

IOI2013 中国国家集训队作业

Codeforces 题目泛做

成都七中 李凌霄

January 14, 2013

Codeforces ID: csimstu

Contents

1	Volume I	4
1.1	7E. Defining Macros	4
1.2	8D. Two Friends	4
1.3	8E. Beads	4
1.4	10E. Greedy Change	5
1.5	15E. Triangles	6
1.6	17E. Palisection	6
1.7	17C. Balance	7
1.8	19E. Fairy	7
1.9	23D. Tetragon	7
1.10	23E. Tree	7
2	Volume II	9
2.1	26E. Multithreading	9
2.2	28D. Do not fear, DravDe is kind	9
2.3	30D. Kings Problem?	9
2.4	30E. Tricky and Clever Password	10
2.5	32E. Hide-and-Seek	10
2.6	35E. Parade	10
2.7	36E. Two paths	10
2.8	37E. Trial for Chief	11
2.9	39C. Moon Craters	11
2.10	39A. C*++ Calculations	11
3	Volume III	12
3.1	39E. What Has Dirichlet Got to Do with That?	12
3.2	40E. Number Table	12
3.3	43E. Race	12
3.4	44J. Triminoes	12
3.5	45G. Prime Problem	13
3.6	45E. Director	13
3.7	46F. HerculePoirot Problem	13
3.8	47E. Cannon	13
3.9	49E. Common ancestor	14
3.10	51F. Caterpillar	14

4	Volume IV	15
4.1	53E. Dead Ends	15
4.2	57D. Journey	15
4.3	226E. Noble Knights Path	15
4.4	217D. Bitonix' Patrol	16
4.5	67E. Save the City!	16
4.6	67C. Sequence of Balls	16
4.7	70D. Professors task	16
4.8	70E. Information Reform	17
4.9	156E. Mrs. Hudson's Pancakes	17
4.10	105D. Entertaining Geodetics	17
5	Volume V	18
5.1	193D. Two Segments	18
5.2	75E. Ships Shortest Path	18
5.3	76F. Tourist	18
5.4	76A. Gift	19
5.5	77E. Martian Food	19
5.6	79D. Password	19
5.7	81E. Pairs	20
5.8	82E. Corridor	20
5.9	83E. Two Subsequences	20
5.10	85E. Guard Towers	21
6	Volume VI	22
6.1	86E. Long sequence	22
6.2	89D. Space mines	22
6.3	91D. Grocer's Problem	22
6.4	93D. Flags	23
6.5	97C. Winning Strategy	23
6.6	97A. Domino	24
6.7	98D. Help Monks	24
6.8	98C. Help Greg the Dwarf	25
6.9	191D. Metro Scheme	25
6.10	164D. Minimum Diameter	26
7	Volumn VII	27
7.1	150E. Freezing with Style	27
7.2	101E. Candies and Stones	27
7.3	103E. Buying Sets	27
7.4	105E. Lift and Throw	28
7.5	107D. Crime Management	28
7.6	113D. Museum	28
7.7	115D. Unambiguous Arithmetic Expression	29
7.8	120I. Luck is in Numbers	29
7.9	123E. Maze	29
7.10	125E. MST Company	30
8	Volumn VIII	31
8.1	193E. Fibonacci Number	31
8.2	145D. Lucky Pair	31
8.3	132E. Bits of merry old England	31
8.4	138D. World of Darkraft	32
8.5	140F. New Year Snowflake	32
8.6	147B. Smile House	32
8.7	152D. Frame	33
8.8	183D. T-shirt	33
8.9	217E. Alien DNA	34
8.10	135E. Weak Subsequence	34

9	Volume IX	35
9.1	163D. Large Refrigerator	35
9.2	167E. Wizards and Bets	35
9.3	232D. Fence	35
9.4	175E. Power Defence	36
9.5	176D. Hyper string	36
9.6	178F. Representative Sampling	36
9.7	178E. The Beaver's Problem II	37
9.8	180B. Divisibility Rules	37
9.9	185D. Visit of the Great	38
9.10	187D. BRT Contract	38
10	Volume X	39
10.1	176E. Archaeology	39
10.2	196D. The Next Good String	39
10.3	198E. Gripping Story	39
10.4	200E. Tractor College	40
10.5	200A. Cinema	40
10.6	201E. Thoroughly Bureaucratic Organization	41
10.7	201D. Brand New Problem	41
10.8	204E. Little Elephant and Strings	41
10.9	207B. Military Trainings	42
10.10	207A. Beaver's Calculator	42
11	Volume XI	44
11.1	167D. Wizards and Roads	44
11.2	209C. Trails and Glades	44
11.3	212B. Polycarpus is Looking for Good Substrings	45
11.4	212D. Cutting a Fence	45
11.5	212C. Cowboys	45
11.6	213E. Two Permutations	46
11.7	217C. Formurosa	46
11.8	229E. Gifts	47

1 Volume I

1.1 7E. Defining Macros

题目大意

给定 $n(n \leq 100)$ 个 `#define name value` 的语句, 其中 `value` 包括 `+-*/()`, 判断另一个语句是否会产生歧义。

算法简述

每个变量设 4 种状态: 有歧义 (S), 多段相加减 (P), 多段相乘除 (M), 一段 (Q)。对于 $A ? B$, 如果 A 或 B 为 S 那么整个结果也为 S。否则分情况讨论:

1. $A+B$: 合法;
2. $A-B$: B 不能是 P;
3. $A*B$: A, B 均不能是 P;
4. A/B : A 不能是 P, B 不能是 P, Q;

用递归可以简化编程复杂度。

1.2 8D. Two Friends

题目大意

平面上三个点 $H(home), S(shop), C(cinema)$ 。两人同时从 C 出发, 路线分别为 $C \rightarrow H$ 和 $C \rightarrow S \rightarrow H$ 。两人均有一个路程的上限值。求两人最长公共路线。

算法简述

如果两人共同经过 $shop$, 那么答案就是 $dist(C, S) + dist(S, H)$, 只要不超过两人路程上限值。否则, 二分答案 x , 那么两人同行的终点一定在半径为 x 的圆内, 设为 p 。同理, p 也要在以 $home, shop$ 为圆心的两个圆内 (对应圆的半径可以根据 x 唯一确定)。于是变成了判断 3 个圆是否有交, 枚举两个圆算交点、各个圆的圆心, 看是否满足。

1.3 8E. Beads

题目大意

将 $1-2^{n-1}$ 的数分组, 同一组的数可以通过二进制下逆序和 01 取反得到, 然后每组保留最小的数。求第 k 大的数。 $n \leq 50, k \leq 10^{16}$ 。

算法简述

先二分答案, 转化成计算不超过 k 的有多少个满足条件的数: 用 x 表示对 x 01 取反, $rev(x)$ 表示逆序。原问题对 x 所属组的定义是 $\{x, -x, rev(x), rev(-x)\}$ 。定义子问题 1 中的组为: $x, rev(x)$, 子问题 2 的组为: $x, rev(-x)$ 。

结合数位 dp 的一般思想和本题的特殊性, 可以从两边往中间进行 dp。具体分类讨论如下:

原问题:

1. $0 \cdots 0 \Rightarrow$ 子问题 1
2. $0 \cdots 1 \Rightarrow$ 子问题 2
3. $1 \cdots 0 \Rightarrow$ impossible

4. $1 \cdots 1 \Rightarrow \text{impossible}$

子问题 1:

1. $0 \cdots 0 \Rightarrow \text{子问题 1}$

2. $0 \cdots 1 \Rightarrow \text{全部合法}$

3. $1 \cdots 0 \Rightarrow \text{impossible}$

4. $1 \cdots 1 \Rightarrow \text{子问题 1}$

子问题 2:

1. $0 \cdots 0 \Rightarrow \text{全部合法}$

2. $0 \cdots 1 \Rightarrow \text{子问题 2}$

3. $1 \cdots 0 \Rightarrow \text{子问题 2}$

4. $1 \cdots 1 \Rightarrow \text{impossible}$

于是可以设计 3 个相似的 dp, 每个 dp 解决不超过二分的答案的合法数的个数。

1.4 10E. Greedy Change

题目大意

给定一个货币系统, 求最小的 w , 使得贪心 (从高价值货币开始能选就选) 使用的货币不是最优解 (即个数不是最少的)。 $n \leq 400$ 。

算法简述

专门有一篇论文针对这题《A Polynomial-time Algorithm for the Change-Making Problem》, 下面仅是对该论文的简要重述。大致思想是找到 $O(n^2)$ 个可能出现反例的 w , 然后每个 $O(n)$ 验证。

首先是几个定义。

定义 1.4.1 定义一个货币系统为 n 维向量 $C = (c_1, c_2, \dots, c_n)$ 。定义 x 在该货币系统下的表达为 $V = (v_1, v_2, \dots, v_n)$, 即 $V \cdot C = x$ 。定义 $|V|$ 为 V 用的货币个数。

定义 1.4.2 x 的贪心表达 $G(x)$ 为 x 的表达中字典序最大的。

显然, $x < y \Rightarrow G(x) < G(y)$ 。

定义 1.4.3 x 的最小表达 $M(x)$ 为最小大小的表达中字典序最大的。

于是, 目标变成了找到最小的 w , 使 $M(w) \neq G(w)$ 。

定理 1.4.1 称 U 是贪心的, 当 $U = G(U \cdot C)$; 是最小的, 当 $U = M(U \cdot C)$ 。那么:

1. 如果 $U \subset V$, 且 V 是贪心的, 那么 U 是贪心的。
2. 如果 $U \subset V$, 且 V 是最小的, 那么 U 是最小的。

证明 1.4.1 注意到向量加法对字典序的保序性:

$$A \leq B \Leftrightarrow A + D \leq B + D$$

设 U' 为 $U \cdot C$ 的任意一个表达。于是

$$\begin{aligned} U' \cdot C &= U \cdot C \\ (V - U + U') \cdot C &= V \cdot C \\ V - U + U' &\leq V \\ U' &\leq U \end{aligned}$$

对于最小的证明, 只需将 \leq 替换成 \sqsubset 即可。

下面开始构造可能的反例集合。设 w 为最小的使 $G(w) \neq M(w)$ 的最小正整数。一个重要的结论是 $G(w)$ 和 $M(w)$ 中非 0 项的交集为空。否则可以同时减去某个 c_i 仍然满足 $G(w) \neq M(w)$, 故 w 不是最小的了。

令 $M(w) = (m_1, m_2, \dots, m_n)$, i, j 为第一个和最后一个非 0 项的下标。由于 $M(w) < G(w)$, 因此 $G(w)$ 第 i 项为 0, 前 $i-1$ 项中有一项非 0。

定理 1.4.2 $M(w)$ 与 $G(c_{i-1}-1)$ 相比, 第 1 到 $j-1$ 项相同, 第 j 项大 1, 其余项全为 0。

证明 1.4.2 因为 $G(w)$ 前 $i-1$ 项有一个非 0, 故 $w \leq c_{i-1}$ 。另一方面, 将 $M(w)$ 第 j 项减 1 得到 $w - c_j$ 的最小且贪心表达。对比 $M(w - c_j) = G(w - c_j)$ 与 $M(w)$, 可以发现前 $i-1$ 项都为 0, 故 $w - c_j < c_{i-1}$ 。

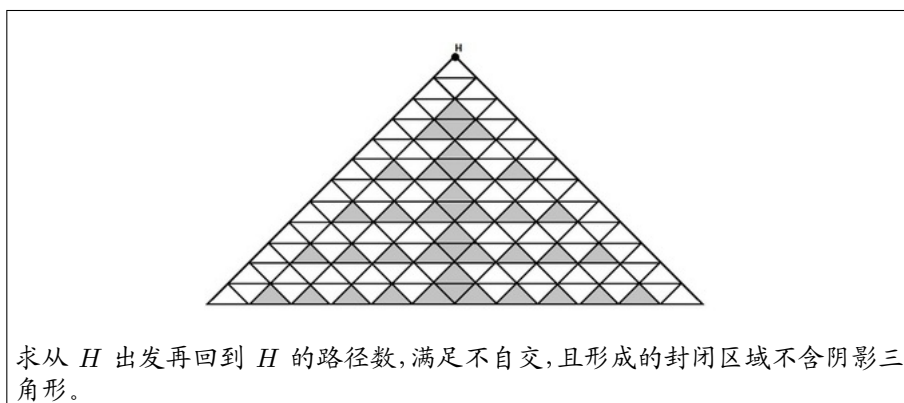
令 $V = (v_1, v_2, \dots, v_n) = G(c_{i-1}-1)$ 。由于 $c_{i-1}-1 \geq c_i, v_i \neq 0$, 可以将 v_i 和 m_i 同时减 1 得到 $G(c_{i-1}-1-c_i)$ 和 $G(w-c_i)$ 。有前面的不等式可知, $G(c_{i-1}-1-c_i) < G(w-c_i)$, 将 i 位置的 1 加回去不影响相对字典序。故 $V < M(w)$ 。

另外, 若将 m_j 减 1, 得到 $G(w - c_j)$ 。因为 $w - c_j \leq c_{i-1} - 1$, 有 $G(w - c_j) \leq V$ 。综合可得 $G(w - c_j) \leq V < M(w)$ 。而 $G(w - c_j)$ 与 $M(w)$ 仅在 j 位置上差 1, 故 V 只能在 j 位置之后与 $M(w)$ 不同。又因为 $m_{j+1}, m_{j+2}, \dots, m_n$ 全为 0, V 只能在 j 位置比 $M(w)$ 小。而 $G(w - c_j) \leq V$, 所以 $m_j - 1 \leq v_j$ 。因此, m_j 只能取 $v_j + 1$ 。

通过枚举 i, j , 就能算得 $M(w)$, 以及 w 。再 $O(n)$ 求出 $G(w)$ 。总复杂度是 $O(n^3)$ 。

1.5 15E. Triangles

题目大意



算法简述

显然一条满足要求的路径必然是左边走一圈, 回到最靠上阴影三角形的上定点, 然后再在右边走一圈。仔细观察可以发现很多递归子问题, 一个是往中间走然后出来, 一个是往下走然后回去。设前者为 $f[i]$, 后者是 $g[i]$, 不难发现递推式:

$$\begin{aligned} f[1] &= 1; f[i] = 2f[i-1] + 3; \\ g[n/2] &= 2; g[i] = 2 + 2f[i] + f[i] \cdot g[i+1]; \end{aligned}$$

答案就是 $2(g[1]^2 + 1)$ 。

1.6 17E. Palisection

题目大意

给你一个长度 $n (1 \leq n \leq 2 \times 10^6)$ 的只由小写字母组成的字符串 s , 问 s 的所有连续回文子串中, 有几对是有公共部分的。

算法简述

先用 manacher 算法求出每个位置的最大回文半径。直接统计讨论比较麻烦, 可以采用补集转化思想, 求不相交的对数。对每个位置求出 $f_1[i]$ 表示以位置 i 开始的回文段数, $f_2[i]$ 表示以

