集训队泛做题解

中山纪念中学 周铭洵

2016 年 1 月 21 日

目录

1 Codechef COUNTARI **17**
- 1.1 题意 . . . 17
- 1.2 关键词 . . . 18
- 1.3 题解 . . . 18
- 1.4 算法流程 . . . 18
- 1.5 复杂度 . . . 19

2 Codechef Martarts **19**
- 2.1 题意 . . . 19
- 2.2 关键词 . . . 19
- 2.3 题解 . . . 19
- 2.4 算法流程 . . . 20
- 2.5 复杂度 . . . 20

3 Codechef MANYLEFT **20**
- 3.1 题意 . . . 20
- 3.2 关键词 . . . 21
- 3.3 题解 . . . 21
- 3.4 算法流程 . . . 22
- 3.5 时间复杂度 . . . 22

4 Codechef EASYEX **22**
- 4.1 题意 . . . 22

4.2 关键词 . . . 22
4.3 题解 . . . 23
4.4 算法流程 . . . 24
4.5 复杂度 . . . 24
4.6 第二类斯特林数简介 . . . 24

5 Codechef HAMILG **25**
5.1 题意 . . . 25
5.2 关键词 . . . 25
5.3 题解 . . . 25
5.4 算法流程 . . . 26
5.5 复杂度 . . . 26
5.6 带花树简介 . . . 26

6 Codechef CBAL **26**
6.1 题意 . . . 26
6.2 关键词 . . . 26
6.3 题解 . . . 26
6.4 算法流程 . . . 27
6.5 复杂度 . . . 27
6.6 离线算法 . . . 27

7 Codechef CHEFBOOK **28**
7.1 题意 . . . 28
7.2 关键词 . . . 28
7.3 题解 . . . 28
7.4 算法流程 . . . 29
7.5 复杂度 . . . 29

8 Codechef GRAPHCNT **29**
8.1 题意 . . . 29
8.2 关键词 . . . 29
8.3 题解 . . . 30
8.4 算法流程 . . . 30
8.5 复杂度 . . . 30

8.6 Dominator Tree 简介 30

9 Codechef BWGAME **32**
9.1 关键词 32
9.2 题解 32
9.3 算法流程 33
9.4 复杂度 33

10 Codechef LPARTY **33**
10.1 题意 33
10.2 关键词 33
10.3 题解 33
10.4 算法流程 34
10.5 复杂度 34

11 Codechef XRQRS **34**
11.1 题意 34
11.2 关键词 34
11.3 题解 35
11.4 算法流程 35
11.5 复杂度 35

12 Codechef RANKA **35**
12.1 题意 35
12.2 题解 35
12.3 算法流程 36
12.4 复杂度 36

13 Codechef FNCS **36**
13.1 题意 36
13.2 关键词 36
13.3 题解 36
13.4 算法流程 37
13.5 复杂度 37

14 Codechef LYRC **37**
14.1 题意 . . . 37
14.2 关键词 . . . 37
14.3 题解 . . . 37
14.4 算法流程 . . . 38
14.5 复杂度 . . . 38

15 Codechef QTREE6 **38**
15.1 题意 . . . 38
15.2 关键词 . . . 38
15.3 题解 . . . 38
15.4 算法流程 . . . 39
15.5 复杂度 . . . 39

16 Codechef SEAORD **39**
16.1 题目大意 . . . 39
16.2 关键词 . . . 39
16.3 题解 . . . 40
16.4 算法流程 . . . 40
16.5 复杂度 . . . 40

17 Codechef TREECNT2 **40**
17.1 题意 . . . 40
17.2 关键词 . . . 40
17.3 题解 . . . 40
17.4 算法流程 . . . 41
17.5 复杂度 . . . 42

18 Codechef QRECT **42**
18.1 题意 . . . 42
18.2 关键词 . . . 42
18.3 题解 . . . 42
18.4 算法流程 . . . 43
18.5 复杂度 . . . 43

19 Codechef RNG **44**
19.1 题意 . . . 44
19.2 关键词 . . . 44
19.3 题解 . . . 44
19.3.1 如何构造一个 k 项线性递推数列的生成函数 . . . 44
19.3.2 关于特征多项式的一些证明 . . . 45
19.3.3 结论 . . . 46
19.3.4 算法 . . . 47
19.4 算法流程 . . . 47
19.5 复杂度 . . . 47
19.6 多项式逆元与多项式整除 . . . 48
19.6.1 定义 . . . 48
19.6.2 求多项式逆元 . . . 48
19.6.3 求多项式整除的商和余多项式 . . . 49

