IOI2015 中国国家集训队第一次作业 第二部分：试题泛做

王逸松

Contents

1 CodeForces 235D Graph Game **13**
1.1 题目大意 . . . 13
1.2 算法讨论 . . . 13
1.3 时空复杂度 . . . 13

2 CodeForces 235E Number Challenge **13**
2.1 题目大意 . . . 13
2.2 算法讨论 . . . 13
2.3 时空复杂度 . . . 13

3 CodeForces 238D Tape Programming **14**
3.1 题目大意 . . . 14
3.2 算法讨论 . . . 14
3.3 时空复杂度 . . . 14

4 CodeForces 240F TorCoder **14**
4.1 题目大意 . . . 14
4.2 算法讨论 . . . 14
4.3 时空复杂度 . . . 15

5 CodeForces 241B Friends **15**
5.1 题目大意 . . . 15
5.2 算法讨论 . . . 15
5.3 时空复杂度 . . . 15

6 CodeForces 241E Flights **15**
6.1 题目大意 . . . 15
6.2 算法讨论 . . . 15
6.3 时空复杂度 . . . 16

7 CodeForces 241F Race **16**
7.1 题目大意 . . . 16

7.2 算法讨论 16
7.3 时空复杂度 16

8 CodeForces 243C Colorado Potato Beetle **16**
8.1 题目大意 16
8.2 算法讨论 17
8.3 时空复杂度 17

9 CodeForces 249D Donkey and Stars **17**
9.1 题目大意 17
9.2 算法讨论 17
9.3 时空复杂度 17

10 CodeForces 249E Endless Matrix **17**
10.1 题目大意 17
10.2 算法讨论 18
10.3 时空复杂度 18

11 CodeForces 253E Printer **18**
11.1 题目大意 18
11.2 算法讨论 18
11.3 时空复杂度 18

12 CodeForces 254D Rats **18**
12.1 题目大意 18
12.2 算法讨论 19
12.3 时空复杂度 19

13 CodeForces 256D Liars and Serge **19**
13.1 题目大意 19
13.2 算法讨论 19
13.3 时空复杂度 19

14 CodeForces 257E Greedy Elevator **19**
14.1 题目大意 19
14.2 算法讨论 20
14.3 时空复杂度 20

15 CodeForces 258D Little Elephant and Broken Sorting **20**
15.1 题目大意 20
15.2 算法讨论 20

15.3 时空复杂度 20

16 CodeForces 260E Dividing Kingdom **21**

16.1 题目大意 21

16.2 算法讨论 21

16.3 时空复杂度 21

17 CodeForces 261D Maxim and Increasing Subsequence **21**

17.1 题目大意 21

17.2 算法讨论 21

17.3 时空复杂度 21

18 CodeForces 261E Maxim and Calculator **22**

18.1 题目大意 22

18.2 算法讨论 22

18.3 时空复杂度 22

19 CodeForces 264E Roadside Trees **22**

19.1 题目大意 22

19.2 算法讨论 22

19.3 时空复杂度 22

20 CodeForces 266D BerDonalds **23**

20.1 题目大意 23

20.2 算法讨论 23

20.3 时空复杂度 23

21 CodeForces 266E More Queries to Array... **23**

21.1 题目大意 23

21.2 算法讨论 23

21.3 时空复杂度 23

22 CodeForces 268D Wall Bars **23**

22.1 题目大意 23

22.2 算法讨论 24

22.3 时空复杂度 24

23 CodeForces 269D Maximum Waterfall **24**

23.1 题目大意 24

23.2 算法讨论 24

23.3 时空复杂度 25

24 CodeForces 273 Dima and Game **25**
24.1 题目大意 25
24.2 算法讨论 25
24.3 时空复杂度 25

25 CodeForces 277D Google Code Jam **25**
25.1 题目大意 25
25.2 算法讨论 26
25.3 时空复杂度 26

26 CodeForces 280D k-Maximum Subsequence Sum **26**
26.1 题目大意 26
26.2 算法讨论 26
26.3 时空复杂度 27

27 CodeForces 283E Cow Tennis Tournament **27**
27.1 题目大意 27
27.2 算法讨论 27
27.3 时空复杂度 27

28 CodeForces 286D Tourists **27**
28.1 题目大意 27
28.2 算法讨论 28
28.3 时空复杂度 28

29 CodeForces 286E Ladies' Shop **28**
29.1 题目大意 28
29.2 算法讨论 28
29.3 时空复杂度 28

30 CodeForces 293B Distinct Paths **29**
30.1 题目大意 29
30.2 算法讨论 29
30.3 时空复杂度 29

31 CodeForces 293D Ksusha and Square **29**
31.1 题目大意 29
31.2 算法讨论 29
31.3 时空复杂度 29

32 CodeForces 293E Close Vertices **30**

32.1 题目大意 30
32.2 算法讨论 30
32.3 时空复杂度 30

33 CodeForces 294D Shaass and Painter Robot **30**
33.1 题目大意 30
33.2 算法讨论 30
33.3 时空复杂度 30

34 CodeForces 295D Greg and Caves **31**
34.1 题目大意 31
34.2 算法讨论 31
34.3 时空复杂度 31

35 CodeForces 301C Yaroslav and Algorithm **31**
35.1 题目大意 31
35.2 算法讨论 31
35.3 时空复杂度 32

36 CodeForces 303E Random Ranking **32**
36.1 题目大意 32
36.2 算法讨论 32
36.3 时空复杂度 33

37 CodeForces 305D Olya and Graph **33**
37.1 题目大意 33
37.2 算法讨论 33
37.3 时空复杂度 33

38 CodeForces 305E Playing with String **33**
38.1 题目大意 33
38.2 算法讨论 34
38.3 时空复杂度 34

39 CodeForces 306C White, Black and White Again **34**
39.1 题目大意 34
39.2 算法讨论 34
39.3 时空复杂度 34

40 CodeForces 306D Polygon **34**
40.1 题目大意 34

40.2 算法讨论 34
40.3 时空复杂度 35

41 CodeForces 309B Context Advertising **35**
41.1 题目大意 35
41.2 算法讨论 35
41.3 时空复杂度 35

42 CodeForces 309D Tennis Rackets **35**
42.1 题目大意 35
42.2 算法讨论 35
42.3 时空复杂度 35

43 CodeForces 309E Sheep **36**
43.1 题目大意 36
43.2 算法讨论 36
43.3 时空复杂度 36

44 CodeForces 311C Fetch the Treasure **36**
44.1 题目大意 36
44.2 算法讨论 36
44.3 时空复杂度 37

45 CodeForces 311E Biologist **37**
45.1 题目大意 37
45.2 算法讨论 37
45.3 时空复杂度 37

46 CodeForces 314E Sereja and Squares **37**
46.1 题目大意 37
46.2 算法讨论 38
46.3 时空复杂度 38

47 CodeForces 316E3 Summer Homework **38**
47.1 题目大意 38
47.2 算法讨论 38
47.3 时空复杂度 38

48 CodeForces 316F3 Suns and Rays **38**
48.1 题目大意 38
48.2 算法讨论 39

48.3 时空复杂度 39

49 CodeForces 317C Balance **39**
49.1 题目大意 39
49.2 算法讨论 39
49.3 时空复杂度 39

50 CodeForces 317E Princess and Her Shadow **39**
50.1 题目大意 39
50.2 算法讨论 40
50.3 时空复杂度 40

51 CodeForces 319D Have You Ever Heard About the Word? **40**
51.1 题目大意 40
51.2 算法讨论 40
51.3 时空复杂度 40

52 CodeForces 319E Ping-Pong **41**
52.1 题目大意 41
52.2 算法讨论 41
52.3 时空复杂度 41

53 CodeForces 323B Tournament-Graph **41**
53.1 题目大意 41
53.2 算法讨论 41
53.3 时空复杂度 41

54 CodeForces 323C Two permutations **42**
54.1 题目大意 42
54.2 算法讨论 42
54.3 时空复杂度 42

55 CodeForces 325D Reclamation **42**
55.1 题目大意 42
55.2 算法讨论 42
55.3 时空复杂度 42

56 CodeForces 325E The Red Button **42**
56.1 题目大意 42
56.2 算法讨论 43
56.3 时空复杂度 43

57 CodeForces 329D The Evil Temple and the Moving Rocks **43**
57.1 题目大意 43
57.2 算法讨论 43
57.3 时空复杂度 43

58 CodeForces 331C3 The Great Julya Calendar **44**
58.1 题目大意 44
58.2 算法讨论 44
58.3 时空复杂度 44

59 CodeForces 331D3 Escaping on Beaveractor **44**
59.1 题目大意 44
59.2 算法讨论 44
59.3 时空复杂度 45

60 CodeForces 332D Theft of Blueprints **45**
60.1 题目大意 45
60.2 算法讨论 45
60.3 时空复杂度 45

61 CodeForces 335D Rectangle And Square **45**
61.1 题目大意 45
61.2 算法讨论 45
61.3 时空复杂度 46

62 CodeForces 338E Optimize! **46**
62.1 题目大意 46
62.2 算法讨论 46
62.3 时空复杂度 46

63 CodeForces 339E Three Swaps **46**
63.1 题目大意 46
63.2 算法讨论 46
63.3 时空复杂度 47

64 CodeForces 342D Xenia and Dominoes **47**
64.1 题目大意 47
64.2 算法讨论 47
64.3 时空复杂度 47

65 CodeForces 348E Pilgrims **47**

65.1 题目大意 47
65.2 算法讨论 48
65.3 时空复杂度 48

66 CodeForces 351D Jeff and Removing Periods 48
66.1 题目大意 48
66.2 算法讨论 48
66.3 时空复杂度 48

67 Google Code Jam World Final 2009 C Doubly-sorted Grid 49
67.1 题目大意 49
67.2 算法讨论 49
67.3 时空复杂度 49

68 Google Code Jam World Final 2009 B Min Perimeter 49
68.1 题目大意 49
68.2 算法讨论 49
68.3 时空复杂度 50

69 Google Code Jam World Final 2009 D Wi-fi Towers 50
69.1 题目大意 50
69.2 算法讨论 50
69.3 时空复杂度 50

70 Google Code Jam World Final 2010 A Letter Stamper 51
70.1 题目大意 51
70.2 算法讨论 51
70.3 时空复杂度 51

71 Google Code Jam World Final 2010 C Candy Store 51
71.1 题目大意 51
71.2 算法讨论 52
71.3 时空复杂度 52

72 Google Code Jam World Final 2011 A Runs 52
72.1 题目大意 52
72.2 算法讨论 52
72.3 时空复杂度 52

73 Google Code Jam World Final 2011 C Program within a Program 53
73.1 题目大意 53

73.2 算法讨论 53
73.3 时空复杂度 54

74 Google Code Jam World Final 2013 E Let Me Tell You a Story 54
74.1 题目大意 54
74.2 算法讨论 54
74.3 时空复杂度 55

75 Google Code Jam World Final 2014 E Allergy Testing 55
75.1 题目大意 55
75.2 算法讨论 55
75.3 时空复杂度 55

76 USACO DEC 05 Cow Patterns 56
76.1 题目大意 56
76.2 算法讨论 56
76.3 时空复杂度 56

77 USACO DEC 07 Best Cow Line 56
77.1 题目大意 56
77.2 算法讨论 56
77.3 时空复杂度 56

78 USACO DEC 08 Fence 56
78.1 题目大意 56
78.2 算法讨论 57
78.3 时空复杂度 57

79 USACO DEC 10 Threatening Letter 57
79.1 题目大意 57
79.2 算法讨论 57
79.3 时空复杂度 57

80 USACO DEC 12 First 57
80.1 题目大意 57
80.2 算法讨论 58
80.3 时空复杂度 58

81 USACO JAN 07 Cow School 58
81.1 题目大意 58
81.2 算法讨论 58

81.3 时空复杂度 58

82 USACO JAN 09 Travel **59**
82.1 题目大意 59
82.2 算法讨论 59
82.3 时空复杂度 59

83 USACO MAR 08 Land Acquisition **59**
83.1 题目大意 59
83.2 算法讨论 59
83.3 时空复杂度 59

84 USACO MAR 09 Cleaning up **60**
84.1 题目大意 60
84.2 算法讨论 60
84.3 时空复杂度 60

85 USACO MAR 12 Cows in a Skyscraper **60**
85.1 题目大意 60
85.2 算法讨论 60
85.3 时空复杂度 60

86 USACO MAR 13 Hill Walk **60**
86.1 题目大意 60
86.2 算法讨论 61
86.3 时空复杂度 61

87 USACO OPEN 07 Connect **61**
87.1 题目大意 61
87.2 算法讨论 61
87.3 时空复杂度 61

88 USACO OPEN 08 Cow Neighborhoods **61**
88.1 题目大意 61
88.2 算法讨论 62
88.3 时空复杂度 62

89 USACO OPEN 09 Tower **62**
89.1 题目大意 62
89.2 算法讨论 62
89.3 时空复杂度 62

90 USACO OPEN 10 Triangle Counting **63**
90.1 题目大意 63
90.2 算法讨论 63
90.3 时空复杂度 63

91 USACO OPEN 13 Photo **63**
91.1 题目大意 63
91.2 算法讨论 63
91.3 时空复杂度 63

1 CodeForces 235D Graph Game

1.1 题目大意

有一个 n 个点 n 条边的连通图，

```
solve(t)（t为一个图）：
    totalCost += t的大小
    随机选择t中的一个点x，并删掉这个点，然后t变成若干个联通块
    对每个联通块调用solve
```

求 totalCost 的期望值。

1.2 算法讨论

先考虑树上的情况，考虑两个点 x 和 y，x 在 `solve(y)` 中有贡献当且仅当 x 到 y 路径上 y 是第一个被删掉的，这个概率是 $\frac{1}{len}$，其中 len 为 x 到 y 路径上点的数量。

再考虑图上的两点，如果在同一棵子树里，x 到 y 只有一条路径，跟树上的一样，

否则 x 到 y 有两条路径，设这两条路径的并的总点数为 len，只在一条路径上出现的点的数量分别为 A 和 B，

那么要么一开始删掉 y，要么一开始一直删只出现在某条路径里的点，然后删掉 y。

这样对答案的贡献为 $\frac{1}{len}+\frac{A}{len-A}+\frac{B}{tot-B}$。

对每个点 dfs 整张图计算以上贡献，时间复杂度 $O(n^2)$。

1.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

2 CodeForces 235E Number Challenge

2.1 题目大意

求 $\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}d(i\cdot j\cdot k)$，其中 $d(x)$ 为 x 的约数个数。

2.2 算法讨论

令 `f[a][b][c][p]` 表示 i,j,k 的上界为 a,b,c，且只能用 $\geq p$ 的质数时原问题的答案，

设 q 为大于 p 的第一个质数，显然 $\texttt{f[a][b][c][p]}=\sum_x\sum_y\sum_z \texttt{f[}\lfloor\frac{a}{p^x}\rfloor\texttt{][}\lfloor\frac{b}{p^y}\rfloor\texttt{][}\lfloor\frac{c}{p^z}\rfloor\texttt{][q]}$。用以上方法 dp，用 hash 来存每个状态，复杂度为 [1,abc] 内只含不大于 $max(a,b,c)$ 的质因子的数的个数，略小于 $O(abc)$。