位置 i 结束的回文段数。所以不合法的答案可以利用前缀和, 由 f_1, f_2 拼凑。补集转化思想在这里非常精妙。

1.7 17C. Balance

题目大意

给定一个长度为 $n (1 \leq n \leq 150)$ 的 abc 字符串, 每次可以将相邻两个字符 c_1, c_2 进行以下变换: $c_1 c_2 \leftarrow c_1 c_1$ 或 $c_1 c_2 \leftarrow c_2 c_2$ 。求进行任意次变换后可能的字符串总数。

算法简述

可以发现, 原字符串中序是不变的。故可设计以下状态: $f[i][j][n_a][n_b]$ 表示构造到第 i 个字符, 目前考察原字符串第 j 个字符, a, b 的数目。转移枚举下一个字符, 为了避免重复, 强制下一个字符由原串中 j 之后最近的来产生。总复杂度为 $O(n^4)$ 。

1.8 19E. Fairy

题目大意

对于一个 n 个点 m 条边的无向图 ($n, m \leq 10^4$), 问删除哪条边后变成二分图。

算法简述

首先建一个生成树森林, 然后进行染色。对于每条非树边, 如果两端点的颜色不同, 则称为 $errEdge$ 。在树上连接 $errEdge$ 两端点的路径称为 $errPath$ 。

首先考虑删除非树边: 如果 $n(errEdge) = 0$ 那么都可以删除; 如果 $n(errEdge) = 1$ 则必须删除这条 $errEdge$; 如果 $n(errEdge) > 1$ 那么都不能删除。

然后考虑树边。对每条树边记 a_i, b_i 分别表示经过这条边的 $errPath$ 数目, 非 $errPath$ 数目。于是可以发现, 第 i 条树边是答案的充要条件是 $n(errEdge) = 0$ or ($a_i = n(errEdge)$ and $b_i = 0$)。用各种数据结构都可以维护每个边 a_i, b_i 的值。

1.9 23D. Tetragon

题目大意

一个有 3 个等边的凸四边形, 告诉你 3 条等边的中点, 求这个凸四边形。

算法简述

一个想法是三分半径 r 。设 3 个已知点为 A, B, C 。计算 3 个圆两两相交的交点。如果存在 P, Q , P 是圆 B 与 A 的交点, Q 是圆 C 与 A 的交点, 且 PQ 经过 A , 那么就能构造出合法的凸四边形。而 PQ 到 A 的距离关于 r 是单峰的。但数据组数太多, 会 TLE。

另一个方法是枚举中间那个点 p_1 (即另外两个已知边长边夹在中间的边的终点), 那么有另外两个四边形上的点在 p_1 与 p_2, p_2 与 p_3 的中垂线上。解二元一次方程即可。

1.10 23E. Tree

题目大意

给定一个树, 可以删除任意数目的边, 使各个连通块大小的积最大。节点个数不超过 700。

算法简述

可以证明,任何一个最优解的残留图中不存在一条长度 ≥ 3 的路径。否则,将中间那条边割开,设分成的两部分大小为 $a, b (a, b \geq 2)$, 于是 $1 \geq 1/a + 1/b \Rightarrow ab \geq a + b$ 。有了这个原则,一个点只有三种情况:

1. 这个点与它一个儿子以及那个儿子的儿子同属一个连通块
2. 这个点与它多个儿子同属一个连通块
3. 这个点与它的父亲同属一个连通块

这可以用经典的树形 dp+ 背包解决。另外,本题需要使用高精度。

2 Volume II

2.1 26E. Multithreading

题目大意

有 $N(N \leq 100)$ 个子过程:

```
repeat n_i times
  y_i := y
  y := y_i + 1
end repeat
```

其中 y 为全局变量, 而 y_i 相互独立。每次可以命令一个子程序运行当前行并跳到下一行。设计一个长度为 $2N$ 的操作序列, 使得最后 $y = W(|W| \leq 10^9)$ 。 $n_i \leq 1000$ 。

算法简述

构造题。首先, 如果不满足 $1 \leq W \leq \sum n_i$ 一定无解。否则有以下三种有解情况:

1. $N = 1, W = n_1$ 。这种情况很显然。
2. $N \geq 2, W \geq 2$ 。可以设计一个 $W = 2$ 的序列: 1, (3 到 n 全部用掉), $(n_2 - 1)$ 次 2, 1, 2, $4(n_1 - 1)$ 次 1, 2。然后对于多出来的 $W - 2$, 将中间的循环移到最后即可。
3. $N \geq 2, W = 1$ 且存在 $n_i = 1$: i, everything else, i.

2.2 28D. Do not fear, DravDe is kind

题目大意

有 n 个人按 $1 \rightarrow n$ 从左到右排成一排。每个人有四个属性 (v, c, l, r) 。第 i 个人要求他左边人的 c 之和恰好为 l , 右边人的 c 之和恰好为 r 。现在要剔除一部分人, 使每个人的需求得到满足, 且 v 之和最大。 $n \leq 10^5$ 。

算法简述

如果选定一个人 k , 那么可以确定最终队列中所有人的 c 之和必为 $(c_i + l_i + r_i)$ 。故可以按每个人的 $c + l + r$ 分组, 不同组之间互不干涉, 这样一来限制就只有 l 了。于是转化成了背包模型。不同的是转移总数只有 $O(n)$ 次, 有用的状态很少。于是可以对状态进行 hash 再按常规的背包 dp 来做。

2.3 30D. Kings Problem?

题目大意

有 $n + 1$ 个平面上的点, 前 n 个都在 x 轴上。现在求从 k 号点出发, 经过每个点至少一次的最短路径。

算法简述

如果当前在 $n + 1$ 号点, 那么之后一定是往还没访问过的最左或最右走, 再到达另一极端。于是可以枚举经过 $n + 1$ 之前的访问区间 $[l, r]$ 。可以发现, $[l, r]$ 必定包含最左端点或最右端点才能最优, 故可用区间只有 $O(n)$ 个, 而验证是 $O(1)$ 的。

2.4 30E. Tricky and Clever Password

题目大意

给定一个加密后的奇数长度的回文串: $A+prefix+B+middle+C+suffix$, 其中 $prefix, suffix$ 为原串的首尾等长的子串, $middle$ 为长度是奇数的回文串。求一种破解方法, 使 $prefix+middle+suffix$, 即原串, 的长度最大。加密后的字符串长度不超过 2×10^5 。

算法简述

首先预处理出每个位置的最大回文半径。可以证明, 若 $middle$ 以某个位置为中心, 那么当 $middle$ 恰好为以那个位置为中心的最长回文串, 得到的解不会更差。再 KMP 预处理另一个数组 $f[i]$ 表示与字符串 $[i, n-1]$ (n 为长度) 逆序匹配的最靠前位置。于是可以枚举每个 $middle$, 注意到当 i 增加时, $f[i]$ 不增, 所以可以二分位置 k 使 $f[k]$ 在 $middle$ 之前且 k 最靠左。整个算法的复杂度为 $O(n \log n)$ 。

2.5 32E. Hide-and-Seek

题目大意

平面上有一堵墙和一面镜子, 以及两个可以视作点的人。问两人是否可以看到对方。只要有公共点即视作相交。

算法简述

最简单的情况: 如果两人连线没有被挡着, 那么可见; 否则, 如果可行一定是通过了镜子。另外, 如果镜子挡到视线那么不能看到对方。通过对称可以确定镜子上一个必然经过的点, 判断这个点与两人的初始位置是否无障碍即可。

2.6 35E. Parade

题目大意

有 $n(n \leq 10^5)$ 个房子坐落在 x 轴上, 每个房子可以视作有 3 个属性的矩形: l, r, h , 分别表示房子的左、右边界以及高度。求一个封闭区域面积最小的轮廓, 将所有房子包含在内。

算法简述

离散化端点, 然后求出每相邻两个离散点的最大高度, 扫一遍得到答案。

2.7 36E. Two paths

题目大意

给定一个无向图, 求两条欧拉路径将所有边无重复地覆盖。点数, 边数不超过 10000。

算法简述

分情况讨论如下:

1. 连通块数 ≥ 3 , 无解
2. 连通块数 = 2, 在两个连通块内分别求欧拉路径。

3. 连通块数 = 1, 如果有 0 个或 2 个奇度数的点, 那么存在一条欧拉路径, 任意将之分成两段即可; 如果有 4 个奇度数的点, 那么任选两个添加一条辅助边, 求出欧拉路径后以这条边为分割即可求出答案。

2.8 37E. Trial for Chief

题目大意

给定一个 $n \times m$ ($n, m \leq 50$) 的黑白棋盘, 每次可以将一个四连通块反色。求最小操作次数使整个棋盘变为白色。

算法简述

不难发现, 最优方案一定是不断反色某个格子所在连通块。否则, 一定能通过调整到只操作一个点来达到不差的效果。于是枚举每个格子, 然后 bfs 求出到边缘的最长黑白交错路径即可。复杂度是 $O(n^2m^2)$ 。

2.9 39C. Moon Craters

题目大意

有 n ($n \leq 2000$) 条 x 轴上的线段 $\{(l_i, r_i) | 1 \leq i \leq n\}$, 要求选出一个集合, 使不存在两条线段相交 (可以互相包含), 求使集合最大的方案。

算法简述

先离散化坐标, 设计状态 $f[i][j]$ 表示 $[i, j]$ 区间的答案。枚举以 j 为尾的线段 seg , 那么 $f[i][j] \leftarrow \max(f[i][j-1], f[i][seg.l] + f[seg.l][r])$ 。如果 $[i, j]$ 本身是一条线段, 那么 $f[i][j] += 1$ 。转移时记录方案, 最后递归输出答案。

2.10 39A. C*++ Calculations

题目大意

定义一种如下的语言:

$expression ::= summand / expression + summand / expression - summand$

$summand ::= increment / coefficient * increment$

$increment ::= a + + / + + a$

$coefficient ::= 0 / 1 / 2 / \dots / 1000$

其中, a 的初始值已知。求一种运算顺序, 使给定表达式的值最大。

算法简述

将表达式解析成很多 $k*(++a)$ 或 $k*(a++)$ 的形式。显然, 因按 k 升序来操作。而 $ka + k(a+2) = 2k(a+1)$, 故 $++a$ 和 $a++$ 的顺序不影响。

3 Volume III

3.1 39E. What Has Dirichlet Got to Do with That?

题目大意

一个游戏, 初始时 $a = a_0, b = b_0$ 。两人轮流操作, 每次可以将 a 或 b 加 1。如果使 $a^b \geq n$, 则输。问谁有必胜策略。 $a \leq 10^4, b \leq 30, n \leq 10^9$

算法简述

如果 $a = 1$, 那么如果 $b \geq \log_2 n$ 则平局;
如果 $b = 1$, 那么当 $a \geq \sqrt{b}$, N, P 状态交替出现;
否则递推就行了。表的大小只有 $O(\sqrt{n} \cdot \log n)$ 。

3.2 40E. Number Table

题目大意

给定一个 k 个格子填有 1 和 -1 的 $n \times m$ 棋盘, 保证 $k < \max(n, m)$ 。一种合法方案指将剩下所有格子填满 1, -1 后满足每行每列乘积为 -1。求方案数。 $n, m \leq 1000$ 。

算法简述

如果 n, m 奇偶性不同, 则无解。否则不妨设 $n \geq m$ 。通过交换行可以使最后一行为空。注意到只要前 $n - 1$ 行填完, 那么最后一行自然确定。所以自由格子数就是每行未确定格子数减一的和 s 。答案就是 2^s 。

3.3 43E. Race

题目大意

在一场直道赛车比赛中, 有 $n (n \leq 100)$ 辆赛车。每辆赛车有 $k (k \leq 100)$ 段行程, 第 i 段行程的速度为 v_i , 时间为 t_i 。求总共的超车次数。

算法简述

枚举两个赛车 i, j , 然后 $O(k_i + k_j)$ 统计答案。

画一个路程与时间的图像, 想象一条竖着的扫描线从左往右扫过去。具体的方法是维护两个指针, 表示对于 i, j 在扫描线前一段的行程。情况有:

- 两线段交点处(非端点)超车 \Rightarrow 线段相交判断。
- 在两条线段的公共端点上超车 \Rightarrow 特判。
- 在某条线段的端点上超车, 另一条线段经过该点 \Rightarrow 特判。

3.4 44J. Triminoes

题目大意

给你 $n \times m (n, m \leq 1000)$ 的棋盘, 其中每个格子可以是黑白两色, 亦可以没有颜色。现在要用 1×3 的骨牌覆盖它, 要求骨牌中间要对应黑色, 两端对应白色。求一种合法方案, 或者判断无解。其中合法方案要求用 4 种颜色去染, 相邻骨牌颜色不能相同。

算法简述

首先, 如果 $n(\text{black}) \neq n(\text{white})/2$, 那么无解; 否则, 如果可以唯一确定的全部加入一个队列。然后每次处理完队列的一个元素可能会导致相邻原来不是唯一确定的变成唯一确定, 于是统统入队。显然, 这个算法一定会停止, 因为不可能所有都是无法确定的。对于 4 染色, 暴力就可以了, 即每次选相邻未出现的颜色染。由于特殊性, 这种方法不会出错。

3.5 45G. Prime Problem

题目大意

将 1 $n(n \leq 6000)$ 的整数分成尽量少的份数, 使每份的和是质数。

算法简述

由哥德巴赫猜想, 任何一个 > 2 的偶数都可以拆成 2 个质数的和; 任何一个 > 7 的质数都可以拆成 3 个质数的和。所以答案不会超过 3。拆成两个的情况比较简单, 三个的话单独将 3 分为一组划归成两个的情况。

3.6 45E. Director

题目大意

有 $n(n \leq 100)$ 个 *name*, n 个 *surname*。现在要将 *name* 和 *surname* 配成 n 对, 然后排成一行。要求首字母相同的对数尽量多。其次, 最后排成一排的字典序最小。

算法简述

将 *name* 和 *surname* 单独排序。从小到大枚举每个 *name*, 考虑跟 *surname* 匹配。如果匹配后不会影响首字母相同的总对数, 就贪心匹配最小的 *surname*。

3.7 46F. Hercule Poirot Problem

题目大意

给定一个犯罪前和犯罪后的场景。场景中有 n 个房间, m 扇门, k 个人。一共有 m 把钥匙, 分别对应每一扇门。初始时所有房间都上了锁。一个人可以打开一扇门只要他有对应的钥匙。同一个房间的人之间可以交换钥匙。一个人可以走到相邻的没有上锁的房间内。以上 3 个动作可以发生任意次。要求判断两个场景是否吻合 (即能否从一个场景转换到另一个场景)。 $n, m, k \leq 1000$ 。

算法简述

注意到, 人的每种行动都是可逆的。也就是说, 只要两个场景可以通过某种方式到达一个完全相同的状态, 那么这两个场景就是吻合的。如果两个房间可以通行, 那么称它们在一个连通块中。于是可以尽可能的膨胀每个连通块, 最后直接比较房间、钥匙、人所属的极大连通块是否一样。这可以用并查集来实现。整个复杂度是线性的。