20 Codechef SIGFIB **50**
20.1 题意 . . . 50
20.2 关键词 . . . 50
20.3 题解 . . . 51
20.4 算法流程 . . . 51
20.5 时间复杂度 . . . 51

21 Codechef DEVLOCK **51**
21.1 题意 . . . 51
21.2 关键词 . . . 52
21.3 题解 . . . 52
21.4 算法流程 . . . 52
21.5 复杂度 . . . 52

22 Codechef PRIMEDST **53**
22.1 题意 . . . 53
22.2 关键词 . . . 53
22.3 题解 . . . 53
22.4 算法流程 . . . 53

22.5 复杂度 53

23 Codechef GERALD08 **53**
23.1 题意 53
23.2 关键词 54
23.3 题解 54
23.4 算法流程 55
23.5 复杂度 55

24 Codechef FIBTREE **55**
24.1 题意 55
24.2 关键词 56
24.3 题解 56
24.4 算法流程 56
24.5 复杂度 56

25 Codechef LMATRIX3 **57**
25.1 题意 57
25.2 关键词 57
25.3 题解 57
25.3.1 转换模型 57
25.3.2 计算答案 58
25.4 算法流程 58
25.5 复杂度 59

26 Codechef GNUM **59**
26.1 题意 59
26.2 关键词 59
26.3 题解 59
26.4 算法流程 60
26.5 复杂度 60

27 Codechef RIN **60**
27.1 题意 60
27.2 关键词 60

27.3 题解 60
27.4 算法流程 61
27.5 复杂度 61

28 Codechef SEAEQ **61**
28.1 题意 61
28.2 关键词 61
28.3 题解 61
28.4 算法流程 62
28.5 复杂度 62

29 Codechef TWOCOMP **62**
29.1 题意 62
29.2 关键词 62
29.3 题解 62
29.4 算法流程 63
29.5 复杂度 63

30 Codechef QUERY **63**
30.1 题意 63
30.2 关键词 63
30.3 题解 63
30.4 算法流程 63
30.5 复杂度 64

31 Codechef ANUDTQ **64**
31.1 题意 64
31.2 关键词 64
31.3 题解 64
31.4 算法流程 65
31.5 复杂度 65

32 Codechef SEAARC **65**
32.1 题意 65
32.2 关键词 65

32.3 题解 66
32.4 算法流程 66
32.5 复杂度 67

33 Codechef GRAPHCNT **67**
33.1 题意 67
33.2 关键词 67
33.3 题解 67
33.4 算法流程 67
33.5 复杂度 67

34 Codechef CARDSHUF **68**
34.1 题意 68
34.2 关键词 68
34.3 题解 68
34.4 算法流程 68
34.5 复杂度 68

35 Sereja and Subsegment Increasings **68**
35.1 编号 68
35.2 题意 69
35.3 关键词 69
35.4 题解 69
35.5 算法流程 69
35.6 复杂度 70

36 The Street **70**
36.1 编号 70
36.2 题意 70
36.3 关键词 70
36.4 题解 70
36.5 算法流程 71
36.6 复杂度 71

37 Count on a Treap **71**
37.1 编号 . . . 71
37.2 题目大意 . . . 71
37.3 关键词 . . . 71
37.4 题解 . . . 71
37.5 算法流程 . . . 72
37.6 复杂度 . . . 72

38 Counting on a directed graph **73**
38.1 编号 . . . 73
38.2 题目大意 . . . 73
38.3 关键词 . . . 73
38.4 题解 . . . 73
38.5 算法流程 . . . 73
38.6 复杂度 . . . 73

39 Counting D-sets **73**
39.1 编号 . . . 73
39.2 题目大意 . . . 74
39.3 关键词 . . . 74
39.4 题解 . . . 74
39.5 算法流程 . . . 74
39.6 复杂度 . . . 74

40 Counting The Important Pairs **74**
40.1 编号 . . . 74
40.2 题目大意 . . . 74
40.3 关键词 . . . 74
40.4 题解 . . . 75
40.5 算法流程 . . . 75
40.6 复杂度 . . . 75

41 To Queue or not to Queue **75**
41.1 编号 . . . 75
41.2 题目大意 . . . 75

41.3 关键词 75
41.4 题解 76
41.5 算法流程 76
41.6 复杂度 76

42 Fibonacci Number **76**
42.1 编号 76
42.2 题目大意 76
42.3 关键词 76
42.4 题解 76
42.5 算法流程 77
42.6 复杂度 77

43 Two Roads **77**
43.1 编号 77
43.2 题目大意 77
43.3 关键词 77
43.4 题解 77
43.5 算法流程 78
43.6 复杂度 78

44 Fibonacci Number **78**
44.1 编号 78
44.2 题目大意 78
44.3 关键词 78
44.4 题解 78
44.5 算法流程 79
44.6 复杂度 79

45 Two k-Convex Polygons **79**
45.1 编号 79
45.2 题目大意 79
45.3 关键词 79
45.4 题解 79
45.5 算法流程 80

45.6 复杂度 80

46 Count Special Matrices **80**
46.1 编号 80
46.2 题目大意 80
46.3 关键词 80
46.4 题解 81
46.5 算法流程 81
46.6 复杂度 81

47 Little Elephant and Colored Coins **81**
47.1 编号 81
47.2 题目大意 81
47.3 关键词 81
47.4 题解 81
47.5 算法流程 82
47.6 复杂度 82

48 A New Door **82**
48.1 编号 82
48.2 题目大意 82
48.3 关键词 82
48.4 题解 82
48.5 算法流程 83
48.6 复杂度 83

49 Sine Partition Function **83**
49.1 编号 83
49.2 题目大意 83
49.3 关键词 83
49.4 题解 83
49.5 算法流程 84
49.6 复杂度 84

50 Hypertrees **84**
50.1 编号 . . . 84
50.2 题目大意 . . . 84
50.3 关键词 . . . 84
50.4 题解 . . . 84
50.5 算法流程 . . . 84
50.6 复杂度 . . . 85

51 Short II **85**
51.1 编号 . . . 85
51.2 题目大意 . . . 85
51.3 关键词 . . . 85
51.4 题解 . . . 85
51.5 算法流程 . . . 86
51.6 复杂度 . . . 86

52 Little Elephant and Boxes **86**
52.1 编号 . . . 86
52.2 题目大意 . . . 86
52.3 关键词 . . . 86
52.4 题解 . . . 87
52.5 算法流程 . . . 87
52.6 复杂度 . . . 87

53 Selling Tickets **87**
53.1 编号 . . . 87
53.2 题目大意 . . . 87
53.3 关键词 . . . 88
53.4 题解 . . . 88
53.5 算法流程 . . . 88
53.6 复杂度 . . . 88

54 Something About Divisors **88**
54.1 编号 . . . 88
54.2 题目大意 . . . 88

54.3 关键词 89
54.4 题解 89
54.5 算法流程 90
54.6 复杂度 90

55 The Baking Business **90**
55.1 编号 90
55.2 题目大意 90
55.3 关键词 91
55.4 题解 91
55.5 算法流程 91
55.6 复杂度 91

56 Dynamic GCD **91**
56.1 编号 91
56.2 题目大意 91
56.3 关键词 92
56.4 题解 92
56.5 算法流程 92
56.6 复杂度 92

57 Evil Book **92**
57.1 编号 92
57.2 题目大意 92
57.3 关键词 92
57.4 题解 93
57.5 算法流程 93
57.6 复杂度 93

58 Different Trips **93**
58.1 编号 93
58.2 题目大意 93
58.3 关键词 93
58.4 题解 93
58.5 算法流程 94

58.6 复杂度 94

59 Ciel and Earthquake **94**
59.1 编号 94
59.2 题目大意 94
59.3 关键词 94
59.4 题解 94
59.5 算法流程 95
59.6 复杂度 95

60 Substrings on a Tree **95**
60.1 编号 95
60.2 题目大意 95
60.3 关键词 95
60.4 题解 95
60.5 算法流程 96
60.6 复杂度 96

61 Annual Parade **96**
61.1 编号 96
61.2 题目大意 96
61.3 关键词 96
61.4 题解 96
61.5 算法流程 97
61.6 复杂度 97

62 Knight Moving **97**
62.1 编号 97
62.2 题目大意 97
62.3 关键词 98
62.4 题解 98
62.5 算法流程 98
62.6 复杂度 98

63 Luckdays **98**
63.1 编号 98
63.2 题目大意 98
63.3 关键词 99
63.4 题解 99
63.5 算法流程 99
63.6 复杂度 99

64 A Game of Thrones **100**
64.1 编号 100
64.2 题目大意 100
64.3 关键词 100
64.4 题解 100
64.5 算法流程 100
64.6 复杂度 101

65 Find a Subsequence **101**
65.1 编号 101
65.2 题目大意 101
65.3 关键词 101
65.4 题解 101
65.5 算法流程 101
65.6 复杂度 101

66 Short **102**
66.1 编号 102
66.2 题目大意 102
66.3 关键词 102
66.4 题解 102
66.5 算法流程 102
66.6 复杂度 102

67 Expected Maximum Matching **103**
67.1 编号 103
67.2 题目大意 103

67.3 关键词 . . . 103
67.4 题解 . . . 103
67.5 算法流程 . . . 103
67.6 复杂度 . . . 104

68 Two Magicians **104**
68.1 编号 . . . 104
68.2 题目大意 . . . 104
68.3 关键词 . . . 104
68.4 题解 . . . 104
68.5 算法流程 . . . 104
68.6 复杂度 . . . 105

