IOI2015 中国国家集训队第二次作业

湖南师大附中 骆轩源

Contents

1 Codeforces 251D **8**
1.1 Two Sets 8
1.2 Solution 8

2 Codeforces 258D **8**
2.1 Little Elephant and Broken Sorting 8
2.2 Solution 8

3 Codeforces 306D **9**
3.1 Polygon 9
3.2 Solution 9

4 Codeforces 253E **9**
4.1 Printer 9
4.2 Solution 10

5 Codeforces 260E **10**
5.1 Dividing Kingdom 10
5.2 Solution 11

6 Codeforces 339E **11**
6.1 Three Swaps 11
6.2 Solution 11

7 Codeforces 288E **12**
7.1 Polo the Penguin and Lucky Numbers 12
7.2 Solution 12

8 Codeforces 305D **12**
8.1 Olya and Graph 12
8.2 Solution 13

9 Codeforces 273D **13**
9.1 Dima and Figure 13
9.2 Solution 14

10 Codeforces 249E **14**
10.1 Endless Matrix 14
10.2 Solution 14

11 Codeforces 286D **15**
11.1 Tourists 15
11.2 Solution 15

12 Codeforces 323C **16**
12.1 Two permutations 16
12.2 Solution 16

13 Codeforces 335E **16**
13.1 Counting Skyscrapers 16
13.2 Solution 17

14 Codeforces 351D **18**
14.1 Jeff and Removing Periods 18
14.2 Solution 18

15 Codeforces 293E **19**
15.1 Close Vertices 19
15.2 Solution 19

16 Codeforces 306C **19**
16.1 White, Black and White Again 19
16.2 Solution 19

17 Codeforces 335D **20**
17.1 Rectangle And Square 20
17.2 Solution 20

18 Codeforces 316E **21**
18.1 Summer Homework 21
18.2 Solution 21

19 Codeforces 261E **21**
19.1 Maxim and Calculator 21
19.2 Solution 22

20 Codeforces 266D **22**
20.1 BerDonalds 22
20.2 Solution 22

21 Codeforces 243C **23**
21.1 Colorado Potato Beetle 23
21.2 Solution 23

22 Codeforces 249D **23**
22.1 Donkey and Stars 23
22.2 Solution 23

23 Codeforces 319D **24**
23.1 Have You Ever Heard About the Word 24
23.2 Solution 24

24 Codeforces 285E **25**
24.1 Positions in Permutations 25
24.2 Solution 25

25 Codeforces 309D **25**
25.1 Tennis Rackets 25
25.2 Solution 25

26 Codeforces 263E **26**
26.1 Rhombus 26
26.2 Solution 27

27 Codeforces 316D **27**
27.1 PE lesson 27
27.2 Solution 27

28 Codeforces 261D **28**
28.1 Maxim and Increasing Subsequence 28
28.2 Solution 28

29 Codeforces 241E **28**
29.1 Flights 28
29.2 Solution 29

30 Codeforces 332D **29**
30.1 Theft of Blueprints 29
30.2 Solution 29

31 Codeforces 295D **29**
31.1 Greg and Caves 29
31.2 Solution 30

32 Codeforces 314E **31**
32.1 Sereja and Squares 31
32.2 Solution 31

33 Codeforces 266E **31**
33.1 More Queries to Array 31
33.2 Solution 31

34 Codeforces 325C **32**
34.1 Monsters and Diamonds 32
34.2 Solution 32

35 Codeforces 293D **33**
35.1 Ksusha and Square 33
35.2 Solution 33

36 Codeforces 338E **33**
36.1 Optimize 33
36.2 Solution 34

37 Codeforces 273E **34**
37.1 Dima and Game 34
37.2 Solution 34

38 Codeforces 323B **34**
38.1 Tournament graph 34
38.2 Solution 35

39 Codeforces 333C **35**
39.1 Lucky Tickets 35
39.2 Solution 35

40 Codeforces 241F **35**
40.1 Race 35
40.2 Solution 36

41 Codeforces 338D **36**
41.1 GCD Table 36
41.2 Solution 36

42 Codeforces 240F **36**
42.1 Torcoder 36
42.2 Solution 37

43 Codeforces 311C **37**
43.1 Fetch the Treasure 37
43.2 Solution 37

44 Codeforces 269D **38**
44.1 Maximum Waterfall 38
44.2 Solution 38

45 Codeforces 152E **38**
45.1 Piglet's Birthday 38
45.2 Solution 38

46 Codeforces 264E **39**
46.1 Roadside Trees 39
46.2 Solution 39

47 USACO Open 13 **39**
47.1 Figure Eight 39
47.2 Solution 39

48 USACO DEC 12 **40**
48.1 First 40
48.2 Solution 40

49 USACO Jan 07 **40**
49.1 Cow School 40
49.2 Solution 40

50 USACO DEC 08 **41**
50.1 Fence 41
50.2 Solution 41

51 USACO DEC 07 **41**
51.1 Best Cow Line 41
51.2 Solution 42

52 USACO Mar 08 **42**
52.1 Land Acquisition 42
52.2 Solution 42

53 USACO Mar 13 **42**
53.1 Hill Walk 42
53.2 Solution 42

54 USACO DEC 05 **43**
54.1 Cow Patterns 43
54.2 Solution 43

55 USACO OPEN 10 **43**
55.1 Triangle Counting 43
55.2 Solution 43

56 USACO Mar 09 **44**
56.1 Cleaning Up 44
56.2 Solution 44

57 USACO Jan 12 **44**
57.1 Cow Run 44
57.2 Solution 44

58 USACO Open 08 **45**
58.1 Cow Neighborhoods 45
58.2 Solution 45

59 USACO Open 13 **45**

59.1 Photo 45

59.2 Solution 45

60 Google Codejam 2014 E **45**

60.1 Allergy Testing 45

60.2 Solution 46

1 Codeforces 251D

1.1 Two Sets

给定一个$n(n \leq 10^5)$个数字($\leq 10^{18}$)的可重复集合N，现在需要将N划分成两个集合A, B，另x_1表示A集合中数字的异或和，x_2表示B集合中数字的异或和。求出一种划分使得$x_1 + x_2$最大。若有多种划分方法，选择x_1最小的。

1.2 Solution

我们另Sum表示N中所有数字的异或和。用$\oplus$ 表示异或运算，那么有$x_2 = Sum \oplus x_1$。我们不妨另Sum和x_1是63位二进制数（不足位在高位补0）。

设$T[i]$表示十进制数T的二进制表示下2^i前系数（也就是二进制第i位是多少）。

如果$Sum[i]$为1，那么不管$x_1[i]$是0还是1，$x_2[i]$都与之相反，也就是$x_1[i]$ 加上$x_2[i]$恒为1。反之如果$Sum[i]$为0, 那么$x_1[i] = x_2[i]$，所以$x_1[i]$加上$x_2[i]$要么是0，要么是2。

如此我们便可以从二进制高位往低位贪心来决定x_1，设当前位为i，如果$Sum[i] = 1$，则跳过。否则先判断$x_1[i]$能否为1，如果能，则让$x_1[i]$为1，否则为0。因为在该位能产生2^{i+1}的贡献，这个贡献是低位无论如何都做不到的。判断$x_1[i]$ 是否可以为1的方法是加入一个异或方程的等式。看看是否与之前的等式矛盾。

这样做完之后得到的x_1和$x_2 = Sum \oplus x_1$就是满足$x_1 + x_2$最大的一种划分了，但是我们还需要让x_1最小。这时候我们需要从高位到低位再贪心一次。保留原有的方程组，设当前位为i，如果$Sum[i] = 0$, 则跳过，否则先加入让$x_1[i] = 0$的方程到原来的方程组，如果不矛盾，则加入，另$x_1[i] = 0$，否则加入让$x_1[i] = 1$的方程组。

最后我们可以得到一个包含n个未知量，不超过63 个方程的异或方程组。可以知道该方程组中至多有63个主元，所以我们完全可以让那些不是主元的未知量为0，这样我们就得到一个63个方程63个未知数的方程组。解完就好。中间的异或方程计算可以使用$bitset$。

设最高二进制位为β

时间复杂度$O(n\beta^2)$（常数可以看成$\frac{1}{32}$），空间复杂度为$O(\beta n)$

2 Codeforces 258D

2.1 Little Elephant and Broken Sorting

给定一个$n(n \leq 1000)$排列,然后按顺序进行$m(m \leq 1000)$次交换操作（交换两个位置上的数字），每个操作被执行的概率为$\frac{1}{2}$，现在问你最终排列的期望逆序对数是多少。

2.2 Solution

计算排列逆序对的方式有很多种，但是因为这里的变化非常大，我们选用的计算方式

应该涉及尽量少的量。

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[p_i > p_j] \tag{1}$$

令$f[i][j]$表示，一开始在位置i, j上的两个数字，在经过m次随机交换操作后，相对位置不变的概率，可以知道，相对位置改变的概率就是$1 - f[i][j]$。

假如已经计算出了$f[i][j]$，我们就枚举$i < j$,如果$p_i > p_i$就让答案增加$f[i][j]$,否则增加$1.0 - f[i][j]$。

现在问题就变成求解$f[i][j]$，我们不妨令$g[x][i][j]$表示现在的位置i, j在剩下的x 次操作后相对位置保持不变的的概率。那么就有$f[i][j] = g[m][i][j]$。至于如何求解$g[x][i][j]$,首先我们可以知道$g[0][i][j] = 1$，然后我们可以用$g[x-1][i][j]$ 来推出$g[x][i][j]$，我们设第$m-x+1$次操作的两个位置为a, b,那么只有形如$g[x][a][k], g[x][k][a], g[x][b][k], g[x][k][b]$ 的答案才会发生改变。所以我们要更新的状态只有$O(n)$ 个。

所以总的时间复杂度为：$O(nm + n^2)$，空间复杂度为：$O(n^2)$

3 Codeforces 306D

3.1 Polygon

找一个凸$n(n \leq 100)$边形，满足所有内角相同，边长互不相同，精度为0.001。无解输出“No Solution”。

3.2 Solution

三角形和四边形是无解的。否则，我们先构造n个绕着原点的单位向量，相邻两个向量的夹角是$\frac{2\pi}{n}$，顺时针给向量标号$V[1...n]$，那么我们可以这么构造多边形，另$P[i]$表示多边形上第i个点（按某种旋转顺序）。则$P[i] = P[i-1] + V[i], P[0] = (0, 0)$。

所以我们可以按照这种方法生成每个角度都相同的多边形，所以现在的问题就是伸缩那n个单位向量，使得它的边长互不相同，而且能够构成封闭多边形。那么如何的这样n个向量才可以构成多边形的，一个必要条件是能够走回来，也就是所有向量相加等于0，可以理解为物理力学题目中正交分解为0。这样我们得到的多边形可能会自交。

可以这么构造，对于$i \leq n-2$我们将原来的$V[i]$ 的长度伸长至$300 + 0.01 * i$，最后留下两个向量$V[n-1], V[n]$，解方程就好（正交分解）。这样做对于$5 \leq n \leq 100$都是可以构造的。

时间复杂度$O(n)$，空间复杂度$O(n)$。

4 Codeforces 253E

4.1 Printer

有一个机器每秒钟完成单位1的任务量，有$n(n \leq 10^5)$个任务，其中第i个任务在时

刻s_i被加进来，它的任务量是t_i，优先级是p_i，每个时刻机器会选择优先级最大的那个任务完成其一个单位的任务量。

现在有一个任务的优先级是未知的，但是你知道它的完成时间T，现在你需要求出该任务的优先级，以及在该优先级下，所有任务的完成时刻，多解输出其一即可。

4.2 Solution

首先我们知道优先级值仅决定相对顺序，所以我们可以先将已知的所有优先级排序，然后选择相邻两个优先级中间的任意一个数值作为候选优先级，这样我们得到了$O(n)$个候选优先级，顺便将它们排好序。

一个性质就是随着未知任务的优先级变高，它的完成时间一定不会增加，这样我们就可以使用二分的方法在候选优先级里面找出最终的答案。

所以我们现在的问题就是高效地对于给定所有任务信息，计算每个任务的完成时刻。可以知道对于计算有用的状态就是当前时刻优先级最高的是哪个任务。这个仅在特定的某些时刻点会发生变化，比如新加入一个任务，或者某个任务完成了。我们可以用一个优先队列S(比如堆）来存储这些时刻点。并用另外一个优先队列F来存储当前已加入的所有任务（按照优先级为关键字）

我们的做法就是先将所有的任务进入时刻加入优先队列S。每次取队首时刻X和队首的下一个时刻Y，加入那些进入时刻为X的任务到F。然后取出F 中优先级最高的任务,如果在Y时刻之前任务可以完成,这个任务就可以顺利地完成，将它从F中删去，记录它的完成时刻，并将完成时刻加入S。如果Y时刻之前完不成，就更新它的任务量。