3.8 47E. Cannon

题目大意

有 $n(n \leq 10^4)$ 个导弹, m 堵墙 ($m \leq 10^5$)。导弹从 $(0, 0)$ 发射, 初速度均为 V , 发射角度各异, 但都 $\leq 45^\circ$ 。墙是指 $(x_i, 0) - (x_i, y_i)$ 的垂直线段。问每个导弹最终坐标。

算法简述

将导弹按与 x 轴夹角排序,墙按 x 坐标排序。如果第 i 枚导弹可以越过第 j 个墙,那么 i 之后的导弹也可以。用两个指针扫一遍即可。

3.9 49E. Common ancestor

题目大意

有两个 DNA 序列 s_1, s_2 (长度 ≤ 50), 求最短公共祖先 DNA 序列。DNA 序列由 26 个小写字母组成。每次衍生可以看做是 $c_1 \rightarrow c_2c_3$, 其中 c_1 是原来某个字符, c_2c_3 是衍生出来的字符, 且在新的 DNA 序列中相邻。一个串 t 的祖先串 p 指通过一系列的衍生 p 可以形成 t 。

算法简述

预处理出 $can[c][l][r]$, 表示 s_1 (或 s_2) 的 $[l, r]$ 子串是否可以由字符 c 衍生出来。再进行 dp: $f[i][j]$ 表示形成 $s_1[1 \cdots i]$ 与 $s_2[1 \cdots j]$ 的最短公共祖先 DNA 序列的长度。转移: $f[i][j] = \min(f[i'][j'] + 1)$, 如果 $s_1[i' + 1 \cdots i]$ 与 $s_2[j' + 1 \cdots j]$ 可以由某个字符衍生得到。复杂度是 $O(n^4)$ 。

3.10 51F. Caterpillar

题目大意

毛毛虫是指一棵树中存在一条路径, 使得所有点到这条路径的距离不超过 1。给定一个无向图, 每次可以合并两个点。问最少需要多少次操作可以将其变为毛毛虫。点数不超过 2000, 边数不超过 10^5 。

算法简述

显然, 对于边二连通分量中的 k 点, 必定要合并 $k - 1$ 次来缩成一个点。这样一来, 就变成了森林。如果两个树都是毛毛虫, 那么可以一定通过某种方法一次操作合并成一个毛毛虫。反之, 如果有一个不是毛毛虫, 那么合并后无论如何也不是毛毛虫。于是原问题简化成了一棵树的问题。

另一个发现是, 毛毛虫的主轴一定是连接两个叶子节点。如果不是可以继续延长至叶子, 同样也是毛毛虫。所以枚举每个叶子, 以它为根做一次 dfs。一边遍历一边统计更新答案。复杂度是 $O(n^2)$ 的。

4 Volume IV

4.1 53E. Dead Ends

题目大意

给定一个 $n (n \leq 10)$ 个点的无向连通图, 删除一些边后使其变成一棵树。要求最后叶子节点个数是 k 。求最后树形态总数。

算法简述

很明显, 添加边比删边要容易得多。由于最多只有 10 个点, 可以考虑强行记录所有状态。为了防止重复统计, 强制定 1 号节点为根。用 $f[s_1][s_2]$ 表示已用节点集合为 s_1 , 叶子节点集合为 s_2 , 其中 $s_2 \subseteq s_1$ 。

对于一个有 t 个叶子的状态, 考虑这个状态由哪些状态转移过来。可以发现, 恰好被重复统计 t 次! 于是就迎刃而解了: 只需要在每个状态转移完成后除以 t 。

还有一个问题, 就是当 1 号节点为叶子时没有统计到。可以多开一维状态来记。

4.2 57D. Journey

题目大意

给定一个 $n \times m (n, m \leq 1000)$ 的棋盘, 其中有些格子不能经过。求从任意一点 (非障碍) 出发, 到任意一点 (非障碍) 结束的最短路期望长度。

算法简述

首先可以 $O(n^2)$ 求出任意两点的曼哈顿距离总和。注意到最短路要么等于曼哈顿路, 要么等于曼哈顿路长度加 2。

```
*X.....
****X.....
*****X.....
*****X????
```

如上图所示, 从 * 到 ? 的最短路需要多加 2。而 * 成不间断的阶梯型。往下是对称的故不讨论。

还有一种是竖着的类似情况, 也完全一样地处理。

注意到后面其实是 $O(n)$ 的。实际上如果前面求和时把绝对值符号拆开讨论, 整个算法也能做到 $O(n)$ 。

4.3 226E. Noble Knights Path

题目大意

给定一棵 $n (n \leq 10^5)$ 个点的树, 每个点有一个权值 $v (v \leq 10^5)$, 同时有 $q (q \leq 10^5)$ 个询问, 每个 i 询问有向路径 (a_i, b_i) 上满足 v 不属于 $[l_i, r_i]$ 的第 k_i 个点。

算法简述

典型的可持久化线段树。对每个点建一个它到根以权值为关键字的线段树。然后预处理倍增数组 anc 。对于每个询问先看是否有解。如果有, 确定是在由 lca 分隔开的哪半条路径上。然后利用倍增数组贪心不断往上跳, 直到不能跳为止, 答案就是那个点。

判断能不能从 u 跳到 p 就要用到可持久化线段树, 询问 u 到根路径上满足条件的点个数和, 再减去 p 的父亲到根路径上的个数和。

4.4 217D. Bitonix' Patrol

题目大意

给出 $n (n \leq 10000)$ 个常数 $a_1 \cdots a_n$, 求去掉一些常数后, 使 $\sum x_i a_i = 0 \pmod{m}$, $x_i \in \{-1, 0, 1\}$ 无解的方案数。 $m \leq 120$ 。

算法简述

显然, 如果按 $a_i \bmod m$ 分类, 那么同类至多只能选 1 个; 由抽屉原理, 最多只能选 6 个数, 否则 $\geq 2^7 \geq 120$ 。于是只有 $\binom{60}{6} \approx 5 \times 10^7$ 种情况。爆搜即可。要用位运算加速。

4.5 67E. Save the City!

题目大意

给定一个简单多边形, 其中一条边 AB 平行于 x 轴。问 AB 上有多少整点可以看到所有所有端点。多边形端点数不超过 1000, 坐标大小在 10^6 内。

算法简述

对每个端点找出 AB 上对应可以看到的线段, 最后求交即可。用向量叉积可以分 4 种情况讨论。

两个向量 \vec{u}, \vec{v} 可以看作将平面分成 4 份。对于平面中的任意一个点 p , 可以通过 $u \otimes p, v \otimes p$ 的正负来判断, 恰好也是 4 个情况。

4.6 67C. Sequence of Balls

题目大意

给定两个序列 A, B (长度 ≤ 4000), 通过以下四种代价不同的操作将 A 变成 B :

1. 删除任意一个字符
2. 添加任意一个字符
3. 交换相邻两个字符
4. 将任意一个字符修改为另一个字符。

其中, 保证两倍 3 操作的代价 \geq 1, 2 操作的代价。

算法简述

对于“保证两倍 3 操作的代价 \geq 1, 2 操作的代价”, 意味着一个字符不可能交换多次。设计状态 $f[i][j]$ 表示 A 的前 i 个跟 B 的前 j 个字符匹配下的答案。1, 2, 4 都比较方便实现。交换比较麻烦:

A: a****b
B: b.....a

除了直接交换外, 还可以将 * 全部删除, 再插入。。

4.7 70D. Professors task

题目大意

动态凸包, 支持插点和询问一个点是否在凸包内。点数 $\leq 10^5$ 。

算法简述

建立循环的极角序,用 set 维护。唯一麻烦的地方在于循环的收尾处,需要仔细处理。另外,前 3 个点也需要特殊处理。

4.8 70E. Information Reform

题目大意

给定一个 $n(n \leq 180)$ 个节点的树,需要在其中选择一些点放置基站,每个基站代价为 k 。其余的点到距离自己 t 的基站的通讯费用为 $d[t]$ 。求一种代价最小的方案。保证 $d[i] \leq d[i+1]$ 。

算法简述

这个题的 dp 不是很好想。首先任意定根。用 $f_1[i]$ 表示以 i 为子树的答案, $f_2[i][j][k]$ 表示在 i 子树中处理完 j 子树,且 i 选择 k 作为基站的答案。转移根据 j 子树是独立出来还是选 k 为基站讨论。

这个 dp 的本质在于强行限定了一些条件,比如选择的基站,以及基站控制的子树。对于这种 dp,需要大胆猜想,小心求证。

4.9 156E. Mrs. Hudson's Pancakes

题目大意

有 n 个调料,编号为 $0 \rightarrow n-1$ 。还有 m 道食谱,每个食谱有 3 个属性 d, s, c 。称一个调料能匹配一个食谱,意味着 d 进制下通过修改 s 中的? 可以得到食谱的编号。一个食谱的价值为它能匹配的调料编号的积再加 c 。求每个食谱价值的最小质因数。 $n \leq 10^4, m \leq 3 \times 10^4, d \leq 16$, 其余数都在 10^{18} 内。

算法简述

首先,预处理出所有 $f[d][s] = \prod(i|i \in s)$ 。 s 不会太多,因为如果位数大了,前面必为 0。又因为 100 以内质数只有 25 个,可以再开一维记模的答案。然后就是各种常数优化,比如每 5 个质数分成一组来记模。

4.10 105D. Entertaining Geodetics

题目大意

有一个 $n \times m(n, m \leq 300)$ 的棋盘,每个格子有一个颜色。某些格子上放有某个颜色标记,同时有一个标记队列。初始时将某个位置的标记入队。然后不断重复以下操作,直到队列为空:取出队首标记。如果该标记所在格子颜色不为 0 且标记与格子不同色,那么所有该格子颜色的格子会按螺旋形重新染成该标记的颜色。如果在重染色的时候格子上有标记,那么将其入队。问总共染色次数。

算法简述

题意异常纠结。可以注意到,一旦进行一次螺旋形染色,那么某种颜色的格子上所有的标记将全部入队。并且,如果某个格子之前被重染色过,那么之后每次染色都会受到影响。记下来统计答案即可。

5 Volume V

5.1 193D. Two Segments

题目大意

给出 $1 \rightarrow n$ 的一个排列 p , 求不同的 (a, b, c, d) 的个数, 满足 $a \leq b < c \leq d$ 且 $\{p[a], p[a+1], \dots, p[b]\} \cup \{p[c], p[c+1], \dots, p[d]\}$ 得到的集合排序后是公差为 1 的等差序列。判断两个四元组是否算重, 看生成的集合是否一样。
 $n \leq 3 \times 10^5$ 。

算法简述

定义一个函数 $f(a, b)$, 表示形成以 a 为首项 b 为末项的数在原序列中的段数。那么答案就是 $\sum_{a,b} [f(a, b) = 2] + [f(a, b) = 1] - n$ 。

如果将 $(a, b) \rightarrow f(a, b)$ 建成一张表, 可以发现以下性质:

1. 每个元素和它相邻的元素差不超过 1。特别的, 每个元素和它下面的元素差不超过 1。
2. 如果将每行减去它下面一行, 那么得到的是如下形式的序列:

$$1, 1, \dots, 1, 0, 0, \dots, 0, -1, -1, \dots$$

, 是随右端点增大而单调不增的。

基于这两个性质, 可以从下往上用线段树实现区间加、减, 询问最小、次小值 (因为全部为正数, 1, 2 必然是最小的两个) 来统计答案。

5.2 75E. Ships Shortest Path

题目大意

在一片海域中, 有一座孤岛 (凸多边形)。一艘船要从 A 到 B , 要求航程最短, 且只能经过岛上或 AB 连线。可以经过岛上, 但航程要按双倍记。

算法简述

求出交点, 然后分情况讨论。为了避免边界精度问题, 可以最后用 floyd 简化。

5.3 76F. Tourist

题目大意

在 x 轴上, 有 $n (n \leq 10^5)$ 个事件点。每个事件点有坐标和发生时刻。一个人从某个点出发, 两个方向都可以选择, 速度不超过 V 。求一种方案使观看的时间个数最多。

算法简述

将每一个事件点 (t_i, x_i) 映射到 (x_i, y_i) 。如果可以从 i 走到 j , 那么 $|y_i - y_j| \leq |x_i - x_j| \cdot V$, 也就是说 i, j 两点的连线斜率绝对值不超过 V 。选择与 x 正半轴斜率为 $+V, -V$ 的两个向量作为基本向量。由平面向量基本定理, 可以转化成经典的二维最长不下降子序列问题。 $O(n \log n)$ 解决。

5.4 76A. Gift

题目大意

给定 n 点 ($n \leq 200$), m ($m \leq 5 \times 10^4$) 条边的无向图。每条边 i 有两个权值 a_i, b_i 。求确定两个参数 S, G , 使得所有 $a \leq S, b \leq G$ 的边连通。使 $k_S \cdot S + k_G \cdot G$ 最小。其中, k_S, k_G 是给定的系数。

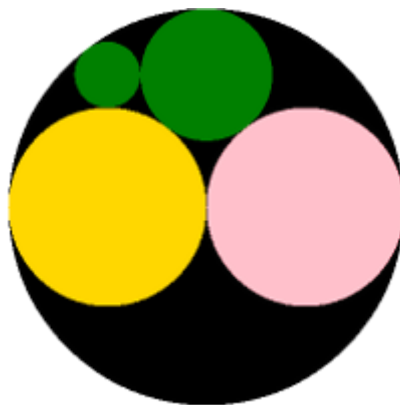
算法简述

基本思路是枚举其中一个关键字, 不妨枚举 S : 将所有边按 a 排序, 然后依次处理。每次添加一条边, 置换出那个环上 b 值最大的边。然后每次暴力统计答案。总复杂度 $O(nm)$ 。

5.5 77E. Martian Food

题目大意

一开始有两个内切的圆, 半径为 R, r ($R > r$)。然后开始在大圆内添加新的圆, 使其与初始时的两个圆一个内切一个外切, 且面积最大。问这样下去第 k ($k \geq 10^4$) 个圆的半径。多组数据 ($\leq 10^4$)。



算法简述

以切点为原点, 过两圆的直径为 x 轴建立极坐标系。对每个点 (d, ϕ) , 进行变换 $(d, \phi) \leftrightarrow (1/d, \phi)$ 。于是过原点的两个圆被变换成两条垂直于 x 轴的直线, 而其他圆依然变换成圆, 添加进去的圆夹在两直线之间。于是, 在新坐标系中第 k 个圆心为 $((R+r)/4Rr, k(R-r)/2Rr)$ 。

注意, 这个圆心不是对应原坐标系中的圆心。应该取三个圆周上的点, 反推回去求三角形外接圆半径。复杂度是常数级别。

5.6 79D. Password

题目大意

一排有 n ($n \leq 10^4$) 个 01 格子, 初始时全为 0。给出一个目标状态, 保证 1 的个数不超过 10, 以及一些操作, 不超过 100 个。每个操作 i 由 a_i 确定, 意味着可以将长度为 a_i 的连续区间取反。每个操作可以进行任意次。求最少总操作次数。

