



牛客竞赛

AC.NOWCODER.COM

# 2025 牛客 暑期多校训练营 8

skywalkert, quality

HR.NOWCODER.COM



AC.

# 概况

- 入门: A, C;
- 简单: B, J;
- 中等: F, G;
- 较难: E, H, I;
- 困难: D, K。

# 私密马赛

- 首先需要抱歉，由于出题组疏忽：
- I 题：没有提前说明每个人最多参加一场比赛。
- J 题：没有提前说明  $|A|$  是  $A$  的字符串长度还是  $A$  在复数意义下的模长。
- G 题：没有提前说明重边只考虑边的两端点不同、方案数却要考虑不同边权，不少选手需要通过分析样例输出才能猜对题意。
- 对此造成的问题我们深表歉意，我们会重视后续的题面审校，保留充分时间打磨题目。

# A - Insert One

## 题意

- 认为没有前导零的十进制整数是合法数字。
- 给一个合法数字  $x$ , 插入一个数位 1 后要求还是合法数字, 最大化修改后数字。
- 多组数据,  $-2^{63} \leq x < 2^{63}$ , 答案绝对值可能超过  $2^{64}$ 。

AC.

# A - Insert One

## 题解

- 因为答案会爆 int64，所以最好用字符串处理输入输出。
- 比较直观的暴力做法是枚举每个位置插入，与当前最优解比大小。比如对于 1024，检查 11024, 10124, 10214 和 10241，找到最大值即可。
- 令  $x$  的十进制长度为  $l$ ，时间复杂度为  $\mathcal{O}(l^2)$ 。考虑到题目定位是签到题，所以没有卡这种做法。
- 更简单的实现需要一定分析，可以对不同位置插入后的数字变化进行讨论。

## A - Insert One

### 题解 (cont'd)

- 如果  $x > 0$ , 则从高到低找到第一个小于 1 的数位, 在其前面插入最优 (例如  $910 > 901$ ), 如果不存在这样的数位则加到末尾最优 (例如  $1231 > 1123, 3211 > 1321$ )。
- 如果  $x = 0$ , 则插入方法唯一, 输出 10。(此情况可以和  $x > 0$  合并考虑)
- 如果  $x < 0$ , 则从高到低找到第一个大于 1 的数位, 在其前面插入最优 (例如  $-12 > -21$ ), 如果不存在这样的数位则加到末尾最优 (例如  $-101 > -110$ )。
- 时间复杂度降低至  $\mathcal{O}(l)$ 。

# C - Bernoulli's Principle

## 题意

- 高中物理题。有一竖直放置在水平平面的圆柱体容器内盛满液体，液面最高点与水平平面高度差保持在  $H$ 。
- 容器侧面开有  $n$  个小孔向外漏水，初速度方向平行水平平面，编号  $i$  小孔中心与水平平面高度差为  $a_i$ 。
- 分析各孔水流到达水平平面时的水平方向距离，按此距离非降序排列小孔编号。
- 多组数据， $1 \leq n \leq 10^5$ 。

# C - Bernoulli's Principle

## 题解

- 这是一道定位签到题的阅读理解题。
- 注意到水平速度与  $\sqrt{H - h_i}$  正相关，竖直自由落体运动时长与  $\sqrt{h_i}$  正相关，于是可知水平距离和  $\sqrt{h_i(H - h_i)}$  正相关，关于此指标排非降序即可。
- 也可以用观察法找到排序指标为  $\min(h_i, H - h_i)$  或者  $|2h_i - H|$  等。
- 注意该数据范围下需要时间复杂度  $\mathcal{O}(n \log n)$  的排序算法。

# B - Inversion Number Parity

## 题意

- 给定随机数生成器参数，基于交换生成一个 0 到  $(n - 1)$  的随机排列，并生成  $(n - 1)$  次区间循环左移的修改，问修改前后每次排列的逆序对奇偶性。
- 多组数据， $1 \leq n \leq 10^7$ ，内存限制 32MB。