所有操作都做完后，弹出S队首。只有一个任务完成才会加入新时刻，所以一共的时刻点数也是$O(n)$级别的，加上二分的代价是$O(\lg n)$ （因为是在候选元素中二分），实现优先队列需要$O(n \lg n)$的时间。

总的时间复杂度为$O(n \lg^2 n)$，空间复杂度为$O(n)$。

5 Codeforces 260E

5.1 Dividing Kingdom

一个名叫Flatland的国家可以看成一个无穷的二维平面。Flatland有$n(n \leq 10^5)$个城市，每个城市可以看做平面上的一个点。国王Circle IV统治着Flatland。Circle IV有9 个孩子，他想让每个孩子分得Flatland的一部分。他想画4 条不重合的直线，其中2条与x 轴平行，另2条与y 轴平行。于是，Flatland 被分成了9部分，每个孩子将得到这其中的一块。Circle IV想了一会，决定第i 个孩子分得的一部分Flatland中包含的城市的个数必须等于a_i。帮助Circle IV找到这样的4条直线，使得Flatland被这些直线分成9部分后，第i个孩子可以得到城市个数为ai的一部分。

5.2 Solution

我们先将所有横坐标从小到大排序之后相邻两坐标x_i, x_{i+1}之间的所有与y轴平行的直线起到的效果都相同，我们不妨就取$x = \frac{x_i+x_{i+1}}{2}$,所以这样我们就只需要考虑$O(n)$条竖线,同理我们也可以将横线缩小到$O(n)$ 级别。

然后我们9!枚举每个孩子得到“井字形”的哪一块，也就确定井字形的每一块有多少点。现在只需要判定其是否可能，我们先看左边这条竖线，它需要满足该线左边的点个数为枚举的井字形最左边三个格子点数之和，这样一条线可以二分得到，如果这样的直线存在（至多1 条）则令其为第一条竖线。然后类似的，我们可以得到第二条竖线，和第一，二条横线。

如果4条线都存在，那么只需判定这四条线是否符合枚举的“井字形”（就是每个格子的点数是否对应），也即需要一个数据结构支持询问有多少个点(xx, yy) 满足$xx < x$且$yy < y$。这个可以用可持久化线段树实现。

设m 为区域数(其实就是9，但把它看成变量)

时间复杂度：$O(m! \lg n + n \lg n)$,空间复杂度$O(n \lg n)$

6 Codeforces 339E

6.1 Three Swaps

有一个序列一开始是$1, 2, 3...n$，然后对其执行了不超过3次翻转操作(选择一个区间$[l, r]$ 翻转)得到新的序列$< a_1, a_2, a_3...a_n >$。现在出a 序列（保证可以3次操作得到）。要你找到一种操作方案（不超过3次翻转）。$n \leq 1000$

6.2 Solution

直接暴力的话，我们需要枚举两次操作（后两次，可以看成从a变成初始序列）的翻转端点$O(n^4)$，然后再使用$O(n)$ 判定剩下序列能否用不超过1次操作得到,这样的最坏复杂度是$O(n^5)$。

但是深入分析，我们会有一种感觉，这样一个序列仅仅只被翻过3次。那么这个最后得到的序列中间肯定有很多部分依然是连在一起的。也就是整体未被大部分破坏，换句话说，我们可以将翻转n个点仅仅看成20多个部分（一个部分就是一个区间）的移动。

这样有什么意义呢？这样可以减少我们的枚举范围，我们只需要枚举割开不同部分的断点就好。那么什么样的点是断点呢？两个位置相邻的点的值差距大于1，则这两个位置以及其位置上的值都是断点。

那么我们就可以设计如下算法，

首先找出$< a >$中的断点，然后枚举最后一次操作的左右端点（从断点中找），并对$< a >$执行该操作得到$< b >$，然后再从$< b >$中找断点，如果断点个数大于20 个，那么前一次枚举肯定是不靠谱的，否则再对$< b >$ 执行该次枚举的操作得到$< c >$，之后再

用$O(n)$判定$< c >$是否已经是初始序列，或者要通过一次操作得到初始序列（仅存在两个断点）。

设最多的断点个数为m，可以知道$m \leq 20$。所以我们得到：

时间复杂度:$O(m^4 n)$，空间复杂度$O(n)$。

7 Codeforces 288E

7.1 Polo the Penguin and Lucky Numbers

定义幸运数字是由4,7构成的数字，给出两个不超过100000位的位数相同的幸运数字$L, R(L \leq R)$。然后将所有L, R之间的幸运数字从小到大排序得到$a_1, a_2 ... a_n$,你需要给出

$$Ans[L, R] = \sum_{i=1}^{n-1} a_i a_{i+1} \mod (10^9 + 7) \tag{2}$$

7.2 Solution

假设L, R都是Len位的幸运数字。那么我们设S是全部由4构成的Len位幸运数字，那么可以知道

$$Ans[L, R] = Ans[S, R] - Ans[S, L] \tag{3}$$

所以我们的问题变成求解$Ans[S, T]$，T为一个Len 位幸运数字。我很幸运地又一次在数位统计中看到幸运数字了。如果我让所有的a_i同时增加X，上求和式变成

$$X^2 \sum_{i=1}^{n-1} 1 + X \sum_{i=1}^{n-1} (a_i + a_{i+1}) + \sum_{i=1}^{n-1} a_i a_{i+1} \tag{4}$$

令$g_1[i]$表示长度为i的所有幸运数字的$\sum_{i=1}^{n-1} 1$,

$g_2[i]$表示长度为i的所有幸运数字的$\sum_{i=1}^{n-1} (a_i + a_{i+1})$,

$g_3[i]$表示长度为i的所有幸运数字的$\sum_{i=1}^{n-1} a_i a_{i+1}$。

有了这些预处理，就跟以往的数位统计做法本质没什么大区别了。

时间复杂度$O(n)$，空间复杂度$O(n)$

8 Codeforces 305D

8.1 Olya and Graph

给定n个点，m条边的有向图，对于每条边(u, v) 满足$(u < v)$，现在你可以加入若干(可以为0)条有向边，使得最终的有向图满足：

1. 所有有向边(u, v)满足$(u < v)$。
2. 从点i出发，可以到达点$i + 1, i + 2, ..., n$。
3. 对于一对点$i, j(i < j)$，若$j - i \leq k$，那么从i到j的最短距离等于$j - i$ 条边。

4. 对于一对点$i,j(i<j)$，若$j-i>k$，那么从i到j的最短距离等于$j-i$或$j-i-k$条边。

询问有多少种加边方式（不产生重边），满足上述条件，输出答案对10^9+7取模结果。

8.2 Solution

对于条件$1,2$，我们可以得到一个必要条件就是所有形如$(i,i+1)$都应该存在于图中。对于条件3，我们可以得到所有$v-u\leq k$ 的边都不能存在于图中。对于条件4，我们可以得到所有$v-u>k+1$的边都不能存在于图中。

所以我们得到，图中只能有跨度为1或者$k+1$ 的边可以存在，先对输入中的边做如此判断，如果存在不合法的边，就输出0。否则的话，我们假设所有的跨度为1 的边都已经加好（因为这是所有方案中都必要的边），然后我们就决定在哪些位置加入跨度为$k+1$的边，可以知道一条边可以由它的终点确定。对于任意一种方案，将所有跨度为$k+1$的边的终点从小到大排序得到$<w_1,w_2...w_t>$，该方案是合法的，当且仅当$w_t-w_1\leq k$。

对于已存在的跨度为$k+1$的边(u,v)，我们在v处打上一个标记。首先我们来看看哪些位置可以成为线段终点，位置i 可以成为终点，当且仅当在$[1..i-k-1]$ 之间不存在标记，在$[i+k+1,n]$之间不存在标记（边界注意一下），然后我们就可以计算出那些位置是有可能成为终点。

这时候再次扫描所有位置，如果某个位置存在有标记但是又不可能成为终点，那么也是无解情况输出0。现在我们就可以来统计答案了，思想是枚举所有跨度为$k+1$的边中位置最小的终点i。如果在$[1..i-1]$中不存在标记（因为i是最小的），而且i可以成为一个终点。那么就可以进行统计，另$j=i+k$，所以在$[i+1,j]$之间的所有位置都是不与i矛盾的，只要统计在$[i+1,j]$之间那些既可以成为终点，也可以不成为终点的点的个数t，让答案加上2^t就好。

最后对于一条跨度为$k+1$的边都不加的情况特殊处理一下。

时间复杂度:$O(n)$,空间复杂度$O(n)$。

9 Codeforces 273D

9.1 Dima and Figure

给定一个$n\times m$的网格，每个格子可以涂上黑也可以涂成白的，现在问题你有多少种染色方案，满足：

1. 所有的黑格子构成四连通块
2. 至少有1个黑格子
3. 任意两个黑格子之间的距离（只准走黑格子）为其曼哈顿距离

输出答案对10^9+7取模的余数。

9.2 Solution

我们的思路应该是去分析怎样的连通块可以满足条件3，一个想法就是连通块要是“凸”的。也即不存在一条竖直的或者水平的扫描线穿过该连通块时出现中间空一段的情况。经过验证发现该条件甚至是一个充分条件。那么我们现在的问题就变成统计“凸”着的连通块个数。

我们模拟一根从上往下的扫描线，可以知道连通块在该扫描线上一定是一个区间。那么考虑使用动态规划来解决，令$f0[k][i][j]$表示考虑到了第k行，且连通块在该行覆盖了$[i,j]$，并且对于第$1..k-1$行的所有区间$[l,r]$，都有$i \le l \le r \le j$。$f1[k][i][j]$表示第k行覆盖了$[i,j]$，且前面行的区间$[l,r]$都满足$r \le j$，但是存在一个区间的$l < i$(我们可以称其为左出头），$f2[k][i][j]$表示第k行覆盖了$[i,j]$，且前面的区间$[l,r]$都满足$l \ge i$，但是存在一个区间的$r > j$（右出头）,$f3[k][i][j]$表示第k行覆盖了$[i,j]$,且前面的区间$[l,r]$至少有一个满足$l < i$且$j < r$ （双出头）。

转移的话处理几个前缀和就好，我们令

$$Ans[k] = \sum_{1 \le i \le j \le m} f0[k][i][j] + f1[k][i][j] + f2[k][i][j] + f3[k][i][j] \tag{5}$$

最后的答案就是$\sum_{1 \le k \le n} Ans[k] * (n-k+1)$。

时间复杂度$O(nm^2)$，空间复杂度$O(m^2)$。

10 Codeforces 249E

10.1 Endless Matrix

有一个矩阵，里面有连续的数字，从1开始。若$a_{i,j} < a_{t,k}(i,j,t,k > 1)$当且仅当

1. $\max\{i,j\} < \max\{t,k\}$
2. $\max\{i,j\} = \max\{t,k\}$且$j < k$
3. $\max\{i,j\} = \max\{t,k\}, j = k$且$i > t$

给定$T(T \le 10^4)$组x_1, x_2, y_1, y_2，求解$\sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} a_{i,j}$

10.2 Solution

不难分析出前6×6的表格应该长成这样：

1	2	5	10	17	26
4	3	6	11	18	27
9	8	7	12	19	28
16	15	14	13	20	29
25	24	23	22	21	30
36	35	34	33	32	31

自然数是按照如下方式被放入无穷表格的，首先$a_{1,1}=1$，然后从第1行第2列开始，往下2 个格子，再往左一个格子，从第1行第3 列开始，往下3个格子，往左2个格子.....