算法简述

首先, 需要简化操作的格子数。有两种方法: 前缀 xor 或者相邻两项 xor, 而后者可以保证变换后 1 的个数依然不超过 20, 故采用后者, 得到一个新数组 b 。于是每次操作变为将距离为 a_i 的两个格子取反。注意到可以将操作看作是从一个格子跳到另一个格子, 于是用最短路可以求出将任意两个 1 格子取反的最少操作数。最后状压 dp 即可。

5.7 81E. Pairs

题目大意

给出一个章鱼型图(每个点有且仅有一条出边),每个边有双关键字的权值。求选出一些边使得没有一个点同属于多个边,且权值和最大。点数 $\leq 10^5$ 。要求输出方案。

算法简述

这题思路还是比较显然的。定义一个关键字类,重载运算符后当一个关键字处理。先在树上 dp,再在环上 dp。输出方案按逆 dp 序来。

5.8 82E. Corridor

题目大意

在平面中, $y = h$ 和 $y = -h$ 是两堵墙。在墙上面有一些窗户,用线段 $(l, h) - (r, h)$ 表示,且关于 $y = 0$ 对称。在 $(0, f)$ 和 $(0, -f)$ 有两个光源。求两堵墙之间被照亮的面积。窗户数不超过 500。

算法简述

注意到每个点不可能被覆盖超过 2 次。可以考虑用简单容斥把覆盖了 2 次的面积减掉。枚举 $y = h$ 上和 $y = -h$ 上两两窗户,求半平面交即得答案。

5.9 83E. Two Subsequences

题目大意

定义一个函数 f :

$f(\text{空串}) = \text{空串}$

$f(s) = s$

$f(s_1, s_2) = \text{最短字符串, 前缀为 } s_1, \text{ 后缀为 } s_2.$

$f(a_1, a_2, \dots, a_n) = f(f(a_1, a_2, a_n - 1), a_n).$

给定等长字符串序列 $a_1 \dots a_n$, 需要将其分成两个子序列 s_1, s_2 , 使得 $|f(s_1)| + |f(s_2)|$ 最小。 $n \leq 10^5$, 字符串为 01 串, 且长度 ≤ 20 。

算法简述

考虑一个暴力的 dp: $f[i][j]$ 表示到了第 i 个字符串, 另一个序列以第 j 个字符串结尾。转移枚举第 $i+1$ 个字符串是接到哪个序列上去。尽管转移为 $O(1)$, 但光状态就是 $O(n^2)$ 。于是想办法简化状态。可以设 $f[i]$ 表示 i 与 $i+1$ 属于不同序列的答案。那么转移就需要枚举一段区间: $f[i] = \min\{f[j] + w[j+1][i] - g(j, i+1)\}$, 其中 $w[l][r]$ 表示将 l 到 r 的字符串压缩后的长度, $g(i, j)$ 表示将第 i 和第 j 个字符串接起来节省的长度。进一步观察可以发现, $g(j, i+1)$ 的取值只有 20 种。故可以枚举 j 与 $i+1$ 的公共长度, 用位运算来记录后缀为某个串的答案。于是整个算法复杂度只有 $O(20n + 2^{20})$ 。

5.10 85E. Guard Towers

题目大意

平面上有 $n (n \leq 5000)$ 个点, 可以将其染成黑色或者白色。求一个 p 的最小值, 使最远黑点距离 $\leq p$ 且最远白点距离 $\leq p$ 。坐标范围 ≤ 5000 。距离指曼哈顿距离。

算法简述

算出两两之间的曼哈顿距离, 然计数排序。从大到小枚举曼哈顿距离 (不超过 10000), 看答案能否为当前枚举的值。

判断方法是动态维护一个二分图, 支持动态加边。这可以用并查集暴力 $O(n)$ 合并实现。总复杂度为 $O(n^2)$, 常数稍大。也可以二分出错位置, 再用二分图匹配验证。由于边数与点数同级, 复杂度是 $O(n^2 \log n)$ 。实测后者速度要快于前者。

6 Volume VI

6.1 86E. Long sequence

题目大意

要求构造一个无限 01 序列, 输出该序列的前 k 项和 k 阶递推式, 满足最小周期为 $2^k - 1$ 。 $k \leq 50$ 。

算法简述

先考虑如何验证一个序列的最小周期是否为 T 。方法可以类比原根的判定方法, 即依次验证以 T 除以它自身所有质因数为周期看等不等于 T , 同时 T 需要为周期。由于 $k \leq 50$, 用矩阵乘法 + 快速幂可以做到 $O(k^3(\log_2 T)^2)$ 。

注意到输入只有一个数, 可以采用随机化 + 打表。直接随机化也是可以过的。

6.2 89D. Space mines

题目大意

三维空间中布置着 $n(n \leq 100)$ 个地雷, 每个地雷由球形雷体和若干引线组成。雷体的半径为 r , 球心为 O ; 每条引线可以被看作连接球心与球外某个点 P 的线段。保证 $|OP| \leq \frac{3}{2}r$ 。有一个半径为 R 的三维球体, 称为死星。初始时死星位于 A 点, 死星以一定的速度 v 移动。问死星是否会碰到地雷(包括引线)。如果会, 求碰撞时刻。

算法简述

由于题目限制, 死星如果碰到引线一定是在端点处。于是问题转化成了判断球体能否与空间中的点碰撞。设经过的时间为 t , 于是与点 P 碰撞的时刻可以通过以下方程解得:

$$(A_x + v_x t - P_x)^2 + (A_y + v_y t - P_y)^2 + (A_z + v_z t - P_z)^2 = R^2$$

6.3 91D. Grocer's Problem

题目大意

给定一个 $n(n \leq 10^5)$ 的排列, 每次可以选择不超过 5 个数进行任意重排。求最少操作次数。

算法简述

首先将所有的循环节找到。对于长度大于 5 的循环节, 可以每次操作使其长度减少 4。于是只用考虑长度为 1-5 的循环节。

1. 长度为 1: 已经在正确的位置上的, 不管;
2. 长度为 4, 5: 操作一次全部到位;
3. 长度为 2, 3: 显然, 先将 2, 3 尽可能多的配对, 一次消去一对。最后只剩 2 或者 3:
 - (a) 剩 2, 尽可能多的两个一消去, 如果有剩余再操作一次;
 - (b) 剩 3, 这个情况比较坑。实际上可以操作两次消去三个, 如果有剩就再做一次。

6.4 93D. Flags

题目大意

设将 n 个排成一排的木块 4 染色 (BWRY), 设满足以下条件的方案数为 $f(n)$:

1. 相邻两木块不能同色
2. 不存在相邻两个木块颜色为 WY
3. 不存在相邻两个木块颜色为 RB
4. 不存在 BWR 或 RWB 的组合
5. 对称的只算一次

给出 $L, R (L, R \leq 10^9)$, 求 $\sum_{L \leq i \leq R} f(i)$ 的值模 1000000007。

算法简述

首先题目满足区间减法, 转化成求 $1 \dots n$ 的方案和。

如果不考虑对称算重的问题, 那么将相邻 3 个木块的颜色压缩成大小为 $4^3 = 64$ 的状态, 记录合法转移在一个矩阵中, 多开一维记录方案和, 最后用快速幂加速矩阵乘法即可。

对称也很好处理。由于“相邻两木块不能同色”, 只有当 n 为奇数时才会出现。这时候要减掉 $f((n+1)/2)$ 。而减掉的部分也是求一段答案和, 再做一边矩阵乘法就行。

6.5 97C. Winning Strategy

题目大意

给出 $n+1$ 个概率 p_0, p_1, \dots, p_n , 满足 $p_i \leq p_{i+1}$ 。求一个长度无限的 $\{a_k | 0 \leq a_k \leq n\}$ 序列, 满足:

$$(\forall k) a_k \leq \sum_{i=1}^{k-1} (n - 2a_i)$$

, 且最大化:

$$\lim_{m \rightarrow \infty} \frac{\sum_{i=1}^m p_{a_i}}{m}$$

。只需输出最大值。 $n \leq 100$ 。

算法简述

先撇开无限不讲, 设计一个 dp 来计算答案。用 $f[i][j]$ 表示当前在构造 a_i 项, 第 a_i 项大小不超过 j 。那么转移就是:

$$\forall k, 0 \leq k \leq \min(n, j), f[i][j] + p_k \rightarrow f[i+1][j+n-2k]$$

。注意到第二维状态如果大于 $2n$ 就没有用了, 否则可以交换它和之后的元素来强行限制。如果把每个第二维的状态对应到一个有向图中的点, 那么问题转化成了从某个点出发, 走无数步, 使得平均每步的权值最大。不难发现, 经过有限步后一定是在一个环上走。

于是问题变成了经典的求最小平均权环, 用分数规划 + spfa 判负环解决。

6.6 97A. Domino

题目大意

有一副 2×1 的骨牌, 共 28 块, 骨牌的每一半包含一个 0 到 6 的数字。
如下所示:

0-0 0-1 0-2 0-3 0-4 0-5 0-6
1-1 1-2 1-3 1-4 1-5 1-6
2-2 2-3 2-4 2-5 2-6
3-3 3-4 3-5 3-6
4-4 4-5 4-6
5-5 5-6
6-6

一个包含了 28 个骨牌的图被叫成 *magic*, 仅当能被 14 个互不相交的 2×2 的方块覆盖, 每个方块由 4 个相同的数字组成。

现给出一个 $n \times m$ 的棋盘, 上面放了许多 1×2 的矩形小片。每个小片恰好占领了棋盘上的两个相邻方格, 总共有 28 个小片。现在, 需要把每个小片换成骨牌, 拼出 *magic* 图片。不同的小片应被替换成不同骨牌。计算合法的方案数并给出一个方案。

算法简述

爆搜。先找出所有 14 个 2×2 的方块。如果限制搜索顺序, 即 1 在 2 之前, 2 在 3 之前, 以此类推, 那么本质不同的方案数的上限是 $\frac{14!}{7!(2!)^7} = 135135$ 。于是就搞定了。

6.7 98D. Help Monks

题目大意

考虑一个加强版的汉诺塔问题: 依旧是三根柱子, 不过每个盘子有一个直径, 可能相同。相同直径的盘子之间顺序可以互换。要求最终盘子的顺序与初始时完全相同。求最少移动次数以及方案。盘子数不超过 20。

算法简述

定义两个状态: $f[i]$ 表示将前 i 个盘子移动到另一根柱子上, 且完全按原顺序的最少次数; $g[i]$ 表示将前 i 个盘子移动到另一根柱子上, 顺序任意的最少次数。

首先考虑 g 的转移。设 $L[i]$ 表示在 i 之上且与 i 大小相同的盘子个数。那么最优决策显然是将前 $i - L[i]$ 个盘子挪开, 移动 $L[i]$ 次, 再将 $i - L[i]$ 个盘子挪回来。故 $g[i] = 2g[i - L[i]] + L[i]$ 。

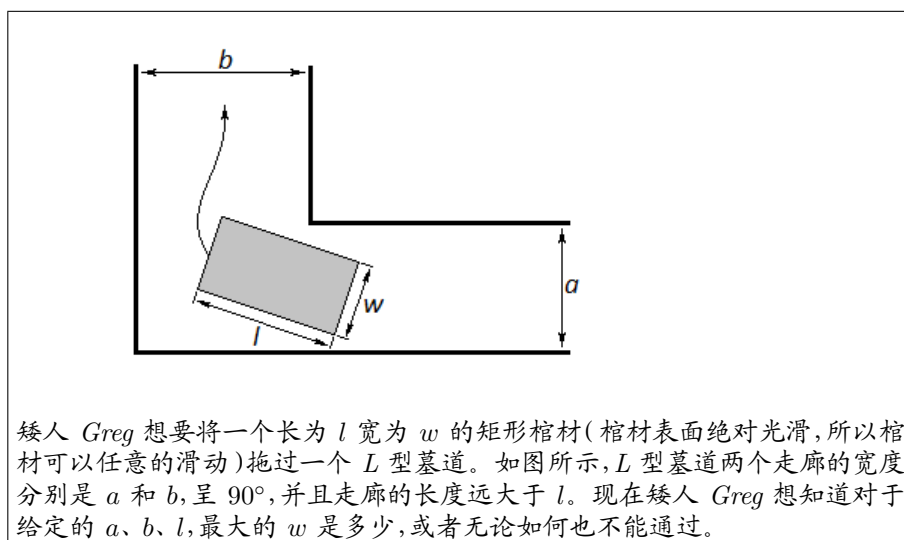
对于 f , 有 2 种情况:

- 像经典汉诺塔一样, $f[i] = 2f[i - 1] + 1$;
- 将最下面 k ($2 \leq k \leq L[i]$) 个整体考虑, 先将前 $i - k$ 个按 g 乱序移开到目标柱, 然后把剩下 k 个移到辅助柱上。注意此时顺序是倒过来的。然后, 将目标柱上的 $i - k$ 个“倒带”, 倒回原柱, 再将辅助柱上的 k 个移到目标柱, 此时顺序已经正确。最后, 按 f 的方案将原柱上的 $i - k$ 个移到目标柱完成整个过程。所以总共是 $2g[i - k] + f[i - k] + 2k$ 步。

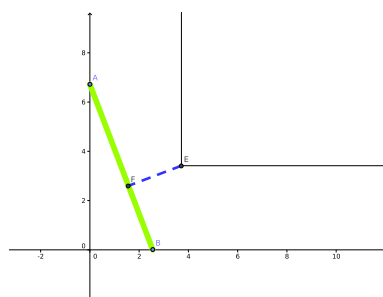
最后记录转移过程递归输出解。

6.8 98C. Help Greg the Dwarf

题目大意



算法简述



如图所示, 矩形一边沿墙角滑下, 可以发现, 内拐点到这边的距离是关于时间的单峰函数。三分得到其最小值就是答案。

6.9 191D. Metro Scheme

题目大意

给定一个 n 个点 m 条边的仙人掌图 (每个点至多属于一个环), 可以用链或状将所有边覆盖, 且每条边只经过一次。求最少需要的链和环的总数。
 $n \leq 10^5, m \leq 3 \cdot 10^5$ 。

算法简述

求双连通分量然后 dp 当然是可以做的, 不过非常麻烦。我们不妨考虑原问题的简化版本, 即原图是树而不是仙人掌。考虑答案最小值, 每条路径链接两个奇度数的点。于是答案最小值为 $\sum_{x \in V} [d(x) \text{ is odd}] / 2$ 。这个值实际上是可以达到的, 而且随便连只要合法就可以得到这个下界。假设有一条边未被覆盖, 那么将这条边断开得到的两个子树中都是偶数个奇度点, 算加上这条边就是奇数个奇度点, 肯定会一边剩一个奇度点, 于是这条边会被覆盖。

对于原问题中的环:

1. 环上的点度数都是 2, 那么 $\text{ans}++$;
2. 环上有一个点度数大于 2, 同样 $\text{ans}++$;
3. 环上有两个以上的点大于 2, 这时候可以通过延长这些有外来路径的点来将环完全覆盖, 具体的分奇偶度数点来讨论即可。这种情况下答案不变。