## B - Inversion Number Parity

### 题解

- 简单算下 32MB 可以存什么。
- $n$  个 bool 约 10MB,  $n$  个 int 约 40MB, 所以不可以存  $n$  个 int、不可以把序列完全存下来分析。
- 尝试从题目性质入手, 思考逆序对奇偶性与什么有关。
- 考虑交换的两个数  $p_i$  和  $p_j$  ( $i < j$ ) 如何影响逆序对数:
- 对于下标不涉及  $i$  或  $j$  的元素对, 其贡献不变;

## B - Inversion Number Parity

### 题解 (cont'd)

- 对于下标不在区间  $(i, j)$  的元素，他们与  $p_i$  和  $p_j$  的相对关系均不变，因此相关的元素对贡献不变；
- 对于下标在区间  $(i, j)$  但是值域不在区间  $(\min(p_i, p_j), \max(p_i, p_j))$  的元素，他们与  $p_i$  和  $p_j$  的相对关系恰好各发生一次相反的变化，相互抵消；
- 对于下标和值域都在对应区间内的元素，他们与  $p_i$  和  $p_j$  的相对关系恰好各发生一次相同的变化，虽然整数意义上没有抵消，但是模 2 意义下抵消；
- 对于  $p_i$  和  $p_j$ ，由于这是一个排列，只要  $i \neq j$  就会发生一次变化。

## B - Inversion Number Parity

### 题解 (cont'd)

- 因此，每次交换两个不相等的数字只会让逆序对数的奇偶性发生一次变化。
- 初始生成排列是从逆序对数为 0 的情况出发做一系列修改，可以直接维护奇偶性。
- 区间循环左移一位相当于交换区间长度次，直接计算即可。
- 时间复杂度  $\mathcal{O}(n)$ ，空间复杂度  $\mathcal{O}(1)$ 。
- 对于输出效率不高的语言，可以每攒一批字符（例如 1MB）再打印并刷新缓冲区。

# J - Multiplication in Base the Square Root of -2

## 题意

- 定义根号 -2 进制表示为  $(c_{n-1} c_{n-2} \dots c_0)_{\sqrt{-2}} = \sum_{i=0}^{n-1} c_i \sqrt{-2}^i$ 。
- 给定两个根号 -2 进制下的数字  $A$  和  $B$ , 求它们的乘积在根号 -2 进制下的表示。
- 多组数据,  $1 \leq \text{len}(A) + \text{len}(B) \leq 2 \times 10^5$ 。

# J - Multiplication in Base the Square Root of -2

## 题解

- 这是一道 FFT 裸题，考虑到比赛定位，不熟悉 FFT 也可以压位暴力实现高精度乘法。
- 唯一需要注意的是如何实现进位，注意到奇数位和偶数位是互相线性无关的，所以可以改成  $a + b\sqrt{-2}$  表示，分别对  $a$  和  $b$  进行  $-2$  进制的进位。
- 而  $-2$  进制进位比较经典，每次让最低位调整至  $\in \{0, 1\}$ ，使得额外的部分是  $-2$  的倍数，直接可以进位。

## J - Multiplication in Base the Square Root of -2

### 题解 (cont'd)

- 对于压位，注意到  $a$  和  $b$  可以分别压 24 位（总 48 位），单次乘法引入的中间值最大  $\frac{2 \times 10^5}{2 \cdot 48} \cdot 2^{48} < 2^{63}$ ，这里除 2 是因为和更短的那边长度有关。
- 压位之后怎么进位呢？也不难，考虑低 24 位能取到的数字范围，形如  $[-2P, P]$ ，其中  $3P + 1 = 2^{24}$ ,  $P = \frac{2^{24}-1}{3}$ ，于是直接取二进制低 24 位的值检查是否在此范围内，不在则只需要微调一次即可让额外部分是  $(-2)^{24}$ （也即  $2^{24}$ ）的倍数然后进位。这里取每 24 位一组而不是每 25 位一组，也是考虑到  $(-2)^2 = 2^2$  方便用位运算加速。
- 整个复杂度就是  $\mathcal{O}\left(\frac{n^2}{B^2}\right)$ ，这里  $B = 96$ ，跑得比大多数 FFT 快。