所以我们可以将第一行每一列看成是一个**连续段**的起点。

问题完全可以等价于求解$\sum_{i=1}^{x}\sum_{j=1}^{y}a_{i,j}$，然后再容斥一下即可得到答案。

然后对于求解上述求和，我们需要分几种情况，

1. $x=y$

   此时答案就是$1+2+....x^2=(2x^2+1)/2$

2. $x<y$

   此时答案分为两部分，一部分就是前$x\times x$的正方形里面的和，算法与上述相同，剩下的就是$y-x$ 个被截断了的**连续段**（一个直角）的前x项的求和。也很容易得到。

3. $x>y$

   同样答案也可以分为两部分计算，类似上述做法。

然后计算的时候可以使用高精度，设数字最大位数为β。时间复杂度$O(T\beta^2)$，空间复杂度为$O(\beta)$。

11 Codeforces 286D

11.1 Tourists

有$n(\leq 10^5)$个移动的点从原点出发以每秒一单位的速度向数轴正方向运动，第i个点在q_i时刻出发。还有m 个障碍线段。其中第i条线段(l_i,r_i)在时刻t_i出现（出现之后不消失），对于每个点，询问其运动过程中与障碍线段有交的时间之和。$(0\leq l_i,r_i,q_i\leq 10^9)$

11.2 Solution

先将坐标离散化，将所有出现过的坐标排序，去掉相同的，然后相邻的两个坐标构成一个小线段（看成一个单元），然后对于输入中的m个障碍线段，将其覆盖的所有单元用t_i更新最小值（可以用线段树），这样就计算出每个小线段成为障碍线段的最早时刻（如果这个线段不属于某个障碍线段，可以认为这个时刻为无穷大）

再将q_i从小到大排序。然后对于每个单元(L,R,t)。其对那些$q_i+L>t$的动点，该单元的贡献为$R-L$。可以二分出最小的x满足$q_x+L>t$，然后给$q_{x..n}$全部加上$R-L$，可以使用树状数组实现。对于那些$(q_i+L\leq t)\bigwedge(q_i+R\geq t)$的动点，该单元的贡献为$R-t+q_i$，也可以二分出$i$的上下界，然后用两个树状数组统计答案。最后将答案输出就好。

时间复杂度$O(m\lg m+n\lg n)$,空间复杂度$O(m+n)$。

12 Codeforces 323C

12.1 Two permutations

你有两个各包含$n(n \leq 10^6)$个元素的排列p和q，和$m(m \leq 10^5)$个由l_1, r_1, l_2, r_2组成的询问。每次询问在p 中位置在$[l_1, r_1]$, 在q中位置在$[l_2, r_2]$ 中的数的数量。

一个n元素的排列是指n个不同的数，每个数在$1 \to n$之间,数字v的位置在排列$g_1, g_2, ,, g_n$中是使得$g_i = v$ 的i。

强制在线。

12.2 Solution

我们令$k[i]$表示元素$p[i]$在q序列中的位置，然后用可持久化线段树预处理$k[1..i]$的位置信息。

对于每个询问,询问$k[1], k[2]...k[r_1]$中在$[l_2, r_2]$中的元素个数减去$k[1], k[2]...k[l_1 - 1]$中在$[l_2, r_2]$中的元素个数即可。

时间复杂度:$O(n \lg n + m \lg n)$，空间复杂度：$O(n \lg n)$。

13 Codeforces 335E

13.1 Counting Skyscrapers

大街上建好了一排摩天大楼。摩天大楼的数量是在2 到314!（314的阶乘，一个非常大的数）中均匀随机选择的。每座摩天大楼的高度（即楼层数）是被独立地随机选择的：对于每个正整数i，楼层数为i 的概率为$2 - i$。如果一座摩天大楼有i 层，那么它的楼层被编号为0 到$i - 1$。

为了加快中转运输的效率，摩天大楼间修建了一些滑索。一座摩天大楼的第i 层和另一座摩天大楼的第i 层之间有滑索当且仅当两楼之间没有摩天大楼有第i 层。Alice和Bob想数一数有多少座摩天大楼。

Alice是个严谨认真的人，她想知道摩天大楼数量的准确值。于是她把计数器初始化为1，从最左边的摩天大楼开始往右走，每次走到一个摩天大楼就把计数器加上1，直到她到达最右边的摩天大楼。

Bob很没耐心，他想尽快完成任务。于是他把计数器初始化为1，从最左边的摩天大楼开始利用滑索从一座摩天大楼滑到另一座。因为恐高，每次他会忽略那些高度（即所在楼层编号）大于h 的滑索，然后在剩下的往右滑的滑索中选一个最高的。由于Bob使用滑索时滑得太快以至于他无法数清经过了多少座摩天大楼，所以他直接将计数器加上$2i$，其中i 是他当前所在的楼层编号。他会一直持续这个过程直到他到达最右边的摩天大楼。

当Alice和Bob到达最右边的摩天大楼时，他们会比较计数器的值。现在给出Alice的计数器的值或者Bob的计数器的值，请你求出另一人的计数器的值的期望。

13.2 Solution

1. **已知Alice求Bob**

   Alice的计数就是楼房的个数n。也就是我们需要计算所有滑索的期望贡献，设$[i,j,d]$来表示一个滑索，它连接了$i,j(i<j)$号楼房，且高度标号为d，那么它对答案的贡献就是2^d 再乘以它能成为滑索的概率。一个滑索是合法的，当且仅当中间的楼房都严格小于d，两边不同时存在高度大于d的楼房（我们把高度大于h的楼房看成高度为h)。

   先枚举d，但仅枚举到$h-1$,令g表示一栋楼房的高度标号小于等于d的概率，然后枚举i,j。令s_1表示i 之前的所有楼房小于等于d 的概率，s_2表示i之前存在楼房大于d的概率,t_1 表示j之后所有楼房小于等于d的概率，t_2 表示j 之后存在楼房大于d的概率。

   设$p[d]$表示一栋楼房高度标号为d的概率(通过简单的计算可以发现，严格大于d的概率也是$p[d]$),$x[d]$表示一栋楼房高度严格小于d 的概率。

   则有：$s_1=d^{i-1}, s_2=1-s_1, t_1=d^{n-j}, t_2=1-t_1$。

   然后再枚举i,j的高度情况：

   (a) i**为**d，j**为**d

   概率增加$p[d]^2x[d]^{j-i-1}(1-s_2t_2)$

   (b) i**严格大于**d，j**为**d

   概率增加$p[d]^2x[d]^{j-i-1}t_1$

   (c) i**为**d，j**严格大于**d

   概率增加$p[d]^2x[d]^{j-i-1}s_1$

   最后让总概率乘以2^d即可，这样做的复杂度有$O(hn^2)$,不过我们可以发现当从小到大移动j的时候，我们可以用类似前缀和的思想，设$s_1[i],s_2[i]$表示左端点为i是计算出来的s_1,s_2，那么我们维护$sum_1[j]$,$sum_2[j]$，分别表示

   $$sum_1[j]=\sum_{i<j}s_1[i]x[d]^{j-i-1}$$

   $$sum_2[j]=\sum_{i<j}s_2[i]x[d]^{j-i-1}$$

   很容易在$O(1)$的时间从$sum_1[j],sum_2[j]$转移到$sum_1[j+1],sum_2[j+1]$。所以可以在$O(n)$ 的时间内完成刚才$O(n^2)$ 做的事情，算法最后对$d=h$的情况另加讨论，讨论方法类似。

   所以总复杂度为$O(nh)$。

2. **已知Bob求Alice**

   对于一条高度为h的绳索，在该绳索的中间期望产生多少个点（算上一个端点）呢？我们设一栋房子高度严格小于h的概率为x，则期望值为：

   $$\lim_{n\to\infty}\sum_{i=0}^{n}x^i=\frac{1}{1-x}$$

令$ex[h]$来表示这个结果

我们先让$n-1$，并认为高度大于h就是等于h

然后我们设$f_1[i][t]$表示Bob的计数还剩下i时，最右边的楼房高度为t，前面不存在严格大于t的楼房的期望值（注意这个期望值并不是所求，因为下面方法计算的全局条件不是已知Bob，所以最后答案还要除以一个概率），$f_2[i][t]$ 表示Bob 的计数还剩下i 时，最右边的楼房高度为t,前面存在至少一个严格大于t的楼房的期望。

令$p_1[i][t]$表示与$f_1[i][t]$相同条件的概率，$p_2[i][t]$表示与$f_2[i][t]$相同条件的概率。先枚举i,再枚举t,最后枚举最后一条绳索的高度d,令$j=i-2^d$

则有

$$f_1[i][t]=\sum_{d=0}^{t} p[t]*(f_1[j][d]+ex[d]*p_1[j][d]) \tag{6}$$

$$\begin{aligned} f_2[i][t]&=p[t]*(f_2[j][t]+ex[d]*p_2[j][t])\\ &+\sum_{u=t+1}^{h} p[t]*(f_1[j][u]+f_2[j][u]+ex[d]*(p_1[j][u]+p_2[j][u])) \end{aligned} \tag{7}$$

注意$f_2[i][t]$仅在$t=d$时才会有值。

然后$p_1[i][t]$ 和$p_2[i][t]$可以类似转移，最后答案即为

$$ans=\frac{\sum_{t=0}^{h}(f_1[n][t]+f_2[n][t])}{\sum_{t=0}^{h}(p_1[n][t]+p_2[n][t])}$$

时间复杂度为$O(nh^2)$

14 Codeforces 351D

14.1 Jeff and Removing Periods

给定一个序列$a[1..n]$,每次询问其一个子段$a[l..r]$最少经过多少次操作可以被消除,每次操作我们可以做2个动作，第一个动作是选择一种颜色，满足这种颜色在区间内的下标构成等差数列，然后把这种颜色删掉。第二个动作是重新排列该区间。

$n\le 10^5$，询问数$m\le 10^5$。

14.2 Solution

对于一个序列，如果其存在一个颜色下标构成等差数列，那么最小操作次数就是颜色数，否则就是颜色数+1。

所以问题就变成询问区间颜色数和区间是否存在至少一个颜色下标构成等差序列。

我们可以先预处理$f[i],g[i]$分别表示，以i 为右端点，往左最远能拓展到的点，和以i为左端点，往右最远能拓展到的点，能拓展的意思是在$[f[i],i],[i,g[i]]$中所有颜色为$a[i]$的点构成等差数列。

用$pre[i]$表示i之前最近的颜色为$a[i]$的位置,$nex[i]$表示i之后最近的颜色为$a[i]$ 的位置。

然后就可以用莫队算法解决，唯一的小问题就是如何时时维护区间中是否存在构成等差数列的颜色，我们不妨统计这样的颜色数量，并用一个标记数组记录每种颜色是否是等差数列颜色，因为移动端点的时候，只有新旧端点的颜色信息会改变，比如从区间$[L,R]$变成$[L-1,R]$，只需看看$g[L]$ 和$g[L-1]$是否大于等于R, 然后更新相关信息即可。

时间复杂度$O((m+n)\sqrt{n})$，空间复杂度$O(n)$。$n,m,a[i]\leq 10^5$

15 Codeforces 293E

15.1 Close Vertices

你得到了一棵包含$n(n\leq 10^5)$个点的树，树上的每条边有一个非负边权，树上两点间路径的长度是该路径包含的边数，树上两点间路径的权重是指该路径包含的边的边权之和。

我们说两点是“相邻”的，当且仅当，存在一条连接该两点的路径，满足该路径的长度小于等于L,且权重小于等于W。统计有多少个点对(u,v)，满足$u<v$，且u,v 是相邻的。

15.2 Solution

可以使用点分治算法解决。当分治到每一棵子树的时候，计算所有点到该子树$root$的权值和$w[x]$和路径长度$l[x]$,按照$w[x]$从小到大的顺序统计每个点的贡献，这里可以使用树状数组解决。

时间复杂度$O(n\lg^2 n)$，空间复杂度$O(n)$。

16 Codeforces 306C

16.1 White, Black and White Again

有$n(\leq 4000)$天，每天可以发生一些好事或者发生一些坏事（不能同时发生好事和坏事）。而且这n天满足先发生若干几天好事，再发生若干天坏事，再发生若干天好事（若干>0)。一共有$w(\leq 4000)$件两两不同的好事，和$b(\leq 4000)$件两两不同的坏事在这n 天发生，每天发生事情的顺序也可以不同。统计时间发生的方法数 $\bmod M(M=10^9+9)$。

16.2 Solution

枚举多少天发生坏事（设为Y），那么发生坏事的方法数即为$b!$乘以$x_1+x_2...x_Y=b$的正整数解的个数，也即$\binom{b-1}{Y-1}b!$。令$X=n-Y-1$，那么发生好事的方法数即为$w!$乘以$\binom{w-1}{X-1}$ 再乘以$X-1$。所以综上，对于枚举的$Y(1\leq Y\leq n-2)$, 其贡献为

$$cur=\binom{b-1}{Y-1}b!\binom{w-1}{X-1}(X-1)w!$$

因为模的数是质数，所以可以先预处理阶乘再求逆元来计算组合数。将所有cur求和即为答案。

时间复杂度$O(n \lg M)$,空间复杂度$O(1)$。

17 Codeforces 335D

17.1 Rectangle And Square

给定$n(n \leq 10^5)$个互不相交的矩形（边平行坐标轴），求一个由矩形构成的集合A，使得A中的矩形构成一个正方形。满足坐标范围为$[0, 3000]$的整数。

17.2 Solution

因为矩形互不相交，而且坐标的范围不大，所以我们可以将输入信息转化成一个表格，用$a[i][j]$表示左上角为$(i-1, j-1)$右下角为(i, j)的小正方形被编号为几的矩形覆盖，如果没被覆盖$a[i][j] = 0$。

得到这个表格的方法就是，一开始让所有$a[i][j] = 0$，每次输入一个矩形(假设编号为k)$(x_1, y_1) \to (x_2, y_2)$，就将所有满足$x_1 + 1 \leq i \leq x_2$，$y_1 + 1 \leq j \leq y_2$的$a[i][j] = k$。