2.3 时空复杂度

时间复杂度：略小于$O(abc)$

空间复杂度：略小于$O(abc)$

3 CodeForces 238D Tape Programming

3.1 题目大意

有一种编程语言，它的程序是一个由 “<”, “>” 和 “0” ∼ “9” 构成的字符串。

程序运行时维护两个指针：当前位置的指针 (CP) 和方向指针 (DP)，其中 DP 可以指向左或右。

一开始 CP 指向这个串的第一个字符，DP 指向右。

每一步操作，如果 CP 指向一个数字，那么输出这个数字并将其减一，如果这个数字已经是 0 就把它从串中删掉，

如果 CP 指向的是 “>” 或 “<”，则 DP 改为该字符所指的方向（“<” 为左，“>” 为右），如果此时上一个字符也是 “<” 或 “>”，就把上一个字符从串种删掉，如果 CP 不在串中程序停止。

现在有一个长度为 n 的串 s，每次可以询问 $s_l, s_{l+1}, ..., s_r$ 构成的子串作为一个单独的程序运行，每个数字会被输出几次。

$n <= 10^5, q <= 10^5$

3.2 算法讨论

首先对于一个长度为 m 的串，把它作为程序运行时，操作的次数是 $O(m)$ 级别的，

并且在任意时刻，已经访问过的位置一定是这个串的一个前缀。

如果询问的左端点都是 1，那么只要知道第一次访问到 $r+1$ 的时间，就可以知道程序结束的时间。

如果询问的左端点不是 1，我们就先对原串模拟一遍整个程序，然后 [l,r] 作为程序时，第一次访问到 l 的时间就是程序开始的时间，

在这个时间之后的第一次访问到 [l,r] 区间以外的位置的时间就是程序结束的时间。

这个时间可以将所有询问按程序开始的时间排序后，用 BIT or 线段树求出。

3.3 时空复杂度

时间复杂度：$O((n+q)\log n)$

空间复杂度：$O(n+q)$

4 CodeForces 240F TorCoder

4.1 题目大意

有一个长度为 n 的字符串，只包含小写字母，要对它进行 m 次操作，每次操作将子串 [l_i,r_i] 中的字母重新排列成一个字典序最小的回文串，如果不能这么做就忽略这个操作。

输出做完所有操作的字符串。

4.2 算法讨论

每次操作最多将一段子串排列成 53($26 \times 2 + 1$) 段由相同字母组成的子串，

所以可以对每个字母开一棵线段树（注意这些线段树形态相同，所以可以只开一棵线段树，并在每个节点上开一个 `int[26]` 的数组），

然后统计出 [l,r] 中每个字母的个数，然后用线段树的区间覆盖操作维护。

4.3 时空复杂度

时间复杂度：$O(26m\log n)$

空间复杂度：$O(26n)$

5 CodeForces 241B Friends

5.1 题目大意

有 n 个数，求他们两两之间异或值的前 m 大的和。

$n \leq 50000, m \leq n*(n-1)/2, A[i] \leq 10^9$。

5.2 算法讨论

对这 n 个数建一棵 trie 树，然后在 trie 上二分就能在 $O(n\log n)$ 时间求出第 m 大的值。

在一次二分中，如果是往左子树走，就要把右子树里的所有数 `xor A[i]` 并将和计算进答案，这时要在每个子树里记录一个数组 `sum[31]`，`sum[i]` 表示这棵子树中第 i 位为 1 的数字的个数。

时间复杂度和空间复杂度都是 $O(n\log^2 n)$，可能会 MLE。

注意到不同的 `sum[]` 数组最多只有 $2n$ 个，空间复杂度就可以做到 $O(n\log n)$。

5.3 时空复杂度

时间复杂度：$O(n\log^2 n)$

空间复杂度：$O(n\log n)$

6 CodeForces 241E Flights

6.1 题目大意

给定一个 n 个点 m 条边的有向无环图，边权都是 1，要求将一部分边权改成 2，使 $1 \sim n$ 的每一条路径的长度都相等。

$n <= 1000, m <= 5000$。

6.2 算法讨论

先去掉 1 号点走不到的点和走不到 n 号点的点，然后设 1 号点到 i 号点的距离为 $dist_i$，对于每条边 (i,j)，就要满足 $1 \leq dist_j - dist_i \leq 2$，这是一个差分约束系统，直接跑 SPFA 就行了。

6.3 时空复杂度

时间复杂度：$O(nm)$

空间复杂度：$O(nm)$

7 CodeForces 241F Race

7.1 题目大意

TOC 是一个由 $m * n$ 个街区组成的城市，每一个街区可以是建筑物，或双向直线道路，或交叉路口。

没有两个不同的道路或两个不同的交叉路口相邻（有公共边）。

一个人要从 (r_s, c_s) 开始走，经过 $s_1, s_2, ..., s_{|s|}$ 这些交叉路口，然后走到 (r_e, c_e)，每个交叉路口要花费 1 分钟的时间，另外也知道每个街区花费的时间。

问这个人开始走后第 k 分钟在什么位置。

7.2 算法讨论

显然只能沿直线走，所以直接模拟这个人走的过程即可。

7.3 时空复杂度

时间复杂度：$O(nm)$

空间复杂度：$O(nm)$

8 CodeForces 243C Colorado Potato Beetle

8.1 题目大意

Old MacDonald 有一个农场和大小为 $(10^{10} + 1) \times (10^{10} + 1)$ 平方米的巨大的土豆种植地。这块田地被分成正方形的小块，每块占地一平方米。

Old MacDonald 知道科罗拉多土豆甲虫将要入侵，并且会破坏收成。Old MacDonald 想在一些土地上喷上杀虫剂。

所以，Old MacDonald 去了田里，站在田地中间的那个小块中间并将这个格子撒上杀虫剂。现在，他将要通过一系列的移动来在更多田地喷洒杀虫剂。

每次移动的时候，Old MacDonald 会在上下左右四个方向中选择一个并移动整数米。当他移动的时候，他会在走过的每个格子上喷洒杀虫剂。也就是说，和 Old MacDonald 的移动轨迹有任何交点的小块都会被撒上杀虫剂。

当 Old MacDonald 停止喷洒杀虫剂后，他将他所有的移动记录在一张纸上。现在，他想知道，有多少小块不会受到科罗拉多甲虫入侵的影响。

我们知道，科罗拉多甲虫的入侵按照以下顺序展开。一开始，一些土地边缘的小块被入侵。接着，任何没有撒过杀虫剂且有一个相邻（与其有公共边的）小块被感染的尚未被感染的土地也会被感染。

帮助 Old MacDonald 计算有多少小块不会受到科罗拉多甲虫的感染。

移动次数 $n \leq 1000$，每次移动的距离 ≤ 1000000。

8.2 算法讨论

离散化每次移动的坐标，然后 bfs 就行了。

8.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

9 CodeForces 249D Donkey and Stars

9.1 题目大意

给定平面上的 n 个点和两个角度 α, β。

从原点出发，每次从当前点开始作两条射线，与 x 轴的夹角分别为 α, β，然后可以走到这两条射线之间的某个点（不包括边界），直到不能再走。

问最多可以走多少步。

9.2 算法讨论

设 $tan\alpha = k1$,$tan\beta = k2$，然后将所有点 (x, y) 变换成 $(k2 \cdot x - y, y - k1 \cdot x)$，再按 x 坐标排序，就是个经典的 LIS 问题了。

9.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

10 CodeForces 249E Endless Matrix

10.1 题目大意

将所有正整数排列成这样的矩阵：

```
 1   2   5  10  ...
 4   3   6  11  ...
 9   8   7  12  ...
16  15  14  13  ...
... ... ... ... ...
```

有 t 个询问，每个询问是求 $x1 \sim x2$ 行，$y1 \sim y2$ 列的和的最后 10 位。

10.2 算法讨论

我们只要求出 $1 \sim x$ 行 $1 \sim y$ 列的和 $sum(x, y)$。

设 $f1(x) = x * (x + 1)/2$，$f2(x) = x * (x + 1) * (2 * x + 1)/6$，

如果 $x > y$，$sum(x, y) = -(x - y) * f1(y - 1) + y * (f2(x) - f2(y)) + f1(y * y)$，

否则 $sum(x, y) = (y - x) * f1(x) + x * (f2(y - 1) - f2(x - 1)) + f1(x * x)$。

每次询问时间复杂度 $O(1)$。

10.3 时空复杂度

时间复杂度：$O(t)$

空间复杂度：$O(1)$

11 CodeForces 253E Printer

11.1 题目大意

有一个打印机，要打印 n 份文件。

第 i 份文件在 t_i 时刻提交给打印机，有 s_i 页，优先级为 p_i（所有文件优先级互不相同）。

对于每个时刻，打印机都找一个当前没打印完的文件中，优先级最高的文件，然后打印一页这个文件。

现在有一个文件 x 的优先级未知（其他都知道），但是知道这个文件被打印完的时刻，要求输出这个文件的优先级和每个文件被打印完的时刻。

11.2 算法讨论

显然答案满足二分性质。

二分答案后用个 `priority_queue` 模拟即可。

11.3 时空复杂度

时间复杂度：$O(n \log^2 n)$

空间复杂度：$O(n)$

12 CodeForces 254D Rats

12.1 题目大意

$n * m$ 的网格中有若干只老鼠，还有若干格子有障碍物，

现在要用两个手榴弹炸死所有老鼠。

每个手榴弹在爆炸时会清除它所在的格子，接下来 d 秒，对于每一个第 $i(0 \leq i < d)$ 秒被清除的格子，与它相邻的格子都会在第 $i + 1$ 秒被清除（不能清除有障碍物的格子）。

求一种用两个手榴弹炸死所有老鼠的方案，或输出无解。

$n, m \leq 1000, d \leq 8$。

12.2 算法讨论

任意选一只老鼠，假设它被第一个手榴弹炸死，

那么第一个手榴弹只能放置在 $O(d^2)$ 个格子里（准确的说是 $2d^2 - 2d + 1$ 个格子），

枚举第一个手榴弹放置的位置，然后剩下的老鼠用类似的方法判断。

12.3 时空复杂度

时间复杂度：$O(nm + d^6)$

空间复杂度：$O(nm)$

13 CodeForces 256D Liars and Serge

13.1 题目大意

有 n 个人，其中有些人说谎，每个人都知道这 n 个人中有多少人说谎，

现在问每个人“有多少人说真话”，说真话的人一定会回答正确的答案，说谎的人会回答 1 到 n 之间除正确答案以外的任意一个整数。

每个说谎的人回答的答案与其他说谎的人的答案是无关的。

问有多少种这 n 个人的答案序列，能判断出“恰好有 k 个人一定在说谎”。

$1 \leq k \leq n \leq 256$，保证 n 是 2 的整数次幂。

13.2 算法讨论

记 cnt_i 为答案为 i 的人数，那么对每个 x，如果 $cnt_x = x$，那么这 x 个人可能在说真话，否则这 cnt_x 个人一定在说谎。

于是可以用 `dp[i][j][k]` 表示只考虑 $x < i$，$\sum_{x=1}^{i-1} cnt_x = j$，有 k 个人一定说谎的方案数，然后枚举 cnt_i 来转移即可。

复杂度是 $O(n^4)$ 的，但是 n 一定是 2 的整数次幂，所以 n 的数量很少，打个表就行了。

13.3 时空复杂度

时间复杂度：$O(n^4)$

空间复杂度：$O(n^3)$

14 CodeForces 257E Greedy Elevator

14.1 题目大意

m 层高的大楼里的一部电梯，在第 0 个时刻停在第 1 层。

有 n 个人，第 i 个人要从第 s_i 层坐电梯到 f_i 层，他会在 t_i 时刻开始等电梯。

对每个时刻，如果电梯里没人且没有人在等电梯，电梯不动，

否则假设电梯在第 x 层，我们计算 p_{up} 为在大于 x 楼层中等电梯的人数加上电梯中要去大于 x 楼层的人数，p_{down} 为在小于 x 楼层中等电梯的人数加上电梯中要去小于 x 楼层的人数，

如果 $p_{up} \geq p_{down}$，电梯往上走一层，否则往下。

我们忽略电梯开关门和人进出的时间，要求模拟电梯工作的过程，输出每个人走出电梯的时刻。

$n \leq 10^5, m \leq 10^5, 1 \leq t_i \leq 10^9$

14.2 算法讨论

用线段树或平衡树维护每个楼层等电梯的人数和电梯中要去每个楼层的人数，直接模拟就行了。

14.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

15 CodeForces 258D Little Elephant and Broken Sorting

15.1 题目大意

有一个 $1 \sim n$ 的排列，然后有 m 个操作，每个操作是交换 i 和 j 位置上的数，

但是有 0.5 的概率不执行这个操作。

问最后这个序列的期望逆序对数量。

15.2 算法讨论

用 `P[i][j]` 表示当前第 i 个数比第 j 个数大的概率。

在一次交换 x 和 y 时，对于所有 $t(1 \leq t \leq n, t \neq x, t \neq y)$，

`P[t][x]` 和 `P[t][y]` 会变为 $0.5 * (\texttt{P[t][x]} + \texttt{P[t][y]})$，

`P[x][t]` 和 `P[y][t]` 会变为 $0.5 * (\texttt{P[x][t]} + \texttt{P[y][t]})$，

然后 `P[x][y]` 和 `P[y][x]` 都变为 0.5，因为有 0.5 的概率不交换。

最后 $\sum_{i<j}$ `P[i][j]` 就是答案。

15.3 时空复杂度

时间复杂度：$O(n^2 + nm)$

空间复杂度：$O(n^2)$

16 CodeForces 260E Dividing Kingdom

16.1 题目大意

平面上有 n 个点，要求用两条平行 x 轴的直线和两条平行 y 轴的直线，将这 n 个点分为 9 部分，每部分的点数排序后为 $a[1], a[2], ..., a[9]$，并输出一组解。

16.2 算法讨论

用 $O(9!)$ 的时间暴力枚举 9 个部分的点数，然后用主席树判断是否合法。

16.3 时空复杂度

时间复杂度：$O(n \log n + 9! * 9 * \log n)$

空间复杂度：$O(n \log n)$

17 CodeForces 261D Maxim and Increasing Subsequence

17.1 题目大意

有一个数列 $b_1, b_2, ..., b_n$，把它复制 t 份作为数列 a，求数列 a 的最长上升子序列长度。

有 k 组数据。

$1 \le k \le 10, 1 \le n, maxb \le 10^5, 1 \le t \le 10^9, n * maxb \le 2 * 10^7$。

