min 陈林峰
- 正确率: 151 corrects/ 216 unaccepted

E. 肉夹馍

- 这个题应该就是个乐学题的魔改版？
- 简单的按题义模拟就行了，建议不要用 $O(n^3)$ 时间的做法。
- 花絮：该题的idea是出题人在西安回北京的火车上想出来的。

F. 狂乱

- 出题人: zhanggengchen
- 题义: 对手有一揽子随从, 计算使用一张狂乱在运气最好的情况下的最小场攻。
- 预期难度: Medium Hard.
- 首次提交: 17min 林恺
- 首次通过: 166min 林恺
- 正确率: 2 corrects/ 22 unaccepted



F. 狂乱

- 可以枚举对哪一个随从使用狂乱
- 如果选定的随从可以把其他随从全打死且自己不死，可以直接得到答案
- 如果这个选定的随从一定会死，可以使用动态规划计算其可以消灭的随从的最大攻击和。
- 性质1 如果打一个随从，肯定要把它打死，要不然等于浪费生命值。
- 性质2 击败某个随从需要的生命值和消耗的生命值并不一定一样。

F. 狂乱

- 定义 $dp[i][j]$ 表示前 i 个随从花费了 j 的生命值所击败的随从的攻击和。
- 对于当前的随从，只要血量达到某个阈值便可以击败某对手，但是有时候生命值会变成负的，以此设计转移即可。
- 解决方案：将所有随从按攻击力排序即可，可以证明后打攻击力大的随从总是更好的（留做作业）。
- 每个随从的动态规划做完了取个最佳答案输出即可。

G. 理财大师

- 出题人: shenyunhan;
- 题意: 给一个价格切片序列和最大交易次数限制, 求最大收益;
- 预期难度: Medium.
- 首次提交: 69min 胡楹浩
- 首次通过: 70min 朱子林
- 正确率: 15 corrects/ 29 unaccepted

G. 理财大师

- 显然，在同一时刻既进行买又进行卖是没有意义的，因此每个时刻要么进行买交易，要么进行卖交易，要么什么也不做；
- 为了最大化收益，每次买交易都应该进行交易量为1的交易，同理，每次卖交易也是全部卖出，因此每时刻要么持有1bitc，要么不持有bitc；
- 考虑 $dp[i][j][c]$ 表示第 i 时刻进行了 j 次交易，且持有1bitc($c = 1$)或不持有bitc($c = 0$)时的最大收益，则有如下转移方程：
$$\begin{cases} dp[i][j][0] = \max(dp[i-1][j][0], dp[i-1][j-1][1] + p[i]) \\ dp[i][j][1] = \max(dp[i-1][j][1], dp[i-1][j-1][0] - p[i]) \end{cases}$$
- $\max_{j=0 \dots k} \{dp[n][j][0]\}$ 就是最终答案。
- 合理的贪心解法也可通过本题的测试样例。

G. 理财大师

Delayi的题解：

- 贪，就硬贪
- 理财大师，不贪心理什么财！
- 首先有一个前置条件，每一时刻要么是满仓（1个bitc），要么是空仓。所以整个购买流程实质上是将整个时间轴切分成一些区间，交替满仓和空仓。
- 因此贪心策略实质上就是寻找这些区间的划分。
- 对于只要买卖一次的情况，显而易见，我们贪心的在最大涨幅的区间中满仓，然后卖出，就可以得到最大的收益。那么对于买卖两次的情况呢？在最大涨幅区间外去找第二大涨幅？显而易见这是理财菜鸟的作为，这连样例都过不了。实质上在贪心的划分更多的区间时，是有可能要拆分在策略中已经满仓的区间的，比如样例中的例子。

