2020牛客暑期多校训练营（第九场）

乐清市知临中学

牛客竞赛
AC.NOWCODER.COM

A-Groundhog and 2-Power Representation

- 朴素的递归思想，模拟从里到外去括号的过程，加上高精度即可AC。

B-Groundhog and Apple Tree

- **推论:在根节点恢复HP够了再出发一定更优**
- **答案即补足访问所有节点的过程中HP最小值**
- 设访问子树 i 需要的总HP为 a_i ，访问过程中最坏情况需要HP为 b_i
  由于访问u的子树所需要的总HP a_u 是一个定值，故我们只考虑子树访问顺序对于 b_u 的影响
- 考虑排布子树的访问顺序，得到
- $b_u = \max\left\{\min_{1\le i\le cnt(son)}\left\{\left(\sum_{j=1}^{i-1} a_j\right) + b_i\right\}\right\}$
- 考虑对儿子访问排序，访问顺序应满足相邻交换最优原则
- 设有相邻对 i , j $(i = j - 1)$，若不交换 i， j 更优，则有

$$\min\{b_j, a_j + b_i\} < \min\{b_i, a_i + b_j\}$$

B-Groundhog and Apple Tree

- 我们将min, max转换为简单的逻辑表达式，即
- $\left(b_i > b_j \text{ or } 0 > \mathrm{a_j}\right) \text{ and } \left(a_i > 0 \text{ or } a_i + b_j > a_j + b_i\right)$
- 若 $a_i > 0$ 且$a_j < 0$ 肯定成立，所以我们先对于 a_i 是否小于0讨论，将儿子的集合分成两部分 A, B
- 其中A为所有满足$a_i \geq 0$的所有a_i形成的集合，B为所有满足$a_i < 0$的所有a_i形成的集合。
- 这样把 A 中的某个元素排在 B 中的另一个元素之前一定满足相邻交换原则
- 然后我们对于 A 内的顺序考虑，一定有比较式后项成立，只需满足 $b_i > b_j$
- 我们再对于 B 内的顺序考虑，一定有比较式前项成立，只需满足 $a_i - b_i > a_j - b_j$
- 综上，问题得到了解决

C-Groundhog and Gaming Time

- 线段的交取决于最大的左端点以及最小的右端点，同时维护两个东西比较困难。
- 所以我们先按照线段左端点从大到小排序，那么排序后的线段的交取决于最小的右端点，以及第一个被选择的线段的左端点。
- 考虑到直接维护右端点比较麻烦，所以考虑在一开始就钦定一个点 X 作为最小的右端点。
- 所有右端点大于等于 X 的线段都可以选择，反之不能选择。
- 其次被选择的线段中至少有一个线段的右端点等于 X 那么这个方案就是合法的。

C-Groundhog and Gaming Time

- 所以可以写出一个 $O(n^2)$ 的 dp。
- dp(i, j, 0/1) 代表前 i 个线段中钦定的 X 为 j，是/否 有一个线段的右端点为 X。
- dp(i, j, 0/1) = dp(i - 1, j, 0/1) * 2   (j < L[i] or j > R[i])
- dp(i, j, 0/1) = dp(i - 1, j, 0/1) * 2 + (j – L[i]) ^ 2   (L[i] <= j < R[i])
- dp(i, R[i], 0) = dp(i - 1, R[i], 0)
- dp(i, R[i], 1) = dp(i - 1, R[i], 0) + dp(i - 1, R[i], 1) * 2 + (R[i] – L[i]) ^ 2
- 最后使用线段树或其他数据结构就可以将该 dp 优化到 $O(n\log n)$。

D-Groundhog and Golden Apple

- **题意：**
- 给一棵大小为 n 的树，第 i 个节点上站着一个编号为 i 的人，第 j 条边只允许编号在 $[L_j, R_j]$ 间的人通过，第 i 个人有 K_i 次机会强行通过一条边，分别求每个人可以到达的点的个数，$2 \le n \le 100000$，$0 \le K_i \le 1$。
- 只要存在一种合法路径，就认为这个点是可到达的

D-Groundhog and Golden Apple

- 考虑第 i 个人时，如果 K_i 等于 0 ，那么答案就是 i 号点所在的连通块的大小；如果 K_i 等于 1 ，则答案还需加上所有与该连通块相邻的连通块的大小和。
- 考虑如何加入一条边并维护信息:
- 可以直接用并查集来维护联通块以及大小
- 以一个点作为整棵树的树根，对于一个连通块，把相对树根深度最浅的点作为该连通块的顶点
- 对于相邻联通块分，按照位置，分为父亲联通块，儿子联通块
- 在合并时额外维护儿子联通块大小之和，加上唯一的父亲联通块即可