69 Billboards **105**
69.1 编号 . . . 105
69.2 题目大意 . . . 105
69.3 关键词 . . . 105
69.4 题解 . . . 105
69.5 算法流程 . . . 105
69.6 复杂度 . . . 106

70 Children Trips **106**
70.1 编号 . . . 106
70.2 题目大意 . . . 106
70.3 关键词 . . . 106
70.4 题解 . . . 106
70.5 算法流程 . . . 106
70.6 复杂度 . . . 107

71 To challenge or not **107**
71.1 编号 . . . 107
71.2 题目大意 . . . 107
71.3 关键词 . . . 107
71.4 题解 . . . 107
71.5 算法流程 . . . 107

71.6 复杂度 107

72 Stepping Average **108**
72.1 编号 108
72.2 题目大意 108
72.3 关键词 108
72.4 题解 108
72.5 算法流程 108
72.6 复杂度 109

73 Deleting numbers **109**
73.1 编号 109
73.2 题目大意 109
73.3 关键词 109
73.4 题解 109
73.5 算法流程 110
73.6 复杂度 110

74 Closest Points **110**
74.1 编号 110
74.2 题目大意 110
74.3 关键词 110
74.4 题解 110
74.5 算法流程 111
74.6 复杂度 111

75 Similar Graphs **111**
75.1 编号 111
75.2 题目大意 111
75.3 关键词 111
75.4 题解 111
75.5 算法流程 112
75.6 复杂度 112

1 Codechef COUNTARI

1.1 题意

长度为 N 的数列 A，统计这样的三元组数量：对于 $1 < i < j < k <= N$，满足 $A_j - A_i = A_k - A_j$。范围是 $N <= 10^5, A_i <= 30000$。

1.2 关键词

分块，FFT

1.3 题解

除了分块之外好像没什么其他的好方法。把数列分成 T 块，枚举每一块。有三种情况：

- 三个数都在当前块
- 两个数在当前块，另一个数在前面的块或后面的块
- 一个数在当前块，一个数在前面的块，一个数在后面的块

这三种情况可以不重不漏计算出答案。

显然我们如果知道了两个数，第三个数可以算出来。利用这条性质可以这样统计（这里的桶 A_i 表示当前范围内数字 i 出现次数）：

- 三个数都在当前块：在当前块内枚举两个数作为前两个数（第三个数可以算出来），然后维护后面的数字扔进桶里面，就可以直接算出来累加。
- 两个数在当前块：同样在当前块中枚举两个数作为前（后）两个数，维护之前和之后的块的数字的桶，也可以算出来累加。
- 一个数在当前块：注意到 $2A_{mid} = A_i + A_j$。我们可以维护当前块的桶 A，之前块的桶 B，之后的桶 C, 令 $D_i = \sum_{j+k=i} B_j * C_k$, 显然对于每一个 i，答案能累计上 $A_i * D_{2*i}$。算 D 时用 FFT 就好了。

1.4 算法流程

- 分块
- 枚举块
- 在当前块中枚举两个数算第一第二种情况。
- 前后桶乘起来然后和当前的桶进行运算，算出第三种情况。

1.5 复杂度

$O(N^2/T + TClogC)$
T 表示分成 T 块，C 表示数字上限。实际 T 取 30 左右就可以了。

2 Codechef Martarts

2.1 题意

一个完全二分图，边有两个权值 $A_{i,j}$ 和 $B_{i,j}$，要进行匹配。令匹配边的 A 值总和为 H，B 值总和为 G。

对手的目的是最大化 $G - H$，其次最大化 G，他会在知道了匹配之后选择是否去掉一条匹配边，使得该边的权值不算入 H 和 G。

任务是找一个完全匹配，最大化 $H - G$，其次最大化 H。

点数在 100 以内。权值在 10^{12} 以内。

2.2 关键词

二分图匹配，匈牙利算法，KM 算法

2.3 题解

为了方便，我们定义每条边的价值 $W_{i,j} = A_{i,j} - B_{i,j}$

这样我们的任务是最大化价值和，其次最大化 H。对手的任务是最小化价值和，其次最大化 H（当价值和一定之后要最大化 G 与最大化 H 实际上等价）。

只考虑最大化价值和：对方会删掉一条边，当且仅当这条边是匹配边中最大的而且价值为正值。直接用 KM 算法似乎不好做。考虑枚举对手会

删掉哪一条边。将边按关键字从小到大排序一条一条加入图中。这样这条边对于对手来说删掉它会优于删掉之前加入的所有边。如果我们强制匹配了这条边，那么我们最终获得的价值和就是除去这条边所连的两个点后剩下的点的最大匹配。

所以我们的任务就是高效地实现“加入一条边”并且“强制匹配这条边”。

可以这样实现：首先我们令图中所有边的价值均为负无穷。然后要加入一条边时首先令该边价值为正无穷（使它一定被匹配上），维护出整幅图的最大匹配，这样就可以算出该边作为被删边时的最优答案。然后我们再令该边的价值为它原来的价值，再次维护图的最大匹配。这样就为接下来的操作做好了准备。

问题在于我们怎么处理“更改一条边权”后维护最大匹配呢？

我们使用 KM 算法中的每个点的 $Label$ 值。设改变的边所连的点分别是 i, j。我们先将 i 和它之前的匹配点 $mate(i)$ 在匹配集中删掉，然后维护 $Label_i$，令 $Label_i = max_j(W_{i,j} - Label_j)$，再从 i 点出发找一条增广路使匹配完全。这样就可以实现改变边权之后无须整幅图重构而能高效地维护最大匹配了。

再考虑两个关键字的情况。由于边变成了双关键字，我们可以用个 pair 之类的来进行加减法和大小比较。这样就解决了双关键字的情况。对边排序时应注意，对手同样希望最大化 H，对于对手来说如果有两条价值相同的边，他会删掉 $A_{i,j}$ 更小的那条。

最后就是常数问题了。我自己的方案是倒过来做可以最优性剪枝。

2.4 算法流程

- 初始化 $Label$ 和边权
- 枚举被对方删掉的边，强制匹配上，维护 $Label$，计算当前的答案
- 将这条边正常地加入图中，维护 $Label$
- 重复这个过程