G. 理财大师

- 在只买卖一次的情况下，当然是在第一时刻购入，第四时刻售出。而对于可以再买卖一次的情况，也可以在第二时刻售出，再在第三时刻购入。对于这样的情况，实际的收益增加是在满仓区间内的最大跌幅。
- 所以贪心策略就是将整个时间轴切分成满仓区间和空仓区间，每增加一次买卖的机会，就分别在各个区间中找到满仓区间的最大下降和空仓区间的最大上升，他们中的最大的值就是这一轮增加的买卖机会的最大收益。
- 贪心正确性证明:略(验题人说它是正确的它就是正确的)
- 区间求最大上升或下降:我解释不清但我觉得可以百度。

G. 理财大师

- 赛时有人写了网络流，我大受震撼。

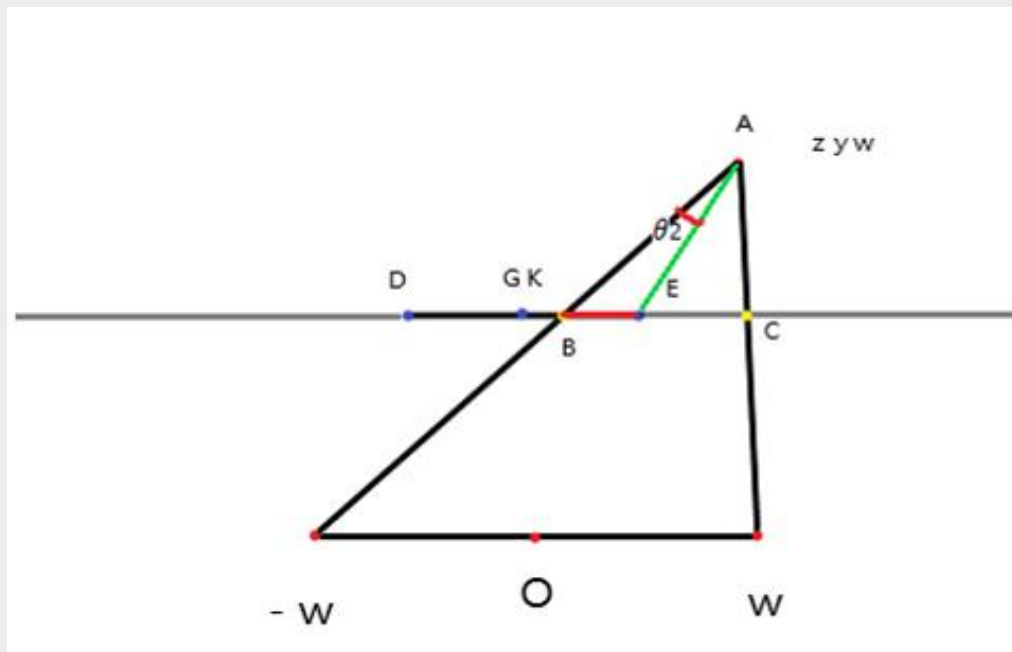
```
int main() {
    scanf("%d%d", &n, &k);
    k /= 2;
    tot = n;
    int s = ++tot;
    int t = ++tot;
    int ss = ++tot;
    add(ss, s, k, 0);
    for(int i = 1; i <= n; i++) {
        double x;
        scanf("%lf", &x);
        int xx = x * 100 + 0.5;
        add(s, i, 1, -xx);
        add(i, t, 1, xx);
        if(i > 1) {
            add(i, i - 1, 1, 0);
        }
    }
    int cost, maxF;
    solve(ss, t, cost, maxF);
    printf("%.10f\n", (-cost) / 100.0);
    return 0;
}
```

H. 守门员

- 出题人: zhanggengchen;
- 题意: 给定球门的范围和守门员的防守范围, 问能进球的角度大小。
- 预期难度: Medium Easy.
- 首次提交: 24min 于瀚文
- 首次通过: 33min 于瀚文
- 正确率: 26 corrects/ 213 unaccepted