17.2 算法讨论

如果 $t \le maxb$，那么答案一定是 b 中不同的数字个数，否则 a 最多只有 $n * maxb$（最多 $2 * 10^7$）个元素。

直接用 LIS 的经典做法（树状数组）的复杂度是 $O(n * maxb * \log n)$，会 TLE。

求 LIS 还有一个用单调栈的做法，复杂度同上，但是可以优化。

这个做法是，每次加入一个数 x 时，二分查找栈中第一个不小于 x 的数，把它改成 x，如果栈中所有数都小于 x，就把 x 压入栈顶，最后栈的大小就是答案。

我们维护 `pos[i]`，表示栈中第一个不小于 i 的数的位置，就能不用二分查找。

插入一个数的时候暴力维护 `pos[]`，栈中每个数每次减一，都要修改一次 `pos[]`，所以修改的总次数是 $O(n * maxb)$ 的，是靠谱的复杂度。

17.3 时空复杂度

时间复杂度：$O(k * n * maxb)$

空间复杂度：$O(n + maxb)$

18 CodeForces 261E Maxim and Calculator

18.1 题目大意

有 a, b 两个变量，一开始 $a = 1, b = 0$，每次操作可以使 $a = a * b$ 或 $b = b + 1$。

问有多少正整数 $x(l \le x \le r)$ 满足存在一种方式从初始状态操作不超过 $p(1 \le p \le 100)$ 步使 $a = x$。

$2 \le l \le r \le 10^9$, $1 \le p \le 100$

18.2 算法讨论

$[1, 10^9]$ 内最大质因子小于 100 的数只有约 $300 * 10^4$ 个，所以可以先找出这些数并从小到大排序，

然后考虑 `dp[i][j]` 表示第 i 个数在 $b = j$ 时最少要操作几次才能达到，

只要枚举 j 就能在 $O(3000000 * p)$ 的时间内求出所有 `dp[i][j]`。

然后第 i 个数被统计进答案当 $min\{$`dp[i][j]`$\} \le p$ 且第 i 个数在 $[l, r]$ 内。

18.3 时空复杂度

时间复杂度：$O(3000000 * (p + \log 3000000))$

空间复杂度：$O(3000000)$

19 CodeForces 264E Roadside Trees

19.1 题目大意

在一条直线有 n 个位置可以种树，自西向东标号 $1 \sim n$。每个月，每棵树都会长高 1 米。

每个月初，你都会收到一个要求。要求有两种类型：

1、在位置 p 种一棵高度为 h 的树

2、砍掉从西向东数的第 x 棵树，当这个位置的树被砍掉后，倒下的树会占据这个位置，之后这个位置不能再种树。

在做完这个要求后，你需要求出当前树高的最长上升子序列。

任何时刻树的高度是不同的。初始没有任何树。

$n \le 10^5$，要求的个数 $m \le 2 * 10^5$，$1 \le h \le 10$, $1 \le x \le 10$。

19.2 算法讨论

由于插入和删除一个数时只会影响到最多 10 个数的答案，直接用线段树暴力更新这 10 个值就行了。

19.3 时空复杂度

时间复杂度：$O(10 * m * \log n)$

空间复杂度：$O(n + m)$

20 CodeForces 266D BerDonalds

20.1 题目大意

给一个带权无向图，求一个点（可以在节点上，也可以在某条边上），使这个点到其他所有节点的最大距离最小，求这个距离。

$n \leq 200$

20.2 算法讨论

首先可以用 floyd 算法求出两两节点之间的最短距离。

然后考虑我们要选的点在某条边 (i, j) 上时，

其他所有节点离我们选的点的距离是关于 “我们选的点到节点 i 的距离” 的一个分段一次函数。

将这些分段一次函数按段的边界排序后，就可以得到答案。

20.3 时空复杂度

时间复杂度：$O(n^3 \log n)$

空间复杂度：$O(n^2)$

21 CodeForces 266E More Queries to Array...

21.1 题目大意

有一个包含 n 个整数的数组 $a[1], a[2], ..., a[n]$，有 m 个操作，为以下两种：

1、将 $a[l], a[l+1], ..., a[r]$ 都赋值为 x。

2、询问 $\sum_{i=l}^{r} a[i] * (i - l + 1)^k$，$0 \leq k \leq 5$，输出答案 $mod 10^9 + 7$ 的值。

21.2 算法讨论

$(a[i] - l + 1)^k = \sum_{j=0}^{k} C(k, j) * a[i] * i^{k-j} * (-l)^j$

所以只要用 $(k+1)$ 棵线段树维护 $a[i] * i^j (0 \leq j \leq k)$ 的区间和。

21.3 时空复杂度

时间复杂度：$O((n+m)k \log n)$

空间复杂度：$O(nk)$

22 CodeForces 268D Wall Bars

22.1 题目大意

一个梯子的中间那根管子的高度为 n。

在高度为 $1, 2, ..., n$ 的地方，恰好有一根水平的横杆从中间的杆子连向四个方向中的某一个预先固定好的杆子上。

如果两根横杆的距离不超过 h，且方向相同，那么一个孩子可以从一个一根横杆爬到另一根上。

在地上的孩子，可以爬到任何一根高度在 $1 \sim h$ 之间的横杠上。

一个从地面出发的孩子至少能到达一根高度在 $n-h+1, n-h+2, ..., n$ 的横杠。

问有多少种摆放横杠的方案满足要求。

$n \leq 1000, h \leq 30$。

22.2 算法讨论

考虑 `dp[i][h1][h2][h3][h4][f1][f2][f3][f4]` 表示高度为 i，4 个方向的横杠的最大高度与 i 的差分别为 $h1, h2, h3, h4$，这 4 个方向是否有高度 h 的一段没有横杠的状态为 $f1, f2, f3, f4$ 的方案数。

表示一个状态时可以保证 $h1 \leq h2 \leq h3 \leq h4$，那么一定有 $h1 = 0$，于是实际有效的状态数为 $O(n * h^3)$ 的，这样复杂度也是 $O(n * h^3)$，要注意常数优化才能过。

（tsinsen 上的时限实在太小了于是我把 n 在 [200,1000]，h 在 [25,30] 的答案打了表才过）

22.3 时空复杂度

时间复杂度：$O(n * h^3)$

空间复杂度：$O(h^3)$

23 CodeForces 269D Maximum Waterfall

23.1 题目大意

平面上有 n 条线段，每条线段连接点 (l_i, h_i) 和 (r_i, h_i)，

最上面和最下面也分别有一条线段，分别连接 $(-10^9, 10^9)$ 和 $(10^9, 10^9)$，$(-10^9, 0)$ 和 $(10^9, 0)$。

当满足以下条件时，水可以从两条线段 (i, j) 之间流过：

$max(l_i, l_j) < min(r_i, r_j)$（即两条线段横向投影重叠）

$h_j < h_i$

不存在一个 $k(h_j < h_k < h_i)$，使线段 k 满足以上条件。

$i \to j$ 的流量为 $min(r_i, r_j) - max(l_i, l_j)$。

求从最上面到最下面最小流量最大的一条路径。

23.2 算法讨论

将所有线段按左端点排序。用一个 `set<int>` 来维护所有线段的 h 值。

每次加入一条线段时（记这条线段为 x），在 `set` 中找到它上面一条线段和下面一条（记为 i, j），

然后我们只要将 i 向 x 连边，x 向 j 连边，并删除 i 到 j 的边，最后在这张图上 dp 即可。

由于同时有加边和删边操作，要用一个 map 来存边表。

23.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

24 CodeForces 273 Dima and Game

24.1 题目大意

有 n 个数对 $(l_i, r_i)(1 \le l_i < r_i \le p)$，两个玩家轮流做以下操作：

选一个数对 (l_r, r_i)，满足 $r_i - l_i > 2$；

将这个数对替换为 $(l_i + \lfloor \frac{r_i - l_i}{3} \rfloor, l_i + 2 * \lfloor \frac{r_i - l_i}{3} \rfloor)$ 或 $(l_i, r_i - \lfloor \frac{r_i - l_i}{3} \rfloor)$。

没有操作可做的玩家就输了。

给定 $n(1 \le n \le 1000)$ 和 $p(1 \le p \le 10^9)$，问有多少种方案写出 n 个数对 $(l_i, r_i)(1 \le l_i < r_i \le p)$ 使得两人都用最优策略时第一个人能赢。

24.2 算法讨论

考虑 $sg[i]$ 为 $r - l$ 为 i 的数对的 sg 值，$sg[i] = mex(sg[\lfloor \frac{i}{3} \rfloor], sg[i - \lfloor \frac{i}{3} \rfloor])$。

我们只要分别计算 $1 \le i \le p$, $sg[i] = 0, 1, 2$ 时的 $n - i$ 的和。

打表后发现，$sg[i]$ 在 $[1, 10^9]$ 上可以分成 102 段，每一段都是相等的值，这样上面的式子就可以很容易算出来。

然后只要用 $dp[i][j]$ 表示 i 个数对，sg 值为 j 的方案数就行了。

24.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(1)$

25 CodeForces 277D Google Code Jam

25.1 题目大意

一场 GCJ 比赛有 n 道题和 t 个单位时间，每道题有小数据和大数据，小数据的提交结果能实时知道，大数据必须要比赛结束才能知道结果。

一场比赛的罚时为最后一次提交正确程序的时间。

第 i 题的小数据需要花 $timeSmall_i$ 个单位时间，保证能得到 $scoreSmall_i$ 的分数。

第 i 题的大数据需要在小数据通过之后再花 $timeLarge_i$ 个单位时间，能得到 $scoreLarge_i$ 的分数，但是有 $probFail_i$ 的概率会 WA 掉而不得分。

可以在做完一题的小数据后先做别的题。

求最大的期望得分以及此时最小的期望罚时。

25.2 算法讨论

如果做哪些题已经确定，那一定是先做所有小数据，然后再按一定顺序做大数据。

对于第 i 题和第 j 题，令 $a_1 = timeLarge_i, a_2 = timeLarge_j, f_1 = probFail_i, f_2 = probFail_j$，

先做 i 再做 j，大数据的期望罚时为 $f_2 * a_2 + f_1 * f_2 * a_1$，

先做 j 再做 i，期望罚时为 $f_1 * a_1 + f_1 * f_2 * a_2$，

只要比较以上两式就能知道第 i 题, 第 j 题的大数据做的顺序。

可以先对所有题目按以上方法排序，然后 `dp[i][j]` 表示前 i 题，花 j 个单位时间，最大的期望得分以及此时最小的期望罚时。

最后 $ans = max$`dp[n][i]`$(0 \le i \le t)$。

25.3 时空复杂度

时间复杂度：$O(n \log n + nt)$

空间复杂度：$O(nt)$

26 CodeForces 280D k-Maximum Subsequence Sum

26.1 题目大意

有一个长度为 n 的序列，要求支持以下操作：

修改一个位置上的数；

询问在 l 到 r 中，最多选 k 段不重复的连续子序列（k 可以等于 0），这些子序列最大的和。

$n \le 10^5$，操作个数 $m \le 10^5$，询问操作个数 $q \le 10^4$，$k \le 20$。

26.2 算法讨论

算法一

$k = 1$ 时是经典问题，只要在线段树每个点上记录左边开始最大值，右边开始最大值和这个区间的答案，就能每个操作复杂度 O(logn)。

把这个算法扩展一下，线段树每个点上开三个数组 `L[],R[],ans[]`，分别表示选不超过 x 个子序列时，第一个子序列包含区间第一个数的答案，最后一个子序列包含区间最后一个数的答案，无特殊条件的答案，

然后两个区间的信息可以 $O(k^2)$ 合并，每个操作的复杂度 $O(k^2 \log n)$。

无法通过 $n = 10^5, m = 10^5$ 的数据。

算法二

对于单次询问（即询问一个序列 `a[1],a[2],...,a[n]` 选 k 段的答案），考虑另一种做法：费用流。

对每个 i，i 号点向 $(i+1)$ 号点连流量为 1，费用为 `a[i]` 的边，再分别向 S 和 T 连流量为 1，费用为 0 的边。

那么我们用费用流的 SPFA 算法，增广 k 次后就能得到选 k 段的答案。

注意到每次增广的实质是找一段和最大的连续子序列，然后将其加入答案，再将这一段的数取反，这可以用线段树维护。

只要维护各最大值以及取到最大值的子段位置，各最小值以及取到最小值的子段位置，就能支持区间取反操作。

26.3 时空复杂度

时间复杂度：$O(n + m \log n + qk \log n)$

空间复杂度：$O(n)$

27 CodeForces 283E Cow Tennis Tournament

27.1 题目大意

有 n 头牛进行比赛，每头牛有一个参数 $s[i]$（所有 $s[i]$ 都不相同），

每一对牛都比赛一次，第 i 头能赢第 j 头牛当且仅当 $s[i] > s[j]$。

FJ 打算选出 k 个区间，对于一个区间 $[a, b]$，我们改变所有 (x, y) $(x \neq y; a \leq s[x], s[y] \leq b)$ 的比赛结果。