# G - Changing Minimum Spanning Tree

## 题意

- 给定一张  $n$  点、 $m$  边带权无向简单图，且边权要求为  $[1, k]$  整数。
- 再加一条边进去，边权  $[1, k]$ ，使得新图还是简单图，但其存在至少一个生成树，且所有最小生成树里都包含这条新边。
- 统计方案数，模  $(10^9 + 7)$ 。
- 多组数据， $1 \leq n \leq 10^5$ ， $0 \leq m \leq 2 \times 10^5$ ， $1 \leq k \leq 10^9$ 。

# G - Changing Minimum Spanning Tree

## 题解

- 注意加入一条边后 MST 可能依然不存在，所以首先需要考虑原图最后剩  $1, 2, \geq 3$  个连通块的情况，1 个连通块时是替换 MST 里一条边，2 个连通块时是在这两个连通块之间任意加一条边， $\geq 3$  个连通块无解。只有替换 MST 里一条边的情况需要展开分析。
- 首先对 MST 中边权进行排序，按照边权分段考虑，假设当前考虑了权值最大值为  $w$  的 MST 子集，不在子集里的 MST 边权最小值是  $w'$ ，那么新加边权在  $[w, w')$  时，需要满足新加边是在当前子集形成的连通块之间，且不和子集外的边重复。这个可选边数可以用一些方式维护到并查集里。

## G - Changing Minimum Spanning Tree

### 题解 (cont'd)

- 方法 1：维护当前不可选边数，每个连通块维护其向外连的边集，合并连通块时启发式合并边集，并修正不可选边数。
- 方法 2：维护可选边数数组，并查集按秩合并不压缩，支持查询同连通块内两点间首次连通使用的边权，枚举到同连通块边时修正原来的可选边数。
- 时间复杂度都是  $\mathcal{O}(n \log n)$ ，空间复杂度都是  $\mathcal{O}(n)$ ，其中方法 2 避开 hash，常数上会快一些。

# F - Broken LED Lights

## 题意

- 给定数位 0 到 9 的 7 根 LED 灯管表示，考虑  $n$  个长度  $m$  数字的灯管表示，保留最少的灯管，使得仅用这些灯管即可区分这  $n$  个数。
- 对于任意两个数，如果在剩下的灯管里，存在一个灯管状态在两数中不同，就能区分这两数。
- 输出最少灯管数。
- 多组数据， $1 \leq n \leq 100$ ， $1 \leq m \leq 3$ 。

# F - Broken LED Lights

## 题解

- 最暴力做法是搜索用哪些灯管后检查  $n$  个数不同，时间复杂度  $\mathcal{O}(2^{7m}n)$ ，复杂度跑满是过不了的，加一点最优化剪枝就可以过。
- 但不要乱加剪枝，例如在没有优化分支数量时，每一层增加  $\mathcal{O}(n)$  或  $\mathcal{O}(7m)$  的代价去找更优分支就会被卡。
- 进一步发现主要是剪枝分析代价过高，可以每个位置的  $2^7$  种候选整体分析，有一定效果但是不大。

# F - Broken LED Lights

## 题解 (cont'd)

- 注意到如果只有一个位置，那么最多用 5 个灯管即可，以及  $2^7$  很多重复情况没必要，具体来说：
- 如果两种候选方式对任意数位集合的等价类划分结果一样，那么只需要留一个使用灯管数最小的；
- 如果一种候选方式能比另一种候选方式划分得更细（等价类不相交或者包含），那么另一种候选方式留下的前提是它使用灯管数严格小于该候选方式。
- 缩一下只有 58 种情况，时间复杂度降低至  $\mathcal{O}(58^m n)$  就可以过了。

AC.

## I - Tennis Match

## 题意

- $m$  个人选出  $n$  组打  $n$  场比赛，每组 2 人，每人只多一组。
- 每个人有能力值  $e_i$ ，每场比赛要求组队成员能力值之差不超过定值  $d$ ，每场额外还有个能力值上限  $l_j$ 。
- 每个人有初级、高级类型，对于  $t = 0, 1, \dots, 2n$ ，统计恰好  $t$  个初级人被选的前提下， $2n$  个人能力值之和最大值。
- 多组数据， $1 \leq n \leq 20$ ， $2n \leq m \leq 10^5$ ， $0 \leq e_i, d, l_j \leq 10^9$ 。

AC.