注意这个流程是正着做。也可以反过来做。

2.5 复杂度

$O(N^4)$

3 Codechef MANYLEFT

3.1 题意

一个 $N*N$ 棋盘，有些棋子，类似于跳棋，相邻的可以跳过去（假如跳过去那格是空的）然后被跳过的就消失了。然后就不停地跳直到再也跳不了，求尽量优的方案让剩下的棋子最多（不要求最优解）。范围是 $10 <= N <= 30$。

3.2 关键词

搜索，A*，Beam Search，随机算法

3.3 题解

这题大致有两个思路：搜索和随机。

搜索

搜索的话，由于不是找最优解，而且时间非常有限，考虑 A* 算法或者 Beam Search 算法。A* 算法大家比较熟悉。后面那个算法大致是强制令搜索树的每一层节点最多只有 B 个（B 是一个合适的常数）。两个算法的核心都是估价函数。其实有一个靠谱的估价函数无论是哪种算法都能得到不错的解。这题估价函数有很多种。一种效果很不错，而且比较简单的是对于一个局面的每个棋子，如果有棋子与它相邻就令当前局面的得分减去常值 C。然后这样搞基本上能拿到 90 分左右（雾）。

随机

其实仔细考虑一下的话，在这题中随机可能会比搜索要好。因为每个局面的决策实在是太多了，而且会有大量价值相差不大的局面，我们难以在时间和空间都比较紧张的情况下保证质量高的解出现在决策树当中。

那么随机调整在这题中效果非常好。因为我们可以用不多的调整来尝试使一个解变优。我们可以一开始随机一些解，找到一个较优解，然后每次重构解后面的部分，或者是去掉解中间的某些移动，然后再重构。当然如何重构解决定了我们最终解的质量。一种比较简单的方法是每次选择能使局面的相邻棋子数目减少最多的那个移动，然后一直动下去。在所给时间内尽可能的尝试，找到更优的解。目前最优的算法好像都是随机（本题中）。

其他

还有一个值得注意的是如果将棋盘黑白染色，那么黑格的棋子永远是通过移走一个白色的棋子然后跳到黑格子上。反之亦然。所以一种不错的策略是尽量移动一种颜色的棋子。实践证明这个策略还是比较有效的。

3.4 算法流程

- 求一个不错的解
- 随机砍断或去掉一些移动
- 剩下的移动先动，构造一个新解
- 比较最优性，更新答案
- 重复这个过程

3.5 时间复杂度

没法算。

4 Codechef EASYEX

4.1 题意

有 N 个骰子，每个有 K 个面，每个面都有不同的颜色。然后将这 N 个骰子分别随机投掷，然后统计所有骰子朝向上的颜色。设每种颜色有 A_i

个骰子是该种颜色。给定整数 L, F，求下面的式子的期望（在模 2003 下运算）：

$$\prod_{i=1}^{l} A_i^F$$

范围是 $0 < N, K <= 10^9, 0 < F <= 1000, 0 < L * F <= 50000$。

4.2 关键词

组合数学，第二类斯特林数

4.3 题解

直接做是不好做的。设 $X_{i,j}$，若第 j 个骰子颜色是 i，则 $X_{i,j}$ 值为 1，否则为 0。那么可以这样推：

$$\begin{aligned} ans &= E(\prod_{i=1}^{L} A_i^F) \\ &= E(\prod_{i=1}^{L} (\sum_{j=1}^{n} X_{i,j})^F) \end{aligned}$$

然后考虑展开式子。

$$\begin{aligned} ans &= E(\prod_{i=1}^{L} (\sum_{j=1}^{n} X_{i,j})^F) \\ &= E(\sum_{all} X_{1,..}X_{1,..}\cdots X_{2,..}X_{2,..}\cdots X_{L,..}X_{L,..}\cdots) \\ &= \sum_{all} E(\underbrace{X_{1,..}X_{1,..}\cdots}_{F\ times}\underbrace{X_{2,..}X_{2,..}\cdots}_{F\ times}\underbrace{X_{L,..}X_{L,..}\cdots}_{F\ times}) \end{aligned}$$

这个式子有挺多性质的：

- 每一项要么是 0，要么当该项中所有的 X 全部是 1 时该项是 1，所以它的期望就是它的概率。
- 每一项均由 L 组组成，每组有 F 个 X_i 组成。

- 一个骰子不会有两种颜色，所以一个骰子，即第二下标 j 只会在同一组出现，若出现在两组中这一项一定是 0。
- 若一项中总共出现 P 个不同骰子，那么它对答案贡献是 $\frac{1}{K^P}$（就是它的概率）

所以我们考虑枚举 P，所有出现了 P 个不同骰子的项对答案的贡献都是一样的。那么只需要知道它们的个数就可以了。个数等于：

$$\binom{N}{P} \cdot P! \cdot count_P$$

对上式解释：首先我们从 N 个骰子中选出 P 个，然后安排顺序（以它们第一次出现的顺序）。接下来，我们设 t_i 表示第 i 组中出现了 t_i 个不同的骰子。那么一组中的方案就是将 F 个变量分配到 t_i 个等价类中去。这个就是第二类斯特林数 $S(F, t_i)$。总的就是所有方案数乘起来。因为会有很多种 t 的安排，所以:

$$count_P = \sum_{t_1+t_2+\cdots+t_L=P} (\prod_{i=1}^{L} S(F, t_i))$$

$count$ 的计算可以通过 DP 或者卷积处理（其实一样）。
所以最终我们有：

$$ans = \sum_{P} \binom{n}{P} \cdot P! \cdot count_P \cdot \frac{1}{K^P}$$

可以发现当 $P > 2003$ 之后的项都变成 0 所以不用计算。

4.4 算法流程

- 预处理第二类斯特林数
- 读入数据
- 卷积算出 $count$
- 枚举 P，计算 ans。

4.5 复杂度

$O(F^2 + 2003^2 \cdot logL)$

4.6 第二类斯特林数简介

$S(n,k)$ 表示 n 个物体分配到 k 个等价类中。

$$S(0,0)=1$$
$$S(n,0)=0\ (n>0)$$
$$S(n,k)=k*S(n-1,k)+S(n-1,k-1)$$

5 Codechef HAMILG

5.1 题意

有个无向图，一开始棋子在某一个点上。接下来按先手，后手，先手，后手……这样的顺序移动棋子，每次可以移动到一个相邻的且没有被走过的节点。如果没有能走下去的节点就算输。求有多少个起始点能让后手获胜。范围是点数在 1000 以内，边数在 10^6 以内。

5.2 关键词

一般图最大匹配，带花树

5.3 题解

这种每次移动向下个节点的都要往匹配的方向靠。假设已经得到了一个最大匹配。如果这个图有完美匹配，那么先手每次都走匹配边，然后后手一定会走一条非匹配边（或者走不了），一直下去后手一定输。如果没有完美匹配，那么从任意一个未被匹配的点走出去，先手一定走非匹配边，后手一定走一条匹配边，最终同理上面的情况，先手一定输。注意这里不会有其他情况，因为如果最终是先手走了一条非匹配边之后就走不了了，证明最终点是非匹配点，那么从起点到终点的路径就是一条增广路，不符合最大匹配的定义。