注意一场比赛的结果可能会被改变多次。

现在问有多少无序三元组 (p, q, r)，使最终比赛结果中，p 赢 q，q 赢 r，r 赢 p。

27.2 算法讨论

如果一个三元组 (p, q, r) 不满足条件，那么 p, q, r 中一定有且仅有一个能打败剩下两个。

所以用 $w[i]$ 表示 i 能打败的牛的数量，答案即为 $C(n, 3) - \sum C(w[i], 2)$。

我们用一棵线段树来维护一头牛是否能被打败，

假设所有牛的 $s[i]$ 递增，考虑求 $w[i]$，只要将右端点不小于 i 的区间在线段树中做区间取反操作（0 变为 1，1 变为 0），

然后询问 $[1, i-1]$ 区间和，就能知道编号小于 i 且能被 i 打败的牛的数量，编号大于 i 的类似可以做。

27.3 时空复杂度

时间复杂度：$O(n + k \log n)$

空间复杂度：$O(n + k)$

28 CodeForces 286D Tourists

28.1 题目大意

在一个平面直角坐标系上，某个时刻 t 会有两个人从 $(-1, 0)$ 和 $(1, 0)$ 开始向 y 轴正方向走，速度均为 1。

某个时刻 t 会出现一堵墙（一条 $(0, l)$ 到 $(0, r)$ 的线段），墙出现后不消失。

一共有 n 对人和 m 堵墙，给定每对人开始走的时刻和每堵墙出现的位置和时刻，问对于每对人，有多少时间他们不能互相看到（即被墙挡住）。

28.2 算法讨论

如果只有一堵墙，它对每对人的答案贡献关于人开始走的时刻是一个分段一次函数。

如果所有墙不重叠，每堵墙的贡献也是分段一次函数，可以把函数系数改变的事件预处理出来，

对于一对人，如果开始走的时刻为 t，只要计算时间不大于 t 的事件的和，就知道这时的函数。

不考虑排序的时间这个是 $O(n+m)$ 的。

考虑有重叠的情况，在一段已经被覆盖的区间上，后出现的墙没有效果，

所以可以把所有墙按出现时间从大到小排序，然后依次进行区间覆盖，最后求出每段区间最早被覆盖的时间。

把这些区间作为新的墙，就不会有重叠的情况，可以用上述做法解决。

区间覆盖可以用线段树或平衡树解决。

28.3 时空复杂度

时间复杂度：$O(n+m\log m)$

空间复杂度：$O(m)$

29 CodeForces 286E Ladies' Shop

29.1 题目大意

有 n 个包，容量分别为 $a[1],a[2],...,a[n]$。

要求构造 k 个物品的重量 $p[1],p[2],...,p[k]$，在满足以下条件的情况下使 k 尽量小（无解输出 NO）：

1、对于所有包，都存在一种方案把这个包装到正好满（每个物品可以使用多次）；

2、对于所有总重量 $\leq m$ 的物品的集合（一个物品可以出现多次），这个总重量一定是某个包的重量。

$n,m\leq 10^6$

29.2 算法讨论

将每一对包的容量两两相加的和一定是某个包的重量（或者这个和 $>m$），否则无解。

然后如果某个包不能由其他包的容量和表示，就要有一个重量等于这个包的容量的物品。

用 FFT 来计算哪些重量可以是两个包的容量和，然后判断是否无解就行了。

29.3 时空复杂度

时间复杂度：$O(m\log m)$

空间复杂度：$O(m)$

30 CodeForces 293B Distinct Paths

30.1 题目大意

有一个 $n*m$ 的网格，一些格子已经被涂上 k 种颜色中的一种。

现在要求将剩下的格子涂上颜色，并使从左上角到右下角的每条路径上都不会有两个颜色相同的格子，

求方案数 %1000000007。

$n,m \leq 1000$,$k \leq 10$。

30.2 算法讨论

首先 $n+m-1>k$ 时肯定无解，所以 n,m 的范围很小。

可以考虑直接搜每个格子的颜色，但是这样会 TLE。

只要搜的过程中保证所有格子的颜色是一个最小表示，搜完后乘上一个排列数，

搜索量就变得非常小，能通过所有数据。

30.3 时空复杂度

时间复杂度：$O(k^{k^2})$（很难达到这个上界）

空间复杂度：$O(k^2)$

31 CodeForces 293D Ksusha and Square

31.1 题目大意

有一个面积不为 0 的凸多边形，现在随机选择两个凸多边形内部或边界上的格点，并以它们之间的连线为对角线作一个正方形，求这个正方形的面积的期望。

凸多边形点的数量 $n \leq 10^5$，坐标范围 $[-10^6,10^6]$。

31.2 算法讨论

考虑两个点 $(x1,y1)(x2,y2)$，以它们的连线作为对角线的正方形的面积为 $0.5*((x1-x2)^2+(y1-y2)^2)$，所以 x 坐标和 y 坐标可以分开考虑。

只要对每个 i，求出有多少个格点的 x 坐标为 i，以及有多少个格点的 y 坐标为 i，就可以了。设坐标范围为 L，复杂度为 $O(n+L)$。

31.3 时空复杂度

时间复杂度：$O(n+L)$

空间复杂度：$O(n+L)$

32 CodeForces 293E Close Vertices

32.1 题目大意

求一棵 n 个点的树上经过边数 $\leq L$ 且边权和 $\leq W$ 的路径条数。

32.2 算法讨论

如果没有路径上经过边数的限制，可以点分治，计算一棵子树的答案时将每个点到根的距离排序，扫一遍。

有边数限制时扫一遍过程中维护一个树状数组。（可以理解为一个二维数点问题）

32.3 时空复杂度

时间复杂度：$O(n \log^2 n)$

空间复杂度：$O(n)$

33 CodeForces 294D Shaass and Painter Robot

33.1 题目大意

有一个 $n * m$ 的网格，一开始每个格子都是白色。

一个机器人从网格边界上的 $(x0, y0)$ 点开始，每次先将所在的格子涂成黑色，然后移动一步。

移动的方向与网格边缘呈 45 度角，机器人碰到网格边缘会镜面反弹。

现在问机器人经过多少格子，才能使整个网格被黑白染色（即任意相邻两格中一定有一格黑色一格白色），无解输出 -1。

33.2 算法讨论

有一个结论：只要网格的边界被黑白染色，整个网格就被黑白染色。

（我不会证明这个结论，但是很容易想到）

这样的话我们只要在边界上暴力走，每一次都是从某个边界上的点走到另一个，

如果过程中边界全部被黑白染色，整个网格就被黑白染色了。

因为最坏情况下先是走若干步然后退回起点，再走到其他点，每个点最多被访问两次，

所以暴力走 $2 * (n + m)$ 步后如果没有全部黑白染色就无解。

33.3 时空复杂度

时间复杂度：$O(n + m)$

空间复杂度：$O(n + m)$

34 CodeForces 295D Greg and Caves

34.1 题目大意

统计一个 $n*m$ 黑白网格中满足以下条件的形状的个数：

存在一个区间 $[l,r]$，使 $[l,r]$ 行中只有两个黑格子而其它行只有白格子；

定义 $S[i]$ 为第 i 行中在第 i 行两个黑格子之间的数对 (i,j) 所形成的集合，存在一个行号 $t(l \le t \le r)$，使对任意 $i,j(l \le i < j \le t)$，$S[i]$ 都是 $S[j]$ 的子集，类似的，对任意 $i,j(t \le j < i \le r)$，$S[i]$ 都是 $S[j]$ 的子集。

34.2 算法讨论

先用 $f[i][j]$ 表示 $t=l$ 的情况下，区间长度（行数）为 i，第一行两个黑格子之间的格子数（包括这两个格子）为 j 的时候的答案，

然后答案为 $\sum_{i=1}^{n}\sum_{j=1}^{m} f[i][j]*(n-i+1+\sum_{k=1}^{n-i}\sum_{p=1}^{j-1} f[k][p]*(n+1-i-k)*(j+1-p))$，

暴力枚举 i,j，然后后面部分用二维前缀和计算。

34.3 时空复杂度

时间复杂度：$O(nm)$

空间复杂度：$O(nm)$

35 CodeForces 301C Yaroslav and Algorithm

35.1 题目大意

有一个算法，由一些命令组成，每个命令的形式为 “`s[i]>>w[i]`” 或 “`s[i]<>w[i]`”，

其中 `s[i]` 和 `w[i]` 为长度不超过 7 的字符串（可以为空），由数字或 “`?`” 组成。

这个算法输入一个字符串 a，

然后每次找一个最小的编号 k，使 `s[k]` 是 a 的子串，

然后 a 中 `s[k]` 的第一次出现的会被 `w[k]` 替换，

如果这个命令是 “`s[k]>>w[k]`”，那么这个算法继续执行（即继续寻找最小的编号 k），否则输出当前的字符串 a，并终止。

现在有 n 个不超过 10^{25} 的正整数，要求构造不超过 50 个指令，使每个正整数看作十进制字符串输入这个算法后，都能输出这个数 $+1$ 的字符串。

要求该算法对每个输入执行不超过 200 步。

35.2 算法讨论

题意是要求实现十进制加一。

我们先在最后一个数字后放上一个 “`?`”，表示这个数将要被加一，

具体方法是，现在第一个数后放上 “`??`”（命令为 “`x>>x??`”，其中 $0 \le x \le 9$），

然后找形如 “`??x`” 的串修改为 “`x??`” 直到无法找到（命令为 “`??x>>x??`”，其中 $0 \le x \le 9$），

最后将 “??” 改为 “?”。

然后考虑 “?” 前的数，如果是 9，就将 9 改成 0，将 “?” 放到 9 前面（命令为 “9?>>0?”），

如果不是 9（假设是 x），就直接改为 $x+1$ 并终止算法（如 “4?<>5”），

如果 “?” 前面没有数，就把 “?” 直接改为 1 并终止算法（命令为 “?<>1”）。

35.3 时空复杂度

时间复杂度：$O(1)$

空间复杂度：$O(1)$

36 CodeForces 303E Random Ranking

36.1 题目大意

有 n 个人参加了一场比赛，第 i 个人的得分在区间 $[l_i, r_i]$ 中均匀随机分布（得分可以是实数），求对所有 i, j，第 i 个人排名为 j 的概率。

$n \leq 80$

36.2 算法讨论

先考虑一个子问题，有 n 个人，每个人的得分范围都在 $[0,1]$ 之内，那么第 1 个人排名 $i(1 \leq i \leq n)$ 的概率均为 $\frac{1}{n}$。

然后可以将 $(-inf, +inf)$ 分为 $2n$ 段，在每一段上对每个人计算答案，

具体是这样：计算某个区间 $[L, R]$ 对第 x 个人答案的贡献时，先把 x 从 n 个人中删掉，然后 $dp[k][i][j]$ 表示加入了 k 个人，有 i 个人的得分一定比 i 小（即有 i 个人的得分一定 $< L$），有 j 个人的得分在 $[L, R]$ 范围内的概率，通过枚举下一个人的得分在 $[L, R]$ 左边还是右边还是 $[L, R]$ 内来转移。最后对每个 $dp[n-1][i][j]$，都对 $ans[x][i+1..i+j+1]$ 有 $\frac{1}{j+1} * \frac{R-L}{r_i - l_i}$ 的贡献。

这样每个人每一段的复杂度为 $O(n^3)$，总复杂度 $O(n^5)$，会 TLE。

可以尝试反着 DP，即根据 $dp[n]$ 来推出 $dp[n-1]$。在考虑第 x 个人时，如果满足 $l_x < L < R < r_x$，则转移的每个系数都 > 0，就可以从 $dp[n]$ 推出 $dp[n-1]$，如果不满足一定有 $l_x = L$ *or* $R = r_x$，这种情况对每个人只会出现两次，从 $dp[0]$ 开始暴力推一遍即可。总复杂度 $O(n^4)$，但是这题的区间范围很大，有 10^9，这种方法中多次乘法和除法会导致精度不够，在 $n = 10$ 时就会有 0.1 的误差。

注意到我们枚举每段后求的是这样的东西：有 n 个元素，可以在 $O(n^2)$ 的时间内将一个元素的集合“加”上一个元素，这个“加”操作满足交换律和结合律，但是我们不能将一个集合“减”去一个元素（精度问题），或快速地将两个集合“相加”，现在要求的是对每个 i，除掉第 i 个元素，其他元素的并集。

这个可以用分治的思想做，$solve(l, r)$ 表示当前集合第 l 个到第 r 个元素没有被加进去，要求第 l 个第 r 个元素的答案，只要令 $mid = \lfloor \frac{l+r}{2} \rfloor$，然后将 l 到 mid 加进集合，$solve(mid+1, r)$，恢复原来的集合后再将 $mid+1$ 到 r 加入集合，$solve(l, mid)$。

“加”操作的次数为 $T(l, r) = T(l, \lfloor \frac{l+r}{2} \rfloor) + T(\lfloor \frac{l+r}{2} \rfloor + 1, r) + O(r-l)$，解得 $T(1, n) = O(n \log n)$，注意这本质上是在线段树上 dfs 的过程。所以每一段的复杂度为 $O(n^2 \log n)$，总复杂度 $O(n^4 \log n)$，能通过本题。

36.3 时空复杂度

时间复杂度：$O(n^4 \log n)$

空间复杂度：$O(n^2 \log n)$

37 CodeForces 305D Olya and Graph

37.1 题目大意

有一个 n 点 m 边的有向图，对于所有边 (u,v) 满足 $u<v$，没有重边。

现在要向图中加入若干条边（可以是 0 条），要求满足：

从每个点 i 开始，可以到达 $i+1,i+2,...,n$；

每条边 (u,v) 满足 $u<v$；

没有重边；

对于两个点 $i,j(i<j)$，如果 $j-i\le k$，i 到 j 的最短路上的边数为 $j-i$，否则为 $j-i$ 或 $j-i+k$。

求方案数。

37.2 算法讨论

显然对每个 $i(1\le i<n)$，i 和 $i+1$ 之间要有一条边，否则无解。

这时所有 $i,j(i<j)$ 的最短路长度为 $j-i$，要出现 $j-i-k$，一定是有某个 $t(i\le t\le j-k-1)$ 向 $t+k+1$ 连边，且图上除了 i 和 $i+1$ 的边以外只有这种边，

因为如果有边 (u,v) 满足 $v-u\neq 1$ && $v-u\neq k+1$，那么 u 到 v 的最短路长度为 1，这个不会等于 $v-u$ 或 $v-u-k$。

接着我们发现对于所有 $(i,i+k+1)$ 的边的 i，最大值减最小值小于等于 k，否则会有某对 (u,v) 的最短路长度为 $v-u-2*k$。

然后我们只要枚举加边后图上 $(i,i+k+1)$ 的边中 i 的最小值，然后把 $2^{(}$当前情况还可以加的边的数量) 加进答案即可。

37.3 时空复杂度

时间复杂度：$O(n+m)$

空间复杂度：$O(n)$

38 CodeForces 305E Playing with String

38.1 题目大意

有一个字符串 s，两个人轮流操作，不能操作的人输。

操作步骤如下：选择一个字符串 t，再选一个 $i(1\le i\le |t|)$ 使得存在正整数 k 满足 $t[i-k\sim i+k]$ 是回文串，将 t 分成 3 段：$t[1\sim i-1],t[i],t[i+1\sim |t|]$。

两人都采取最优策略，问先手是否能赢，如果能赢输出先手第一步操作，有多个输出最小的。

38.2 算法讨论

考虑一段子串 $s[l \sim r]$，满足 $l \sim r$ 都是回文中心，$l-1$ 和 $r+1$ 不是。

显然该问题只跟这样的子串的长度有关，跟内容无关，

所以可以用 $sg[i]$ 表示长度为 i 的子串的 sg 函数值，枚举下一步操作来转移。

最后从小到大枚举第一步操作来求最小的答案。

38.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

39 CodeForces 306C White, Black and White Again

39.1 题目大意

有 w 件不同的好事和 b 件不同的坏事要在 n 天中发生，其中一天中至少发生一件事情。

这 n 天会先有若干天只发生好事，然后若干天只发生坏事，最后若干天只发生好事（若干 >0）。

问这些事件发生的方案数（每天发生的事的顺序不同算不同方案）。

39.2 算法讨论

可以计算 $fw_i = w! * C(w-1, i-1)$，表示 w 件好事在连续 i 天中发生的方案数，

同理可以计算 $fb_i = b! * C(b-1, i-1)$，表示 b 件坏事在连续 i 天中发生的方案数。

那么答案 $ans = \sum_{i=1}^{min(w,n-1)} fw_i * fb_{n-i} * (i-1)$，即考虑好事发生的总天数和一开始若干天只发生好事的天数。

39.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

40 CodeForces 306D Polygon

40.1 题目大意

要求构造一个 n 边形，每个内角都一样，但每条边的长度都不一样。

输出每个点的坐标。

$n \leq 100$。

40.2 算法讨论

先构造一个正 n 边形，然后将每条边都随机延长一些长度。

随机延长的方法是，对于顺时针第 i 条边，将第 $i-1$ 条边和第 $i+1$ 条边同时随机延长同样的长度。

40.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

41 CodeForces 309B Context Advertising

41.1 题目大意

有 n 个单词，要求选择第 i 到第 j 个单词，把它们写在 r 行中，每一行最多 c 个字符（相邻单词之间用空格隔开，不能改变单词顺序，可以有空行），求最多能选择多少个单词。