D-Groundhog and Golden Apple

- 连接一条边容易，但是直接删除一条边比较困难，所以可以考虑**线段树分治**简化回撤问题
- 并查集回撤可以使用**按秩合并并查集**解决
- 时间复杂度: $O(n\log^2 n)$。
- 也可以使用Splay维护括号序列的方法维护此题，这里就不做赘述

E-Groundhog Chasing Death

- 一个比较显然的思路是，对给出的数分解质因数，然后对于每个质因数分别讨论其幂次。那么问题就从求乘积问题转化为$O(\log x + \log y)$个子问题。
- 对于每个子问题，在幂次上，形如给出两个数x', y'，求$\sum_{\mathrm{i}=a}^{b} \sum_{j=c}^{d} \min\{x'i, y'j\}$
- 那么可以枚举这个最小值是多少，是在$x'i$取到还是在$y'j$取到，并可以$O(1)$求出每种情况的方案数。
- 注意处理好$x'i = y'j$的情况，注意如果要对幂次取模，按照欧拉/费马小定理，模数应该取$\varphi(998244353)$。
- 令n=max{b,d}，这种算法的最终复杂度是$O(n(\log x + \log y))$的，通过本题已经绰绰有余。
- 当然，如果你追求更优秀的算法，可以使用类欧几里得问题的算法做到更优秀的复杂度（如果使用这种做法，复杂度瓶颈在分解质因数，这里不详细展开）。

牛客竞赛
AC.NOWCODER.COM

F-Groundhog Looking Dowdy

- 由于要最小化最大值和最小值的差值，因此我们可以把所有衣服按照dowdiness从小到大排个序。
- 排序之后，设最终选出的m件衣服最小覆盖区间为[L,R]，则答案为downdiness[R]-downdiness[L]
- 则一个合法的区间至少需要包含m种不同的日期
- 可以对于每个L求出最小的合法的R，这就转化为一个简单的尺取问题了。
- 若使用基数排序，可以做到在$O(\sum k_i)$时间求解

G-Groundhog Playing Scissors

- 首先，容易发现旋转多边形和旋转直线是没有本质差别的，设直线的垂线与x轴夹角为$\theta, \theta \in [0,2\pi)$，直线在凸包内的长度为$\mathrm{f}(\theta)$，所以问题变为求：$\mathrm{f}(\theta) > \mathrm{L}$ 的θ范围
- 我们发现，随着直线的旋转，和直线相交的凸包的两条边也会定向改变。可以用两个指针维护，求出$O(n)$个θ的区间，每个区间内相交的两条边是相同的（可以忽略凸包的顶点，因为落在定点上的概率可以算作零）。这样就把原问题化为$O(n)$个子问题
- 发现每个子问题中的$\mathrm{f}(\theta)$是一个严格的单峰函数，所以可以用三分法求出极值点，再在两侧二分出θ的边界。复杂度是$O(nt)$的，其中t是三分/二分次数，但是由于有大量计算几何的浮点运算，所以常数较大。

G-Groundhog Playing Scissors

- 还有一种比较暴力的解法，就是直接暴力模拟直线旋转，把整个2π分为很多个小角度，然后每次用均摊 $O(1)$ 的复杂度求出和哪两条边相交。如果把旋转次数开到很大，可以近似求出解。不过由于是比较近似的写法，可能需要调一些参数。事实上，这种算法的计算几何运算较少，精度和常数并不一定弱于前一种。
- 由于精度要求只有四位小数，也许还存在一些乱搞算法可以通过此题，这里不做讨论。

H-Groundhog Speaking Groundhogish

- 考虑简单的判定子序列的算法：
- 用s贪心地匹配p中最前面的对应位置的字符，能匹配就匹配，最后得到匹配位置为关键点。
- 合法情况下，每两个关键点之间的位置所放的字符不能与下一个关键点字符相同
- 所以视构造s的过程为：先确定关键点，然后在关键点之间插入若干字符的过程
- 关键点之间的方案，都是允许插入$m-1$或m种字符（允许插入多次），和具体取值无关。

H-Groundhog Speaking Groundhogish

- 而对于每种字符带权的问题，把$m-1$或m种字符的选择 改为 可选字符的权值总和 即可
- 因此，不妨设$v_i=\sum_{j\neq s_i} a_j$，问题就变为一个分组的完全背包问题：第i个物品的权值是v_i，总权值是所有物品的权值乘积。
- 可以直接用动态规划求解，复杂度为$O(nk)$。