## I - Tennis Match

## 题解

- 由于每一场的  $d$  一样，当要选的队员已经确定，配对是可以贪心做的，即按能力值排序后相邻两两配对，如果这种方案合法，那么其他方案也合法，反之不存在其他方案合法。
- 因此直接 DP 即可，将队员排序，令  $f[i][j][k]$  表示前  $i$  位队员里选了  $j$  个（含第  $i$  个位、总共  $k$  个初级队员时这  $j$  位队员的能力值之和最大值， $c_i = [t_i = 1]$ ），于是有：
  - $f[i][2j][k] \leftarrow f[i'][2j-1][k - c_i] + e_i$  需要满足  $e_i - e_{i'} \leq d$  且  $e_i \leq l_j$ ；
  - $f[i][2j+1][k] \leftarrow f[i'][2j][k - c_i] + e_i$  需要满足  $e_i \leq l_j$ 。

AC.

## I - Tennis Match

## 题解 (cont'd)

- 对于第一种情况，满足条件的  $i'$  是一个滑动区间，且随着  $i$  增大，区间左右端点同向移动，可以用单调队列维护 DP 最大值。
- 对于第二种情况，满足条件的  $i'$  是一个前缀区间，直接维护前缀 DP 最大值。
- 整体时空复杂度为  $\mathcal{O}(nm^2)$ 。如果觉得空间不够，可以将  $j$  那一维度滚动掉。

## I - Tennis Match

## 题解 (cont'd)

- 另外还有一种时空复杂度相同但是常数更小的做法。
- 注意到如果一组配对队员为排升序后第  $L$  位配第  $R$  位 ( $L < R$ )，且存在一个  $M$  ( $L < M < R$ ) 使得第  $M$  位队员和第  $L$  位队员属于相同类别，那么选第  $L$  位总是不优的，可以在枚举第  $R$  位时只选择前面每种类别队员最靠后的，一次性转移两位队员。

# H - Interval LRU

## 题意

- 给一个长度为  $n$  的数组  $[a_1, a_2, \dots, a_n]$ , 以及  $q$  个询问, 查询有两种:
- 给定  $l, r, k$ , 表示按顺序访问编号为  $a_l, a_{l+1}, \dots, a_r$  的页面, 使用容量为  $k$  的 LRU 缓存, 统计缓存命中次数;
- 给定  $l, r, k$ , 表示按顺序访问编号为  $a_l, a_{l+1}, \dots, a_r$  的页面, 使用 LRU 缓存, 要求缓存命中次数最少为  $k$ , 问最小所需容量, 无解回答  $-1$ 。
- 多组数据,  $1 \leq n, q \leq 10^5$ 。

# H - Interval LRU

## 题解

- 首先需要分析出 LRU 命中的特点，找到满足  $j < i$  且  $a_j = a_i$  最大的  $j$ ，只有访问  $a_i$  前  $a_j$  一直在 LRU 中才能得到命中，这意味着只要  $a_{j+1}, a_{j+2}, \dots, a_{i-1}$  去重后的数量小于 LRU 容量即可。
- 设  $p_i = j$ ，这个数量为  $f_i$ ，询问一转化为对于  $l \leq p_i < i \leq r$  统计有多少满足  $f_i < k$ ，询问二转化为求  $f_i$  第  $k$  小值。
- 这是一组经典问题，可以用树状数组套可持久化线段树或者莫队算法结合值域分块解决，预期时间复杂度  $\mathcal{O}(n \log^2 n)$  或  $\mathcal{O}(n\sqrt{n})$ ，这里假设  $\mathcal{O}(q) = \mathcal{O}(n)$ 。

# H - Interval LRU

## 题解 (cont'd)

- 这里简单讲一下莫队算法结合值域分块怎么做。
- 莫队算法把请求排一定顺序，变成  $\mathcal{O}(q\sqrt{n})$  次  $[p_i, i]$  区间贡献的插入、删除。
- 对于  $f_i$  的取值范围  $[0, n)$ ，分  $\sqrt{n}$  段，每段统计出现次数，每个值也单独统计出现次数。
- 对于询问一，整段  $< k$  的直接统计，不足整段的暴力枚举每个值计数；
- 对于询问二，从小到大枚举整段，直到第  $k$  小坐落于此段，再暴力枚举每个值。