那么问题在于一个点只要它在某一个最大匹配中未被匹配到就一定可以作为后手必胜点。枚举所有可能的最大匹配显然是不科学的。暴力的方法是强制一个点不匹配然后令剩下的点去匹配看看匹配数是否不变。更好的方法是我们任意找到一个匹配，然后可知一条备用路径（长度为偶数，从一个未匹配点开始，于一个匹配点结束的交错路）的两个端点都可以是

非匹配点（翻转备用路径可以使任意一个端点变成非匹配点）。所以我们可以从当前每一个非匹配点开始，做一次找带花树算法中增广路的过程（肯定找不到的）。然后把所有长度为偶数的交错路的终点以及花上的点（花上的每个点可以通过翻转花上使它成为可行的交错路的终点）加入答案。然后这样所有可能不被匹配的点就可以找出来了。

5.4 算法流程

- 找出任意一个最大匹配
- 从当前匹配的非匹配点出发找增广路，把符合条件的点加入答案。

5.5 复杂度

$O(N^3)$

5.6 带花树简介

带花树算法的本质思想还是从未匹配点找增广路（就是匈牙利算法里面那个）。我们用 BFS 实现。问题在于它的一条交错路可能出现奇环（二分图里面不会有）。我们就将这个奇环上的边变成双向的（本来是从搜索树的根到树底方向的），然后把环上所有的点缩成一个超级节点（本质就是缩点）。这样这个环就等价成一个节点了，就可以继续找增广路了。然后可能会出现环套环，由于缩了点了就不用怕了。

6 Codechef CBAL

6.1 题意

一个长度为 N 的小写字母字符串 S，如果一个连续子串里面所有的字母出现次数都是偶数，那么我们称这个子串为平衡字符串。给出 Q 组询问，每次给出 $L, R, type$，要求给出区间 $[l, r]$ 中所有平衡字符串长度的 $type$ 次方的和。$type = 0, 1, 2$。询问要求强制在线。范围是 $1 <= N, Q <= 10^5$。

6.2 关键词

分块, 预处理

6.3 题解

先转化模型。设 $count_{i,c}$ 表示 1 到 i 位置中 c 字符出现的次数的奇偶性（用 0，1 表示）。那么如果一个子串 $s_{i,j}$ 是平衡子串，那么肯定对于所有的 c, 都有 $count_{i,c} = count_{j-1,c}$。又因为只有 26 个字母，$count_i$ 可以用一个 26 位的二进制整数就可存下。所以问题可以大概转化成：

给出一个长度为 N 的数组，每次询问区间 $[l, r]$ 中的相同数字对的相隔的长度的 *0*，*1*，*2* 次方。

考虑如何暴力，对于每个询问，我们可以从 l 到 r 按顺序扫，维护 $t0[i], t1[i], t2[i]$ 分别表示 i 这个数字之前出现位置的 $0, 1, 2$ 次方和。然后和当前的做下运算就可以更新答案，然后维护这三个数组也很简单。暴力是 $O(QN)$ 的。

暴力的话并没有任何的预处理，实际上我们可以预处理非常多的信息来优化算法。我们可以每隔 $\sqrt{N}$ 定一个“观测点”，总共会有 $\sqrt{N}$ 个观测点。然后预处理出每个观测点向前向后拓展出的区间的答案，用暴力就可以在 $O(N^{1.5})$ 的复杂度下预处理出来。

然后看看怎么询问。设当前询问的区间是 $[l, r]$。如果区间长度小于 $\sqrt{N}$ 就直接暴力。否则区间内一定会有观测点。设区间中最左边的观测点是 lp, 最右边的是 rp，那么 $ans[lp][r], ans[l][rp]$ 这两个值都是已经预处理过的了。把它们相加，那么中间 $ans[lp][rp]$ 的部分重复了一次，减去即可。最后漏掉的部分是开头在 $[l, lp)$，结尾在 $(rp, r]$ 的那些数字对。由于这两个区间长度均小于 $\sqrt{N}$，直接用类似暴力的方法做就可以了。最终一次询问可以做到 $O(\sqrt{N})$ 的复杂度。

6.4 算法流程

- 预处理，从每个观测点分别向前向后处理
- 询问, 小区间暴力，大区间利用预处理信息加部分暴力做

6.5 复杂度

$O(N^{1.5} + QN^{0.5})$

6.6 离线算法

好像可以做到 $O(NlogN)$?

7 Codechef CHEFBOOK

7.1 题意

给出一个有向图，每条边有个权值 $L_{x,y}$，要给每个点分配两个非负权值 P_x 和 Q_x，最终一条边的权值是 $W_{x,y} = L_{x,y} + P_x - Q_y$，给出每条边的约束条件：$S_{x,y} <= W_{x,y} <= T_{x,y}$。求最优权值分配下最终权值和的最大值或输出无解。点数小于 100。

7.2 关键词

线性规划，对偶，网络流

7.3 题解

这里讲的比较形式化，具体的式子一定要写下来，推不然可能看不懂。

这题如果将每个约束拆开两个，稍微调整一下式子，把 P,Q 当作变量就是一个经典的线性规划问题：

- 约束条件：$\sum A_{i,j}x_j \le B_i,\ 1 \le i \le m$
- 变量：$x_j \ge 0, 1 \le j \le n$
- 目标函数：最大化 $z = \sum C_j x_j$

观察此题中的约束条件：矩阵中每个横行中一个 +1 一个 -1，然后每一个约束都有一个对应的约束（系数相反，常数项不同）。

直接做不好做。我们考虑它的对偶问题 (注意下面的 A 是转置后的 A)：

- 约束条件：$\sum A_{i,j}y_i \ge c_j,\ 1 \le j \le n$
- 变量：$y_i \ge 0, 1 \le i \le m$
- 目标函数：最小化 $z = \sum b_i y_i$

对偶后发现将所有的约束条件的左边加起来是 0，右边加起来也是 0。这样的意思就是所有的不等式约束全部变成等式了! 然后再看，一个变量 y 只会出现在两个约束中，而且一个系数是 1，一个系数是 -1，如果我们把一个变量 y 看作是一条边，一个约束看作是一个点，那么我们将一个变量 y 出现的两个约束连起来，然后将一个约束常数项当作是连向原点或汇点，然后因为是要求费用最小，跑一遍最小费用最大流就可以求出原问题的目标函数值。