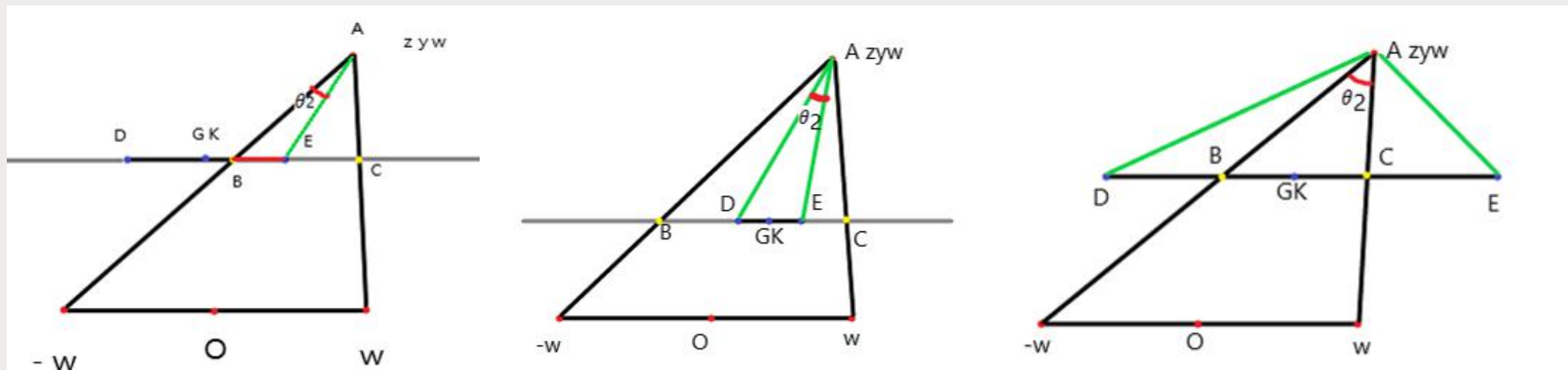
H. 守门员

- 如图所示zyw在A点，设球门范围在 $(-w, 0)$ 到 $(w, 0)$ 的范围内。我们首先可以用相似三角形求出A点与两个门柱的连线与守门员 (x_{GK}, y_{GK}) 所在水平直线 $y = y_{GK}$ 的两个交点B, C的横坐标 x_B 和 x_C 。三角形 $\triangle ABC$ 在A点的角即为没有守门员是zyw能够进球的角度 θ_1 。设守门员水平扑救的范围为D到E横坐标范围是 (x_D, x_E) 。求出区间 (x_D, x_E) 和 (x_B, x_C) 的交 $(\max(x_D, x_B), \min(x_E, x_C))$ ，形成一个三角形 $\triangle A(\max(x_D, x_B), y_{GK})(\min(x_E, x_C), y_{GK})$ ，用余弦定理求出该三角形A点的角即为守门员能够封堵的角度 θ_2 ，zyw能够进球的角度 θ 为 $\theta_1 - \theta_2$ 。



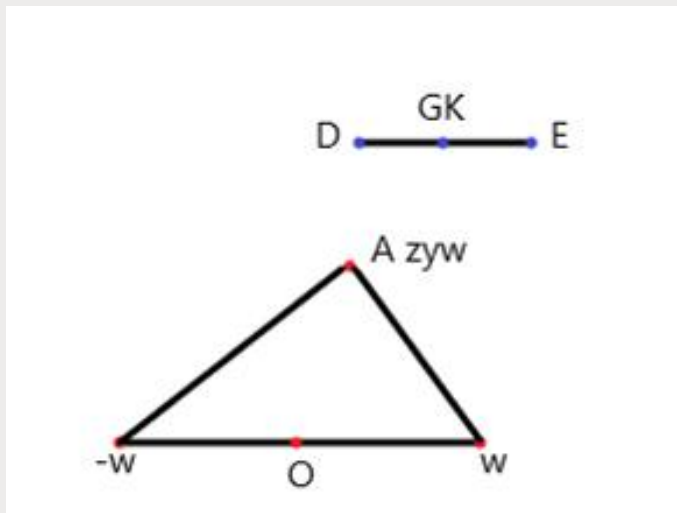
H. 守门员

- 提示1: 守门员站位基本有如下几种情况



H. 守门员

- 如果zyw射门前已经过掉了守门员，那他一定可以进球，如下图所示。



- 提示2： 请注意`acos`函数接收参数是范围在-1到1的浮点数，在计算余弦时的浮点误差可能会导致余弦值小于-1或大于1。若计算出的余弦值小于-1则将余弦值置为-1，若计算出的余弦值大于1则将余弦值置为1。