$n \leq 10^6$

41.2 算法讨论

首先可以扫一遍对每个 i 求出“如果某一行的第一个单词的编号是 i，那么这一行最后一个单词的编号”，记这个编号为 $next(i)$，于是我们要求的就是 $next^r(i) - i$ 的最大值。

由于 $next(i)$ 构成了一棵树，我们可以直接 dfs 这棵树，就能求出答案。

41.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

42 CodeForces 309D Tennis Rackets

42.1 题目大意

给你一个正三角形，每条边上有 n 个等距的点，把这条边分成 $n+1$ 段，现在要求在每条边的 n 个点上各选一个点（不能选每条边上两侧离顶点最近的 m 个点），使这三个点能连成一个钝角三角形，求方案数。

$n \leq 32000$, 时限 3s。

42.2 算法讨论

先假设钝角在底边上，枚举另外两个点，根据点积小于 0 的条件算出底边上的点的范围，然后使劲卡常数，就能 AC 了。

42.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(1)$

43 CodeForces 309E Sheep

43.1 题目大意

有 n 个区间，求一个排列，使每一对有交的区间在排列中的排名的差的最大值最小。

$n \leq 2000$。

43.2 算法讨论

先二分答案，然后可以贪心构造，具体是这样：

依次加入每个区间，用 $R[i]$ 表示第 i 个区间必须在不超过 $R[i]$ 时刻被插入，

我们找一个最小的 t，满足 $R[i] \leq t$ 的 i 的个数正好等于 t（如果大于 t 就无解），那么下一次要插入的区间一定是 $R[i] \leq t$ 的区间。

我们找一个右端点最小的区间插入，因为这样对其他区间的 R 值影响最小。

插入后维护 $R[]$ 即可。

43.3 时空复杂度

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n^2)$

44 CodeForces 311C Fetch the Treasure

44.1 题目大意

一条数轴上有 $n(1 \leq n \leq 100000)$ 个宝藏，第 i 个宝藏的位置为 $x_i(0 < x_i \leq 10^{18})$，价值为 c_i。

你一开始在 $x = 1$ 的位置，一开始每次只可以向正方向走 $k(1 \leq k \leq 10000)$ 个单位距离。

要求支持以下操作：

1、增加一种向正方向走的长度 $x(x > 0)$。

2、将第 x 个宝藏的价值减少 y。

3、询问从 $x = 1$ 开始走能得到的宝藏中价值的最大值，并删除这个宝藏。如果一个宝藏都不能得到，输出 0。如果有多个满足，删掉编号最小的。

第一种操作最多只有 20 个。

44.2 算法讨论

先考虑询问在 1, 2 操作后的情况。

我们记 $dist[i](0 \leq i < k)$ 为 $x = pk + i(p \leq 0)$ 且能走到的位置中的最小的 x，然后枚举每次走的长度转移，这相当于一个最短路问题，用 dijkstra 实现。（这个做法叫做“剩余类”）

这样就能 $O(n + 20 * k \log k)$ 找出能得到的宝藏。

我们将这些宝藏放入线段树 or 平衡树 or 堆里，然后询问时直接找最大值删除即可。

44.3 时空复杂度

时间复杂度：$O(n \log n + 20 * 20 * k \log k)$

空间复杂度：$O(n + k)$

45 CodeForces 311E Biologist

45.1 题目大意

有 n 条狗，第 i 条狗的性别为 $s[i]$ (0 或 1)，可以花 $v[i]$ 的钱改变它的性别。

现在有 m 个人，对于第 i 个人，如果给定的 $k[i]$ 条狗的性别都是 $f[i]$，那么你会得到 $w[i]$ 的钱。

求最多能得到的钱。

45.2 算法讨论

这显然是一道最小割的题目。

对每条狗，如果一开始性别为 0，则 S 向它连边，否则它向 T 连边，流量为 $v[i]$（即改变性别的代价）。

对每个人，将和它关联的狗与他连边，流量为 $+INF$（即不能被割掉），然后如果 $f[i]$ 为 0，S 向这个人连边，否则这个人向 T 连边。

然后 $\sum w_i - mincut$ 即为答案。

45.3 时空复杂度

时间复杂度：$O(maxflow(n, n + m * k))$

空间复杂度：$O(n + m * k)$

46 CodeForces 314E Sereja and Squares

46.1 题目大意

Sereja 在平面上画了 n 个点，点 i 在坐标 $(i, 0)$。然后，Sereja 给每个点标上了一个小写或大写英文字母。Sereja 不喜欢字母 “x”，所以他不用它标记点。

Sereja 认为这些点是漂亮的，当且仅当：

- 所有的点可以被分成若干对，使得每个点恰好属于一一对之中。
- 在每对点中，横坐标较小的会被标上小写字母，较大的会被标上对应的大写字母。
- 如果我们在每对点上画一个正方形，其中已知的一对点会作为正方形的相对的顶点，它们间的线段会成为正方形的对角线，那么在所有画出的正方形中不会有相交或触碰的情况。

小 Petya 擦掉了一些小写字母和所有大写字母，现在 Sereja 想知道有多少种方法来还原每个点上的字母，使得还原后这些点是漂亮的。

$n \leq 10^5$，时限 4s。

46.2 算法讨论

首先 n 是奇数时无解。

然后标字母没有什么意义，我们要求的答案就是把除 "?" 外的字符看作左括号，"?" 为一个没确定的字符，这样的条件下的合法的括号序的个数乘以 25 的 (没有确定字母的括号个数) 次方。

用 `dp[i][j]` 表示考虑前 i 个字母，多出了 j 个左括号的方案数，这个 dp 是 $O(n^2)$ 的，但是 $j > min(i, n-i)$ 和 $i+j$ 为奇数的情况不用考虑，所以常数很小，可以通过这题。

46.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

47 CodeForces 316E3 Summer Homework

47.1 题目大意

维护一个数列 $a[n]$，支持以下操作：

1 x v：将 a_x 赋值成 v

2 l r：询问 $\sum_{x=0}^{r-l}(f_x * a_{l_{i+x}})$，其中 $f_0 = f_1 = 1$,$i >= 2$ 时 $f_i = f_{i-1} + f_{i-2}$

3 l r d：区间 $[l, r]$ 的数都加上 d

一共 m 个操作

47.2 算法讨论

线段树。维护区间 $[l, r]$ 的 $\sum_{x=0}^{r-l}(f_x * a_{l_{i+x}})$ 和 $\sum_{x=0}^{r-l}(f_{x+1} * a_{l_{i+x}})$。

47.3 时空复杂度

时间复杂度：$O(n + m \log n)$

空间复杂度：$O(n)$

48 CodeForces 316F3 Suns and Rays

48.1 题目大意

给你一张只有两种颜色的图片，大小为 $n * m$ 像素，上面有若干个太阳，

太阳是一个大小为 $40 \sim 200$ 的圆或椭圆（椭圆的轴可能与坐标轴不平行）。

每个太阳上连接着若个条光线（长度为 $10 \sim 30$，宽度为 3 的线段）。

任意两个太阳不重叠，任意两条光线不相交。

要求输出每个太阳上光线的条数。

48.2 算法讨论

首先要找到有一个太阳只要 bfs 整张图就行了。

然后可以把太阳的椭圆部分“删除”，这样的话每个太阳的剩下部分的联通块个数就是光线的条数。

一种可行的做法是，先将离背景中的点距离 $\leq K1$ 的点删掉，然后将背景中离太阳距离 $\leq K2$ 的点标记为太阳的部分。（显然 K2 要大于等于 K1）

48.3 时空复杂度

时间复杂度：$O(nm)$

空间复杂度：$O(nm)$

49 CodeForces 317C Balance

49.1 题目大意

有 n 个注水容器，由 m 条管道相连，每个容器的容积是 V 升。

每次可以将整数升的水在管道连接的容器之间进行传输。

给定每个容器初始状态和目标状态的水量，要求构造一种输水方案，总步骤数不能超过 $2 * n^2$。

49.2 算法讨论

考虑每个联通块，如果初始状态和目标状态的水量和不同，则无解，

否则可以这样构造：设这个联通块的大小为 $size$，我们随便找一棵生成树，树上的叶子节点显然可以用其他点来使他达到目标状态，

具体来说只要以它为根，然后 $balance(x, d)$ 表示我们要从 x 节点取出（当 $d < 0$ 时为注入）d 升水时，x 节点最多能满足取出（或注入）多少升，然后枚举 x 的子树递归处理。

平衡好后删掉这个叶子节点，问题的规模缩小为 $size - 1$。

每次平衡一个点的复杂度是 $O(n)$，所以总复杂度为 $O(n^2)$。

49.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

50 CodeForces 317E Princess and Her Shadow

50.1 题目大意

公主和影子所在的森林可以被表示为平面上一系列的整数方格。在这里，公主和影子只能沿上下左右一格一格移动。一些格子长有树木，而影子和公主都不允许进入这样的格子。

最开始公主在 (vx, vy) ，而影子藏在 (sx, sy) 。公主，影子和树木都在不同的格子上。

影子在捉弄公主。一旦公主移动一格，只要可行，影子也在同一方向移动一格（如果这个格子没有树木）；否则，影子不移动。

影子是虚无的，所以她们并不互相干扰。

如果在某次移动后影子和公主在同一格子，那么我们说影子被抓住了。

求一种抓住影子的方案（移动步数不能超过 10^6），或输出无解。

树的数量 $m \leq 400$，坐标的范围为 $[-100, 100]$。

50.2 算法讨论

首先如果公主和影子不在同一个连通块，那一定无解，否则一定有解。

如果能走到地图外（即坐标的绝对值大于 100），那可以把公主移到地图外很远的地方（比如 $(1000, 1000)$），然后将影子移到地图外很远的地方，最后用地图上某个障碍物即可把公主和影子移到同一位置。

如果不能走到地图外，可以证明，假设坐标的范围是 L，每次将公主向影子方向移动（需要每次 bfs 得到移动的方向），最多只要 $O(L)$ 步就可以把公主和影子移到同一位置。

50.3 时空复杂度

时间复杂度：$O(L^3)$

空间复杂度：$O(L^2)$

51 CodeForces 319D Have You Ever Heard About the Word?

51.1 题目大意

你有一个字符串，一个重复块由一个字符串与自身连接而成，每一步你要找到所有子串中最短且最靠左的重复块，将其重复的部分删掉，重复该操作直到字符串中没有重复块，输出最后的字符串。

字符串长度 $L \leq 50000$。

51.2 算法讨论

我们发现，每次删掉的子串的长度是递增的，所以被删掉的子串最多有 $O(\sqrt{L})$ 种不同的长度，那么删除并重建字符串的复杂度为 $O(L\sqrt{L})$。

现在问题转化成，给定一个字符串 S，如何快速判断 S 中是否有长度为 k 的重复块。

我们将 S 分为 L/k 段，第 i 段长度为 $min(k, L-(i-1)*k)$，记第 i 段为 S_i，则一个重复块一定是由某个 S_{j-1} 的后缀,S_j,S_{j+1} 的前缀构成，且 $LCP(S_j, S_{j+1}) + LCS(S_{j-1}, S_j) >= k$。

求 LCP 和 LCS 的总次数为 $\sum_{i=1}^{L} L/i = O(L \log L)$。

可以用 hash+ 二分求 LCP 和 LCS。

51.3 时空复杂度

时间复杂度：$O(L \log^2 L + L\sqrt{L})$

空间复杂度：$O(L)$

52 CodeForces 319E Ping-Pong

52.1 题目大意

你有一个区间的集合，每次可以从一个区间 (a,b) 移动到 $c<a<d \mid\mid c<b<d$ 的区间 (c,d)，要求支持：加入一个区间、询问第 x 个区间能否移动到第 y 个区间。保证每次加入的区间的长度大于前面加入的任何区间。

一共有 n 个操作，$n \leq 100000$。

52.2 算法讨论

先考虑所以区间都已经加入的情况。

如果所有区间不存在包含关系，那么所有区间都能互相移动到，

否则将极大的互相移动到的区间的集合合并后，要判断区间 x 能否移动到区间 y，

只要先判断是否在同一个集合里，否则设 y 所在集合的 $minl=L,\ maxr=R$，

只要区间 x 能移动到区间 (L,R)，区间 x 就能移动到区间 y。

因为每次加入的区间都比前面加入的长，所以新加入的区间不会被任何区间包含，所以只要将与它相交的区间和它合并。

可以用线段树套 `vector`，维护每个点被哪些区间覆盖，合并的时候再用一个并查集维护每个集合的 $minl,\ minr$。

52.3 时空复杂度

时间复杂度：$O(n\log n)$

空间复杂度：$O(n\log n)$

53 CodeForces 323B Tournament-Graph

53.1 题目大意

构造一个 $n(3 \leq n \leq 1000)$ 个点竞赛图，满足任意两点间距离 ≤ 2。

竞赛图是没有自环且任意两个不同的点之间都有且仅有一条有向边的图。

53.2 算法讨论

$n=3$ 时答案为三元环，$n=4$ 时无解，其他情况可以这样构造：

如果 n 为奇数，对每个点 $i(1 \leq i \leq n)$，向点 $(i+k*2-2)\%n+1(1 \leq k<n/2)$ 连边（简单的说就是从 $i+1$ 开始，每次连的点的编号 $+2$）；

如果 n 为偶数，先构造 $n-1$ 的竞赛图，然后 3 向 n 连边，n 向其他奇数编号点连边，所有偶数编号点向 n 连边。

53.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

54 CodeForces 323C Two permutations

54.1 题目大意

有两个 $1 \sim n$ 的排列 A 和 B，有 m 个询问，每次问有多少个数同时在 $A[l1..r1]$ 和 $B[l2..r2]$ 中出现。

54.2 算法讨论

考虑一个 $n * n$ 的矩阵 a，$a[i][j] = 1$ 当 $A[i] = B[j]$，那么询问 $l1, r1, l2, r2$ 的答案为 $i = l1..r1,\ j = l2..r2$ 的和。

由于 a 中为 1 的元素只有 n 个，所以可以用主席树来求这个和。

54.3 时空复杂度

时间复杂度：$O((n+m)\log n)$

空间复杂度：$O(n \log n)$

55 CodeForces 325D Reclamation

55.1 题目大意

有一个 $r * c$ 的地图，把左边界和右边界粘起来使得形成一个圆柱，现在要不断地挖去其中的格子，要求任何时候都存在一条从最上方到最下方的路径 (四联通)，如果某次操作不满足要求则不做，问最后有多少次操作是成功的。