题目还要求一个解。我们可以这样想：我们最终构出来的解一定会恰好被某些约束“卡”住（我们称之为检验约束），即这些不等式约束最终应该是等式。我们跑一遍费用流之后，那些被流过的边所对应的约束（注意这里已经对偶回原问题）就是检验约束。我们可以这样想：假如原本没有求最大值的要求，那么这个问题就是一个差分约束问题。现在我们通过对偶之后做费用流知道了哪些约束是检验约束能使得目标函数最优。所以我们最终把原问题的约束和那些检验约束一起加入进来做差分约束，这样得到的解一定能满足目标函数的最优性。检验约束的等式就加多一条不等式就可以了。

7.4 算法流程

- 将原问题转成线性规划问题，然后对偶
- 根据对偶问题构图，跑费用流，得到目标函数最优值
- 根据费用流的图的情况得到检验约束
- 将检验约束和原问题约束一起做差分约束得到一组解

7.5 复杂度

$O(Maxflow(N, M) + N^3)$

8 Codechef GRAPHCNT

8.1 题意

一个 N 个点，M 条边的有向图，求有多少个无序点对 (x, y)，满足能找到一条从 1 到 x 的路径和一条 1 到 y 的路径除了 1 之外没有公共点。范

围是 $1 \leq n \leq 10^5$, $1 \leq m \leq 5 * 10^5$

8.2 关键词

Dominator Tree

8.3 题解

这题就是找必经点而已。建出 Dominator Tree 之后用 1 点的子树搞搞就好了。

8.4 算法流程

- 建 Dominator Tree
- 计算答案。

8.5 复杂度

$O(N + M)$

8.6 Dominator Tree 简介

Dominator Tree 是这样的一棵树：一个有向图上，从一个点 S 出发到每个点，到任意一个点 T 的所有路径的共同点叫做 S 到 T 的必经点。离 T 的最近的必经点称为最近必经点 $dom[T]$。每个点以它的最近必经点作为父亲，连起来的树就是 Dominator Tree。

首先我们从原点 S 出发，DFS 一遍图，得到 DFS 序。$dfn[x]$ 表示 x 点的 DFS 序编号，$id[x]$ 表示 DFS 序编号为 x 的是哪个点。给出以下定义：

- 一个点 x 的祖先指它在搜索树中的祖先。
- 一个点 x 的半必经点 $semi[x]$ 表示点 x 的能不经过其它祖先到达的深度最浅的祖先。
- 一个点 x 的必经点 $dom[x]$ 已经定义过了。

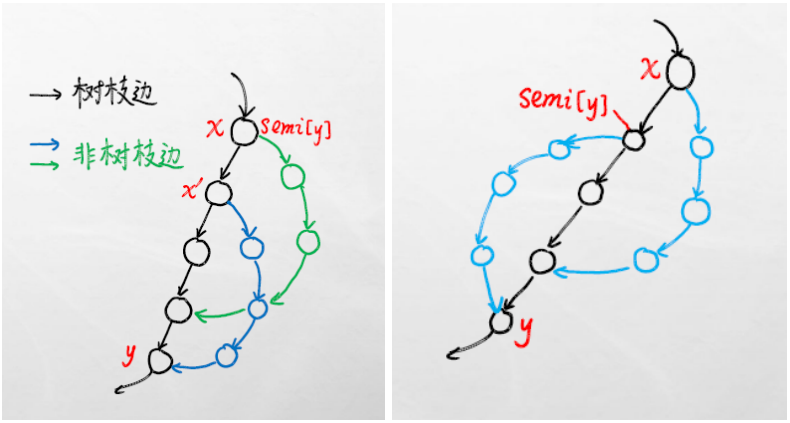

图 1: 左图：半必经点的定义；右图：半必经点不一定是必经点

半必经点定理：对于点 y 来说，考虑所有 $x \in pre[y]$：

- $dfn[x] < dfn[y]$：将点 x 加入考虑范围
- $dfn[x] > dfn[y]$：对于所有 x 的祖先 z，如果有 $dfn[z] > dfn[y]$，将 $semi[z]$ 加入考虑范围（这样能保证 $semi[z]$ 是 x 的一个祖先）

所有在考虑范围内的点中 dfn 值最小的就是点 y 的半必经点。

必经点定理：对于点 y 来说，考虑搜索树中 $semi[y]$ 到 y 的路径中除了端点外的所有点的 $semi$。

- 如果这些点中所有点的 $semi$ 都等于 $semi[y]$ 或者在 $semi[y]$ 之下的话，那么有 $dom[y] = semi[y]$
- 如果有点 z 的 $semi$ 在 $semi[y]$ 之上的话，那么如果 $dfn[semi[z]]$ 是满足条件的点中最小的，那么 $dom[y] = dom[z]$。

证明不会啊。

那么我们的算法流程是：

- DFS 序从大到小加入，用并查集维护点 x 到其祖先的路径上最小的 $dfn[semi[z]]$。从大到小可以保证满足半必经点的条件。
- 设当前加入点是 y，枚举所有 x 满足 $semi[x] = y$，用必经点定理判断 $dom[x]$ 是否等于 y。

- 枚举所有 $x \in pre[y]$，用半必经点定理判断 y 的 $semi$。
- 再由 DFS 序从小到大更新 dom。

知道了所有点的 dom，建树就很简单了。

9 Codechef BWGAME

满足下列条件的排列中：$\forall i \in [1,n], P_i \in [L_i, R_i]$，设逆序对数是偶数的排列是偶序列，奇序列类似。问偶序列多还是奇数列多。范围是 $1 \le n \le 10^5$

9.1 关键词

行列式，可并堆

9.2 题解

原题会比这个简化版题意更加直观。我们可以这样想：有一个矩阵 A，每一行的 L_i 到 R_i 列是 1，其他是 0。它的行列式就是满足题目所述条件的偶数列数减去奇数列数（直接用最简单的定义可以推得）。那么知道这个行列式的值就知道答案了。那么怎么快速地求这个矩阵行列式的值呢？

这个矩阵特殊在于它是 01 矩阵，而且一行内的 1 都挨在一起。一行可以用两个数 L_i, R_i 表示。我们用平常消成上三角矩阵的方法求行列式的值，消完之后对角线一乘就是行列式了。我们一列一列的消。如果枚举到当前行是 i，然后 $A_{i,j} = 0$ 的话要考虑换行，找到 $L_i = i$ 的行中 R_i 最小的那行与当前行交换，然后行列式变号。如果无论怎么换行都无法使得 $A_{i,j} = 1$，那么显然行列式是 0 了。然后考虑用当前行去消别的行，因为保证了当前行是所有相同 L_i 的行中 R_i 最小的，那么别的相同 L_i 的行消完长这个样子：$L_j = R_i + 1$，R_j 不变。一直消下去就可以了。