那么一个边长为k的正方形，就是$a[i][j]$中的一个$k \times k$的子矩阵。那么怎么判断一个从(x, y) 开始到$(x + k, y + k)$这样一个正方形子表格是可以用输入集合拼出来的呢？

首先要满足$a[x..x + k][y..y + k]$中间没有0，且所有出现在正方形中的编号代表的矩形的横坐标最小不能小于x，最大不能大于$x + k$, 纵坐标最小不能小于y，最大不能大于$y + k$。

我们用$L[i][j]$，表示最大的k满足在矩阵$a[i..i + k - 1][j..j + k - 1]$中，编号代表的矩形横坐标最小不小于$i$, 纵坐标最小不小于$j$，且矩阵中不出现0。

关于$L[i][j]$可以二分+ST得到，也可以直接递推得到，总的思想就是，先分别求出只满足横坐标限制，纵坐标限制，0 限制的k的上界，然后再从这三个值中取最小值得到$L[i][j]$。复杂度就少了个lg。

同样我们求出$R[i][j]$，表示最大的k满足在矩阵$a[i - k + 1..i][j - k + 1..j]$中，编号代表的矩形横坐标最大不大于$i$,纵坐标最大不大于$j$，且矩阵中不出现0。求解方法类似$L[i][j]$。

可以知道能够构成正方形的左上角(x_1, y_1)和右下角(x_2, y_2)要满足$x_2 - x_1 = y_2 - y1$也就在同在一条斜率为1的线上，我们找出所有的斜线（一共2*3000-1)条，每条的长度不超过3000。然后现在考虑如何在一条斜线上统计答案。

我们设斜线上横纵坐标最小的点为(x, y)（x, y中一定有一个为1），然后从小到大枚举i,现在我们考虑以$(x+i, y+i)$为右下角是否有可能找到正方形，也就是是否存在$j <= i$满足$\min(L[x + j][y + j], R[x + i][y + i]) \geq i - j + 1$，换句话说在$j \in [i - R[x + i][y + i] + 1, i]$的范围内是否存在$L[x + j][y + j] + j \geq i + 1$,也就是询问一个区间的最大值，可以用$ST$预处理得到。

如果在某条斜线上找到了正方形，就可以轻松地统计答案，否则即可输出无解。设坐标最大值为m。

时间复杂度$O(m^2 \lg m)$，空间复杂度$O(m^2)$。

18 Codeforces 316E

18.1 Summer Homework

给定一个序列$a[1...n](n \leq 2 * 10^5)$,以及$m(m \leq 2 * 10^5)$个以下三种操作:

1. 输入x和v，将$a[x]$改成v。
2. 输入l和r，计算
$$Ans = \sum_{x=0}^{r-l} f_x \times a[l+x] \tag{8}$$
3. 输入l, r和d，将$a[l...r]$同时加上d。

18.2 Solution

可以将问题放大一点点，我们考虑矩阵
$$F = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \tag{9}$$
则f_k的值就是矩阵F^k的第一行第一列的数字。

所以原问题可以看成求解矩阵
$$Ans = \sum_{x=0}^{r-l} a[l+x]F^x \tag{10}$$
因为F是可逆的，所以不难求出其逆元
$$F^{-1} = \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \tag{11}$$
所以得到
$$Ans = F^{-l} \sum_{x=l}^{r} a[x]F^x \tag{12}$$
所以我们只要维护$a[i]F^i$的信息，然后预处理所有F^{-i}，这样就可以用线段树来完成题目要求的所有操作。

时间复杂度$O(m \lg n)$，空间复杂度$O(n)$。

19 Codeforces 261E

19.1 Maxim and Calculator

Maxim 得到了一个计算器，这个计算器有两个整数单元，一开始，第一个单元包含数字1，第二个单元包含数字0。这个计算器支持一以下两种操作：

1.假设第一个单元的数字为a，第二个单元的数字为b，那么将第二个单元的数字改成$b+1$。

2.假设第一个单元的数字为a，第二个单元的数字为b，那么将第一个单元的数字改成$a*b$。

现在Maxim 想知道，有多少个正整数$x(l <= x <= r)$满足，存在一种方式从计算器初始状态开始，操作不超过$p(\leq 100)$ 步之后使得第一个单元中的数字为x。$l \leq r \leq 10^9$

19.2 Solution

首先我们知道因为$p \leq 100$，所以满足要求的x 的最大质因子不超过100，那么这样的数字在10^9之内只有290万个。先把这些数字弄出来，然后我们知道如果x满足要求当且仅当其存在一种因子分解$x = a_1a_2...a_k$满足$k + \max\{a_i\} \leq p$。

那么我们枚举最大因子Max，然后令$f(i)$表示在最大因子不超过Max的限制下，i的因子分解最少的因子个数。从1 到p枚举Max，假设$f(i)$表示最大因子不超过Max的限制下，i的因子分解最少的因子个数，那么令$g(i)$表示最大因子不超过$Max+1$的限制下，i分解的最少因子个数。

则如果$(Max+1)$是i的因子，那么$g(i) = \min\{f(i), f(\frac{i}{Max+1}) + 1\}$，否则$g(i) = f(i)$，所以我们可以很轻松的求出每一层的$f(i)$, 然后再看看$Max + f(i)$是否小于等于$p$，就可以确定$i$是不是合法的。

具体实现上，就不能使用$f(i)$来表示了,不过我们可以先将所有的质因子小于等于p的$[1, R]$之内的数排个序，设第i 个为c_i，就用$f(i)$来表示原来的$f(c_i)$, 再使用一个单调移动的指针就可以轻松找到满足$c_j = c_i/Max$的j。

设所有质因子小于等于p的$[1, R]$之内的树有M 个。

总的时间复杂度为$O(pM)$，空间复杂度为$O(M)$。

20 Codeforces 266D

20.1 BerDonalds

有一个城市的交通网络由$n(n \leq 200)$个节点和m 条双向道路组成，每个节点有一个咖啡厅。现在要选择某个位置(可以是节点，也可以是某条道路上的任意一点)建一个餐馆，使得餐馆到最远的咖啡厅的距离最近，输出这个距离。简洁版题意：给定一个无向带权联通图，求图的直径。无重边，无自环。

20.2 Solution

先可以用任何最短路算法求出两两点之间的最短路$f[x][y]$，然后枚举每条边(u, v, w)，用$d_1[i]$ 表示$f[u][i]$，$d_2[i]$表示$f[v][i]$，我们假设从u开始不断向v移动答案点，设移动了x距

离。那么直径长度就应该是

$$\max_{i=1}^{n}\{\min\{d_1[i]+x, d_2[i]+w-x\}\}$$

我们用集合A_x来记录对应x时，$d_1[i]+x \le d_2[i]+w-x$的点i,用B_x来记录A_x的补集。

假设我们已知答案x，那么直径长度$len(x)=\min\{\max\{A_x\}+x, \max\{B_x\}+w-x\}$

我们只需要对于每个点i求出其最早从A集合中去掉的时刻，也就是$d_1[i]+x > d_2[i]+w-x$ 的最小时刻X_i。然后我们就从小到大扫描这些时刻点，对于相邻的两个时刻点(lx,rx)中间的时刻，其A,B集合中的元素都是一样的，不妨设A,B集合中的最大值分别为c_1,c_2,方程$c_1+x=c_2+w-x$的根x 就是极点，当然还需要判断x 是否在$[lx,rx]$中，如果不在就在lx 和rx中取最大值。

A集合不断删除元素，B集合不断加入元素，所以B 集合的最大值可以$O(1)$维护，而A集合可以用堆在$O(\lg n)$时间维护，或者提前对每个点到所有点的最短路从大到小排序，就可以$O(1)$维护A的最大值。

时间复杂度$O(mn\lg n)$,空间复杂度$O(n^2)$。

21 Codeforces 243C

21.1 Colorado Potato Beetle

你位于一个$(10^{10}+1)*(10^{10}+1)$的棋盘的正中间格子上。然后你从初始位置出发，给定$n(n \le 1000)$次移动操作，每次操作可以是向上，向下，向左，向右移动给定数量的格子（不会碰到边界），路径上经过的点都被打上农药，现在一堆的虫子在边界上走四连通格子，但不能走有农药的格子，问最后还剩下多少个格子没被虫子走过。

21.2 Solution

好像离散化之后BFS就行啦 ，时间复杂度$O(n^2)$，空间复杂度$O(n^2)$。

22 Codeforces 249D

22.1 Donkey and Stars

给定一个点集合P,满足所有点都在第一象限或者x,y的正半轴上，然后要你找到一个最长点序列$<(x_1,y_1),(x_2,y_2).....(x_k,y_k)>$ 满足，向量$(x_{i+1}-x_i, y_{i+1}-y_i)$的正切值$\in (\frac{a}{b},\frac{c}{d})$ $|P| \le 10^5$

22.2 Solution

首先可以知道对于一个合法的序列，必须满足$x_i < x_{i+1}$,那么先对点按照x从大到小排序，令$f(i)$ 表示以i 号点开始的最长序列长度。可知$f(i)=\max f(j)+1$,其中j 必须满足正切值的限制。

形式化表达正切值的限制就是：

$$\frac{a}{b} < \frac{y_j - y_i}{x_j - x_i} < \frac{c}{d}$$

整理一下就是

$$by_j - ax_j > by_i - ax_i$$

$$dy_j - cx_j < dy_i - cx_i$$

令$w_1(i) = by_i - ax_i$,$w_2(i) = dy_i - cx_i$

也就是询问所有满足$w_1(j) > w_1(i) \bigwedge w_2(j) < w_2(i)$的$f(j)$最大值，这个可以用树状数组套线段树实现。时间复杂度$O(n \lg^2 n)$，空间复杂度$O(n \lg^2 n)$

23 Codeforces 319D

23.1 Have You Ever Heard About the Word

一个字符串的子串是该字符串的一段连续子序列，如bca是abcabc的子串，而cc不是。

一个重复块(repeating block)由一个字符串与自身连接而成，如abcabc是一个重复块，而abcabd, abababb不是。

你有一个由拉丁字符组成的字符串。每一步你要找到它的子串中最短的重复块，如果有多于一个，你必须选择最左边的那个。你要将那个形如XX(X - 某个字符串)的重复块替换成X，换句话说你要删除其中的一个X。重复以上步骤直到字符串中不存在重复块。

23.2 Solution

首先我们可以知道，去掉的重复块的大小不会递减。所以一个暴力点的做法就出来了，我们枚举重复块的长度L，然后判定是否存在长度为L的重复块（比如abababab的长度为2），如果存在则去掉所有的长度为L 的重复块。

这样的复杂度为$O(n^2)$。因为去除的长度在递增，所以我们最多去除$O(\sqrt{n})$次。只要我们可以做到高效的判定，那么复杂度就可以做到$O(n\sqrt{n})$

我们现在来分析如何判定长度L。有一篇讲后缀数组的论文中提到过一种方法。

我们先弄一些等距间隔点$< 1, L, 2L, 3L....\lfloor \frac{n}{L} \rfloor L >$,我们假设$s$串中存在一个长度为$L$的重复块$l, mid, r$，满足$mid = l + L - 1$,$r = mid + L$,$s[l...mid] = s[mid + 1...r]$。则一定存在两个等距间隔点$x, y$满足$x = iL, y = (i + 1)L$。

算出最长的长度l_1满足$s[x - l_1 + 1...x] = s[y - l_1 + 1...y]$,最长的$l_2$满足$s[x...x + l_2 - 1] = s[y...y + l_2 - 1]$，则有$l_1 + l_2 \geq L$。

所以我们的算法就是枚举所有间隔点，然后算出相邻间隔点的l_1, l_2。再判断其是否大于L，可以使用二分+$hash$，复杂度$O(\lg n)$，总共有n/L 个间隔点，所以这样的总复杂为$O(\lg n) * O(n(1 + \frac{1}{2} + \frac{1}{3}...\frac{1}{L})) + O(n\sqrt{n}) = O(n \lg^2 n + n\sqrt{n})$

时间复杂度：$O(n \lg^2 n + n\sqrt{n})$,空间复杂度：$O(n)$。

（话说，暴力方法加一些优化也可以水过）。

24 Codeforces 285E

24.1 Positions in Permutations

给定$n(1 \le n \le 1000)$，定义位置i是完美的，当且仅当$|P[i] - i| = 1$，求出所有n排列中，完美位置恰好为$k(1 \le k \le n)$个排列个数模$10^9 + 7$的值。

24.2 Solution

我们的思想是先放好那些完美位置，再把剩下的位置填好。令$f[i][j][x][y]$表示在前i个位置中，有j个完美位置，且数字i的使用状态为x(0表示没用过，1表示用过)，数字$i+1$的使用状态为y(0没用过，1用过)，转移十分简单。