55.2 算法讨论

将这个地图水平复制两份（长为 r，宽为 $2 * c$），每次挖掉格子 (x, y) 时，如果 (x, y) 和 $(x, y + c)$ 能通过挖掉的格子联通（八连通），那么这个格子就不能挖掉，否则就把 (x, y) 和 $(x, y + c)$ 挖掉。用并查集维护挖掉的格子的连通性即可。

55.3 时空复杂度

时间复杂度：$O(n\alpha(r * c))$

空间复杂度：$O(r * c)$

56 CodeForces 325E The Red Button

56.1 题目大意

构造一个长度为 $(n + 1)$ 的数列 a，要求 $a[1] = 0,\ a[i] = (a[i-1] * 2)\%n \ ||\ a[i] = (a[i-1] * 2 + 1)\%n$，

除了 0 必须在开头和结尾出现，其他数只能出现一次。

56.2 算法讨论

首先 n 为奇数一定无解。

证明：令 $n = 2k + 1$，则倒数第二项（0 的前一项）一定是 k（否则只能是 0），但是 $n - 1$ 的前一项一定为 k（否则只能是 $n - 1$），所以 n 为奇数无解。

n 为偶数时，新建 $n/2$ 个点，编号为 $0 \sim n/2 - 1$，连 n 条有向边（编号为 $0 \sim n - 1$），编号为 i 的边连接编号为 $i/2$ 的点和编号为 $min(i, (i + n/2)\%n)$ 的点，然后这张有向图中的一条欧拉回路就是一个合法的答案。由于每个点的出度和入度都是 2，这张图一定有一条欧拉回路，把它求出来构造答案就行了。

56.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

57 CodeForces 329D The Evil Temple and the Moving Rocks

57.1 题目大意

有一个 $n * n$ 的网格，每个格子要么为空，要么可以放上石头，每块石头有一个方向 (上下左右)。

当一块石头被激活时，它会沿着它的方向移动，直到撞到网格边缘（墙壁）或其它石头，撞到的石头会被激活。

如果一块石头发生撞击之前至少移动了一个格子，那这次撞击会发出响声。如果撞击的次数达到 10^7 或没有石头被激活，则游戏结束。

一开始你只能激活一块石头。要求构造一种网格布局和一开始激活的石头，使响声次数不小于一个给定的数 x。

$x \leq 1/8 * n^3 - 1/4 * n^2$。

57.2 算法讨论

显然每块石头要发生 $O(n)$ 次撞击，那么一种可行的构造是这样：

先将整个网格里的石头按蛇形排列，具体来说，第一列方向向上，对于其他列，第 $(2k + 1)$ 行向右，第 $2k$ 行向左，边界处还要有一个向下的，这样撞击次数能保证。

为了同时发出 $O(n^3)$ 次响声，可以将每行的前 $n * t$ 个石头保留，剩下的隔一个删一个，t 约为 0.5 时发出响声次数最多，稍微调整参数即可通过。

57.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

58 CodeForces 331C3 The Great Julya Calendar

58.1 题目大意

有一个数 n，每次可以减去 n 的某个数位上的数，问最少操作几步能把 n 变为 0。

58.2 算法讨论

首先我们发现每次减去所有数位上的数中最大的是正确的。然后，用 `f[i][j][k][x]` 表示 $n = i*10^j-k$，且 n 的左边最大数位为 x，将 n 减到小于 0 的操作次数，`g[i][j][k][x]` 表示以上数字减到小于 0 后的绝对值，则 `f[i][j][k][x]=f[10][j-1][k][max(x,i-1)]+f[i-1][j][g[10][j-1][k][x] ][x]`, `g[i][j][k][x]=g[i-1][j][ g[10][j-1][k][x] ][x]`，还要考虑一些边界情况。读入 n 后只要从低位开始把 n 的每一位都减到 0 以下就可以了。

58.3 时空复杂度

时间复杂度：$O(\log n)$

空间复杂度：$O(\log n)$

59 CodeForces 331D3 Escaping on Beaveractor

59.1 题目大意

$m*m$ 的网格上有 n 个箭头，每个箭头都平行于坐标轴，箭头直接没有公共点。

有 q 个询问，每个询问是从 (x_i, y_i) 开始，每秒钟移动一个单位距离，且在箭头上时要沿着箭头所指方向走，问 t_i 秒后走到哪里。

$n, m, q \leq 10^5$

59.2 算法讨论

把所有箭头看成点，每个箭头向它所指方向第一个碰到的箭头连边，我们要求的就是从图上某个点开始走 k 个单位距离后走到哪里。

建图：沿着一个箭头向前走可能有三种情况，没有碰到箭头，碰到一个与它共线的箭头，碰到一个与它垂直的箭头。

比如说向上的箭头，与它垂直的箭头是横向的，先把横向的箭头按 y 坐标从大到小排序，然后按顺序在线段树上进行区间覆盖，将 $y \geq y0$ 的横向箭头覆盖完后，当前线段树的第 i 个点就表示从 $(i, y0)$ 开始向上走，最先碰到的横向箭头。

使用可持久化线段树就可以做到在线询问（也可以将询问和箭头一起排序去掉可持久化）。

对于共线的箭头，如竖向的，对每一列竖向的箭头按 y 坐标排序，然后查询一个点向上走最先碰到的竖向箭头时只要在数组中二分。

每个箭头向第一个碰到的箭头连边，就能建出一张 n 个点 n 条边的有向图。

这张图的每个联通块一定是一个环 + 若干棵内向树，一次询问一定是在树上走一段，然后在环上走若干圈。

这样求答案略麻烦，可以考虑直接使用倍增，预处理每个点在图上走 2^i 条边后到达的点的编号和走过的距离。

询问时先查找第一个碰到的箭头，然后在图上用倍增得到答案。

时间复杂度 $O((n+q)\log n)$，空间复杂度 $O(n\log n)$

59.3 时空复杂度

时间复杂度：$O((n+q)\log n)$

空间复杂度：$O(n\log n)$

60 CodeForces 332D Theft of Blueprints

60.1 题目大意

给出一个 n 个点的带权无向图，满足对于任意一个大小为 k 的顶点集合 S，恰好有一个点与 S 每一个点都有边。

令这个点为 $v(S)$，并且对 S 进行操作的代价是 S 中每个点与 $v(S)$ 的边权之和。

现在求对于一个大小为 k 的子集操作代价的期望。

60.2 算法讨论

可以证明，$k=1$ *or* $k=2$ *or* $k=n-1$，于是就可以直接计算答案了（$k=1$ *or* $k=n-1$ 时答案为两倍边权和除以 n，$k=2$ 时对每一对点计算即可）。

当然这题也可以直接枚举 $v(S)$，然后用组合数计算答案。

60.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

61 CodeForces 335D Rectangle And Square

61.1 题目大意

你有 n 个矩形（编号从 1 到 n）。所有矩形的四个角都是整点，并且两组对边分别平行于 X 和 Y 两坐标轴。不同的矩形可能接触，但是不会重叠。（换言之，不存在一个点严格处在多个矩形的内部）

现在你的任务是：判断是否存在一个构成正方形的子集。换言之，是否存在一个矩形的子集和一个正方形，使得：

1. 每个处在正方形边缘或内部的点都存在于至少一个该子集内的矩形的边缘或内部；
2. 每个处在该子集内某一矩形的边缘或内部的点都处在那个正方形的边缘或内部。

坐标范围为 $[0,3000]$，$n\le 10^5$。

61.2 算法讨论

枚举正方形左上角的矩形，再枚举边长，然后 $O(1)$ 判断某个正方形是否合法。

判断的方法有很多，我是给每个点随机一个范围在 $[0, W]$ 里的权值，然后保证输入的每个矩形内的权值的异或和为 0（只要修改每个矩形的一个值即可），如果一个正方形内权值的异或和不等于 0，那一定不合法，否则有 $1/W$ 的概率不合法，只要 W 足够大（如 2^{32} 或 2^{64}）就可以。

记坐标范围为 L，这样做的复杂度是 $O(nL + L^2)$，可能会被卡常数。枚举边长时沿着正方形边界上的矩形枚举，就不会 TLE 了。

61.3 时空复杂度

时间复杂度：$O(nL + L^2)$

空间复杂度：$O(n + L^2)$

62 CodeForces 338E Optimize!

62.1 题目大意

有两个数组 $a[1..n]$ 和 $b[1..len]$ 和一个数 h，求有多少个 i，使得对于数组 $a[i..i + len - 1]$ 和 $b[1..len]$，存在一种匹配使每对数之和 $\leq h$。

62.2 算法讨论

对于一对数 x, y，如果 $x + y \geq h$，则 $h - x \leq y$，所以可以考虑数组 $A[i] = h - a[i]$，将 A 数组和 b 数组的数放在数轴上，$A[]$ 中的数记为 $+1$，$b[]$ 中的数记为 -1，那么 $A[]$ 和 $b[]$ 匹配当且仅当对每个 X，$(-inf, X]$ 上的数的和 ≥ 0。用线段树维护前缀和的 RMQ 即可。

62.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

63 CodeForces 339E Three Swaps

63.1 题目大意

有一个 $1 \sim n$ 的排列，一开始为 $1, 2, 3, ..., n$，每次操作可以选择一个 $[l, r]$，使 $[l, r]$ 中的所有数反向排列。给定一个排列 A，问能否在三个操作内从初始状态达到 A。

63.2 算法讨论

三次操作后序列中最多会有 7 段连续上升或连续下降的子序列，因此可以将 $1 \sim n$ 的排列分为 7 段进行整体操作。

但是由于子序列的长度等于 1 时的情况会使上述方法出现 bug，应该把整个排列分成 21 段，即每个子序列的第一个和最后一个数单独拿出来作为一段。

然后暴力枚举每次操作并判断，复杂度是 $O(21^7)$，会 TLE。

用 meet in the middle 思想优化这个算法，先反向搜索一步，用 hash 存下来，然后正向搜索两步即可，复杂度 $O(21^5)$，可以通过所有数据。

63.3 时空复杂度

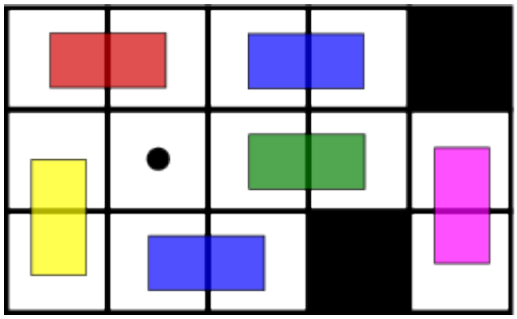
时间复杂度：$O(21^5)$

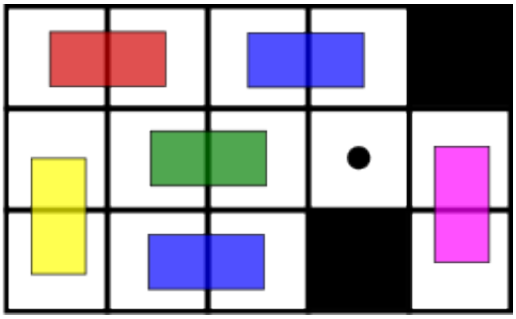
空间复杂度：$O(21^2)$

64 CodeForces 342D Xenia and Dominoes

64.1 题目大意

有一个 3 * n 的拼图（如图），有些格子为禁止块。





你要用多米诺骨牌覆盖这个拼图，且满足:

每个多米诺骨牌覆盖正好两个非禁止块；

没有两个多米诺骨牌覆盖同一块区域；

有且仅有一个非禁止块没被任何多米诺骨牌覆盖。

问你有多少种覆盖方案，满足以上条件，且至少有一块多米诺骨牌可以沿着它的方向（比如横向的多米诺骨牌只能横向移动）移动一格到没被覆盖的点上。

64.2 算法讨论

只考虑恰好完全覆盖，可以用状压 DP，令 $dp[i][mask]$ 为 $1 \sim (i-1)$ 列完全覆盖，且第 i 列未覆盖的状态为 mask 的方案数，转移时枚举下一列的覆盖情况，$dp[n][0]$ 即为答案。

对于有移动的情况，可以先将能移动的那两个格子标记为禁止块，然后做状压 DP。

这样可能会重复统计一些情况，要用容斥原理排除掉。

64.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

65 CodeForces 348E Pilgrims

65.1 题目大意

给一棵 n 个点的带边权的树，其中有 m 个点是黑点，剩下的是白点。现在要删掉一个白点。

一个黑点会不高兴，当且仅当删掉那个白点后，离这个黑点最远的所有黑点都与这个黑点不连通。

求删掉一个白点最多能使几个黑点不高兴，以及使最多黑点不高兴的删除方案数。

$2 \leq m < n \leq 10^5$

65.2 算法讨论

容易证明，对于每个黑点 i，每个离 i 最远的黑点到 i 的路径一定经过距离最远的那对黑点之间的路径上的中点（记为 A）。

然后考虑每个黑点，它会不高兴当且仅当删掉的白点在它到 离它最远的所有黑点的 lca 的路径上。

由于这条路径一定经过点 A，所以这个 lca 就是以 A 为根，A 的子树中不包含这个黑点的子树里的离 A 距离最远的黑点的 lca。

然后只要路径加一个数，求单点的值即可。

65.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

66 CodeForces 351D Jeff and Removing Periods

66.1 题目大意

一个序列 $a[1], a[2], ..., a[n]$，每次做以下操作：

选择 3 个数 $v, t, k (1 \leq v, t \leq n;\ 0 \leq k;\ v + t * k \leq n)$, 满足 $a[v] = a[v + t] = a[v + 2 * t] = ... = a[v + t * k]$，

删除 $a[v], a[v + t], ..., a[v + t * k]$，将剩下的数重新标号成 $a[1..n - k + 1]$ 并重新排列。

给你一个序列 $b[1..n]$，每次询问 $b[l..r]$ 用以上的操作最少要几次才能删光。

66.2 算法讨论

重新排列后，剩下的最少操作次数就是序列中不同的数的个数，

所以每个询问的答案就是 $a[l..r]$ 中不同的数的个数 $+$(存在一个数 x 在 $a[l..r]$ 中出现的下标为等差数列 $?\ 0 : 1$)。

将询问按右端点排序后用线段树/树状数组就可以做了。

66.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

67 Google Code Jam World Final 2009 C Doubly-sorted Grid

67.1 题目大意

我们称一个 $N * M$ 的小写字母网格是双排序的，当任意一行从左到右都不下降，任意一列从上到下也不下降。

在下面的例子中，前两个网格是双排序的，而后两个不是。

```
a b c       a c e       a c e g       b a s e
d e f       a d e       c d e f       b a s e
g h i       b d g       x x y y       b a s e
```

给定一个 $N * M$ 的网格，其中有些格子已经填了字母，问有多少种方案把剩下的格子填满并使这个网格是双排序的。

$1 \leq N, M \leq 10$

67.2 算法讨论

算法一

假设 c 是一个小写字母，则网格中所有 $\leq c$ 的字母与其他字母的界线一定是一条从左下角到右上角的路径（这样的路径最多有 $C(20, 10) = 184756$ 条）。