但问题在于怎么快速的消和在相同 L_i 的行中找到最小的 R_i。我们考虑搞 n 个可并堆。第 i 个堆存所有相同 L_i 的行的 R_i 的值。这样解决了第二个问题。对于第一个问题，如果当前消完了 i，直接把当前堆中剩下的元素并到第 $R_i + 1$ 个堆就好了。这样就不用显式地消了。

9.3 算法流程

- 构出 n 个堆
- 然后按照一列一列的消，消完之后并到其他堆。
- 统计对角线答案。

9.4 复杂度

$O(nlogn)$

10 Codechef LPARTY

10.1 题意

给出 N 个 01 变量的 M 种取值方案，已知这 M 种取值都是“坏”的。一个组定义为某几个变量的一种取值方案。比如一个组是 $x1 = 0, x2 = 1, x3 = 1$。那么假如一种取值方案满足了任意一个组，那么这个取值方案就是坏的，而且我们称这个组可以代表这个取值方案。定义基表示那些只要出现就一定能让方案变坏的组（注意我们必须从 M 个取值方案中推得）。现在我们想要找到一些基，使得这些基能够代表所有 M 中取值方案，基的大小就是它所包括变量的个数，我们希望找到大小和最小的解。范围是 $N \leq$

10.2 关键词

搜索

10.3 题解

题意很奇（gui）怪（chu）。先理清楚它的定义：一个组能是一个基的必要条件是在它出现的那些取值方案里，其它变量无论取什么值，这种取值方案都要出现在 M 个取值方案中。就是说我们要“证据充分”才能确定一个组是一个基。然后知道这个之后我们可以枚举所有可能的组，暴力判它可不可能是一个基。然后假如一个基存在另一个基能代表的取值方案能够包括它所代表的那些，那么我们显然取更优的那个。这样能够把可能有

用的基压在 32 个以内。然后就开始考虑搜索了（本来就是 NP）。各种优化都用上，位运算，最优性剪枝，合法性剪枝，以及记忆化，然后挤挤就过了。

细节上：一定要注意记忆化和别的优化的冲突。比如说：最优性剪枝之后有些状态实际没有拓展出去，然后却把这种状态记下来了，然后会导致记下了无用的状态，再次调用就调出了那个无用的状态。

10.4 算法流程

- 枚举所有可能的基，判断，去掉不会成为最优解的基
- 搜索

10.5 复杂度

上界是 $O(2^{3^N/N})$，然而实际远远达不到。

11 Codechef XRQRS

11.1 题意

有 M 个操作，$M \leq 5 * 10^5$，操作有五种类型。

- 在数列 A 最后加一个数 x。
- 询问 $[L, R]$ 中与 x 异或起来最大的 A_i。
- 删掉数列最后的 k 个数。
- 询问 $[L, R]$ 中小于等于 x 的数有多少个。
- 询问 $[L, R]$ 中第 k 小。

11.2 关键词

可持久化线段树，可持久化 trie

11.3 题解

询问最大的 xor，一般考虑二进制上按高位到低位找，把所有的数字把它的二进制用 trie 存下来，询问时在 trie 上走就可以了。最后两个询问都是线段树的基础操作了。询问 $[L,R]$ 区间，在末尾增减数之类的直接可持久化两个数据结构就可以了。

11.4 算法流程

按操作来做就可以了。

11.5 复杂度

$O(MlogN)$，N 是所有 x 的最大值

12 Codechef RANKA

12.1 题意

让你构造一种下围棋的方案，在 10000 步之内游戏不能结束。这里的围棋可以弃疗（不下子）。而且不能重复出现同一种游戏状态（状态包括当前是黑子下或是白子下，以及棋盘上棋子的状态）。

12.2 题解

奇葩的提答题。

一种简单的方法是我们不停塞满左上三角。给出定义：称黑子塞满的腰长为 K 的左上直角三角形为黑 K 三角。比如说我们想构造一个黑 K 三角，考虑先构一个白 $K-1$ 三角，然后沿着三角形的底边放黑子，白子一直不动，这样可以吃掉白 $K-1$ 三角。然后再构出黑 $K-1$ 三角。这样我们可以递归的做下去。

构出一个黑 K 三角之后，我们想要扩展新的状态。我们可以令白子放斜线，黑子弃疗来吃掉这个黑 K 三角，然后再构一个白 K 三角。这样我们就得到一个白 $K+1$ 三角了。这个过程其实与上面非常类似。这样我们就可以不重复的做下去。

当然局限于左上那个腰长为 9 的三角是做不到 10000 步的。但是我们只是要塞满左上边就可以了。所以我们要塞满一个 K 三角，且 $K > 9$ 时，实际上我们塞满的是腰长为 K 的三角被 9×9 棋盘所截的部分。所以这样我们又可以继续做下去了。这样做就能轻松的做到 10000 步。注意弃疗的时机，否则构造出来的方案会不合法。

12.3 算法流程

- 构造一个 K 三角。然后用一条 $K+1$ 斜线吃掉它。
- 在已经有了一条 $K+1$ 斜线的情况下构造一个 K 三角。得到一个 $K+1$ 三角。
- 重复这个过程。

12.4 复杂度

$O(n)$，没什么意义……

13 Codechef FNCS

13.1 题意

给出一个长度为 N 的数列 A，再给出 N 个函数，函数 $F[i] = \sum_{j=L[i]}^{R[i]} A[j]$，$L[i]$ 和 $R[i]$ 给定。再给出 M 个操作。第一种操作是令 $A[x]$ 修改为 y, 第二种是询问标号为 $[L, R]$ 区间内的函数值的和。范围是 $N, M \leq 10^5$。

13.2 关键词

分块，数据结构。

13.3 题解

每个函数可以写成前缀和的形式：$F[i] = S[R[i]] - S[L[i] - 1]$。我们对函数按标号分成 $\sqrt{N}$ 块，然后每一块内维护一个数组 F 和 G，$F[i]$ 表示组内所有函数 $S[i]$ 的系数和，G 是 F 的后缀和。

修改操作：我们可以把修改操作改成加法操作。对每个组保存一个变量 sum，表示当前组函数的总值。若当前修改操作是 $A[x] = A[x] + y$，等价于对于 $i \geq x$，都有 $S[i] = S[i] + y$。那么我们可以枚举每一个组，发现当前操作对该组函数的影响就是 $y * G[x]$，修改 sum 即可。每一个修改操作都是 $O(\sqrt{N})$ 的。

询问操作：由于分了段，那么一个区间 $[L, R]$ 可以被切成很多段和前面后面长度小于 $2 * \sqrt{N}$ 的部分。段的个数小于 $\sqrt{N}$，把它们的 sum 加起来就好了。剩下的就暴力询问每个函数的值，用树状数组或者线段树实现。每个询问操作是 $O(\sqrt{N}logN)$ 的。