再令$F[j] = \sum f[n][j][x][y]$，那么我们现在来想想$F[j]$里面究竟存了什么东西。为了方便，我们用$cnt[j]$表示$n$排列中**恰好**有$j$个完美位置的排列数。

那么就可以得到

$$F[j] = \sum_{i=j}^{n} \binom{i}{j} cnt[i] \tag{13}$$

不难得出$F[n] = cnt[n]$，所以我们可以让j 从$n..1$利用已知的$F[i]$推出所有的$cnt[j]$。

时间复杂度：$O(n^2)$，空间复杂度：$O(n^2)$。

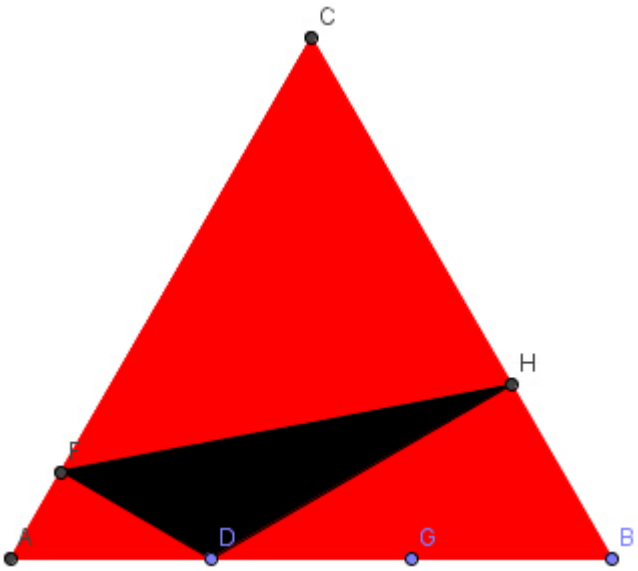
25 Codeforces 309D

25.1 Tennis Rackets

给定一个正三角形，然后每条边有等距离的$n(n \le 31000)$个点将边分成$n+1$份，每条边上，离顶点距离不超过m的点都是障碍点，现在问你，从每条边上选择一个点（不能选障碍点）构成钝角三角形的方法有多少种。

25.2 Solution

一个三角形$a, b, c(a, b < c)$是钝角三角形的充分必要条件是$a^2 + b^2 < c^2$



令$L = m + 1, R = n - m$, 我们先枚举AD长度$a(L \le a \le R)$(另D成为钝角)，设AF长为x，HB长为y。让用$n + 1$来代替n, 则由余弦定理可知：

$$FD^2 = a^2+x^2-ax, DH^2 = (n-a)^2+y^2-y(n-a), FH^2 = (n-x)^2+(n-y)^2-(n-x)(n-y)$$

令

$$FD^2 + DH^2 < FH^2$$

整理可以得到:

$$x < \frac{2na - ay - 2a^2}{n - a + y}$$

不妨再从$[L, R]$枚举y，然后就可以$O(1)$算出有多少个x满足。最后让答案乘以3就好。

这样做的时间总复杂度是$O((n - 2m)^2)$，空间复杂度是$O(1)$，当然想在3s内跑出来，需要一点优化，比如三角形是对称的，AD的长度只需要枚举到$(L + R)/2$,另一半跟前一半相等，特别处理一下中间的点。仅这个优化就够了。

26 Codeforces 263E

26.1 Rhombus

给出一个$n \times m$的表格，第i行第j列上有一个非负整数a_{ij},在$x \in [k, n - k + 1]$,$y \in [k, m - k + 1]$上定义函数

$$f(x, y) = \sum_{i=1}^{n}\sum_{j=1}^{m}(a_{ij} \times \max\{0, k - |i - x| - |j - y|\}) \tag{14}$$

求出一个(a, b)，满足$f(a, b)$最大，$n, m \le 1000$。

26.2 Solution

先把那个不好看的max给弄掉,也就得到

$$f(x,y)=\sum_{|i-x|+|j-y|<k}(a_{ij}\times(k-|i-x|-|j-y|)) \tag{15}$$

满足$|i-x|+|j-y|<k$的点(i,j)构成一个斜着的正方形，其对角线长度为$2k-2$，所以我们只需要维护这个正方形内我们需要的信息就好，我们不妨将其拆开来计算(另$P(i,j)$表示满足上述条件)

$$f(x,y)=\sum_{P(i,j)}ka_{ij}-\sum_{P(i,j)}(|i-x|a_{ij})-\sum_{P(i,j)}(|j-y|a_{ij}) \tag{16}$$

去掉绝对值的方法也就是分类讨论，分别讨论$i\le x$和$i>x$的情况就可以算出第二部分求和，第三部分也可以用类似的方法。其中$i\le x$就是正方形的上半部分（一个直角三角形），依次类推，我们需要知道四个大三角形的信息。

总的方法就是先枚举x，初始另$y=k$，计算出这个正方形的上下左右四个三角形的有用信息：a_{ij}的和，ia_{ij}的和，ja_{ij} 的和，该步计算的代价是$O(n)$（可以通过预处理每一列的前缀和来实现），得到这些信息之后，我们就可以计算出$f(x,y)$,再考虑让$y+1$ 的时候四个三角形的变化，不难发现，移动之后无非就是多一列，少一列，或者多一条斜边，少一条斜边，这些都是可以通过预处理之后$O(1)$ 得到。

这样就可以求出所有的$f(x,y)$，再比较最大值。

时间复杂度：$O(n^2)$，空间复杂度：$O(n^2)$。

27 Codeforces 316D

27.1 PE lesson

输入$n(n\le10^5)$，和一个序列$a[1..n](1\le a[i]\le2)$，现在问你有多少个排列可以由初始状态$(1,2,3...n)$，通过一些合法的交换（交换两个元素）得到，说一个交换序列是合法的，当且仅当位置i的交换次数不超过$a[i]$,输出答案模10^9+7的结果。

27.2 Solution

首先可以知道答案仅仅与$a[1..n]$中1的个数t_1 和2的个数t_2，因为任意一个排列都可以描述成若干个环（这种关系是一一对应的），然后就有个结论，如果一个排列是合法的，那么它的“环表示”中，每个环最多包含2个$a[i]=1$的点（不难证明）。

有了这个之后，我们先求一个$f[i]$表示将i 个元素划分成若干块，使得每一块的元素个数不超过2的方法数。很简单

$$f[i]=\begin{cases}1 & 0\le i\le1\\ f[i-1]+(i-1)f[i-2] & i\ge2\end{cases} \tag{17}$$

所以，我们先确定那些$a[i]=1$的位置所处的环的情况，然后再确定剩下的$a[i]=2$的元素在环上的下一个点是谁。所以我们可以得到答案为

$$Ans = f[t_1] \times \frac{n!}{t_1!} \tag{18}$$

时间复杂度：$O(n)$，空间复杂度：$O(1)$。

28 Codeforces 261D

28.1 Maxim and Increasing Subsequence

给定一个序列$a[1..n](n \leq 10^5)$满足任意正整数$a[i] \leq maxb$，$(maxb \times n \leq 2 * 10^7)$, 然后现在你可以将这个序列重复$t$次之后得到一个新的序列，求这个序列的最长严格上升子序列。有不超过10组多测。

28.2 Solution

我们设答案为$Answer$,因为要求严格上升，所以有$Answer \leq maxb$,又因为最长的长度不超过n，也即$Answer \leq n$，所以有

$$Answer \leq \min\{n, maxb\} \tag{19}$$

又因为$maxb \times n \leq 2 * 10^7$，所以有$\min\{n, maxb\} \leq 4500$，所以$Answer \leq 4500$。我们设不同的数字个数为$m$。如果$maxb \leq n$，则$maxb \leq 4500$，否则$n \leq 4500$。二者都能推出$m \leq 4500$。

所以我们设计一个DP做法，先将所有数字离散化成$1..m$。然后令$f[i][j]$，表示以数字$i(i \leq m)$ 结尾，严格上升子序列长度为j 的最小结束位置$(1 \leq f[i][j] \leq nt)$, 再令$s[i][j] = \min_{k=0}^{i} f[k][j]$。则$f[i][j]$可以在$O(1)$的时间内从$s[i-1][j-1]$ 转移得到（具体实现需要预处理一堆东西,而且这两维还可以滚动）。有了$f[i][j]$，就可以轻松得到答案。设一共有K组测试。则

时间复杂度：$O(n * maxb * K)$，空间复杂度：$O(n)$。

29 Codeforces 241E

29.1 Flights

给出一个$n(n \leq 1000)$个点$m(m \leq 5000)$ 的拓扑图（没有重边和自环），每条边的长度一开始为1，现在你可以将一些边的权值修改成2，使得任意从1到n的路径的长度相同，输出方案。

29.2 Solution

如果某个点可以从1走到，而且可以走到n，那么我们就说这个点是“有意义”的。

如果满足任意一条从1到n的路径长度相同。则有对于任意“有意义”的点u，所有从1到u的路径长度都相同，不妨设为$d[u]$。

如果满足对于任意一条输入的边$(u,v)(u,v$ 都是有意义的)，$d[v] \leq d[u]+2$且$d[v] \leq d[u]+1$。则可以推出任意一条从1 到n 的路径长度都相同。那么就是求解一个合法的$d[]$,或者输出无解。

这个问题可以用差分约束来做。

时间复杂度：$O(e(n+m))$，空间复杂度：$O(m+n)$。

30 Codeforces 332D

30.1 Theft of Blueprints

给出一个$n(n \leq 2000)$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，输出答案的整数部分。

30.2 Solution

总的期望等于每条边的期望之和。对于一条边（看成有向的）(u,v)，如果u的度数$d[u] \geq k$,则该条边的贡献为$a_{u,v}\binom{d[u]-1}{k-1}/\binom{n}{k}$。另$s[u]$表示$u$连出的所有边的边权和。

所以我们可以得到最终的答案为：

$$E = \frac{1}{\binom{n}{k}} \sum_{d[u] \geq k} s[u] \binom{d[u]-1}{k-1} \tag{20}$$

另$W[d] = \binom{d}{k-1}/\binom{n}{k}$, 可以知道$W[n] = \frac{k}{n-k+1}$，且$W[i] = \frac{i-k+2}{i+1}W[i+1]$,所以我们可以按照$n, n-1, ...k-1$的顺序求解$W[i]$，因为每次递推是乘以一个小于等于1的实数，所以$W[i]$单调不上升。考虑到我们只需要得到答案的整数部分，所以只要$W[i]$小于某个特别小的实数(例如10^{-15})的时候,我们就可以让它等于0了。

最后计算

$$E = \sum_{d[u] \geq k} s[u]W[d[u]-1] \tag{21}$$

时间复杂度：$O(n^2)$，空间复杂度：$O(n^2)$。

31 Codeforces 295D

31.1 Greg and Caves

输入一个$N, M(\leq 2000)$,然后问你有多少种$N \times M$的黑白棋盘中间有一个洞。

一个棋盘是有洞的，当且仅当其满足一下条件：

1. 每一行要么全部是0，要么有且仅有两个1 （可以用区间(L_i, R_i)表示这样的行）。
2. 所有非0行都是连在一起的，设最上面的行为T,最下面的行为B。
3. 存在一行t满足，则对于任意两行$T \le i \le j \le t$，都有(L_i, R_i)被包含在(L_j, R_j) 内。
   对于任意两行$t \le i \le j \le B$，都有(L_j, R_j)被包含在(L_i, R_i)内。

31.2 Solution

用$f[i][j]$表示在第i行，且该行的区间长度$R_i - L_i = j$的时候，往上扩展形成类似三角形的形状的方法数。再令$g[i][j]$表示跟$f[i][j]$类似的信息，只是要求第$i-1$行的线段长度严格小于j。

则转移非常明显：

$$f[i][j] = 1 + \sum_{k=1}^{j} (j-k+1) \times f[i-1][k] \tag{22}$$

$$g[i][j] = 1 + \sum_{k=1}^{j-1} (j-k+1) \times f[i-1][k] \tag{23}$$

如果我们预处理

$$s_1[j] = \sum_{k=1}^{j} f[i-1][k] \tag{24}$$

$$s_2[j] = \sum_{k=1}^{j} k \times f[i-1][k] \tag{25}$$

那么

$$\begin{aligned} f[i][j] &= 1 + \sum_{k=1}^{j} (j-k+1) \times f[i-1][k] \\ &= 1 + (j+1)s_1[j] - s_2[j] \end{aligned} \tag{26}$$

$$\begin{aligned} g[i][j] &= 1 + \sum_{k=1}^{j-1} (j-k+1) \times f[i-1][k] \\ &= 1 + (j+1)s_1[j-1] - s_2[j-1] \end{aligned} \tag{27}$$