I. 创业园的共享单车

- 出题人: `chenhongta0`
- 题义: n 个人 m 辆自行车行走 L 的距离的最短时间。
- 预期难度: Medium Easy.
- 首次提交: 7min 陈雪峰
- 首次通过: 7min 陈雪峰
- 正确率: 91 corrects/ 687 unaccepted

I. 创业园的共享单车

- 结论: $t_{min} = \begin{cases} \frac{L}{vb}, n \leq m \\ \frac{L}{vb} * \frac{m}{n} + \frac{L}{vp} * \left(1 - \frac{m}{n}\right), n > m \end{cases}$
- $n \leq m$ 时显然。
- $n > m$ 时的一种感性理解: 最佳方案下肯定把 m 辆车都骑到终点, 且 n 个人同时到达终点, 这就要求我们把车经过的总路程 $L * m$ 平均地分配给每个人来骑, 所以对于每个人来说有 $L * \frac{m}{n}$ 的路程是以车速前进的, 其余路程是以人速前进的, 这样就能得出上面的结论, 顺利通过此题了。下面给出一种通用的构造方案。

I. 创业园的共享单车

- $n > m$ 时的一种构造方案：建立一个 $m * n$ 的矩阵 A , $A_{ij} = (i + m * j) \% n$, 将整条路 n 等分, 记作路段0到路段 $n - 1$; 将 m 辆单车记作单车0到单车 $m - 1$; 将 n 个人标号0到 $n - 1$ 。那么单车 i 在路段 j 由 A_{ij} 号人来骑即为一种可行方案。以下为 $n = 5$, $m = 3$ 时的矩阵 A 。

	路段0	路段1	路段2	路段3	路段4
单车0	0	3	1	4	2
单车1	1	4	2	0	3
单车2	2	0	3	1	4

J.zyw 范数

- 出题人: strawberry
- 题意: 给出一个大小不超过3的方阵和一个整数k, 求将其中一个位置替换为k所能得到的方阵最大行列式值;
- 预期难度: Easy.
- 首次提交: 23min 凌铭志
- 首次通过: 23min 凌铭志
- 正确率: 117 corrects/ 335 unaccepted

J.zyw 范数

- 对于 $n=1, n=2, n=3$ 三种情况的行列式，分情况按照公式计算即可；
- 可能出现的问题：
- 读错了题，把没有进行替换（题目要求必须替换）的原行列式值也纳入考虑了；
- 行列式值可能为负，所以用于记录所有方案中最大值的变量，初始应当赋值为一个足够大的负数（例如 -10^9 ）；