经过一系列简化后,连双连通都不用求。实现的时候可以将度数为 2 的环上点一个个删去,不影响答案。

6.10 164D. Minimum Diameter

题目大意

给定平面上的 n 个点。你需要去删掉恰好 k 个点 ($k < n$),使得剩下的 $n - k$ 个点所构成的集合的直径尽量小。一个点集的直径是指集合中最远点对的距离。

算法简述

如果直接二分答案,只能问题转化为一般图的匹配问题。不妨枚举两点作为最远点。设它们的距离为 d 。那么形成的两圆之外的点是一定要删除的。而在两圆交的部分内的点,需要满足:不存在两个点距离大于 d 。而交部分的点可以分成两部分,从而变成二分图的匹配问题。理论复杂度上限非常高,不过蕴含的常数很小的。

7 Column VII

7.1 150E. Freezing with Style

题目大意

给定一个 n 个点 ($n \leq 10^5$), 边上带权的树。求一条长度在 l 到 r 之间的路径, 使路径上边权的中位数最大。对于一个长度为 m 、下标从 0 开始的有序数列 a_i , 它的中位数是 $a_{\lfloor m/2 \rfloor}$ 。

算法简述

典型的树分治。于是我们只用考虑在一个子树中经过根的路径。首先二分答案 (局部二分比全局二分要快) w 。如果一条边边权大于等于 w , 那么令这条边新的边权为 1, 否则为 0。问题转化成了判断是否存在一条路径上新边权和大于等于 0, 即求新权值最大的路径。将儿子按其子树深度从大到小排序后, 利用单调性这一步可以做到 $O(n)$ 。总复杂度为 $O(n \log^2 n)$ 。

7.2 101E. Candies and Stones

题目大意

给定一个 $n \times m$ 的棋盘, 每个格子有一个权值。现在要从 $(1, 1)$ 走到 (n, m) , 使经过的格子权值总和最大。求方案。要求空间复杂度做到 $O(n+m)$ 。格子的权值以某种压缩的形式给出。 $n, m \leq 20000$, 时限 15s。

算法简述

如果没有空间限制, 那么很容易得到 $O(nm)$ 的 dp 算法。而如果只算答案, 用滚动数组也可以优化空间至线性。不如将这两者结合起来, 考虑运用分治。在传统的 dp 算法中, 阶段是按对角线划分的。那么对于规模是 $n \times m$ 的棋盘, 通过合并前 $\frac{n+m}{2}$ 和后 $\frac{n+m}{2}$ 可以得到整个的答案以及第 $\frac{n+m}{2}$ 个阶段上的方案。于是可以分成左上和右下两个子矩形递归下去做。

复杂度分析: 每次矩形的规模 $n \times m$ 至少缩小至 $nm/4$ 。所以 $T(nm) = O(nm) + 2T(nm/4) = O(nm)$ 。

7.3 103E. Buying Sets

题目大意

给定 n 个集合, 每个集合包含若干个 1 到 n 的整数, 另外还有一个整数权值。保证任意 k 个集合的并的元素个数不小于 k 。求一个集合的集合, 使得集合的集合的大小等于所选集合并的元素个数, 且权值和最大。 $n \leq 300$ 。

算法简述

首先建立二分图, 左边对应集合, 右边对应数。如果第 i 个集合包含数 j , 那么连有向边 $u_i \rightarrow v_j$ (u 是左边点, v 对应右边点)。由 Hall 定理, 以及题目的特殊性, 该二分图有完备匹配。求出这个完备匹配。这样以来相当于给每个数找了一个配对的集合。然后, 将左边的点和右边的点按完备匹配合并成一个点, 保留对应的边, 得到一个 DAG。可以发现, 任何一个合法的解在 DAG 中都不存在出边, 反过来也成立。于是问题转化为最大权闭合子图, 可以用最大流最小割模型解决。

7.4 105E. Lift and Throw

题目大意

三个人初始时站在数轴整点上。任意一个人可以进行以下操作不超过一次：

1. 移动一段距离, 不超过一个定值。
2. 抓起另一个人, 要求两人距离恰好为 1。一旦抓起一个人后, 将不能移动。被抓起的人不能进行任何操作。
3. 将抓起的人扔一段距离, 不超过一个定值。

求三人中任意一人能到达的最远距离。输入所有数均为整数且不超过 10。

算法简述

爆搜 + 记忆化剪枝, over。

7.5 107D. Crime Management

题目大意

构造一个长度为 n 的字符串, 由大写字母组成。对于每个字母有个系数 c_i , 要求 i 字母出现次数为 c_i 的倍数。满足所有 c 的最小公倍数不超过 123。求合法构造方案数模 12345。

算法简述

由于所有 c 的最小公倍数不超过 123, 可能的状态总数也不超过 123。dfs 预处理出所有状态以及相互的转移, 最后用矩阵快速幂加速。

7.6 113D. Museum

题目大意

有一个 $n(n \leq 22)$ 个点的无向图。一开始 Alice 和 Bob 分别站在 a, b 点。每一回合, Alice 和 Bob 将等概率随机走向某个相邻点。求对于每个点 Alice 和 Bob 在此相遇的概率。注意: 一旦相遇则停止游戏。

算法简述

按照常规思路, 未知量用二维信息来表示, 可以得到 $O(n^2)$ 个变量的方程组。用高斯消元一次是 $O(n^6)$, 做 n 次将达到 $O(n^7)$, 无法承受。

原来计算的是 $Ax = b$, 其中只有 A 是 $n^2 \times n^2$ 的矩阵, 而 b 是 $n^2 \times 1$ 的向量。实际上每次换一个终点计算, 仅仅改变了 b 。那么可以将 b 换成 $n^2 \times n$ 的矩阵, 一次消元就能算出所有解。于是复杂度变为了 $O(n^6)$ 。

7.7 115D. Unambiguous Arithmetic Expression

题目大意

定义 UAE 为:

1. 单独一个非负整数是 UAE。
2. 如果 X 和 Y 是两个 UAE, 那么 $(X) + (Y)$, $(X) - (Y)$, $(X) * (Y)$, $(X)/(Y)$ 都是 UAE。
3. 如果 X 是 UAE, 那么 $-(X)$ 和 $+(X)$ 也是。

给定一个仅包含数字和 $+*$ 的字符串, 求添加任意括号后能形成的合法 UAE 数目。长度不超过 2000。

算法简述

因为第三个限制, 可以考虑从右到左进行构造。

令 $f[i][j]$ 表示进行到了第 i 个字符, 第 i 个字符之后是 j 个块相加的形式 (因为一旦套上括号后就跟数字无异了)。为了方便处理, 将所有数字替换为 a 。如果 $i-1$ 是 a , 那么显然 $i-1$ 和 i 要按第二个规则构造。转移到 $f[i-2][k]$, $\forall 1 \leq k \leq j+1$ 。否则, 如果 i 是 $+-$, 那么按第三个规则构造, 转移到 $f[i-1][k]$, $\forall 1 \leq k \leq j$ 。答案就是 $f[0][1]$ 。

这样的复杂度是 $O(n^3)$ 。通过部分和优化可以做到 $O(n^2)$ 。

7.8 120I. Luck is in Numbers

题目大意

对于一个位数为偶数的数字, 将它从中间切开, 分成长度相同的两部分。然后重叠在一起。那么交的个数称为这个数字的幸运度。给定一个 $k(k \leq 2 \times 10^5)$ 位数, 求位数相同且大于这个数, 同时幸运度也更大的最小的数。

算法简述

主要思路是贪心。预处理出任意两数码交的个数。注意到 8 这个数字非常 imba。从高位到低位处理, 改变当前位, 之后全部填 8, 看是否满足条件。如果满足就确定这一位。否则无解。通过部分和预处理可以使验证是 $O(1)$ 的。总复杂度为 $O(10k)$ 。

7.9 123E. Maze

题目大意

一个 maze 由一个树表示。考虑一个如下的子过程:

```
DFS(x)
  if x == exit vertex then
    finish search
  flag[x] <- TRUE
  random shuffle the vertices' order in V(x)
  for i <- 1 to length[V] do
    if flag[V[i]] = FALSE then
      count++;
      DFS(y);
  count++;
```

现在给出每个点选为起点和终点的概率, 求 count 的期望值。点数不超过 10^5 。

算法简述

设起点为 s , 终点为 t , s 到 t 路径上点数为 l 。如果以 s 为根建树, $sz[x]$ 表示 x 的子树大小 (不含 x)。那么 t 子树中的点一定不会被经过。而 $s \rightarrow t$ 对答案的贡献为 $l-1$ 。称 $s \rightarrow t$ 为主轴, 那么父亲在主轴上, 本身又不在主轴上的点所代表的子树对答案的贡献又是什么呢?

注意到, 一旦偏离主轴进入某个点 u , 那么一定是 u 子树中所有点都要走遍, 这些点对答案的贡献是 2 (一进一出)。设 u 的父亲是 p , p 主轴上的儿子是 v 。进入 u 的概率等于 p 儿子的所有排列中 u 在 v 前面的概率, 即 $\frac{1}{2}$ 。于是, 每个不在主轴上且不在 t 子树中的点对答案的贡献恰好是 1。设总点数为 n 。那么以 s 为起点, t 为终点的期望答案就是 $n-l-sz[t]+l-1=n-sz[t]-1$, 只与 t 的子树大小有关。于是乎, 在一遍 dfs 过程中动态修改根, 维护所有点 sz 与作为终点概率的乘积和即可。

7.10 125E. MST Company

题目大意

求满足度限制的最小生成树。即要求某个点 u 的度恰好为 k 。点数 $n \leq 5000$, 边数 $m \leq 10^5$, $k \leq 5000$ 。

算法简述

传统最小度限制生成树的做法基于贪心: 先将 u 删掉, 用 kruskal 求最小生成森林。再考虑将与 u 关联的边往上加。如果加上会形成环, 就置换掉最大的边 (如果可以使答案变小的话)。复杂度是 $O(m \log m + nk)$ 。

还有一种基于实数二分的方法。将所有与 u 关联的边加上一个实数 w 。令 $f(w)$ 表示加上 w 后与 u 关联的边在最小生成树中的条数。显然, $f(w)$ 是单调不增函数。当 $f(w) = k$ 时, 新图的最小生成树就是答案。反证很容易证明。复杂度为 $O(m \log m \log w)$, 与 k 无关。

8 Column VIII

8.1 193E. Fibonacci Number

题目大意

考虑 *fibonacci* 数列每项模 10^{13} 后得到的数列。问 x 在这个数列中是否出现过, 如果出现过, 求最早出现位置。

算法简述

如果 x 模 10^i 为 d , 那么 x 模 10^{i+1} 的结果必然是以 d 为后缀的。于是基本思路就是由模 10^i 的答案列表推出模 10^{i+1} 的答案列表, 最后得到模 10^{13} 的答案。

将 10^i 推到 10^{i+1} 时, 需要将列表按循环节长度平铺多次, 再通过矩阵乘法快速幂验证。因此, 我们需要知道 *fibonacci* 数列模 n 的循环节, 设为 $L(n)$ 。这里有一个定理:

设 n 的质因数分解为 $n = p_1^{m_1} p_2^{m_2} \dots p_k^{m_k}$, 那么 $L(n) = \text{lcm}(L(p_1^{m_1}), L(p_2^{m_2}), \dots, L(p_i^{m_i}) = L(p_i) p_i^{m_i-1}$ 。

为了避免边界情况, 先暴力构造出 10^3 的列表。注意到 $L(10^k) = 10L(10^k - 1), k \geq 4$, 那么每次平铺 10 次, 然后验证即可。复杂度比较诡异, 实测非常快。

8.2 145D. Lucky Pair

题目大意

幸运数字是那些仅由 4 和 7 构成的数字。有一个长度为 n 的非负整数序列 A , 要从整个序列中选出两个互不相交的子段 $A[l_1, r_1], A[l_2, r_2] (1 \leq l_1 \leq r_1 < l_2 \leq r_2 \leq n)$, 使得不存在某个幸运数字既在 $A[l_1, r_1]$ 出现, 又在 $A[l_2, r_2]$ 出现。求有多少种选择方案。保证 A 序列中幸运数字的总出现次数不会超过 1000 次。 $n \leq 10^5$ 。

算法简述

注意到非幸运数字都是等价的, 可以先把幸运数字找出来。设 $X[i]$ 为从左数第 i 个幸运数字。再将非幸运数字合并到一块。

因为幸运数字不是很多, 可以枚举左区间。具体的, 先枚举左区间最靠右的幸运数字是第 r 个。然后从左往右枚举左区间最靠左的是第 l 个。注意到一旦 $X[l]$ 在 $X[l+1] \dots X[r]$ 不再出现, 那么右区间就可以包含这个数了。于是用并差集维护所有极大合法右区间, 其任意子区间都合法。一旦一个新的幸运数字变为合法, 那么合并两个极大合法右区间。算法复杂度为 $O(k^2)$, k 是幸运数字个数。

需要注意的是, 实现上细节很多。问题主要在左右区间交界处, 需要仔细斟酌, 耐心编码。

8.3 132E. Bits of merry old England

题目大意

你被要求输出一个由 n 个数 ($n \leq 250$) 组成的序列。有 m 个变量 ($m \leq 26$) 给你用, 变量必须是单个小写英文字母。要求给出一个操作序列, 输出这 n 个数。操作只有 2 种:

1. 对某个变量赋值, 代价为该值二进制表示中 1 的个数。
2. 输出某个变量, 无代价。

变量没有初值, 也就是, 你必须给变量赋过值才能输出这个变量。要求最小化操作序列的代价。输出这个最小代价, 以及能达到最小代价的满足条件的操作序列。

算法简述

首先可以确定需要用多少个变量。设 n 个数中不同的数的个数为 z 。那么恰好需要 $t = \min\{m, z\}$ 个变量。建一个 $n+t$ 个点带权的 DAG。前 n 个点对应序列中的数。从第 i 个点向第 j 个点连边 ($i < j \leq n$)。边权的话,如果序列中 i, j 位置的值相同,那么为 0。否则为 j 二进制表示中 1 的个数。对于附加的 t 个点,从第 $n+i$ 向 j 连边 ($j \leq n$),权值为 j 二进制表示中 1 的个数。于是乎,这个 DAG 的最小路径覆盖数恰好为 t 。最小路径覆盖可以由二分图匹配解决。那么既然控制了匹配数,求出的最佳匹配(或最小费用流)就是满足最小代价的方案。

8.4 138D. World of Darkraft

题目大意

给定一个 $n \times m$ ($n, m \leq 20$) 的棋盘。棋盘上有 3 类格子 (L, R, X)。两个人轮流进行游戏。每轮每人选择一个未被标记的格子。根据不同格子触发相应的效果:

L — 从这个格子出发往左下和右上方向走,每遇到一个未被标记的格子则标记之,直到遇到一个已标记的格子或者超出棋盘边界。

R — 与 L 类似,唯一差别在于扩展方向是左上和右下。

X — 结合 L 和 R 的效果。

不能移动者输。问是否先手必胜。

算法简述

由于所有操作都是斜对角线上,可以将棋盘黑白染色单独处理,最后通过求独立游戏的和(即异或)来得到整个游戏的答案。

然后将棋盘转 45° 。每次操作变成在行和列上进行。而每次操作会划分成若干个子棋盘,互为独立游戏。于是可以动态规划来求出 sg 函数的值。状态记录当前矩形。复杂度是 $O(n^4 m^4)$ 。