有了这些预处理，我们就可以枚举最上面的横轴线i，和轴线长度j。然后它对答案的贡献就是

$$\sum_{j=1}^{m-1} g[i][j](\sum_{k=1}^{n-i+1} g[k][j]) \tag{28}$$

如果预处理

$$s[i][j] = \sum_{k=1}^{i} g[k][j] \tag{29}$$

则对于每个枚举的最上方的横轴线，我们就可以$O(1)$ 计算出贡献。

时间复杂度：$O(nm)$,空间复杂度：$O(nm)$。

32 Codeforces 314E

32.1 Sereja and Squares

给定一个长度为n的由‘?’和小写字母组成的字符串s，两个点$(i,j):(i<j)$可以配对，当且仅当$s[i]$ 和$s[j]$分别是同一个字母的小写和大写。一个字符串是合法的当且仅当存在一种配对方式使得每个点仅属于一个配对，且所有的配对不相交。

现在需要你用大写或小写英文字母(除了x,X) 来替换输入串中的‘?’。问有多少种方法可以得到合法串，除以2^{32}取余数。

32.2 Solution

一个简单的性质就是一个字符串如果存在配对，那么这个配对方式唯一，证明很简单，我们只看一种字母，能使得它们不相交的配对方式仅有一种，所以平行地看所有字母都是如此，所以组合起来的字符串的配对方式也是唯一。

所以我们可以一开始不管颜色的问题，我们只需要管是左括号还是右括号，设左括号代价为1，右括号代价为-1。然后令$f(i,j)$表示到了第i个位置，前缀和为$j(j\geq 0)$的方法数。

$$f(i,j)=\begin{cases} f(i-1,j-1)+f(i-1,j+1) & s[i]='?' \\ f(i-1,j-1) & \text{else} \end{cases} \tag{30}$$

令Cnt表示输入串中不是'?'的字符串个数，则答案即为

$$ans=25^{\frac{(n-2*cnt)}{2}}f(n,0)$$

时间复杂度:$O(n^2)$，空间复杂度$O(n)$。当然需要强大的优化和卡常数技巧。

33 Codeforces 266E

33.1 More Queries to Array

给定一个序列$a[1..n](n\leq 10^5)$，以及m个操作，每个操作分为两种类型：

1. $?L,R,k(0\leq k\leq 5)$ 询问$\sum_{i=L}^{R}(a[i]-L+1)^k$
2. $=L,R,x$ 将$a[L...R]$都赋值为x

以上所有运算都是模10^9+7下进行的。

33.2 Solution

首先考虑询问L,R,k，令$l=L-1$，则询问的答案就是

$$\sum_{i=L}^{R}(a[i]-l)^k \tag{31}$$

将二项式展开，不难发现我们只需要维护

$$\sum_{i=L}^{R} a[i]i, \sum_{i=L}^{R} a[i]i^2, \sum_{i=L}^{R} a[i]i^3 ... \sum_{i=L}^{R} a[i]i^5$$

由这6个东西，可以“拼”出所有我们要的东西，而这6个式子的维护可以使用线段树。

时间复杂度$O(m \lg n + n)$。空间复杂度$O(n)$。

34 Codeforces 325C

34.1 Monsters and Diamonds

Piegirl 发现一只怪物和一本关于怪物和馅饼的书。当她在读这本书的时候，她发现有n 种怪物，每种都有一个唯一的1 到n 的编号。如果你喂怪物一块饼，它就会分成一定数量的怪物（可能为零），以及至少一个多彩的钻石。怪物们可能存在多种分裂方式。

最开始Piegirl 有且只有一只怪物。她先喂了一块饼，它随之分裂。对于分出来的怪物，继续喂饼，直到它们都分裂为钻石。然后她把所有钻石收集起来。

你将得到一系列规则(m个)描述不同怪物的分裂方式，每种怪物至少有一种分裂方式。分裂的时候，如果有多种方式，Piegirl 可以选择任意一种。

你的任务是：对于每种怪物，确定以其为起始，Piegirl 可以得到的钻石最少最多分别是多少。Piegirl 有无限多的饼。

$n, m \leq 10^5$，所有规则中到的怪物次数之和小于等于10^5。

如果以某种怪物开始，分裂无法停止，认为其最大最小值都为-1。

如果以某种怪物开始分裂可以停止，而且可以得到无限多的钻石，则认为其最大值为-2。

除了以上两种情况，如果最终结果大于314000000 ，则用314000000 代替。

34.2 Solution

1. **无法停止的怪物**。

   假设怪物x是可以停止的，那么一定存在其一种分裂方式，满足其中涉及到的怪物也都是可以停止的（或者没有怪物），而且我们知道，到了最后，一定是剩下那些只有钻石的怪物停止，所以这些怪物（至少存在一种仅得到钻石的分裂方式）肯定是可以停止的。我们可以借用SPFA 的思想，先将这些点加入队列，然后用这些点去加入更多点进入队列，这样到了最后队列中的所有点都是可以停止的。

   具体实现的话，对于规则c，设要分裂的怪物为m_c,分裂后所有怪物为$x_1x_2..x_k$,然后从x_i向m_c 连一条边权为c的有向边。这样我们就知道每个怪物可以用来更新哪些怪物了。

2. **求最小值**

只要能够停止的怪物就存在最小值。所以类似上面的做法（建图不变，或者删掉那些不停止的怪物重新建图也行），令$Min[x]$ 表示怪物x 的最小值，先将所有存在仅得到钻石的分裂方式的怪物加入队列，令它们的$Min[x]$为x所有仅分裂出钻石的分裂方式的钻石数最小值。然后类似SPFA 的方式去更新其它怪物。

3. **最大值为无穷大的怪物**
   一个怪物x能得到无穷大的钻石，当且仅当，去掉那些不停的怪物，按照上面说过的方式建图后，存在一个环(包括自环)上的点y 能够走到怪物x。所以可以先找出大小大于等于2的强连通块和那些自环中的点，然后从这些点开始bfs，能遍历到的点就是无穷大的怪物。
4. **求最大值**
   删掉无穷大的点，重新建图，类似求最小值的方法求解。

时间复杂度：主要为SPFA的复杂度，通常比较快。

35 Codeforces 293D

35.1 Ksusha and Square

给定一个凸$n(n \le 10^5)$边形,坐标的绝对值小于等于10^6,现在求解随机选取两个在多边形内部(或者边界上)的两个整点，以这两点为对角线的正方形的期望面积。

35.2 Solution

对于两个点(x_1, y_1),(x_2, y_2)，其贡献为$(x_1 - x_2)^2 + (y_1 - y_2)^2)$。所以呢，我们可以分开统计$x$的期望贡献和$y$的期望贡献。先考虑$x$的贡献如何计算，然后$y$ 的计算与x 对称。我们可以枚举一条垂直扫描线X，不难计算出多边形与X相交的情况，然后我们可以得到在该扫描线上，有多少个整点在多边形内部。用$tX[X]$记录。有了$tX[X]$数组之后，我们就可以轻松地计算出所有整点对的横坐标差的平方和。类似我们可以计算出纵坐标差的平方和，设总整点数为cnt，答案就是之前的两个求和除以$cnt * (cnt - 1)$。

设坐标最大值减去最小值为m 时间复杂度：$O(m + n)$。空间复杂度$O(m + n)$。

36 Codeforces 338E

36.1 Optimize

给定两个序列$a[1..n], b[1..m](1 \le m \le n \le 150000)$，统计有多少个起点$i$,满足以下条件：

我们令一个新序列$s[1..m] = a[i..i + m - 1]$，然后构造一个二分图$(L, R)$，左边右边都是$m$个点。如果$s[i] + b[j] \ge H$，则加入一条边$(L_i, R_j)$

要求这个二分图存在完备匹配。

36.2 Solution

由**霍尔定理**可以知道一个左右等势的二分图(L,R)存在完备匹配的充分必要条件是对于任意子集$A \subseteq L$,都有$|N(A)| \geq |A|$，其中$N(A)$是R中所有可以连向A的点构成的集合。

在这道题目中，如果$s_i < s_j$，那么一定有$N(\{s_i\}) \subseteq N(\{s_j\})$,所以我们可以知道，对于所有大小为$k$的子集中，约束条件最难满足的一定是将$s_i$从小到大排序之后前$k$个元素构成的集合。

将$s[i]$从小到大排序，得到的新序列为$q[i]$，再另$w[i]$表示有多少个$b[j]$满足$b[j]+q[i] \geq H$。

所以原图有完备匹配的充分必要条件就是对于任意的i，都有$w[i]-i \geq 0$。

有了这个性质，我们就可以用线段树做这个判定了。

时间复杂度：$O(n \lg n)$，空间复杂度：$O(n+m)$。

37 Codeforces 273E

37.1 Dima and Game

一个基于n对整数(L_i,R_i)的游戏，两个人交替操作，每次每个人可以选择一个整数对(L_i,R_i)满足$R_i-L_i>2$，将它变成$(L_i+\lfloor\frac{R_i-L_i}{3}\rfloor, L_i+2\lfloor\frac{R_i-L_i}{3}\rfloor)$ 或者$(L_i, R_i-\lfloor\frac{R_i-L_i}{3}\rfloor)$。

谁不能操作谁输，假设两个人都使用最优策略，那么有多少种整数对(要求$1 \leq L_i < R_i \leq p$)是先手必胜的？输出答案 $\bmod 10^9+7$。满足$1 \leq p \leq 10^9$。

37.2 Solution

因为这是一个SG组合游戏，所以可以将每个整数对看成一个子游戏，设$sg[i]$表示第i个整数对的SG值，那么整个游戏SG值就等于所有$sg[i]$的异或和。

可以知道$sg[i]$仅仅跟R_i-L_i的值有关。所以我们不妨另$f[i]$表示长度为i的线段的SG值，暴力求解$f[1...10^9]$可以发现SG值仅仅分成102段。

所以剩下的事情就可以用DP解决。

时间复杂度：$O(n)$，空间复杂度：$O(n)$。

38 Codeforces 323B

38.1 Tournament graph

你要构造一个有$N(\leq 1000)$个结点的竞赛图，使得对任意两个结点u和$v(u \neq v)$，从u到v的最短距离不超过2。竞赛图就是基图为无向完全图的有向图（每对结点之间有一条有向边相连，且无自环）。如果无解输出-1。

38.2 Solution

我们假设A集合中的点是一个满足距离要求的图$G=(A,E)$，那么我们现在构造一个新的多4 个点的集合$B=A\cup\{a,b,c,d\}$ 的满足距离要求的图$G'=(B,E')$。其中

$$E'=E\cup\{(a,b),(b,d),(d,a),(a,c),(c,d)\}\cup_{x\in A}\{(x,a),(x,b),(c,x),(d,x)\}$$

我们知道$n=4,n=2$的时候是没有解的，而$n=1,n=3,n=6,n=8$都是有解(可以手工构造)的，所以由以上构造方法可以构造所有$n\geq 5$ 的满足距离要求的图，然后再将图随便加边补成竞赛图就好。

时间复杂度$O(n^2)$，空间复杂度$O(n^2)$。

39 Codeforces 333C

39.1 Lucky Tickets

给定$k(1\leq k\leq 10^4)$，求构造$n(1\leq n\leq 3*10^5)$个不同的8位数（可以有前导零），使得每个8位数都可以通过加入一些括号和$+,-,*$符号满足计算出来的结果等于k。

39.2 Solution

这题目呢，似乎就是用各种方法去凑$3*10^5$个数。我用的是以下方法：

1. 枚举$a(0\leq a\leq k)$,然后令$b=k-a$，然后把a,b写在最左边，然后看还剩下多少位，再枚举剩下的位去凑0（只要满足存在一个0，或者有相邻的位相同就好）。
2. 枚举$a(0\leq a\leq k),b(0\leq b\leq k-a)$,然后令$c=k-a-b$，然后连续在最左边写下$a,b,c$，假如还剩下一位空位，那么我们有两种弄法，一种把0弄到开头去，另一种就是枚举最后一位$l(0\leq l\leq 9)$,然后让a或者b 或者c加上l（目的是凑出$a+b+c-l=k$）。
3. 枚举k的因子a，令$b=k/a$，然后把a,b写在最左边，用1中的方法填满剩下位置。
4. 枚举$a(1\leq a\leq k)$，然后令$q=k/a$，另$r=k \mod a$, 然后连着写a,q,r就行。