令 $a[i]$ 为第 i 列上最下面的 $\leq c$ 的字符的行号（如果不存在则为 0），则 $a[i] \geq a[i + 1]$（$1 \leq i < M$）。

所以可以用 $dp[S][c]$ 表示 $\leq c$ 的字母的界线是 S 时，原问题的答案。

枚举所有满足 $a[i] > a[i + 1]$ 的 i，再考虑 $(a[i], i)$ 这个位置是否有字母 c 来转移，需要枚举子集 + 容斥，复杂度为 $C(20, 10) * 2^{10} * 26$。

算法二

用 $dp[S][c][r]$ 表示 $\leq c$ 的字母的界限为 S，且字母 c 只在列 $1 \sim r$ 出现的答案，转移时就不用枚举子集，复杂度为 $C(20, 10) * 10 * 26$，能通过所有数据。

67.3 时空复杂度

时间复杂度：$O(C(20, 10) * 10 * 26)$

空间复杂度：$O(C(20, 10) * 10 * 26)$

68 Google Code Jam World Final 2009 B Min Perimeter

68.1 题目大意

给一个整数坐标的点集，求点集中最小的三角形的周长（面积为 0 的三角形也可以）。

$n \leq 100000$

68.2 算法讨论

乱搞：用 k-d tree 求出离每个点最近的 $5 \sim 10$ 个点，然后在这 $5 \sim 10$ 个点中找一个周长最小的三角形，不知道是否正确，但是能通过 GCJ 的官方数据和 tsinsen 上的数据。

正解：这题类似于最近点对问题。考虑分治，每次用一条平行于 y 轴的直线将点集划分为等大的两部分，先递归处理左右两个部分，设这时的答案为 p，然后将离分割线的距离不超过 $p/2$ 的点按 y 坐标排序，然后按顺序扫一遍，对每个点只考虑一个长为 p，宽为 $p/2$ 的长方形内的点与它构成三角形。

可以证明，这样的长方形内最多只有 16 个点。排序可以用下一层分治的结果归并，所以复杂度是 $O(n \log n)$ 的。

68.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

69 Google Code Jam World Final 2009 D Wi-fi Towers

69.1 题目大意

给你一个无线发送塔的网络。每个塔有一个范围，可以将数据发送给附近的塔，只要那个塔的距离小于或等于塔的范围。

塔用一个古老的通信协议 A，但有一个新且更好的协议 B 可用。我们正在考虑将一些塔的通信协议升级到 B，以实现更好的带宽。

有一个重要的限制: 如果一个塔 T 使用新的协议 B，每个在塔 T 的范围内的大楼也必须使用该协议，这样他们就可以理解从 T 送出的数据。反之没有必要——使用新的协议 B 的塔可以理解使用旧的协议 A 的塔送出的数据。

你的任务是选择最佳的升级方案，升级塔有一些好处，但同时也有安装成本。所以每个塔都有正或负的得分 (即升级塔的利润)。选择一种升级的方案使总分最大。

有 T 组数据，每组数据塔的数量为 n，$T \leq 55$, $n \leq 500$。

69.2 算法讨论

如果选了第 i 个塔就必须选第 j 个塔，那么就从 i 向 j 连一条边，然后就是个最大权闭合子图问题了，复杂度 $O(T * maxflow(n, n^2))$，由于网络流的复杂度上界很难达到，这个算法不会 TLE。

实际上我们可以在不改变图的连通性的情况下，优化边的数量。

考虑三个点 A, B, C，如果 $A \to C$, $B \to C$ 都有边，且 $\angle ACB < 60°$，那么可以删掉 AC 和 BC 中较长的那条边（假设是 AC），因为这时 A 一定能通过 B 走向 C。

这样做以后，每个点的入边最多有 6 条，所以边数就是 $O(n)$ 的了，复杂度为 $O(T * maxflow(n, 8n))$。

69.3 时空复杂度

时间复杂度：$O(T * maxflow(n, 8n))$

空间复杂度：$O(n)$

70 Google Code Jam World Final 2010 A Letter Stamper

70.1 题目大意

有一个栈式的打字机，支持下面三种操作：

(1) 把一个字母加到栈顶。

(2) 打印栈顶的字母。

(3) 删除栈顶字母。

要求打印一个只包含 “A” 或 “B” 或 “C” 的字符串，打印开始前和打印结束后栈必须都是空的，问最少操作次数。

字符串的长度 $n \leq 2000$。

70.2 算法讨论

考虑一个最优解，在任何时候，打字机的栈一定满足以下性质：

1、不会有两个连续的字母。这很显然。

2、不会有形如 `ABA` 的一段。如果有的话，一定是在某个时刻，栈顶为 `AB`，要求输出 `A`，这时可以先弹出 `B`，再输出 `A`，操作数量不会比压入一个 `A` 要多。

3、一定是形如 ABCABCABC... 的串。由第一和第二个性质可以得到。

所以不同的栈的数量只有 $O(n)$ 级别（准确的说是 $6 * n$），

可以用 `dp[i][j]` 表示打了前 i 个字符，栈的状态是 j 的最少操作次数，复杂度是 $O(n^2)$ 的。

70.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

71 Google Code Jam World Final 2010 C Candy Store

71.1 题目大意

你在经营一家糖果店，你可以从你的供应商处买来装有任意整数克糖果的盒子。

每天有至多 k 位顾客到你的店里来。从第 1 个人开始，每个人会买整数克的糖果，这个整数的范围为 $[1, C]$。

如果一个人想要买 4 克糖果，你可以给他 1 个 4 克的盒子，也可以是 1 个 2 克的盒子和 2 个 1 克的盒子。

你最少需要买几个盒子才能保证，不管每个人想买多少糖果，你总是可以给他们对应质量的糖果？

注意：当一个人选择自己想买多少糖果后，你知道之前的人已经买了多少糖，但不能预知之后的人打算买多少糖。

有 T 组数据。$T \leq 100,\ k \leq 1000,\ C \leq 10^{12}$。

71.2 算法讨论

有一种贪心的做法是，设一个人要买的糖果的质量为 x，每次找一个质量最大且不超过 x 的盒子给他，直到 $x = 0$。

构造每个盒子的质量的方法是，记当前所有盒子的质量的和为 S，加入一个质量为 $\lfloor \frac{S}{k} \rfloor + 1$ 的盒子，反复这么做知道 $S \geq k * C$。

可以证明这样贪心和这样构造都是最优的。

71.3 时空复杂度

时间复杂度：$O(T * k * \log C)$

空间复杂度：$O(1)$

72 Google Code Jam World Final 2011 A Runs

72.1 题目大意

有一个由小写字母“a” - “z” 组成的字符串。每一个极大连续相同的子序列被称为一个“run”。比如说，“bookkeeper”有 7 个“run”。

给定一个字符串，问将其重新排列，有多少种不同的排列使得它和原字符串有相同的“run”?

两个字符串不同的定义是两个字符串在某个相同位置上的字母不同。

字符串长度 n 不超过 450000，run 的个数 m 不超过 100。

72.2 算法讨论

考虑用 $f[i][j]$ 表示只用了前 i 种字母，有 j 个 run 的方案数。

我们记 L 为前 i 种字母的个数和，那么对于只用前 i 种字母且有 j 个 run 的字符串，插入一个第 $i+1$ 种字母时，有 $j+1$ 个插入位置会使 run 的个数增加 1，另外 $L-j$ 个插入位置会使 run 的个数增加 2，

所以我们枚举 $j+1$ 个位置中有几个位置至少插入了一个字母（记为 a），另外 $L-j$ 个位置中有几个位置至少插入一个字母（记为 b），

于是 $f[i][j]$ 就对 $f[i+1][j+a+2*b]$ 产生 $f[i][j]*C(j+1,a)*C(L-j,b)*C(cnt[i+1]-1,a+b-1)$ 的贡献，其中 $cnt[i]$ 为第 i 种字母的总个数。

按照这个 DP 就可以了，复杂度是 $O(26 * m^3)$，还要用 $O(n * m)$ 的时间预处理组合数。

72.3 时空复杂度

时间复杂度：$O(n * m + 26 * m^3)$

空间复杂度：$O(n * m)$

73 Google Code Jam World Final 2011 C Program within a Program

73.1 题目大意

在一个数轴上有一个机器人，该机器人一开始在原点，需要走到 n 的位置并停下来。

数轴上每个位置都有一个标记，一开始所有标记都是 0。

机器人本身有一个状态，用一个整数表示，一开始为 0。

机器人的程序由若干条命令组成，每条命令是形如 "<S> <M> -> <action>" 的字符串，表示机器人的为 S，机器人当前位置的标记为 M 时，将执行 action 这个操作，不能有两条命令的 S 和 M 都相等。

若 action 是形如 "<D> <NS> <NM>" 的字符串，则机器人会先将当前位置的标记改为 NM，然后将状态改为 NS，然后向 D 表示的方向移动一个单位长度（用 "W" 表示负方向，用 "E" 表示正方向）。

若 action 是 "R"，则程序结束。

现在给定 n，要求构造出一个最多包含 30 条命令的程序，使机器人能在 $1.5 * 10^5$ 步内走到 n 的位置并结束程序。

$0 \leq n \leq 5000$

73.2 算法讨论

可以考虑把 n 表示成二进制数，然后记在数轴上，每次将这个数减一并向右移动一个单位，如果这个数等于 0 就结束程序。

具体来说可以这样：

把 n 表示成 13 位的二进制数（我们用 100 表示二进制的 0，用 101 表示二进制的 1），放在数轴的 $0 \sim 12$ 的位置，然后把机器人移到 13 的位置。

每次减一时，先向左找到第一个标记为 101 的位置，将这个位置的标记改为 100，找的过程中把碰到的 100 都改成 101，如果没找到 101，就移到这个数的最高位所在位置，然后结束程序，否则将每一位的标记都向右移一个单位长度，并继续将这个数减一。

这样构造出来的程序有 29 条命令，机器人移动的步数大约为 $29n$，满足题目中的条件。

$n = 5000$ 的程序如下：

```
0 0 -> E 120904 101
120904 0 -> E 110904 100
110904 0 -> E 100904 100
100904 0 -> E 90392 101
90392 0 -> E 80136 101
80136 0 -> E 70008 101
70008 0 -> E 60008 100
60008 0 -> E 50008 100
50008 0 -> E 40008 100
40008 0 -> E 30000 101
30000 0 -> E 20000 100
20000 0 -> E 10000 100
10000 0 -> E -11 100
-11 0 -> W -1 0
```

```
-1 100 -> W -1 101
-1 101 -> W -2 100
-2 100 -> W -2 100
-2 101 -> W -2 101
-2 0 -> E -1000 0
-1000 100 -> E -1100 0
-1000 101 -> E -1101 0
-1100 100 -> E -1100 100
-1100 101 -> E -1101 100
-1101 100 -> E -1100 101
-1101 101 -> E -1101 101
-1100 0 -> E -11 100
-1101 0 -> E -11 101
-1 0 -> E 233 0
233 101 -> R
```

73.3 时空复杂度

时间复杂度：$O(\log n)$

空间复杂度：$O(1)$

74 Google Code Jam World Final 2013 E Let Me Tell You a Story

74.1 题目大意

有一个长度为 n 的序列，每次操作可以删掉一个数，直到这个序列是不上升的。

问合法的操作序列的个数（操作序列可以为空）。

$n \leq 2000$。

74.2 算法讨论

不妨考虑计算不合法的操作序列的个数。

对于一个长度为 L 的操作序列，如果在第 L 个操作之前，这个序列就是不上升的，那么这个操作序列不合法。

所以考虑原序列的每一个不上升的子序列，设长度为 i，那么不合法的操作序列个数就要加上 $(n-i)! * i$。

另外要考虑原序列的每一个非不上升子序列，操作序列不能在这时结束，设长度为 i，那么不合法的操作序列个数就要加上 $(n-i)!$。

所以我们只要求出原序列的每一种长度的不上升子序列的个数。

用 $f[i][j]$ 表示最后一个数是 i，且长度为 j 的不上升子序列个数，

$f[i][j] = \sum_k f[k][j-1] \ \ where\ a_k >= a_j$。

只要用 BIT 优化 DP 即可。

74.3 时空复杂度

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n^2)$

75 Google Code Jam World Final 2014 E Allergy Testing

75.1 题目大意

有 n 种食物，其中一种会导致过敏，但是不知道是哪种，要通过实验来确定。

每次实验可以选取若干种不同的食物全吃下去，A 天之后就可以知道是否过敏，

如果没有，就说明吃下去的食物都不会导致过敏，否则还要花 $(B-A)$ 天等过敏反应褪去。

每一个实验必须在上一个实验完成后开始，开始时可以根据前面的实验结果来选择要吃的食物集合。

问最坏情况下，最少要多少天才能找到会导致过敏的食物。

$n \leq 10^{15},\ 1 \leq A \leq B \leq 10^{12}$。

75.2 算法讨论

考虑一个简单一些的问题：在 T 天内，最多能从几种食物中找到会导致过敏的食物。

我们把每次实验后决策的过程画成二叉树。根节点的权值为 T，每次实验时吃掉大小为右子树叶子节点个数的食物集合，如果没有过敏，就向左子树走，同时点的权值减少 A，否则向右子树走，同时点的权值减少 B，如果某个时刻点的权值小于 B，那么就不能继续决策了，即这个节点为叶子节点。

我们要统计的是，根节点的权值为 T 的决策树上有多少个叶子节点。

假设从根到某个叶子节点的路径上有 p 次向左子树走，有 q 次向右子树走，那么 p,q 满足 $T-B < pA+qB \leq T$，这样的叶子节点有 $C(p+q,q)$ 个。

我们要求的就是对于所有满足 $T-B < pA+qB \leq T$ 的 p,q 的 $C(p+q,q)$ 的和。

由于 q 的大小是 $O(\log T)$ 级别，我们可以先枚举 q，然后求出 p 的范围 $[p1,p2]$，由于 $\sum_{p=p1}^{p2} C(p+q,q) = C(p2+q+1,q+1) - C(p1+q,q+1)$，可以在 $O(\log T)$ 的时间内计算出，所以我们能在 $O(\log^2 T)$ 的时间内求出“在 T 天内，最多能从几种食物中找到会导致过敏的食物”。

对于原问题，我们二分答案然后再求这个即可。

75.3 时空复杂度

时间复杂度：$O(log^3 n)$

空间复杂度：$O(1)$

76 USACO DEC 05 Cow Patterns

76.1 题目大意

有两个由 $1 \sim S$ $(1 \leq S \leq 25)$ 的数字组成的串，要求出所有第一个串与第二个串匹配的位置。

两个串匹配的定义是，这两个串每个位置上的数在整个串中的排名构成的串正好相同。

串的长度 $N \leq 100000$。

76.2 算法讨论

一共只有 25 种不同的数字，那么只要对每个数字预处理 hash，然后对每个起始位置枚举每个数字用 hash 判断。

76.3 时空复杂度

时间复杂度：$O(N * S)$

空间复杂度：$O(N * S)$

77 USACO DEC 07 Best Cow Line

77.1 题目大意

有一个字符串 A，和一个空串 B，每次你要从头或尾取出（并删掉）一个字符，然后加到 B 的最后，要求删完 A 的所有字符后 B 的字典序最小。

77.2 算法讨论

记 A 串当前剩下的部分为 S，长度为 n。如果 $A[1] \neq A[n]$，那么只要选较小的那个，否则比较 A 串和翻转过的 A 串，如果 A 串较小，就选 $A[1]$。