H-Groundhog Speaking Groundhogish

- 而这种类型的背包是显然可以使用生成函数优化的。
- 对于每个物品的完全背包问题，可以列出其对应的普通型生成函数是$F_i(x) = \sum_{j=0} {v_i}^j x^j = \frac{1}{1-v_i x}$。
- 答案就是$\sum_{i=0}^{k} [x^i] \prod F_i(x)$
- 可以先分治NTT求出倒数积再求逆求出$\prod F_i(x)$，若认为n, k同阶，复杂度$O(n\log^2 n)$。

I-The Crime-solving Plan of Groundhog

- 把当前的数字拆成4个数 $a, b, c, d(a \leq b \leq c \leq d)$,那么我们有两种决策：两位数×两位数，或者三位数×一位数。
- $(10a + d) \cdot (10b + c) = 100ab + 10ac + 10bd + cd$
- $(100b + 10c + d) \cdot a = 100ab + 10ac + ad < (10a + d) \cdot (10b + c)$
- 同理类推，
- 可以证明留一个最小的正整数作为第一个数，剩下的所有数字排成最小的数作为第二个数时，答案取到最小值。
- 注意高精度细节和“正整数”、“整数”的区分，以及前导0的处理。

J-The Escape Plan of Groundhog

- 如果需要做到 $O(n^3)$，套路一般都是：枚举上下行边界，对于列扫一遍，用前缀和等维护。
- 那么这里四条边上都要为1，那么枚举上下边界后，肯定要找一段在这两行都是1的连续的列区间。然后在这个区间里找。
- 枚举每一列，如果这一列也都是1，就可以统计进去。用一个前缀和维护，在原矩阵中为0则当做-1，否则当做1。那么每次找到合法的列，查询前面的前缀和是否有和它相差1以内的，计入答案。然后把自己的前缀和加入统计。
- 注意前缀和不能算上边界的1。
- $O(n^2m)$ 或 $O(nm^2)$

K-The Flee Plan of Groundhog

- Solution#1：
  - 二分一个时间 t，然后判断在 t 内 Groundhog 是否会被 Orange 追上。
  - 以 Orange 所在的寝室为根建树，从 1 到 n 枚举所有 Groundhog 能够到达的点，然后判断在走的过程中会不会被追上即可。
  - 时间复杂度: $O(n \log n)$
- Solution#2：
  - 由于 Orange 的决策是单一的，所以对于每个点，如果 Orange 想去则一定会径直走过去不会绕弯路。因此，可以直接以 Orange 为起点，dfs 出 Orange 到每个点的最短时间。在此之后，再以 Groundhog 为起点 bfs，能到达的最远点即为二人时间第一次重合的点。
  - 时间复杂度:$O(n)$

L-The Shopping Plan of Groundhog

- 下面，将每条边的左侧和右侧都算作一个“半边”。
- 因为不带修，容易联想到可以离线处理问题。一种自然的思路是，将共计 $2n-2$ 个半边按照礼物价格排序，对于每一种礼物，计算在只走那些礼物价格小于等于它的半边时，两点之间的最短距离，并加上这种礼物的价格作为答案的候选值来更新答案（如果边权比较大，有可能两点是不连通的，此时不更新答案即可）。动态的维护计算最短路用的边权即可。这种算法的正确性是显然的，但是对于每个半边都要重新计算所有询问，时间复杂度是$O(n^2)$或者$O(n^2\log n)$的（具体取决于计算路径长度的复杂度），难以通过。

L-The Shopping Plan of Groundhog

- 考虑用数据结构解决问题。这里提供一种优化思路：可以把所有点对应到它的dfs序，这样每组询问就对应二维平面上的一个点，而每条边影响的范围就是二维平面上的$O(1)$个矩阵。
- 由于维护边权的时候，每次只会更新一条边的边权，所以我们可以每次对二维平面上的$O(1)$个矩阵进行加减，而需要额外加上的礼物价格就对应一个矩阵全局的加减修改。
- 这样，我们把更新转化为了二维平面的矩阵加减操作。
- 对于答案的计算，可以维护某些点的历史最值来实现。std使用了单次矩阵加减操作复杂度为$O(\sqrt{n})$的KD树实现，总复杂度为$O(n\sqrt{n})$。这样实现的常数较大，可能需要一些卡常。

Thanks