综上差不多可以凑出$3*10^5$种方案，最主要的是第二种，中间可以用map判重复。设我们一共弄出来了m个候选数字。

时间复杂度：$O(m\lg n)$，空间复杂度:$O(n)$。

40 Codeforces 241F

40.1 Race

给定一个$n\times m(n,m\leq 100)$的表格。表格上有一些字符。分为以下几种：

1. $'\#'$ ：障碍物。
2. $'1'..'9'$：道路的一部分，数字大小代表通过的时间。

3. $'a'..'z'$：十字路口，通过时间为1。

保证道路就是一条水平或者竖直的$x \times 1$ 或$1 \times x$的形式，不会有两个十字路口连在一起，而且所有十字路口的字母不同。

然后再输入一个起点坐标，一个终点坐标以及一个序列$s(|s| \leq 1000)$。

现在你从起点出发，沿途经过的十字路口按顺序在s串中记录。要求走最短路。现在问你从开始到第$K(K \leq 10^5)$ 分钟的时候你在哪个位置（如果到了终点就不动了）。

40.2 Solution

因为走的路线都是确定的，所以直接模拟。

时间复杂度：$O(nm + |s| \max\{n, m\})$。空间复杂度:$O(nm + |s|)$。

41 Codeforces 338D

41.1 GCD Table

输入$n, m(1 \leq n, m \leq 10^{12})$,和一个长度为$k(1 \leq 10000)$的序列$a[1...k]$。询问是否存在一个正整数对$(X, Y)$满足$(1 \leq X \leq n, 1 \leq Y \leq m)$，使得对于任意$j \in [1, k]$满足：

$$a[j] = \gcd(X, Y + j) \tag{32}$$

如果存在输出YES，否则输出NO。

41.2 Solution

首先设序列中全部数字的最小公倍数为LCM，那么让$X = LCM$肯定是一个理智的选择，因为首先X肯定要是LCM的倍数，然后可以证明如果对于任意$X = wLCM(w > 1)$，Y有解的话，则可以将X替换成LCM。

然后就是Y的问题，我们不妨将X分解质因数（弄出其所有的质因子，可以知道质因子不会超过20个），然后Y 需要满足的条件就是：

对于任意的$1 \leq j \leq k$，$Y + j - 1 \mod a[j] = 0$。同样将$a[j]$分解（按照X的质因子）。然后可以列出一些线性同余方程。

用中国剩余定理(CRT)求一个合法的最小的Y。然后检验Y，如果Y不行就试试第二小的$(Y + M)$，其中M是所有质因子的乘积.....

这样做的复杂度不稳定，不过可以过。

42 Codeforces 240F

42.1 Torcoder

给出一个长度为$N(N \leq 10^5)$的字符串（仅包含小写字母），以及$M(M \leq 10^5)$个操作。每个操作给出一个区间$[l, r]$，你需要将$[l, r]$ 中的字符重排，使得这个子串成为一个回

文串——如果有多种方案，你要使得字典序最小。每个操作结束后都不撤销，也就是将字符串依次进行M次变换。如果某个操作不可能排出回文串，则直接无视这个操作。

你需要输出经过M次操作后的字符串。

42.2 Solution

一个字符串是可以成为回文串的，当且仅当其出现次数为奇数的字母至多一个。所以我们可以使用线段树来维护区间每个字母的出现次数。

对于每个操作$[l,r]$，先用线段树询问$[l,r]$中每个字母出现的次数，然后看看出现次数为奇数的有几个字母。

如果是0个，那么我们从$'a'$到$'z'$开始放每个字母，假设$'a'$字母有x个，那么我就在$[l,l-1+x/2]$和$[r+1-x/2,r]$这两个区间全部放$'a'$，依次类推去放$'b'...'z'$。

如果是1个，那么也类似地去放，只是碰到那个次数为奇数的字母的时候，留一个先不放，最后再放到正中间就好。

设字符集大小为$\sum$ 时间复杂度：$O(n\sum \lg n + m\sum^2 \lg n)$,空间复杂度：$O(n\sum)$。

43 Codeforces 311C

43.1 Fetch the Treasure

给定$n(n \le 10^5)$个位置$ai(1 \le ai \le 10^{18})$，每个位置上恰好有一个宝物价值为wi,假设你有技能集合$\{b1,b2..bk\}$，那么你能够走到的位置是所有的$1+\sum_{i=1}^{k} vibi$，其中vi是非负整数。

初始的时候你拥有技能$\{h\}(1 \le h \le 10^4)$,然后有$m(m \le 10^5)$个操作，每个操作分为三种类型：

1. 1 x：在技能集合中新加入技能x。保证该操作不超过20个。
2. 2 x y：让wi减去y，保证减去之后非负。
3. 3：询问能够到达的位置中的宝藏最大值，并取走该宝藏。

43.2 Solution

本题目的难点就是算出每个位置第一次能够被到达的时刻，可以知道有用的时刻点不超过20个，因为仅在新加入技能的时候才会使得状态变化。

考虑h很小，对于一个位置a,我们的想法是一开始全部用h，直到最后剩下$r(0 \le r < h)$，然后呢，再使用其他的技能来调整，调整的方式就是退掉一部分使用过的h。具体实现如下：

建立h个点，分别代表 $\bmod h$的余数，然后对于任意一个点$u(0 \le u < h)$,枚举所有技能bi，然后令$v = u + bi \mod h$，从u向v连一条权值为$(u+bi) \div h$的边。

然后从0出发，求单源最短路。对于任意有宝藏的位置ai，另$r = ai \mod h$,$q = ai \div h$,如果$dist[r] \le q$那么位置ai就是可以到达的。

这样我们就可以计算出每个位置的加入时刻，剩下的操作用堆就可以实现了。

设1操作的个数为c。

时间复杂度：$O(c^2n + m\lg n)$，空间复杂度：$O(cn)$。

44 Codeforces 269D

44.1 Maximum Waterfall

给定$n(n \leq 10^5)$条同一平面上的水平线段，每条线段用(h_i, L_i, R_i)表示，h_i表示高度，Li, R_i分别表示左右端点。最上面的线段为$(t, -\infty, \infty)$，最下面的线段为$(0, -\infty, \infty)$。

现在要找一个从最上面的线段到最下面的线段的路径，路径是合法的，当且仅当对于路径中任意两个相邻线段i, j 满足$h_i < h_j$且$\max(L_i, L_j) < \min(R_i, R_j)$，而且不存在$h_i < h_k < h_j$,且线段$i, k$和$k, j$都相交。

路径的权值为任意两个相邻位置的相交长度最小值。

44.2 Solution

假设从线段i到线段j可以走，就连边，最后再跑最短路。这是最直观的想法。实际上也是问题的解法，不要幻想着边会很多，可以使用势能分析知道边是$O(n)$的。

然后再使用线段树来实现连边就好，最后可以跑一遍最短路或者DP。

时间复杂度：$O(n\lg n)$，空间复杂度：$O(n)$。

45 Codeforces 152E

45.1 Piglet's Birthday

给定n个架子，第i个架子上有$a_i(\leq 100)$个糖罐子，一开始所有糖罐子都是装满了糖的。然后有m个操作，每个操作表示为$x, y, k(1 \leq x, y \leq n, 1 \leq k \leq 5)$。意思是从架子$x$上随机取$k$个糖罐子，并吃光这$k$ 个罐子的所有糖，然后再把它们放到架子y上。

我们说一个架子是空的，当且仅当架子上的所有糖罐子都是空的。

现在需要你计算在每次操作后，期望的空架子个数。

45.2 Solution

首先每个架子上的**非空罐子个数**是不会增加的,而且每次操作后i号架子的总罐子数($^\circ b_i$)是确定的。所以我们令$f[i][j]$表示架子i上恰好有$j(0 \leq j \leq a_i)$个非空罐子的概率。

也就是我们每次的答案就是$\sum_{i=1}^{n} f[i][0]$。现在就考虑如何维护每次操作后的$f[][]$和$b[]$，可以知道执行完操作(x, y, k)之后，仅仅只有$f[x][j]$才会改变。转移可以在$O(a[x]k)$的时间做到。最后让$b[x]$ 减去k，让$b[y]$加上k，然后让答案减去上次的$f[x][0]$，再加上更新后的$f[x][0]$就行。

时间复杂度：$O(n + m\max\{a_i\})$,空间复杂度：$O(n\max\{a_i\})$。

46 Codeforces 264E

46.1 Roadside Trees

一条直线上$n(n \leq 10^5)$个位置可以种树。一共有m个月，每个月所有树都会长高1米。每个月要么在某个位置种一棵高度为$h_i(\leq 10)$ 的树,要么砍掉从左往右数第$x_i(\leq 10)$棵树。同一时刻，所有树高度不同。一个位置砍掉之后不能再种树。对于每个月输出所有树的最长上升子序列长度。

46.2 Solution

一棵t时刻插入的高度为h的树在第i个月的高度为$h+i-t$。那么我们令第i个月新种的树的高度为h_i-i，这样就避免了树的长高问题。另外可以发现，第i个月插入的树的高度一定小于第$j(j \leq i-10)$ 个月插入的树。

设$f(i)$表示以i位置开头的最长上升子序列长度。当在x_i位置插入h_i-i的时候，由于在$i-10$个月之前插入的树的高度都严格大于$hi-i$，发生改变的$f()$最多不超过10个，用一棵线段树对位置维护在第$i-10$ 个月之前加入的所有树的$f()$最大值。然后再用一个循环枚举最近10个月中加入的树的$f()$即可完成更新。

删除操作，用另外一棵线段树对高度维护当前从左往右数第10个之后的所有树的$f()$的最大值。然后删除操作最多更新从左往右数的前10个$f()$，他们可以由线段树中的信息更新，也可以由这10个中的信息更新。

令c 为一个小于等于10的变量。时间复杂度就为$O(cm\lg n+mc^2)$，空间复杂度为$O(n\lg n)$。

47 USACO Open 13

47.1 Figure Eight

给定一个$n\times n(n\leq 300)$的黑白棋盘，求一个“8”字形满足以下条件：

1. 数字8由上下两个矩形构成。
2. 数字8的上下两个矩形都满足至少有一个单元格在矩形内部。
3. 数字8顶部的矩形的底边必须为底部矩形顶边的子集。
4. 数字8只能刻在白格子上。
5. 规定数字8的得分为上矩形和下矩形的面积的乘积，它们希望得分能达到最大。

输出最大得分。

47.2 Solution

另$f[k][i][j]$表示在第k行，左右边分别为i,j，向上最多能延伸多少，另$g[k][i][j]$ 表示往下最多能延伸多少。则答案可以由$g[k][i][j]$和$f[k][x][y](i\leq x\leq y\leq j)$ 组合得

到，求解f和g的过程十分容易，关键是如何对于给定的k,i,j，找到一个那个决策最优的$x,y(i \le x \le y \le j)$。这个其实也不难，在枚举的每一行k，预处理$b[l][r]$表示所有$l \le x \le y \le r$中$(g[k][x][y]-1)*(y-x-1)$的最大值，则$b[l][r]$可以由$b[l+1][r]$和$b[l][r-1]$递推得到。

时间复杂度：$O(n^3)$，空间复杂度：$O(n^3)$。

48 USACO DEC 12

48.1 First

给定$n(n \le 30000)$个字符串,现在问你哪些字符串可以通过任意修改字母之间的大小关系使得其成为字典序最小的。字符串总长度不超过300000。

48.2 Solution

用后缀数组对所有字符串排序，然后对于每个字符串$s[i]$，枚举j，再枚举所有字符c，再看所有与$s[i]$最长公共前缀至少为$j-1$的字符串的第j位是否出现过c。维护这个东西很简单。

设字符集大小为$\sum$,字符串总长度为m。

时间复杂度：$O(m\sum \lg n + m \lg m)$。空间复杂度：$O(n + m\sum)$。

49 USACO Jan 07

49.1 Cow School

小美考了$N(N \le 50000)$次试，第i次考试得分为Ti，而满分为Pi。在计算总成绩G前，她的老师先将把分数率Fi最低的d份试卷去掉，其中$Fi = Ti/Pi$。然后计算剩余Ti之和以及剩余Pi之和，最后计算总成绩$G = (\sum Ti)/(\sum Pi)$。

输出所有满足以下条件的d：去掉d份试卷，她的总成绩G可以比老师算出来的更高。

49.2 Solution

假设对于某个d是可以输出的，那么也就是在分数率前$n-d$大的那些试卷中可以替换掉若干个试卷使得总成绩更优，很容易用反证法证明如果能够使得总成绩更优，当且仅当存在一种能交换前$n-d$大中的某张试卷和前d小中的某张试卷使得总成绩更优。

所以我们按照分数率从小到大排序得到新序列T,P。

考虑一个d。假设按照老师计算的方法算出来的结果是$\frac{U}{D}$(不约分) 我们一个朴素的想法就是在$n-d+1...n$ 中枚举一个试卷i，再在$1..d$中枚举一个试卷j，计算$a = Tj - Ti$,$b = Pj - Pi$，然后计算$\frac{U+a}{D+b}$是否大于$\frac{U}{D}$。