8.5 140F. New Year Snowflake

题目大意

一个点集是中心对称的,指存在一个点 X (不一定要属于这个点集),对于任意点集中一点 a ,一定有点集中某点 b (b 可以和 a 相同),使得 b 是 a 关于 X 的对称点。此时称 X 为点集的中心。现给定平面内 n 个点的集合,你可以添加最多 m 个点(也可以不添加),使得添加后的点集是中心对称的。问有多少个不同的点可能成为对称中心,并将他们的坐标输出。如果可能有无穷多个对称中心,输出 -1。 $n \leq 2 \times 10^4, m \leq 10$ 。

算法简述

显然,如果 $n \leq m$,那么输出 -1。

否则,只要能确定所有可能的对称中心,就可以 $O(n)$ 用 Hash 验证了。

将所有点按 x 坐标排序。那么可能的对称中心只可能是前 $m+1$ 个点与后 $m+1$ 个点自由组合得到的 $(m+1)^2$ 个中点。因为即便添加 m 个 x 坐标极小的点,第 $m+1$ 个点也能与倒数第 $m+1$ 个点配对。否则会与后 m 个点配对。倒数第 $m+1$ 个点同理。于是便解决了。

8.6 147B. Smile House

题目大意

给出一个 n ($n \leq 150$) 个点边上带权的有向图。求正权环(环上路径和为正)的最小长度。

算法简述

修改矩阵乘法,原来是:

$$c[i][j] = \sum a[i][k] \times b[k][j]$$

修改为:

$$c[i][j] = \max\{a[i][k] + b[k][j]\}$$

可以验证修改后的矩阵乘法依然满足结合律。于是也能进行快速幂。(类似与 $O(n^3 \log n)$ 的 APSP 最短路)

现在简化问题。设 E 是边权的邻接矩阵。我们要求的是最小的 t , 满足 $F = E^t$ 中存在 k , $F[k][k] > 0$ 。运用倍增即可求出。(类似倍增 LCA)。复杂度为 $O(n^3 \log n)$ 。

8.7 152D. Frame

题目大意

在一个 $n \times m$ 的棋盘上画两个矩形边框, 可以任意重叠, 甚至重合。现在给你最终的图案, 问是否可能通过这样画两个矩形边框的方式得到。
 $n, m \leq 1000$ 。

算法简述

如果一行或者一列有超过 3 个连续的 #, 那么必然是一个矩形的边界。确定两个矩形边界后, 可以 $O(n)$ 验证。

除了一个例外情况: 一个矩形的长或宽为 3。不难发现, 这种情况下, 边界一定是在最大(小), 次大(次小)上。单独处理即可。

8.8 183D. T-shirt

题目大意

有 n 个人, m 种衬衫尺寸。你要带 n 件衣服分配到每个人身上。但是你只知道第 i 个人穿 j 尺寸衬衫的概率是 $p[i][j]$ 。求一种方案, 使期望配对上的衬衫数目最大。

算法简述

由于衬衫尺寸之间是独立的, 我们可以先计算这样一个数组。定义 $f[i][j][k]$ 表示对于 i 尺寸, 前 j 个人中至少 k 个人是这个尺寸的概率。转移:

$$f[i][j][k] = (1 - p[j][i]) \times f[i][j-1][k] + p[j][i] \times f[i][j-1][k-1]$$

答案可以贪心得得到, 即 f 数组中最大的 n 项对应的期望。不过这样的复杂度会达到 $O(n^2 m)$, 无法承受。

注意到只需求出最大的 n 项, 可以先对于所有 i, k , 求出 $f[i][1][k]$ 。因为 $f[i][j][k] \leq f[i][j][k-1]$, 所以每次只需选出当前最大的 i , 然后扩展出下一项。扩展过程可以做到 $O(n)$ 。因为:

$$\begin{aligned} f[i][j][k] &= (1 - p[j][i]) \times f[i][j-1][k] \\ &+ p[j][i] \times f[i][j-1][k-1] \text{ (已知)} \end{aligned}$$

, 而:

$$\begin{aligned} f[i][j-1][k] &= (1 - p[j-1][i]) \times f[i][j-2][k] \\ &+ p[j-1][i] \times f[i][j-2][k-1] \text{ (已知)} \end{aligned}$$

如此下去, 未被计算的只有 $O(n)$ 个状态。于是这样下来, 总复杂度为 $O(n^2)$ 。

8.9 217E. Alien DNA

题目大意

给出一段仅有 A, C, G, T 组成的长为 $n (n \leq 3 \times 10^6)$ 的字符串 s 。一次操作指给出区间 $[l, r]$ $l, r \leq 10^9$, 然后在 r 之后插入交替变换组成的长为 $r - l + 1$ 的字符串: $s_{l+1}s_{l+3} \cdots s_ls_{l+2} \cdots$ 。现在给出所有操作 (总数不超过 5000), 求最终序列的前 $k (k \leq 3 \times 10^6)$ 个字符。

算法简述

倒着考虑原问题。可以确定最后一次操作的位置是 $[l, r]$, 产生的是 $[r+1, r+r-l+1]$ 这一段。那么可以将 $[r+1, r+r-l+1]$ 截去, 得到不包含最后一次操作的序列。于是可以递归做下去。

然后考虑用数据结构来维护。偷懒的办法是用 STL 中的 rope, 支持 string 支持的所有操作, 且复杂度为根号级别。

8.10 135E. Weak Subsequence

题目大意

对于一个字符集大小为 k 的字符串 s , 如果称它是合法的要求其中最长的子串同时也是 s 的弱子序列的长度为 w 。如果说长为 n 的字符串 a 是长为 m 的字符串 b 的弱子序列, 只存在 $1 \leq i_1 < i_2 < \cdots < i_n \leq m$, 且:

$$\forall 1 \leq k \leq n, a_k = S_{i_k};$$
$$\exists k, i_{k+1} - i_k > 1。$$

求合法字符串的个数模 $10^9 + 7$ 。 $k \leq 10^6, w \leq 10^9$ 。

算法简述

弱子序列的定义比较奇怪, 可以从这里入手。容易发现, 强化其定义, 即设从 l 到 r 的子串是原串的弱子序列, 等价与存在 $k > r$, 且 $s[k] = s[r]$, 或者存在 $k < l$, 且 $s[k] = s[l]$ 。这样以来只与 l 和 r 中的一个有关了。即, 为原串的前缀或后缀, 且前缀 (以前缀为例) 的最后一个元素在之后还有出现。

进一步放宽限制, 最长合法前缀的长度不超过 t 等价于字符串后 $n - t$ (n 为长度) 个元素互不相同。设计函数 $f(n, a, b)$ 为长度是 n , 前 a 个字符互不相同, 后 b 个字符互不相同的字符串个数。 $f(n, a, b)$ 很容易通过组合基本知识预处理后在 $O(1)$ 求出。于是, 枚举长度 n , 答案为 $f(n, n - w, n - w) - f(n, n - (w - 1), n - (w - 1))$ 。这种放宽限制的思想没有任何特殊情况需要考虑, 实在巧妙。

9 Volume IX

9.1 163D. Large Refrigerator

题目大意

对于确定体积的长方体,求其中表面积最小的三边长。体积 $V \leq 10^{18}$, 以质因数分解的形式给出。

算法简述

设所求长方体三边长为 $a, b, c, a \leq b \leq c$ 。首先枚举 a 的取值。问题变成了对于 V/a , 求 b, c , 满足 $bc = V/a$ 且 $b+c$ 最小。这里可以对 V/a 记忆化剪枝。最优化剪枝即当 $b=c$ 时可以取得 $b+c$ 的最小值, 如果仍然大于当前答案, 则没必要继续下去。枚举 a 最好从大到小枚举, 便于最优化剪枝。然后差不多就可以过了。

9.2 167E. Wizards and Bets

题目大意

有一个 n 个点 m 条边的 DAG。没有入边的点为源, 没有出边的点为汇, 保证源和汇的数目是相同的。现在要从图中选出 k 条路径, 分别从某个源出发到达某个汇。每个源汇都恰被一条路径覆盖一次, 任何路径不在顶点处相交。
假设终止于 i 号汇的路径是由源 a_i 出发的, 我们称汇对 (i, j) 是一个逆序对当且仅当 $i < j$ 且 $a_i > a_j$ 。如果所有汇对 (i, j) ($1 \leq i < j \leq k$) 中的逆序对总数是偶数, 那么计数器加 1, 否则减 1。
现在, 对于所有合法的路径集合, 都要对计数器产生影响。问最后计数器的值是多少。 $n \leq 600, m \leq 10^5$ 。

算法简述

首先要对“任何路径不相交”下手。假设 $u \rightarrow p, v \rightarrow q$ 两条路径于 t 点相交, 那么更改 $u \rightarrow p = u \rightarrow t + t \rightarrow q, v \rightarrow q = v \rightarrow t + t \rightarrow p$, 这两种情况奇偶性不同对答案的贡献恰好为 0。因此这个条件可以直接无视。

dp 预处理出从任意一个源到任意一个汇的方案数。于是最终答案为:

$$\sum_{g \in \text{all permutation}} \text{sgn}(g) \cdot \prod_{1 \leq i \leq n} f[i][g[i]]$$

其中, g 是枚举所有配对方案, $\text{sgn}(g)$ 是 g 中逆序对数的奇偶性对应的权值 $(+ - 1)$, 后面的乘积是情况数。

可以发现, 这个式子就等于 f 的行列式。

行列式 $O(n^3)$ 的求法: 将行列式当成矩阵进行高斯消元。交换两行对应行列式取倒数。将某行乘上一个系数对应行列式乘同样的系数。行与行之间做减法行列式不变。最后消成上三角矩阵。而三角矩阵行列式为对角线乘积。

9.3 232D. Fence

题目大意

给定一个序列 a_1, a_2, \dots, a_n 以及 q 个询问 $(l_1, r_1), (l_2, r_2), \dots, (l_q, r_q)$ 。对于每个询问 (l_i, r_i) , 回答 (l_j, r_j) 二元组的个数, 满足 $r_j - l_j + 1 = r_i - l_i + 1$, 且 $[l_j, r_j] \cap [l_i, r_i] = \emptyset$, 且对于所有 $0 \leq k \leq r_i - l_i$, 使 $a[l_i + k] + a[l_j + k] = a[l_i] + a[l_j]$ 。 $n \leq 10^5$ 。

算法简述

先将原序列相邻两项做差得到长度为 $n - 1$ 的 b 序列,就变成了询问一段取反后相同的且不相交的段数。构造一个 s 序列为 $b\$ - b$, 然后就变成了经典的后缀数组离线询问的问题了。复杂度为 $O(n \log n + q \log n)$ 。

9.4 175E. Power Defence

题目大意

在一个塔防游戏中, boss 从 x 轴的负无穷走到正无穷, 可以在 $y = 1$ 和 $y = -1$ 两条直线上的整点处建三种塔: 火、电、冰。每种塔有一个攻击半径和每秒伤害(除冰塔外)。在某一时刻, boss 走到 k 个冰塔的攻击半径内, 则移动速度减为原来的 $1/(k + 1)$ 。给定每种塔的总数 (≤ 20), 求最大化伤害的方案。

算法简述

首先, 可以把 $y = -1$ 的塔对称到 $y = 1$ 上, 问题变成了在 $y = 1$ 上每个整点可以修不超过 2 个塔。可以发现, 塔必然是连续的一段, 且不能有相邻位置都只有 1 个塔, 所以可以暴力出冰塔的修建方案。之后进行 dp: 用 $f[i][j][k]$ 表示在前 i 个位置修了 j 个火, k 个电塔, $O(1)$ 转移。

优化: 所有塔肯定必须用, 塔连续一段的长度 ≤ 13 (实际上 10 就够了)。

9.5 176D. Hyper string

题目大意

计算一个 *Hyper string* 和一个长度不超过 2000 且仅由小写字母构成的字符串的最长公共子序列。*Hyper string* 是由多个基本字符串连接而成的。每个基本字符串长度不超过 2000, 且长度之和不超过 10^6 。

算法简述

首先考虑一个简化版问题: 计算一个长度不超过 2000 的字符串 s_1 和一个长度不超过 10^6 的字符串 s_2 的 LCS。传统的 dp 方法复杂度将达到 $O(n^2)$, 不能承受。注意到题目特殊性, 答案不是很大, 且仅由 26 个小写字母构成。可以换一种状态定义方式。定义 $f[i][j]$ 表示 s_1 进行到 i , 答案为 j , 在 s_2 中最靠前的位置。转移只需看 $s_1[i + 1]$ 在 $s_2[f[i][j]]$ 之后第一次出现的位置。预处理出 s_2 中每个位置之后第一个出现每个字母的位置, 可以使复杂度做到 $O(n^2)$ 。

回到原问题, 不难发现本质做法一样。唯一区别在于 $f[i][j]$ 记录两个东西, 一个是基本字符串编号, 一个是在基本字符串中的位置。转移类似, 预处理也类似, 复杂度也是 $O(n^2)$ 。

9.6 178F. Representative Sampling

题目大意

有 n ($n \leq 2000$) 个由 26 个小写字母组成的字符串, 要求从中选出 k 个, 使该集合的权值最大。设 S 为选择的集合, 则权值定义如下:

$$\sum_{i,j \in S} i, j \text{ 最长公共前缀的长度}$$

算法简述

初一看, 很容易想到用 trie 这种前缀处理工具。将所有字符串插入 trie 中后, 变成了标记 k 个点, 每个点的权值定义为 $m(m - 1)$, m 是子树中标记点的数目。于是可以设计一个状态 $O(n^2)$, 转移 $O(n)$ 的树型 dp, 总复杂度为 $O(n^3)$ 。走到这里就难以继续走下去。只好另辟蹊径。

联想到后缀数组求 lcp 的方法, 可以将所有字符串按字典序排序。设计状态 $f[l][r][k]$ 表示区间 $[l, r]$ 选 k 个的答案。于是:

$$f[l][r][k] = \max_{l \leq t \leq k} \{f[l][p][t] + f[p+1][r][k-t] + \text{something}\}$$

注意到 p 取值的随意性, 不妨取相邻两字符串 LCP 最小的那个位置。于是:

$$f[l][r][k] = \max_{l \leq t \leq k} \{f[l][p][t] + f[p+1][r][k-t] + t(k-t) \cdot \text{lcp}[p]\}$$

有用的区间只有 $O(n)$ 个。复杂度比较难算, 考虑两类极端情况: $p = l+1$, 转移是 $O(1)$ 的, 总共是 $O(n^2)$; $p = (l+r)/2$, $T(n) = (n/2)^2 + 2T(n/2) = O(n^2 \log n)$ 。是一个十分优秀的算法。