J.zyw 范数

■ 我大受震撼

```
if(n==3){
    ans=c*(a[1][1]*a[2][2]-a[1][2]*a[2][1])+a[0][1]*(a[1][2]*a[2][0]-a[2][2]*a[1][0])+a[0][2]*(a[1][0]*a[2][1]-a[1][1]*a[2][0]);
    ans=max(ans,c*(a[1][2]*a[2][0]-a[1][0]*a[2][2])+a[0][2]*(a[1][0]*a[2][1]-a[2][0]*a[1][1])+a[0][0]*(a[1][1]*a[2][2]-a[1][2]*a[2][1]));
    ans=max(ans,c*(a[1][0]*a[2][1]-a[1][1]*a[2][0])+a[0][0]*(a[1][1]*a[2][2]-a[2][1]*a[1][2])+a[0][1]*(a[1][2]*a[2][0]-a[1][0]*a[2][2]));
    ans=max(ans,c*(a[2][1]*a[0][2]-a[2][2]*a[0][1])+a[1][1]*(a[2][2]*a[0][0]-a[0][2]*a[2][0])+a[1][2]*(a[2][0]*a[0][1]-a[2][1]*a[0][0]));
    ans=max(ans,c*(a[2][2]*a[0][0]-a[2][0]*a[0][2])+a[1][2]*(a[2][0]*a[0][1]-a[0][0]*a[2][1])+a[1][0]*(a[2][1]*a[0][2]-a[2][2]*a[0][1]));
    ans=max(ans,c*(a[2][0]*a[0][1]-a[2][1]*a[0][0])+a[1][0]*(a[2][1]*a[0][2]-a[0][1]*a[2][2])+a[1][1]*(a[2][2]*a[0][0]-a[2][0]*a[0][2]));
    ans=max(ans,c*(a[0][1]*a[1][2]-a[0][2]*a[1][1])+a[2][1]*(a[0][2]*a[1][0]-a[1][2]*a[0][0])+a[2][2]*(a[0][0]*a[1][1]-a[0][1]*a[1][0]));
    ans=max(ans,c*(a[0][2]*a[1][0]-a[0][0]*a[1][2])+a[2][2]*(a[0][0]*a[1][1]-a[1][0]*a[0][1])+a[2][0]*(a[0][1]*a[1][2]-a[0][2]*a[1][1]));
    ans=max(ans,c*(a[0][0]*a[1][1]-a[0][1]*a[1][0])+a[2][0]*(a[0][1]*a[1][2]-a[1][1]*a[0][2])+a[2][1]*(a[0][2]*a[1][0]-a[0][0]*a[1][2]));
    printf("%d\n",ans);
}
```

K.Easy Sigma

- 出题人: bit_liyiwen
- 题意: 求值。
- 预期难度: Hard.
- 首次提交: 11min 刘进宇
- 首次通过: NaN
- 正确率: 0 correct/ 169 unaccepted

K.Easy Sigma

- T的范围达到1e5，所以需要很快的速度来求值，暴力的计算不仅算不出结果而且精度不够。

- 先做一些简单的转化：

$$\begin{aligned}(-1)^x &= 1 - 2(x \bmod 2) \\ &= 1 - 2(x - 2\lfloor \frac{x}{2} \rfloor) \\ &= 1 - 2x + 4\lfloor \frac{x}{2} \rfloor\end{aligned}$$

- 所以

$$\sum_{i=1}^n (-1)^{\lfloor i\sqrt{k} \rfloor} = n - 2 \sum_{i=1}^n \lfloor i\sqrt{k} \rfloor + 4 \sum_{i=1}^n \lfloor \frac{i\sqrt{k}}{2} \rfloor$$

K.Easy Sigma

■ 不妨令

$$f(n, a, b, c) = \sum_{i=1}^n \left\lfloor \frac{a\sqrt{k} + b}{c} i \right\rfloor$$

■ 所以变成了计算

$$n - 2f(n, 1, 0, 1) + 4f(n, 1, 0, 2)$$

■ 令

$$\frac{a\sqrt{k} + b}{c} = d$$

K.Easy Sigma

When $\lfloor d \rfloor > 0$

$$\begin{aligned} f(n, a, b, c) &= \sum_{i=1}^n \lfloor di \rfloor \\ &= \sum_{i=1}^n \lfloor di - \lfloor d \rfloor i + \lfloor d \rfloor i \rfloor \\ &= \sum_{i=1}^n \left\lfloor \frac{a\sqrt{k} + b - c\lfloor d \rfloor}{c} i \right\rfloor + \frac{n(n+1)}{2} \lfloor d \rfloor \\ &= f(n, a, b - c\lfloor d \rfloor, c) + \frac{n(n+1)}{2} \lfloor d \rfloor \end{aligned}$$