# E - Matrix Lottery

## 题意

- 有一个  $n \times n$  矩阵，其中每个格子有一盏灯，初始都为关闭状态。
- 每次固定概率选其中一个位置点亮，可能重复选到之前的位置。
- 有  $q$  个询问，每次希望把  $n$  行、 $n$  列、2 对角线里某些线上的灯都点亮至少一次，问期望操作次数。
- 输入格式希望你注意到每条线都只有  $n$  个格子。
- 多组数据， $2 \leq n \leq 7$ ,  $1 \leq q \leq 10^5$ 。

# E - Matrix Lottery

## 题解

- 考虑一个无限长的抽奖结果序列，设格子  $(i, j)$  出现的下标集合为  $P_{i,j}$ ，对于一个询问集合  $S$ ，所求即  $E(\max_{(i,j) \in S} \min(P_{i,j}))$ ，这里  $E(x)$  表示  $x$  的期望。
- 直接做是有些麻烦的，我们可以先结合期望线性性、最大最小恒等式（即容斥）做一个转换，所求变为  $\sum_{T \subseteq S} (-1)^{|T|+1} E(\min_{(i,j) \in T} \min(P_{i,j}))$ ，这样就好算多了。相当于枚举每种格子集合  $T$ ，求首次出现其中任一格子的时间再乘上容斥系数。令  $X = \sum_{(i,j) \in T} p_{i,j}$ ，该期望时间为  $\frac{W}{X}$ 。
- 于是可以对着  $S$  里的格子做一个背包 DP， $X$  作为状态计算所有对应它的子集  $T$  的容斥系数之和，那么一个请求的复杂度是  $\mathcal{O}(n^2 W)$  的。

## E - Matrix Lottery

### 题解 (cont'd)

- 注意到最多有  $\mathcal{O}(2^{2n+2})$  种请求，远少于请求数上界，因此可以对请求做一次去重，根据  $S$  去重最好。
- 注意到一个请求的 DP 信息可以由另一个请求的 DP 信息增加至多  $n$  个格子转移过来，于是可以按 DFS 顺序处理 DP，均摊节省掉一个  $n$  的时间系数。
- 再一个优化是 DFS 换序，因为注意到对角线的影响总是最大的，放到最前面可以使后面加入物品变少，行列交替类似，然后就可以跑得很快了。
- 当然，由于  $n$  比较小，我们没有刻意卡常数，一些复杂度略高但是常数比较好的做法也是可以通过的。

## D - Christmas Tree

### 题意

- 给一棵  $n$  个点的有根树，第 1 个点作为根，选中第  $i$  个点有  $a_i$  的代价和  $b_i$  的收益，第  $j$  条边连接的两个点  $u_j$  和  $v_j$  都被选中时有  $c_j$  的额外代价。
- 给定  $m$ ，有  $q$  次询问  $r$  和  $k$ ，只在以  $r$  为根的子树里选点，使得每个选出的点都至少与一个未被选出的点相邻，且代价之和  $\bmod m = k$ ，最大化总收益，并计算最大化总收益的方案数模  $(10^9 + 7)$  的值。
- $1 \leq n \leq 100$ ,  $1 \leq m \leq 3 \times 10^4$ ,  $1 \leq a_i, b_i, c_j \leq 10^6$ 。

## D - Christmas Tree

### 题解

- 看上去是一个经典的树上背包问题。考虑  $f[u][i][j]$  表示在以  $u$  为根的子树里选点，选点状态为  $i \in \{0, 1, 2\}$ ，总代价为  $j$  时的最大收益，其中：
  - $i = 0$  表示不选  $u$ ；
  - $i = 1$  表示选  $u$ ，并且存在某个  $u$  的儿子没有被选；
  - $i = 2$  表示选  $u$ ，并且不存在某个  $u$  的儿子没有被选。
- 对于一个询问  $(r, k)$ ，如果  $r$  是根，那么最大收益就是  $f[r][0][k]$  和  $f[r][1][k]$  的最大值，否则最大收益还可以再对  $f[r][2][k]$  取最大值。