现在我们考虑上式子的充分必要条件：

$$(\frac{U+a}{D+b} > \frac{U}{D}) \Leftrightarrow (aD > bU) \tag{33}$$

令x表示T，y表示P，那么等价式子可表示为

$$(xj - xi)D > (yj - yi)U \tag{34}$$

也就是

$$yj - \frac{D}{U}xj > yi - \frac{D}{U}xi \tag{35}$$

所以就是在斜率为$\frac{D}{U}$的时候以每个点(xi, yi)作直线。然后$1..d$中截距的最大值要大于$n-d+1..n$中截距的最小值。

这个可以维护上下凸壳实现。

时间复杂度：$O(n \lg n)$，空间复杂度：$O(n)$。

50 USACO DEC 08

50.1 Fence

给定$n(n \leq 250)$个点,选出最多的点使得这些点能够成凸多边形（不存在三点共线的情况）。

50.2 Solution

给定任意凸多边形，一定存在一个点，使得从这个点出发沿着逆时针经过每条边，它们代表的向量的极角单调递增。

所以我们先求出每条边的极角，然后排序。再枚举起点s，另$f[i][j]$表示以向量(i, j)结尾最多经过的边数。则$f[i][j] = \max\{f[k][i]\} + 1$,其中向量$(k, i)$的极角要小于$(i, j)$的极角。所以我们按照之前排好的序进行转移就好，另外同时维护$g[i] = \max\{f[k][i]\}$,所以$f[i][j]$就可以$O(1)$ 由$g[i]$转移。

时间复杂度：$O(n^3)$，空间复杂度：$O(n^2)$。

51 USACO DEC 07

51.1 Best Cow Line

给定一个长度为$n(n \leq 30000)$的字符串s，现在要得到一个字典序最小的字符串c，你每次可以将当前s串的第一个字符或者最后一个字符从s中删去放到c的最后。

51.2 Solution

我们可以贪心来做，假设当前的串是$s[L...R]$，那么如果字符串$s[L...R]$的字典序**严格小于**$s[R..L]$(倒序)，那么我就把$s[L]$放到c的最后，否则就把$s[R]$放到c的最后，实现上，我们可以用后缀数组来比较两个串的字典序。

时间复杂度：$O(n \lg n)$，空间复杂度：$O(n)$。

52 USACO Mar 08

52.1 Land Acquisition

有$n(n \leq 50000)$个元素，每个元素有两个属性$a[i], b[i]$。现在让你将n个元素分为若干块（随你怎么分），每一块的代价是这一块最大的b乘以最大的a。求最小代价和。

52.2 Solution

首先我们将元素按照$a[i]$从大到小排序。则存在一个最优解中，每一个块中的元素在序列（排序之后）是连续的（也就是区间划分）。同样考虑两个元素如果$a[i] \geq a[j]$且$b[i] \geq b[j]$，那么j号元素是没有意义的，可以删掉。

所以最后得到的序列就满足当i从小到大的时候，$a[i]$递减,$b[i]$递增。然后就可以用动态规划来做。

$$f[i] = \min_{j=0}^{i-1}\{b[i]a[j+1] + f[j]\} \tag{36}$$

由于a, b都有单调性，这就是一个经典的斜率优化问题。

时间复杂度：$O(n \lg n)$，空间复杂度：$O(n)$。

53 USACO Mar 13

53.1 Hill Walk

给定$n(n \leq 10^5)$条斜率大于0的位于第一象限的不相交线段,你现在处于第一条线段（左端点为原点），然后你沿着当前线段往上爬，直到右端点就会掉下去，要么跌入万丈深渊结束过程，要么掉到第一个碰到的线段上，然后这个线段就成为当前线段，继续往上爬，重复上述过程。

现在问你，在掉入万丈深渊之前会在几条线段上爬过。

53.2 Solution

可以使用垂直扫描线先来处理，用一个平衡树来维护在当前垂直扫描线扫到的所有线段的相对位置关系，可以知道时掉落事件和线段进出扫描线事件仅发生在端点时刻，所以让扫描线从左往右移动，只要时时刻刻维护好这个平衡树，直到到了当前所在线段的右

端点时刻，在平衡树找到最近的比我当前线段低的线段，让其成为当前线段，并让答案加1。如果没有，就结束。

时间复杂度:$O(n \lg n)$,空间复杂度:$O(n)$。

54 USACO DEC 05

54.1 Cow Patterns

给定一个长度为$n(n \leq 10^5)$的母串A，和一个长度为$m(m \leq 25000)$的模式串B。字符集大小S为25。定义两个等长度的串s_1,s_2 匹配当且仅当，对于任意i, j，若$s_1[i] < s_1[j]$，则有$s_2[i] < s_2[j]$。若$s_1[i] > s_1[j]$，则有$s_2[i] > s_2[j]$,若$s_1[i] = s_1[j]$，则有$s_2[i] = s_2[j]$。求所有匹配位置。

54.2 Solution

仍然可以借用KMP算法的思想，只是定义两个位置匹配的原则，改成该两个位置元素在各自串相同前缀中的大小关系相同。这里只需要$O((n+m)S)$ 的预处理就可以做到，剩下就是使用KMP求解原问题。

时间复杂度：$O((n+m)S)$,空间复杂度:$O((n+m)S)$。

55 USACO OPEN 10

55.1 Triangle Counting

给定$n(n \leq 10^5)$个点，求有多少个由给定点构成的三角形包含$(0, 0)$，过任意两点的直线不经过原点。

55.2 Solution

三个点构成的三角形包含原点，当且仅当任意两个点与原点的夹角小于平角。对于点(x, y)，我们用它来表示向量(x, y),对所有的向量极角排序存入$P[]$。然后计算每个向量可以在跨度不超过平角的限制下，向左向右最远能越过多少个向量。设$L[A]$，$R[A]$分别表示向量$P[A]$最多向左向右越过多少个向量。

统计的时候我们假设先枚举一个向量$P[i]$,然后在$[0, R[i]]$中枚举第二个向量是由$P[i]$向右越过多少个向量得到，设为枚举的量为l_1，另j表示第二个向量的编号。

另$k = n - L[i]$,如果$l_1 + R[j] \geq k$，则让答案加上$l_1 + R[j] - k + 1$，最后得到的答案除以3就好。

因为条件$l_1 + R[j] \geq k$是有单调性的，所以我们不妨在$[0, R[i]]$中二分，得到$[x, R[i]]$中的长度都是满足条件的，再通过预处理$R[j]$的一些前缀和，既可以很快的地对于一个给定i求出其贡献。

时间复杂度$O(n \lg n)$，空间复杂度$O(n)$。

56 USACO Mar 09

56.1 Cleaning Up

给定一个序列$a[1..n](n \leq 10^5)$,可以将其分成任意连续段，代价是每一段中不同$a[i]$ 的个数的平方。求出划分的最小代价。

56.2 Solution

我们分开每一个位置可以得到代价为n的划分，所以在最优解中一段里面不同元素的个数不能超过$\sqrt{n}$。另$f[i]$表示将$a[1..i]$划分后的最小代价和。另$c[j,i]$表示$a[j],a[j+1]..a[i]$中有多少不同元素，

如何求解$c[j,i]$呢?我们求出每种颜色在i之前最靠右的位置（称为颜色标记），然后$\geq j$的颜色标记个数就是$c[j,i]$。如果我们让i单调增加，则可以用一个平衡树来维护颜色标记，每当i增加1，就删除原来元素$a[i+1]$ 在平衡数中的颜色标记（如果存在的话），然后加入颜色标记$i+1$。

这样我们对于要求解的$f[i]$,就可以在当前平衡树从大到小取$\sqrt{n}$个颜色标记，然后就在这些颜色标记旁边转移。

时间复杂度$O(n\sqrt{n}\lg n)$，空间复杂度$O(n)$。

57 USACO Jan 12

57.1 Cow Run

约翰和贝茜两人在玩一个游戏，这个游戏一共进行$N(N \leq 14)$ 个回合，每个回合会提供8张卡片，首先约翰选择前4张或者后4张，然后把剩下的扔掉，然后贝茜在约翰剩下的4张的卡片中选取前两张，或者后两张，设剩下来的两张卡片中前面的那张为A，后面的那张为B，则将当前位置R变成$R*A+B+R$(初始时$R=0$)。

最终的状态R是合法的，当且仅当$R \leq k$ 或者$m-R \leq k$。k是给定的。已知约翰每一步的操作都是完美的，也就是不管之后贝茜怎么操作，约翰总有办法可以到达一个合法状态。

现在告诉你每一步贝茜的操作（约翰不知道），求一个字典序最小的约翰的操作序列（每一次操作如果约翰选前4张则用T表示,否则用B表示）。

57.2 Solution

从约翰的策略来看，如果在前面的操作能够选择B，则一定比选择T要优。所以我们可以得到一个贪心策略。从1到n顺序考虑每个回合，首先看看选择B 之后，之后的状态是否一定能到一个合法状态（贝茜可以随意选择），如果不行就选择T （因为前面的选择保证了后面一定有解），判定能否到达合法状态的方法是用搜索（当然需要一点优化）。

时间复杂度:$O(n4^n)$（最坏），空间复杂度$O(n)$。

58 USACO Open 08

58.1 Cow Neighborhoods

给定$n(n \leq 10^5)$个点(x_i, y_i),以及一个常数C。定义两个点(x_i, y_i),(x_j, y_j)有边当且仅当

$$|x_i - x_j| + |y_i - y_j| \leq C \tag{37}$$

现在问你一共有多少个连通块，以及最大的连通块大小。

58.2 Solution

假设我们令$a_i = x_i + y_i$，$b_i = x_i - y_i$。一个很常用的性质就是

$$\max\{|a_i - a_j|, |b_i - b_j|\} = |x_i - x_j| + |y_i - y_j| \tag{38}$$

所以我们先把点按照a_i从小到大排序，然后考虑当前点为i，我们将所有$a_i - a_j \leq C$的点j全部加入一个平衡树(以b_j为关键字),然后找到平衡树种与b_i相邻的两个点$p1, p2$，用并查集合并$(p1, i)$，$(p2, i)$就行。

时间复杂度：$O(n \lg n)$，空间复杂度：$O(n)$。

59 USACO Open 13

59.1 Photo

假设有一个01序列$a[1...n](n \leq 10^5)$,以及m 个要求，每个要求形如$[L, R]$，表示$a[L] + a[L+1]....a[R] = 1$。现在问你满足这$m$个要求的所有序列中，$a[1] + a[2]...a[n]$ 最大是多少？

59.2 Solution

我们不妨另$s[1] = a[1], s[i] = s[i-1] + a[i](i > 1)$，则对于限制条件$[L, R]$可以表示为$s[R] - s[L-1] = 1$，就可以用差分约束来做了，因为答案要求最大值，所以我们求解最短路。

时间复杂度$O(e(m+n))$(e是SPFA算法的常数)，空间复杂度$O(m+n)$。

60 Google Codejam 2014 E

60.1 Allergy Testing

要找出n种食品中对哪种（有且仅有一种）过敏，每次可以选择吃一个集合，过A天知道结果，如果过敏了，还需要$B - A$来缓解才可以进行下一次实验。求最坏情况下最少多少天测出那个过敏食品。

60.2 Solution

二分天数，然后在天数限制下最多可以检验多少种食品。

可以使用决策树模型来求解，树中的节点代表一个状态（也即过敏食品候选集合），走左孩子代表不过敏，走右孩子代表过敏。现在即求解在天数限制下，决策树最多可以有多少个叶子。

一棵叶子可以用一条路径唯一表示，比如$LLRR$ 代表先走了两次左子树，再走两次右子树到达的叶子。我们不妨设二分的天数为T,路径中L 的个数为X，R的个数为Y,则在最优解中，路径需满足: $T-B<X*A+Y*B\leq T$ 所以能检验的最多食品的种数即为

$$\sum_X\sum_Y\binom{X+Y}{X}[T-B<X*A+Y*B\leq T]$$

首先我们可以构造一个高度为$\lg n$的决策树，所以得到二分的答案$T\leq B\lg n$，也就是说$Y\leq\lg n$，我们不妨枚举Y,然后计算X的上下界K_1,K_2，于是答案变成：

$$\sum_{X=K_1}^{K_2}\binom{X+Y}{X}=\sum_{X=0}^{K_2}\binom{X+Y}{X}-\sum_{X=0}^{K_1-1}\binom{X+Y}{X}$$

可以使用以下公式来计算：

$$\sum_{X=0}^{K}\binom{X+Y}{X}=\binom{Y+K+1}{K}=\binom{Y+K+1}{Y+1}$$

求出来后与n比较大小即可。所以计算一组答案的复杂度即$O(clgn)$,c即计算上述表达式的复杂度。