9.7 178E. The Beaver's Problem II

题目大意

给出一个黑白图片 (2000×2000), 里面仅包含正方形和圆, 可能经过任意次旋转。每个像素有 20% 概率反色。统计正方形和圆的数目。

算法简述

这是一道非常偏向实际应用的图像处理题。整个算法分两步: 噪音消除和模糊统计。

噪音消除的方法: 对于每个像素, 如果四周有白色像素, 那么在新图中为白色, 否则为黑色。然后再对新图黑白颠倒做一遍。剩下的图形噪音就不是很多了。

然后 bfs 出所有黑色连通块。如果大小太小说明是噪音, 直接删掉。否则判断是正方形还是圆。

这可以利用圆和正方形的差异来解决。首先算出中心, 用平均坐标即可。然后计算半径, 即离中心最远点距离, 设为 r 。对于圆来说, 落在 $\frac{\sqrt{2}}{2}r - r$ 的点数接近总点数的一半, 而正方形则很少。就可以通过 CF 上的数据了。

9.8 180B. Divisibility Rules

题目大意

如果检查一个数能否被 $2, 4, 5, 8, 10$ 整除的时候只需要检查它的最后一位或几位是否满足某个条件。这种规则称为 2 类型规则 (2-type)。如果检查一个数能否被给定的数整除意味着计算这个数的各个数位上的数字之和并判断这个和能否被给定的数整除, 那么称这个规则为 3 类型规则 (3-type)。如果我们要求出一个数的奇数位上的数字之和与偶数位上的数字之和的差值去检查这个差值能否被给定数整除, 那么这个规则被称为 11 类型规则 (11-type)。有些情况下我们应当把除数分解成一些因数然后检查是否满足一些不同类型的规则 ($2\text{-type}, 3\text{-type}, 11\text{-type}$)。这样混合的整除规则被称为 6 类型规则 (6-type)。最后, 有些除数是所有类型的规则都无效的, 称在这种情况下神秘的 7 类型规则 (7-type) 有效。要求出 b 进制下除数为 d 的整除规则类型。 $b, d \leq 100$ 。

算法简述

将一个数 x 写成 $b^0d_0 + b^1d_1 + b^2d_2 \dots$ 的形式。对于每种类型分别讨论:

1. 2-type : 意味着 $\exists k, b^k = b^{k+1} = \dots \equiv 0 \pmod{d}$ 。通过不断乘 b 知道能被 d 整除来判断。
2. 3-type : 意味着 $\forall i, b^i \equiv 1 \pmod{d}$, 即 $b \equiv 1 \pmod{d}$ 。
3. 11-type : 意味着 $\forall i, b^{2i} \equiv 1 \pmod{d}, b^{2i+1} \equiv -1 \pmod{d}$, 或者反过来。即 $b \equiv -1 \pmod{d}$ 。
4. 6-type : 分解开, 然后判断。

9.9 185D. Visit of the Great

题目大意

给出 $l, r, p (l, r \leq 10^{18}, p \leq 10^9)$, 计算 $\text{lcm}(k^{2^l} + 1, k^{2^{l+1}} + 1, \dots, k^{2^r} + 1) \bmod p$, p 是质数。 10^5 组数据。

算法简述

首先尝试简化 lcm 。可以发现, $\text{gcd}(k^{2^i} + 1, k^{2^j} + 1) = \text{gcd}(k^{2^i} + 1, 2)$ 。于是若 k 是奇数, 那么所有数都含 2 这个质因子, 且公共的质因子仅有 2; 若是偶数, 那么两两互质。考虑偶数情况:

$$\text{lcm} = (k^{2^l} + 1) \cdot (k^{2^{l+1}} + 1) \cdots (k^{2^r} + 1) \quad (1)$$

$$= \frac{k^{2^{r+1}} - 1}{k^{2^l} - 1} \quad (2)$$

$$= (k^{2^l} + 1) \cdot (k^{2^{l+1}} + 1) \cdots (k^{2^r} + 1) \quad (3)$$

如果 (2) 中分母模 p 为 0, 那么通过 (3) 式可以得到答案为 $2^{r-l} \bmod p$, 否则直接逆元。奇数情况类似, 不再赘述。

9.10 187D. BRT Contract

题目大意

一条路上有 n 个红绿灯, 共 $n+1$ 段路。起点为左端点, 终点为右端点。给出每段路的长度。已知所有红绿灯的绿灯周期均为 g , 红灯周期均为 r , 且在 $[0, g)$ 时刻 (模 $g+r$ 意义下) 区间内为绿灯, $[g, g+r)$ 时刻区间内为红灯。现在有 q 辆公交车从起点出发, 开向终点, 第 i 辆公交出发时刻为 t_i 。求所有公交到达终点的时刻。 $n \leq 10^5$ 。

算法简述

可以发现, 一旦公交停在某个红绿灯前, 下一次出发又是从 0 时刻开始。因此, 可以预处理一个数组 f , 其中 $f[i]$ 表示 0 时刻在第 i 个红绿灯前, 到达终点的时刻。这个可以通过递推来得到。假设已经算出了 $k+1 \cdots n$ 的 f 值。在计算 $f[k]$ 时, 只需知道从 k 出发第一个遇到的红灯是第 t 个, 那么 $f[k] = k$ 到 t 距离 + 等待时间 + $f[t]$ 。利用前缀和可以算出从 k 到之后任意位置不停留的路程, 找到最靠左的落在 $[g, g+r)$ 区间内就是停靠的红绿灯。这可以用线段树来实现。

对于从起点出发的公交车处理方法是一样的, 同样只需找到第一个停靠位置就能推出答案了。

10 Volume X

10.1 176E. Archaeology

题目大意

给出一棵 $n(n \leq 10^5)$ 个节点, 边带权的树。节点有黑白两色。一开始所有点都是白色。有 $q(q \leq 10^5)$ 个操作, 共三种形式:

1. 将一个节点染为黑色
2. 将一个节点染为白色
3. 询问使所有黑色节点两两互达的最小权边集是多少。

算法简述

显然, 答案就是两两黑色节点之间路径的并。我们需要找一个统计方法, 满足并中每条边都被算到恰好一次, 且不在并中的边都不被算到。

将黑色节点按照 dfs 序排序。那么答案就是每个黑点到它和它前面一个黑点最近公共祖先的路径。第一个点的前面一个点为最后一个点。证明采用反证。显然每条并中的边都会被算到。假设一条边被计算了两次以上, 那么两个本应相邻的黑点中间一定夹着另外一个黑点, 产生矛盾。

充分利用 STL 的 set 可以在 $O(q \log n)$ 内维护增量圆满解决这题。

10.2 196D. The Next Good String

题目大意

给一个仅由小写字母组成的字符串 S 和一个正整数 m 要求一个长度与 S 相同的仅由小写字母组成的字符串 S_1 满足以下要求

1. S_1 的字典序大于 S
2. S_1 不包含长度大于等于 m 的回文子串

$|S| \leq 10^5, m \leq 10^5$ 。

算法简述

这题只是个障眼法。直接搜索就行。因为一旦某个位置比原来大, 那么后面其实是可以直接构造出来的, 就不会回溯了。搜索过程用 hash 来验证合法性比较快, 递归的时候直接记下最后 m 和 $m+1$ 位正着和反着的 hash 值。复杂度是 $O(26|S|)$ 。

10.3 198E. Gripping Story

题目大意

在一个二维平面上, 有 n 块散落的磁铁。一开始你的手中也有一块。每个磁铁都可以抽象成一个点, 目标是吸引最多的散落的磁铁。每一块磁铁都有五个属性: x, y, m, p, r , 分别表示磁铁的横坐标, 纵坐标, 重量, 吸引力和吸引半径。

一块磁铁想要把另一块磁铁吸过来的条件是:

1. 被吸引的磁铁和吸引的磁铁之间的距离小于等于吸引磁铁的吸引半径。
2. 被吸引的磁铁的重量小于等于吸引磁铁的吸引力。

任何被吸过来的磁铁都可以用来吸引新的磁铁。每块磁铁可以吸引无数多次, 同时你的位置 (x, y) 也是不变的。现在你想知道, 你最多可以吸引多少散落的磁铁。

算法简述

将所有磁铁按到 (x, y) 的距离从小到大排序。那么每次满足条件 1 的磁铁是区间 $[1, k]$ 。要找到同时满足条件 2 的磁铁, 应在 $[1, k]$ 中寻找。于是可以用树套树在 $O(n \log^2 n)$ 内解决。

注意到条件 2 的特殊性, 即若区间 $[1, k]$ 中重量最小的磁铁也大于吸引磁铁的吸引力, 那么所有磁铁都不能被吸走。反之, 吸走重量最小的。由于每个磁铁只能被吸走一次, 故若用线段树维护区间最小值, 可以在 $O(n \log n)$ 时间内解决。

10.4 200E. Tractor College

题目大意

给出 4 个常数 c_3, c_4, c_5, W , 求 w_3, w_4, w_5 满足 $w_3 + w_4 + w_5 = W, 0 \leq w_3 \leq w_4 \leq w_5$, 且使下式最小化:

$$|w_3 c_3 - w_4 c_4| + |w_4 c_4 - w_5 c_5|$$

保证 $c_3, c_4, c_5 \leq 1$ 且 $c_3 + c_4 + c_5 \leq 300$, 以及 $w_3 + w_4 + w_5 \leq 3 \cdot 10^5$ 。

算法简述

为了减轻思维负担, 可以先枚举一个 w 。由于 w_4 在式子里出现了两处, 可以枚举它。之后, w_5 可以写成 $W - w_4 - w_3$, 于是只有一个未知量了。

用 x 代替 w_3 , 目标式可以简化成下列函数:

$$y = |k_1 x + b_1| + |k_2 x + b_2|$$

可以看成是 $y_1 = |k_1 x + b_1|$ 与 $y_2 = |k_2 x + b_2|$ 的叠加。不难发现, $y_1 + y_2$ 是个单峰函数, 峰值即为最小值。故可以在整数上三分这个最小值。整个算法复杂度为 $O(M \log M)$ 。

10.5 200A. Cinema

题目大意

给你一个 n 行 m 列的 01 矩阵 A , 每个元素初始值为 0, 再给你 k 个这样的操作: 给出 (x_i, y_i) , 你得按要求找出 (a_i, b_i) , 将矩阵 A 的点 (a_i, b_i) 赋值为 1, 并且输出 (a_i, b_i) 。要求如下:

1. $A(a_i, b_i) = 0$ 。
2. 满足 1 的情况下, (a_i, b_i) 与 (x_i, y_i) 的曼哈顿距离尽可能小。
3. 若存在多个 (a_i, b_i) 满足条件 2, 则选出 a_i 最小的。
4. 若存在多个 (a_i, b_i) 满足条件 3, 则选出 b_i 最小的。

$n, m \leq 2000, k \leq \min(10^5, nm)$ 。

算法简述

暴力枚举的话每次复杂度是 $O(k)$ 。注意到偏移的行数是 $O(\sqrt{k})$ 级别的(斜正方形)。大体思路是, 加快每行枚举速度。具体来说, 维护每个格子往左往右第一个值为 1 的位置。如果使用并差集, 这一步可以做到近似 $O(1)$ 。故整个复杂度为 $O(k\sqrt{k})$ 。

10.6 201E. Thoroughly Bureaucratic Organization

题目大意

有一个长度为 n 的排列。为了确定这个排列,你可以做出若干次询问。每次询问给出 m 个 1 到 n 的数(不能重复),返回这些数在排列中位置的集合。求最少询问次数来确定排列。数据组数 ≤ 1000 , $n, m \leq 10^9$ 。

算法简述

直接求解信息量太少,不妨由反切入。设计一个函数 $f(m, k)$, 返回最大的 n , 使得询问大小为 m , 询问 k 次能确定的最多的数为 n 。如果设计成功, 那么通过二分就能求得 k 了。

然后抽象原问题模型如下:

有一个 $k \times n$ 的 01 矩阵。每行代表一次询问, 每列对应一个排列中的数。若第 (i, j) 为 1, 则表示第 j 个数出现在了第 i 次询问中。可以发现, 确定排列等价与确定每个数的位置, 即任意两列不同。因为一个数 t 的位置可以看作是 t 列所有 1 的行的位置集合的并, 再减去 t 列所有 0 的行。若两列相同, 那么这两个数将无法区分了。 m 的限制即要求每行 1 的个数不超过 m 。

定理 10.6.1 (弱化限制) 每行 1 的个数不超过 m 等价于 1 的总数不超过 km 。

证明 10.6.1 假设 1 的总数不超过 km , 设 x 为 1 的个数最多的行, y 为 1 的个数最少的行。显然, 如果不满足“每行 1 的个数不超过 m ”, 那么 $x > m, y < m$ 。取列 r , 使 $(x, r) = 1, (y, r) = 0$ 。然后交换 (x, r) 与 (y, r) , 仍然满足条件。一直进行下去, 一定能达到“每行 1 的个数不超过 m ”。

接下来的问题就明了了。我们一列一列的贪心构造, 一次放长度为 k , 1 的个数为 $0, 1, 2, \dots$ 的 01 字符串, 直到总共 1 的个数刚好不超过 km 为止。综上, 整个算法复杂度为 $O(\log^2 n)$ 。

10.7 201D. Brand New Problem

题目大意

现在有 n 个单词组成的一句话 s_0 , 以及 m 个由若干单词组成的长句子 s_i , 若 s_0 的某一个排列是长句子 s_i 的子序列, 则称 s_0 同 s_i 相似。定义两者的差异度 p 为满足相似条件的排列的逆序对数的最小值。现在需要你求出在 m 个长句子中与 s_0 差异度最小的句子, 若存在多个, 则取编号最小的句子。 $n \leq 15, m \leq 10$, 所有长句子长度之和不超过 500000。

算法简述

由于 m 很小, 可以每个长句子独立开来做。然后将单词离散化。问题变成了求一种排列, 是长句子的子序列, 且逆序对数最少。

这个可以用状态压缩 dp 来解决。定义 $f[i][s]$ 表示长句子进行到第 i 个数, s_0 中已出现的数字集合是 s , 值为最小逆序对数。但是光状态就太大。注意到答案很小, 可以使用 dp 优化的惯用伎俩: 交换答案和状态。于是定义一个新的状态 $f'[s][k]$, s_0 意义不变, k 为逆序对数, 值为长句子中最早符合条件的位置。转移枚举下一个出现的 s_0 中的数, 通过预处理可以使整个复杂度做到 $O(2^n n^3)$ 。

10.8 204E. Little Elephant and Strings

题目大意

给定 n 个字符串, 询问每个字符串有多少子串是所有 n 个字符串中至少 k 个字符串的子串。 $n, k \leq 10^5$, 且总长不超过 10^5 。

算法简述

先不管三七二十一建后缀数组,再在此基础上思考算法。

后缀数组中每个元素 $sa[i]$ 是某个字符串的后缀,那么所有的子串就对应后缀数组中元素的前缀。

对 sa 中某个元素 i 单独考虑:通过二分我们可以找到一个位置 j ,使得 i 的任意长度不超过 j 的前缀都对答案有贡献。问题就变成了判断前缀 j 是否在至少 k 个字符串中出现。可以找到最大的区间 $[l, r]$, $lcp(l, r) \geq j$ 。 l, r 可以通过二分得到。然后,要求 $[l, r]$ 中出现的不同所属字符串的个数。