## D - Christmas Tree

### 题解 (cont'd)

- 对于  $u$ , 初始化  $f[u][i] = [0, -\infty, \dots, -\infty]$ , 每次转移  $u$  的一个儿子  $v$ :
  - $g[u][0] = f[u][0] \times (f[v][0] + f[v][1] + f[v][2]);$
  - $g[u][1] = f[u][1] \times (f[v][0] + (f[v][1] \ll c_j)) + f[u][2] \times f[v][0];$
  - $g[u][2] = f[u][2] \times (f[v][1] \ll c_j)。$
- $g[u]$  是辅助数组, 完成转移之后令  $f[u] \leftarrow g[u]$ 。这里  $V_1 + V_2$  表示两个(长度相同的)数组对位取  $\max$ ,  $V_1 \times V_2$  表示两个数组的循环  $(\max, +)$  卷积,  $V \ll c$  表示数组循环左移  $c$  位。

## D - Christmas Tree

### 题解 (cont'd)

- 特别地，如果  $u$  是叶子，由于不可能出现存在某个  $u$  的儿子没有被选的情况，要令  $f[u][1] = [-\infty, -\infty, \dots, -\infty]$ 。
- 然后将选  $u$  的代价和收益加入  $f[u][1]$  和  $f[u][2]$ 。
- 如果朴素地使用  $O(m^2)$  的循环卷积，复杂度会达到  $O(nm^2)$ ，无法通过。观察到子树里的有效状态不超过  $2^{\text{子树大小}}$ ，如果维护子树里的有效状态进行转移，复杂度是  $O(nm^2 / \log m)$ ，由于  $m$  比较大，还是难以通过。

## D - Christmas Tree

### 题解 (cont'd)

- 基于循环卷积合并两个背包的做法存在较大的瓶颈，但是往背包里加入一个物品的复杂度是  $O(m)$ ，这篇论文提供了一个通过反复递归加入物品来替代循环卷积的思路。应用到本题的具体过程如下：
- 定义过程  $DP(u, x)$  返回初始背包是  $x$  时在以  $u$  为根的子树里选点的结果背包数组，那么有  $f[u][i] = DP(u, [0, -\infty, \dots, -\infty])[i]$ 。
- 调用  $DP(u, x)$  时，如果  $u$  是叶子，令  $h = [x, [-\infty, -\infty, \dots, -\infty], x]$ ，将选  $u$  的代价和收益加入  $h[1]$  和  $h[2]$  之后返回  $h$  数组。

## D - Christmas Tree

### 题解 (cont'd)

- 否则  $u$  有儿子  $v_1, v_2, \dots, v_d$ , 定义辅助过程  $SDP(u, i, x)$  返回初始背包是  $x$  时处理了  $u$  的前  $i$  个儿子的转移的结果背包数组。调用  $DP(u, x)$  时, 先调用  $SDP(u, d, x)$  得到  $h$ , 将选  $u$  的代价和收益加入  $h[1]$  和  $h[2]$  之后返回  $h$  数组。调用  $SDP(u, i, x)$  时:
- 若  $i = 1$ , 调用  $DP(v_1, x)$  得到  $h$ , 返回  $[h[0] + h[1] + h[2], h[0], h[1] \ll c_j]$ 。

## D - Christmas Tree

### 题解 (cont'd)

- 否则先调用  $SDP(u, i - 1, x)$  得到  $h$ , 然后调用  $DP(v_i, h[0])$ 、 $DP(v_i, h[1])$ 、 $DP(v_i, h[2])$  分别得到  $t[0]$ 、 $t[1]$ 、 $t[2]$ , 令:
  - $s[0] = t[0][0] + t[0][1] + t[0][2]$ ;
  - $s[1] = t[1][0] + (t[1][1] \ll c_j) + t[2][0]$ ;
  - $s[2] = (t[2][1] \ll c_j)$ 。
- 返回  $s$  数组作为调用  $SDP(u, i, x)$  的结果。

## D - Christmas Tree

### 题解 (cont'd)

- 上述反复递归加入物品的过程实际上是不重不漏的，可以顺便统计出取到最大收益的方案数。接下来分析复杂度：
- 调用  $DP(u, x)$  的复杂度记为  $f(n)$ ， $u$  的各个儿子  $v_1, v_2, \dots, v_d$  的子树大小依次是  $n_1, n_2, \dots, n_d$ 。在递归过程中，除了  $v_1$  子树只有一次  $DP$  调用，其余子树都有三次  $DP$  调用，于是有

$$f(n) \leq f(n_1) + 3 \sum_{i=2}^d f(n_i) + O(m)$$

AC.