比较两个串的大小可以用 Hash+ 二分或后缀数组，复杂度都是 $O(n \log n)$（n 为 A 的长度）。

77.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

78 USACO DEC 08 Fence

78.1 题目大意

给定 n 个点，问最多选取多少个点，使它们构成一个凸多边形。

$n \leq 250$，保证任意三点不共线。

78.2 算法讨论

首先我们可以枚举凸多边形中 y 坐标最小的点 x，然后 $dp[i][j]$ 表示凸多边形上最后两个点为 i 和 j 时，凸多边形点数的最大值，然后只要枚举每个在射线 ji 与射线 xi 中间的点转移即可，复杂度是 $O(n^4)$，其中状态数 $O(n^3)$，转移 $O(n)$，这个做法会 TLE。

由于转移相当于权值为 1 的边，我们可以考虑用“左儿子右兄弟”的思想来优化转移。

记 $first[i][j]$ 为在射线 ji 左侧，且射线 ix 与射线 ji 夹角最小的 x。

记 $next[i][j]$ 为在射线 ji 左侧，且射线 jx 与射线 ji 夹角最小的 x。

那么 $dp[i][j]$ 就可以转移到 $dp[first[i][j]][i]$ 和 $dp[next[i][j]][j]$，边权分别为 1 和 0，我们可以用 bfs 来实现这个 DP。

$first$ 数组和 $next$ 数组可以在 $O(n^3)$ 时间内预处理出来，所以总复杂度为 $O(n^3)$。

78.3 时空复杂度

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

79 USACO DEC 10 Threatening Letter

79.1 题目大意

给定两个字符串 A, B ($|A|, |B| <= N$)，问 B 最少能由几段 A 的子串（可以重复）拼接成。

79.2 算法讨论

考虑 $dp[i]$ 表示 $B[1..i]$ 的答案，则 $dp[i] = min\{dp[j]\} + 1$ $(i - len[i] \le j < i)$，

其中 $len[i]$ 表示 $max\{lcs(A[1..x], B[1..i])\}(1 \le x \le |A|)$，即 $B[1..i]$ 与 A 的所有前缀的 lcs 的最大值。

将 A 串和 B 串反过来建后缀数组，然后对每个 B 的前缀 $B[1..i]$ 求它与 A 的前缀中 $rank$ 与它最接近的两个前缀的 lcs，即为 $len[i]$。

79.3 时空复杂度

时间复杂度：$O(N \log N)$

空间复杂度：$O(N)$

80 USACO DEC 12 First

80.1 题目大意

给你 n 个字符串，总长度为 L，问对于每个字符串，是否存在一种改变 $a \sim z$ 字母的字典序的方式，使第 i 个字符串的字典序最小。

$n \le 30000,\ L \le 300000$

80.2 算法讨论

对所有字符串建一棵 trie。对于第 i 个字符串的在 trie 上的某个结点，如果它的出边有 $c_1, c_2, ..., c_n$（$c_1..c_n$ 为小写字母），字符串上这个位置为 c_k，那么在新的字典序中，$c_k < c_i$ $(1 \leq i \leq n,\ i \neq k)$。

我们将这些大小关系看作有向边，那么如果图中没有环，这种字典序就可以构造出来，否则不能。

有一种情况要特判：如果 $s1$ 是 $s2$ 的前缀，那么 $s2$ 不可能成为字典序最小的。

80.3 时空复杂度

时间复杂度：$O(n * 26^2 + L * 26)$

空间复杂度：$O(L * 26)$

81 USACO JAN 07 Cow School

81.1 题目大意

有 n 次考试，每次考试的得分为 T_i，满分为 P_i。

在计算总分时，先将 P_i/T_i 最高的 D 个试卷去掉，然后总分为剩下的 T_i 的和除以剩下的 P_i 的和，求所有满足以下条件的 D：如果选择 D 个试卷去掉，算出的总分能比“把 P_i/T_i 最高的 D 个试卷去掉”算出的总分要高。

81.2 算法讨论

对于某个 D 的值，如果能提高算出的总分，那么一定存在一种方案是从 P_i/T_i 最高的 D 个试卷中选出一个 i，剩下的选出一个 j，交换试卷 i 和试卷 j。

考虑某个 D，记剩下的 T_i 的和为 ST，剩下的 P_i 的和为 SP，则我们要找到一组 $(i,\ j)$，使

$ST/SP\ <\ (ST + T_i - T_j)\ /\ (SP + P_i - P_j)$

即 $ST * SP + SI * P_i - ST * P_i\ <\ ST * SP + SP * T_i - SP * T_j$

即 $ST * P_i - SP * T_i\ <\ ST * P_j - SP * T_j$

将所有试卷按 P_i/T_i 从大到小 sort 后，我们只要对每个 D，判断 $min(ST * P_i - SP * T_i)$ $(1 \leq i \leq D)$ 是否小于 $max(ST * P_j - SP * T_j)$ $(D < j \leq n)$。

考虑分治，设当前处理的区间为 $[l, r]$，$mid = (l + r)/2$，将编号在 $[l, mid]$ 中的试卷按 (T_i, P_i) 建凸包，然后更新编号在 $[mid + 1, r]$ 中的最小值，将编号在 $[mid + 1, r]$ 中的试卷建凸包，然后更新编号在 $[l, mid]$ 中的最大值。需要将点按照 T_i 排序，询问按照斜率排序，这可以由下一层的分治的结果归并得到。

81.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

82 USACO JAN 09 Travel

82.1 题目大意

Description 给一张 n 点 m 边的带权无向图，对于每个 $i(2 \leq i \leq n)$，保证 1 号点到 i 号点的最短路径唯一，求 1 号点到 i 号点不经过最短路径上的最后一条边的最短路径长度。

82.2 算法讨论

先用 dijkstra 求出 1 号点到每个点的最短路径长度以及最短路径树。

对于每个点 i，我们要求的路径一定是从根开始走到某个点，然后经过一条非树边到达 i 的子树内的点（或 i 本身），然后向上走到 i。

我们用可并堆来维护所有非树边 (x, y) 的 $dist_x + dist_y +$ 这条边的长度（其中 x 在 i 的子树内，y 不在 i 的子树内），对每个 i，将它的每个儿子的可并堆都并到 i 这个节点上后，不断 pop 直到找到第一个 y 不在 i 子树内的 (x, y)，这时 $dist_x + dist_y +$ 这条边的长度 $- dist_i$ 就是 i 号点的答案。

82.3 时空复杂度

时间复杂度：$O((n+m)\log n)$

空间复杂度：$O(n+m)$

83 USACO MAR 08 Land Acquisition

83.1 题目大意

有 n 块长方形的土地，第 i 块的长为 X_i，宽为 Y_i。

购买一些土地的价格为 $max(X) * max(Y)$，不论同时买几块。

问买下所有土地最少的花费。

$n \leq 50000$

83.2 算法讨论

显然如果存在 i, j，使 $X_i \geq X_j$ && $Y_i \geq Y_j$，那么可以把土地 j 删掉，这样剩下的土地按 X 从小到大排序后，Y 是不上升的。

于是可以这样 DP：$f[i] = max\{f[j] + Y_{j+1} * X_i\},\ 0 \leq j < i$。

容易证明这个 DP 的决策是单调的，这样就可以用单调栈 + 二分做到 $O(n \log n)$。

83.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

84 USACO MAR 09 Cleaning up

84.1 题目大意

有 n 个数，要将这些数分成若干段，一段的权值定义为这段不同的数的个数的平方。

求最小的权值和。

84.2 算法讨论

答案最大一定是 n，所以每一段不同的数的个数一定不会超过 $\sqrt{n}$。

只要用一个双向链表维护不同的数就行了。

84.3 时空复杂度

时间复杂度：$O(n\sqrt{n})$

空间复杂度：$O(n)$

85 USACO MAR 12 Cows in a Skyscraper

85.1 题目大意

有 N 个物品，问最少要用几个容量为 W 的背包装下。

$N \leq 18,\ W \leq 10^8$

85.2 算法讨论

一个很显然的想法是 $f[S]$ 表示 S 这个集合最少用几个背包，然后枚举子集转移，这样是 $O(3^N)$ 的，会 TLE。

如果我们同时用 $f1[S]$ 表示 S 这个集合在用最少的背包装下时，最后一个背包剩下的容量的最大值，就可以做到 $O(N * 2^N)$。

85.3 时空复杂度

时间复杂度：$O(N * 2^N)$

空间复杂度：$O(2^N)$

86 USACO MAR 13 Hill Walk

86.1 题目大意

有 n 座山，每一座山用一条 $(x1, y1)$ 到 $(x2, y2)$ 的线段表示，其中 $x1 < x2$ && $y1 < y2$，$(x1, y1)$ 这个点属于这座山，$(x2, y2)$ 不属于。保证没有两条线段相交（包括端点重合）。

你从第一座山的 $(0, 0)$ 点（保证第一座山的 $x1 = y1 = 0$）开始走，每次走到这座山的 $(x2, y2)$ 时，向 y 轴负方向掉落直到碰到另一座山。当 y 轴负方向没有别的山的时候结束。

问一共经过了几座山。

$n \leq 100000$

86.2 算法讨论

这题本质上是要求出一个 $next[]$ 数组，$next[i]$ 表示第 i 条线段的 $(x2, y2)$ 正下方的第一条线段的编号。

由于所有线段不相交，所以可以按 x 轴从左到右扫描，把线段插入一个 `set` 里，然后 $next[i]$ 就是在 `set` 中的 i 的前驱。

86.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

87 USACO OPEN 07 Connect

87.1 题目大意

有一个 $n * m$ $(1 \leq n \leq 2,\ 1 \leq m \leq 15000)$ 的网格图，要求支持加边，删边，询问两点之间是否联通。

$(x1, y1)$ 与 $(x2, y2)$ 的“联通”要求经过的路径不能超出列 $y1$ 和列 $y2$ 的范围。

87.2 算法讨论

在线段树的每个区间 $[l, r]$ 上维护一个数组 $f[n][n]$，$f[i][j]$ 表示 (i, l) $(1 \leq i \leq n)$ 与 (j, r) $(1 \leq j \leq n)$ 之间的连通性，再根据 mid 与 $mid + 1$ 的连边将 $[l, mid]$ 和 $[mid + 1, r]$ 的信息合并。

询问 $(x1, y1)$ 和 $(x2, y2)$ 时只要求出 $[y1, y2]$ 区间的 f 数组，输出 $f[x1][x2]$ 即可。

87.3 时空复杂度

时间复杂度：$O(m + q \log m)$

空间复杂度：$O(m)$

88 USACO OPEN 08 Cow Neighborhoods

88.1 题目大意

有 n 只奶牛，第 i 只奶牛在坐标 (X_i, Y_i) 上（保证坐标不重复）。

两只奶牛 i, j 属于同一个群的条件是 $|X_i - X_j| + |Y_i - Y_j| \leq C$，或存在一个 k，使 i, k 在同一个群，j, k 也在同一个群。

问一共有多少个群，以及最大的群有多少奶牛。

88.2 算法讨论

首先把所有坐标 (x, y) 变换成 $(x+y, x-y)$，然后 $(x1, y1)$ 与 $(x2, y2)$ 的距离就是 $max(|x1-x2|, |y1-y2|)$。

这样只要把所有点按 x 坐标排序，加入一个队列里，保证队列头尾的 x 的差不超过 C，然后用平衡树维护所有点的 y 坐标，每次插入一个点时，如果它与上下的点 y 的差不超过 C，那就在这两个点之间连一条边，最后统计答案即可。

88.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

89 USACO OPEN 09 Tower

89.1 题目大意

有 n 堆干草，每一堆有一个宽度 w_i，高度都是 1。

现在要将这些干草按编号顺序堆成若干行，可以将任意多堆干草放在一行，但是每一行的总宽度不能比下面的一行大，一旦把一堆干草放在第二行，就不能把后面的干草放到地基上。

问最多能放多少行。

89.2 算法讨论

显然可以 DP。将这 n 堆干草反序后，f_i 表示 $1 \sim i$ 堆干草堆在一起后，最下面一行最大的宽度。

那么 $f_i = max\{j\}$ $(0 \le j < i \ \&\& \ s_i - s_j \ge f_j)$（其中 s_i 为 $1 \sim i$ 堆干草的宽度的和）

注意到我们要求的是满足 $s_i \ge f_j + s_j$ 的最大的 j，可以对 $(f_x + s_x)$ 建一棵平衡树来优化 DP，复杂度是 $O(n \log n)$ 的。

另一种方法是，考虑一个决策 a，如果存在 b 满足 $b < a$ 且 $f_b + s_b \ge f_a + s_a$，那么 b 一定不比 a 要优。

我们用一个单调栈来维护所有决策，每次询问在栈上二分，复杂度也是 $O(n \log n)$。

注意到询问的值 s_i 是单调递增的，可以在询问时将遍历到的点删掉，于是询问的复杂度为均摊 $O(1)$，总复杂度为 $O(n)$。

89.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

90 USACO OPEN 10 Triangle Counting

90.1 题目大意

给你平面上 n 个点（保证任意两点连线不经过原点），要求统计由其中三个点构成的三角形中，覆盖原点的三角形个数。

90.2 算法讨论

对于一个覆盖原点的三角形，原点和任意一个顶点的连线一定能将剩下两个顶点分到这条直线的两侧；对于其他三角形，三个顶点中满足上述性质的一定只有一个。

于是我们统计过原点和每个点的直线两侧的点数，就能知道答案了。

具体实现只要把所有点按极角排序，扫一遍即可。

90.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

91 USACO OPEN 13 Photo

91.1 题目大意

给你一个 n 长度的数轴和 m 个区间，每个区间里有且仅有一个点，问最多能有几个点。

91.2 算法讨论

$dp[i]$ 表示如果 i 这个位置上有一个点，$1..i$ 中最多的点数，$dp[n+1]-1$ 就是答案。

$dp[i] = max\{dp[j]\}+1$，其中 j 满足 $j+1..i-1$ 中没有完整的区间且 j 和 i 不在同一个区间里。

这样的 j 一定是在一个区间里的，所以可以用线段树来求区间最大值，复杂度 $O(n \log n + m)$。

注意到能转移到 i 的区间的左右端点关于 i 都是单调递增的，就可以用单调队列维护这个区间最大值，复杂度 $O(n+m)$。

91.3 时空复杂度

时间复杂度：$O(n+m)$

空间复杂度：$O(n+m)$