注意到一个条件没有利用: k 是固定的。于是可以想到对后缀数组中每个位置 i 预处理 $f[i]$,即最靠前的位置,使区间 $[f[i], i]$ 不同所属字符串的个数不小于 k 。注意到如果 $i < j$,那么 $f[i] < f[j]$ 。可以使用单调队列来预处理。

最后,判断 $[l, r]$ 是否可行,只需满足 $f[r] \geq l$ 。整个算法的复杂度为 $O(n \log n)$ 。

10.9 207B. Military Trainings

题目大意

有 n 个坦克排成一排。设从左往右第 i 个位置的坦克编号为 $a[i]$ 。每个初始时 $a[i] = i$ 。一共要进行 n 个回合的演练。每次演练如下:

1. 坦克 $a[1]$ 接受到一段信息,开始信息传输。第 i 个坦克能将信息传给从左往右第 j 个坦克的条件是 $i < j$ 且 $i + r[a[i]] \geq j$ 。每次传输耗时为 1。当坦克 $a[n]$ 接受信息后传输停止。
2. 完成传输后,最右侧的坦克将移到最左侧,其余坦克往右顺移。

求最小演练总时间。 $n \leq 2.5 \times 10^5$ 。

算法简述

如果只计算一次传输耗时,可以通过简单的一维 dp 在 $O(n^2)$ 时间内实现:

$$f[i] = \min_{j < i \text{ and } i + r[a[i]] \geq j} \{f[j] + 1\}$$

通过数据结构可以优化至 $O(n \log n)$ 。但是,这种思路已经走到尽头,因为我们没有利用原题的特殊性。

特殊性在于,坦克构成一个环。于是可以考虑破坏成链。将坦克队列扩充成 $2n$,第 $n + i$ 为第 i 个坦克。于是,问题变成了求 $\sum_{1 \leq i \leq n} ans(i, n + i - 1)$ 。

这时候可以运用倍增的思想。运用倍增思想最核心的部分在于确定从第 i 个点往后跳到哪个点最优。由于 i 覆盖的范围是 $(i, i + r[i]]$,那么最优点一定是 j ,使 $j + r[j]$ 最大。因为其它点都不能覆盖到 $j + r[j]$ 之外的点,且其他点下一步能到的点 j 也能到。(除了 (i, j) 内的点,但跳到其中是没有意义的)。

接下来就容易了。定义 $f[k][i]$ 表示从 i 开始跳 2^k 步,到 j 使 $j + r[j]$ 最大。转移是 $f[k][i] = f[k-1][f[k-1][i]]$ 。算答案的时候基于贪心:尽可能往远跳,直到越过结束点。整个算法复杂度是 $O(n \log n)$ 。

10.10 207A. Beaver's Calculator

题目大意

给出一个 $n \times m$ ($n, m \leq 5000$) 的矩阵 A 。要求构造一个同样大小的 B 矩阵。 B 矩阵中的元素能与 $[1, nm]$ 中的数一一对应,且满足 $\forall i, j_1 < j_2, B[i][j_1] < B[i][j_2]$ 。在所有合法 B 矩阵中,选逆序对数最小的。
 B 矩阵的逆序对数定义为:将 A 中元素按 B 编号从小到大取出的序列中逆序对数。若 $nm \leq 200000$,输出 B ,否则输出最小逆序对数。

算法简述

首先想到一个贪心的算法:每行记录一个指针 $p[i]$,表示第 i 行 $p[i]$ 之前的格子已经赋好值了。同时记录上一个赋值的格子在 A 中对应的值 x 。如果 x 大于所有指针指向格子的 A 值,那么选其中 A 值最小的赋值,然后相应的 $p[i]$ 加 1,同时逆序对数加 1;否则找到最小的不小于 x 的 $p[k]$ 加 1,逆序对数不变。用平衡树(STL set)可以实现,复杂度是 $O(nm \log(nm))$ 。若 $nm \leq 200000$ 就可以使用此法。

若 $n, m \leq 5000$,只统计个数,又怎么解决?注意到顺序是不影响。于是,设 $p[i]$ 对应 B 中最小的一个是 $p[r]$ 。那么对每个 i 找出从 $p[i]$ 开始的最长不下降子段,全部添加到答案。由于 $p[r]$ 是最小的,每行的 p 至少都会加 1,因此最多进行 n 次。故可以在 $O(nm)$ 的时间内解决求最小逆序对数。

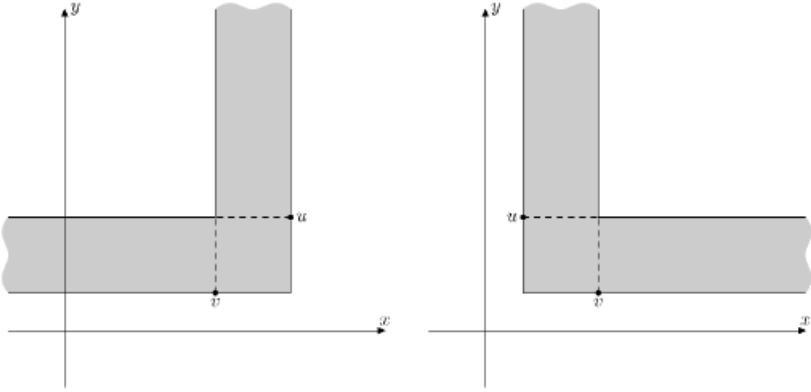
11 Volume XI

11.1 167D. Wizards and Roads

题目大意

给出平面上 $n(n \leq 10^5)$ 个点, 保证点的分布随机, 且任意两点 x, y 坐标均不同。 u, v 两点之间能连边的条件是 $u_y > v_y$, 且任何严格在点 u, v 形成的拐角中的点 w , 都有一个点 s 并不在 u, v 形成的拐角中, 并且 $w_x < s_x < u_x$, 同时 $s_y > v_y$ 。

u, v 形成的拐角的定义如下图阴影部分:



现在给出 $m(m \leq 10^5)$ 个询问 $[L_i, R_i]$, 表示将横坐标在 $[L_i, R_i]$ 的点取出, 按上述规则两边得到的无向图中的最大匹配数。

算法简述

首先, 需要简化连边的条件。不难发现: 对于每个 u , 找到其右第一个纵坐标大于 u_y 的点 s , 那么 u 往右只能跟 s 左边第一个纵坐标小于 u_y 的点 v 连边。由于每个点最多往下连两条边, 所以形成的图是一个以纵坐标最大的点为根的二叉树, 且期望深度为 $O(\log n)$ 。

按以上规则建好树后, 考虑如何回答询问。求树的最大匹配树容易联想到一个简单的 dp: $f[i][0/1]$ 表示以 i 为根的子树中, i 是否匹配的最大匹配数。递推也非常容易。

但原问题还有一个限制: 每次询问的树的形态都是不一样的。但新树可以通过以下操作得到: 对每个在 $[L_i, R_i]$ 内的点 x , 它的父亲修改为最近的一个祖先 y , 且 y 也在 $[L_i, R_i]$ 内。这样修改后有一个重要的性质没有变化: 即答案仍然可以通过合并左右子树来得到。于是, 最终算法就是建一个类似于线段树的树, 修改合并函数, 保证合并的子树都在区间内。整个算法的复杂度为 $O(m \log n)$ 。

11.2 209C. Trails and Glades

题目大意

有一个 n 个点 m 条边的无向图, 可能存在自环, 重边。问是否能从 1 号点出发, 走边所有边正好一次, 并回到 1 号点。如果不能, 问最少添加多少条边能使这样的走法存在。 $n, m \leq 10^6$ 。

算法简述

联系到欧拉回路, 判定一个无向图存在题目描述的合法回路的充要条件是所有非零度点连通且每个点度数为偶数。然后开始设计算法。

先将零度点删掉(除了 1 号点)。求出每个连通块中奇度点的个数, 排序得到 a 数组。记 a 中 0 的个数为 $evenCnt$, 然后从 a 中剔除掉。每次新增一条边连接两个 a 中的连通块 i, j , 会使连通块个数减 2, 且新增一个奇度点数目为 $a[i] + a[j] - 2$ 的连通块。如果每次将 a 最大的两个元素合并, 放在队尾(如果 $a[i] = a[j] = 1$, 直接删掉令 $evenCnt++ = 1$), 可以发现依然是排好序的。

如果 a 中还剩一个,那么答案就是 $(a[0] - 2)/2 + evenCnt + 1$, 否则若 $evenCnt = 1$, 答案不变; 否则增加 $evenCnt$ 。

11.3 212B. Polycarpus is Looking for Good Substrings

题目大意

现在有一个的字符串 s , 由小写字母构成。再给你 m 个集合。对于每个集合, 求以下子串 $s[a, b]$ 的数量:

1. 子串中出现的字母必须出现在集合中。
2. 集合中的字母必须在子串中找到。
3. 不存在比这个子串更长的子串 $s[x, y]$, 使得 $s[x, y]$ 满足 1, 2 条件, 且 $x \leq a \leq b \leq y, y - x + 1 > b - a + 1$ 。

$|s| \leq 10^6, m \leq 10^4$ 。

算法简述

对于某个位置 i , 若以 i 为右端点, 能形成的集合数不超过 26 个。于是预处理每个位置往前的第一个 z 字母在哪个位置, 可以在 $O(26|s|)$ 内预处理出所有答案。有一个问题就是可能不能满足 3 条件, 即 i 还能往后扩展。可以这样解决: 一旦往前碰到 $s[i + 1]$, 就停止扩展。

但是这样空间复杂度高达 $2^{26}B = 256MB$ 。注意到询问不多, 可以先全部记在一个 hash 表里, 再算答案。

11.4 212D. Cutting a Fence

题目大意

有一条护栏, 由 $n(n \leq 10^6)$ 个木板组成。第 i 个木板的高度是 $a[i]$ 。现在要给护栏画矩形。若要在区间 $[x, x + k - 1]$ 画一个宽度为 k 的矩形 ($1 \leq x \leq n - k + 1$), 为了美观, 高度一定是这个区间里高度最低的木板。对于每个 $k(1 \leq k \leq n)$, 求矩形高度的期望。

算法简述

反方向考虑, 对于一个木板 i , 对答案的贡献是多少。首先可以算出 i 往左能扩展的最远距离 l , 往右能扩展的最远距离 r 。令 $m_1 = \max(l, r)$, $m_2 = \min(l, r)$ 。则:

1. 若 $1 \leq k \leq m_2 + 1$, 则贡献为 $k \cdot a[i]$;
2. 若 $m_2 + 2 \leq k \leq m_1 + 1$, 则贡献为 $(m_2 + 1) \cdot a[i]$;
3. 若 $m_1 + 2 \leq k \leq m_1 + m_2 + 1$, 则贡献为 $(m_1 + m_2 + 2 - k) \cdot a[i]$ 。

对于每个 k , 贡献可以写成 $tk + b$ 。维护两个树状数组, 分别记录 t 和 b , 可以在 $O(n \log n)$ 的时间复杂度内解决本题。

11.5 212C. Cowboys

题目大意

对于一个 AB 字符串, 进行一种操作。操作之后, 任意相邻的两个 AB 将变成 BA 。现在给出操作之后的序列, 求原来可能的字符串个数。长度为 3 到 100。

算法简述

这题明显是个 $O(n)$ 的动态规划,只是细节比较难处理。定义 $f[i][0/1]$ 表示构造到 i 位置, $i-2$ 之前的位置已经跟操作之后的序列对上了,第 $i-1$ 位置上是 A 或 B。枚举第 i 位,如果出现 AB 则变为 BA 。

环的处理:枚举首尾,再 dp。长度很小的时候可能会发生边界问题,不如写个暴力的来特殊处理。

11.6 213E. Two Permutations

题目大意

给出两个排列:一个是长度为 n 的 a 排列,另一个是长度为 m 的 b 排列。问有多少个 d ,使得 a 排列所有数加上 d 后是 b 排列的子序列。
 $n \leq m \leq 2 \times 10^5$ 。

算法简述

从小到大枚举 d 。维护一个数组 c 。一开始 $c[b^{-1}[i]] = i$ 仅当 $1 \leq i \leq d$ 。每次枚举到 d 时,将 $c[b^{-1}[d-n]]$ 设为 0, $c[b^{-1}[d]]$ 设为 d 。对 c 整个求 hash,设结果为 p 。设 a 的 hash 为 q 。那么当 $p = q + 11 \cdots 1 \times d$ 时, d 满足要求。用线段树实现,复杂度是 $O(m \log m)$ 。

11.7 217C. Formurosa

题目大意

有 n 个待确定的 01 变量。同时,有一个程序,输入恰好 k 个未知量的编号,可以返回某个算式的答案。
算式包含常数,以及 $|$ (or), $\&$ (and), \wedge (xor) 三种运算符。并且遵从以下语法: $s \rightarrow 0|1|?(s|s)|(s\&s)|(s\wedge s)$ 。? 即填充未知量的槽孔。问是否能通过优先次调用程序来确定所有变量的值。 $n \leq 10^6$, 算式长度不超过 10^6 。

算法简述

容易发现, n 是不影响的。这题的难点在于如何找到一个充要条件。这里直接给出如下:

定理 11.7.1 (充要条件) 存在将算式中每一个槽孔对应到一个 01 字符串 s 。用 $F(s)$ 表示代入 s 程序返回的值。那么能确定所有变量的充要条件是 $F(s) \neq F(-s)$, 其中 $-s$ 是 s 取反得到的值。

证明 11.7.1 如果不存在这样的 s , 那么将所有变量取反, 所有算式答案不变, 因此无法区分。

对于两个变量 a, b , 先将 s 中 0 位置填上 a , 1 位置填上 b 。设得到的答案为 t_1 。然后将 1 位置填上 a , 0 位置填上 b , 得到 t_2 。如果 $t_1 = t_2$, 那么 $a = b$; 否则, 若 $t_1 = F(s)$, 那么 $a = 0, b = 1$, 不然, $a = 1, b = 0$ 。

建立一个表达式树, 在树上 dp。记录三种情况的 s 是否存在:

1. $F(s) = F(-s) = 0$;
2. $F(s) = F(-s) = 1$;
3. $F(s) \neq F(-s)$ 。

复杂度是线性的。

11.8 229E. Gifts

题目大意

给出一个 $n \times m$ 的矩阵的矩阵 A 。你要给出 p 个行号,可能重复。设第 i 行出现了 $t[i]$ 此。接下来,对于行 i ,会从中随机取出 $t[i]$ 不同的列,收益增加 i 行上这些列对应的数。你会选择价值最高的 p 件礼物。求使收益最大的概率。 $n, m, p \leq 1000$, 保证 A 中非零项个数不超过 1000。

算法简述

将所有非零元素取出排序。后面的数对应行是必定要选的。中间可能会出现同一个值太多。设这个值为 w , 统计每行的 w 。预处理二项式系数 $\binom{n}{m}$ 。设计 dp: $f[i][j]$ 表示到了第 i 行, w 已经选了 j 个的概率。逐行递推即可。由于 A 中非零项不是很多, 故复杂度为 $O(np)$ 。