## D - Christmas Tree

### 题解 (cont'd)

- 考虑令  $n_1 \geq n_2 \geq \dots \geq n_d$ , 根据凸性可以分析在  $n_1 = \lceil (n-1)/2 \rceil$ ,  $n_2 = \lfloor (n-1)/2 \rfloor$ ,  $n_3 = \dots = n_d = 0$  时右边最大, 此时有

$$f(n) \leq f(\lceil (n-1)/2 \rceil) + 3f(\lfloor (n-1)/2 \rfloor) + O(m)$$

- 可得  $f(n) = O(n^2 m)$ 。

## D - Christmas Tree

### 题解 (cont'd)

- 现在还要对每个子树求解，实际上做重链剖分之后，对重链顶  $u$  调用  $DP(u, [0, -\infty, \dots, -\infty])$  就可以得到分别以重链上每个点为根的子树的结果，记总复杂度为  $g(n)$ ，则有

$$g(n) \leq f(n_1) + \sum_{i=1}^d g(n_i)$$

- 可得  $g(n) = O(n^2 m)$ 。

# K - Many Sprinklers!!!

## 题意

- 给定二维平面上  $n$  个圆  $(x_i, y_i, r_i)$ , 计算这些圆内部区域的并集的面积。
- $1 \leq n \leq 10^5$ ,  $-10^4 \leq x_i, y_i, r_i \leq 10^4$ 。

# K - Many Sprinklers!!!

## 题解

- 看上去是一个经典的圆面积并问题。
- 一个点  $(x, y)$  属于圆并，当且仅当存在  $i$  使得  $(x - x_i)^2 + (y - y_i)^2 \leq r_i^2$ ，也就是  $\min_{i=1}^n ((x - x_i)^2 + (y - y_i)^2 - r_i^2) \leq 0$ 。
- 考虑令  $(x - x_i)^2 + (y - y_i)^2 - r_i^2$  取到最小值的  $i$ ，由于  $x$  和  $y$  关于  $i$  是常数，也就是令  $-2x_i x - 2y_i y + (x_i^2 + y_i^2 - r_i^2)$  取到最小值的  $i$ 。

AC.

# K - Many Sprinklers!!!

## 题解 (cont'd)

- 取  $n$  个半空间  $z \leq -2x_i x - 2y_i y + (x_i^2 + y_i^2 - r_i^2)$  的交集得到一个上凸壳，每个面投影到  $xOy$  平面的区域就是在对应  $i$  取到最小值的所有  $(x, y)$ 。
- 考虑对偶形式，就是取  $n$  个点  $(-2x_i, -2y_i, x_i^2 + y_i^2 - r_i^2)$  的下凸壳，需要使用高效的三维凸包算法求解，一个相对易于实现的算法是随机增量法，期望复杂度是  $O(n \log n)$ 。
- 为了避免求解下凸壳时出现退化情况，可以事先在平面足够远处的四个角上分别放一个足够小的圆。

# K - Many Sprinklers!!!

## 题解 (cont'd)

- 取出下凸壳之后，考虑凸壳上一个点  $i$ ，其在凸壳上的相邻点就是在半空间交里与平面  $i$  相邻的平面，联立方程可以得到交线在  $xOy$  平面上的投影，也就是两个区域的分割线，对这些分割线用  $O(n \log n)$  的半平面交算法求解得到一个凸多边形区域，就是在  $i$  取到最小值的所有  $(x, y)$ 。
- 根据三维凸包的性质可以知道所有凸多边形区域的总边数是  $O(n)$ ，因此只需要再分别对每个凸多边形区域计算与对应圆的交集的面积，这也是一个经典问题。总复杂度是  $O(n \log n)$ 。

# THANKS!

AC.NOWCODER.COM