K.Easy Sigma

When $\lfloor d \rfloor = 0$

$$\begin{aligned} f(n, a, b, c) &= \sum_{i=1}^n \lfloor di \rfloor \\ &= \sum_{i=1}^n \sum_{j=1}^{\lfloor dn \rfloor} [j < di] \\ &= \sum_{j=1}^{\lfloor dn \rfloor} \sum_{i=1}^n [i > \frac{j}{d}] \\ &= \sum_{j=1}^{\lfloor dn \rfloor} (n - \lfloor \frac{j}{d} \rfloor) \end{aligned}$$

$$\begin{aligned} &= n * \lfloor dn \rfloor - \sum_{j=1}^{\lfloor dn \rfloor} \lfloor \frac{c}{a\sqrt{k} + b} j \rfloor \\ &= n * \lfloor dn \rfloor - \sum_{j=1}^{\lfloor dn \rfloor} \lfloor \frac{ac\sqrt{k} - bc}{ka^2 - b^2} j \rfloor \\ &= n * \lfloor dn \rfloor - f(\lfloor dn \rfloor, ac, -bc, ka^2 - b^2) \end{aligned}$$

K.Easy Sigma

- 之后就可以用递归函数来计算了，复杂度为 $O(\log w \cdot \log n)$ 。
- 总时间复杂度为 $O(T \cdot \log w \cdot \log n)$ 。
- (w 为值域，不大于 $1e18$)
- P.S. 因为long long范围不太够，可以让 a , b , c 同除最大公因数，以防止溢出。

L. 零时困境

- 出题人: fried_chicken
- 题意: 对于一个全排列, 你已经知道了一些信息, 每条信息形如 p_i, p_j, p_k 的中位数的下标是 i 或 j 或 k , 问能否根据所给信息唯一确定原排列;
- 预期难度: ? .
- 首次提交: 10min 胡楹浩
- 首次通过: 139min 林恺
- 正确率: 3 corrects/ 3 unaccepted

L. 零时困境

- 注意到对于长为 n 的排列，将1和 n 交换位置，一定不会改变任何询问的结果，所以总不能确定原排列，输出NO。
- 特别的，对于 $n = 1$ 的情况，因为只有一种排列，所以输出YES。
- 花絮：“零这家伙，一定又在捣什么鬼，我才不信他会有这么好心。”西格玛喃喃自语道。
- 这个 $n = 1$ 的特判不全是出题人的恶趣味，我的想法是，如果你很明确的清楚了为何总是NO，那会更容易注意到这个特例（因为“把1和 n 交换位置”的特例就是 $n = 1$ ），所以这个特例有助于让真正理解了这个原理的人有更大概率通过此题。

M. 客观的校长

- 出题人: fried-chicken;
- 题意: 定义一个人若选择了一个周围k个位置已经有人的位置坐, 则这个人就是低情商的, 现在给出一排座位的就座情况, 输出根据到来先后顺序的不同, 低情商人数最少和最多有多少个;
- 预期难度: Medium.
- 首次提交: 10min 丁致远
- 首次通过: 26min 丁致远
- 正确率: 38 corrects/ 80 unaccepted

M. 客观的校长

- 低情商最多：
- 可以发现，若所有同学的到来顺序就是按照输入串的从前到后顺序的话，就是使低情商贪心最多的一种方案；
- 所以直接按该顺序模拟一遍该过程，统计这样会有多少低情商同学即可；
- 低情商最少：
- 由于总人数一定，高情商最多就代表着低情商最少；
- 如何让高情商最多？其实还是类似的，钦定第一个同学是高情商的，然后从前往后遍历，只要当前这个人能高情商（和前一个高情商相隔已经超过了 k ）就让他是高情商的；
- 最后答案用总人数减去高情商人数即可；

M. 客观的校长

- 花絮：本来这是两道题，一道叫乐观的校长，一道叫悲观的校长，现在合二为一了所以是客观的校长；
- 题目背景本来其实不是坐座位的间隔，是某个其他的背景，不妨猜猜原来的题目背景；