13.4 算法流程

- 预处理出 F,G,sum。
- 分别对询问和修改按上述方法实现。

13.5 复杂度

$O(N\sqrt{N} + Q\sqrt{N}logN)$，有 $O((N + Q)\sqrt{N})$ 的算法。

14 Codechef LYRC

14.1 题意

给出 W 个单词，再给出 N 个文章片段，文章片段由一个个单词组成。求每个单词出现的次数。注意假如文章中某个单词的某个子串是一个给定单词，那么那个给定单词出现次数也要加 1。单词总长度不会超过 2500000，文章总长度不会超过 5000000。

14.2 关键词

AC 自动机

14.3 题解

对所有单词建立 AC 自动机。然后注意到一个节点的 fail 是当前节点的一个后缀。所以我们想要把一个单词的所有子串都打上标记的话，让它

在自动机上运行，标记每一个经过的节点。那么这些节点代表了该单词的所有前缀。然后再逆推回去，把一个点的标记传到它的 fail 去，我们就可以标记它的一个后缀。这样我们标记了一个单词的所有前缀的所有后缀，即标记了它所有的子串。这样答案就很容易统计了。

14.4 算法流程

- 对所有单词建立 AC 自动机，记录代表它的节点是哪一个。
- 对文章中的单词，让它在自动机上运行，标记经过的点。
- 按逆 BFS 序把标记上传，得到所有的标记。
- 最后对所有单词，输出代表它的节点的标记。

14.5 复杂度

$O(\sigma L)$，L 表示所有字符串长度总和。σ 是字符集大小。

15 Codechef QTREE6

15.1 题意

一棵有 N 个节点的树，一开始每个节点都是黑色的。有 M 个操作，操作分两种：一是修改任意一个节点的颜色（黑白翻转），二是询问某一个节点所在联通块的大小，两个节点联通指它们之间路径的所有点颜色一样。范围是 $N, M \leq 10^5$。

15.2 关键词

Link/Cut Tree，链剖

15.3 题解

这个题的关键点在于如何把两种操作变成这个点到根路径上的一些操作，这样就能套上 LCT 或者是链剖了。我们可以定义 $B[x]$ 和 $W[x]$ 表示如果当前点是黑点或者白点时，以它为根的子树中和它一起的联通块大小。

对于询问操作，我们可以沿着当前点一直向上找到和它联通的最上面的节点，然后直接输出答案就可以了。

对于修改操作：假设当前是点 x 由黑变白。先找到当前点一直向上找到和它联通的最上面的节点，然后令 y 是这个最上面节点的父亲（因为我们会假设这个点是黑点来计算它的 B 值），然后对 x 到 y 路径上的 $B[i]$ 减去 $B[x]$。将 x 点变白色后类似于上面，找到 y 后将 x 到 y 路径上的 $W[i]$ 加上 $W[x]$ 即可。这样的路径区间加减用 LCT 或者链剖都可以实现。

还有一个问题就是怎样快速的找到“和一个点联通的最上面的点”，我们可以在 LCT 上或者是链剖中的数据结构中维护区间的黑点数和，然后在 splay 上走或者是二分一下都可以。

15.4 算法流程

以 LCT 为例：

- 读入后遍历树，把 LCT 建起来。
- 按上述方法修改或询问即可。

15.5 复杂度

LCT 是 $O(nlogn)$ 的，链剖多个 log。

16 Codechef SEAORD

16.1 题目大意

有 N 个程序，每个程序要在第一台机器上跑 A_i 时间，在第二台机器上跑 B_i 时间，先后均可。一台机器不可以同时运行两台机器，一个程序也不可以同时在两台机器上跑。求最优的安排方案使得所有程序完成的时间最少。$N \leq 10^4$，多组测试数据的 N 总和不会超过 200000。

16.2 关键词

贪心，玄学

16.3 题解

假如没有空隙地塞满，那么答案应该是 $max(\sum A, \sum B)$，不妨设答案就是 $\sum A$。假如我们已经知道了两台机器分别跑程序的序列表，那么我们把 A 安排后，要安排 B 时，贪心地把程序塞进去，如果有冲突就往后挪。然后这样很容易就得到了整个方案。假如第二台机器上的总时间小于等于第一台的话那么显然当前答案就取第一台完成时间，因为第一台是塞满的，所以肯定得到了最优解。否则我们就换一个序列重新做一次。由于玄学，有很多个合法解，所以我们随机序列就可以了。

有一种特殊情况：若某一个 $A_i + B_i > max(\sum A, \sum B)$，那么不可能无空隙地塞满。答案应该是 $A_i + B_i$。而且安排好 i 之后其它是随便塞的。

16.4 算法流程

- 判断特殊情况
- 随机序列，贪心求解，直到得到最优解

16.5 复杂度

$O(NM)$,M 是期望随机次数（玄学）。

17 Codechef TREECNT2

17.1 题意

一棵 N 个节点的树，边上有权值。求无序点对 (x, y) 满足 x 到 y 上的路径的边权的 gcd 是 1。还要支持 M 个修改，每次修改更改一条边的权值，然后再次询问答案。范围是：$N \leq 10^5, M \leq 10^6$，权值不大于 10^6。

17.2 关键词

预处理，反演

17.3 题解

先不考虑修改。

看到“gcd=1”这个条件，可以想到用反演。下面的 e 是单位函数，当且仅当 $x=1$ 时 $e(x)=1$，否则为 0。然后最常用的反演如下：

$$\begin{aligned} & e(gcd(a,b)) \\ =& \sum \mu(d), d|a, d|b \end{aligned}$$

在本题中我们是路径上的边权 gcd 起来。那么我们考虑反演，然后枚举 d，看看 $\mu(d)$ 出现了多少次，加到答案里边就好了。那么当前 $\mu(d)$ 出现的次数，等价于统计这样的路径数：路径上所有边权都是 d 的倍数。那么我们只考虑边权是 d 倍数的边所构成的子图：$\mu(d)$ 出现的次数就是这个子图里所有的联通块内的点对数量。我们可以把当前有用的边加进去然后用并查集之类的维护下联通块的大小就好了。又因为一个整数的因子不会很多，所以一条边只会出现较少次。枚举所有的 d 就可以算出答案了。

考虑修改。

首先一种简单方法是：考虑到对答案有影响的是经过当前修改边的路径，我们枚举从当前更改的那条边两端点走出去的路径，一边为 A 集，另一边为 B 集，用桶的形式存下来。那么答案就是在 A 集中枚举一个 d，在 B 集中枚举一个 d'，$gcd(d,d')=1$，然后乘一下次数加到答案里面。因为 d 和 d' 都是当前边权的因子，所以有用的 d,d' 不会有很多。

但是上面那种方法要 $O(QNlogC)$，比较危险。注意到 Q 不大，而且上面的方法没有用到反演的性质。先定义 $G(x)$ 表示只有 x 倍数的边组成的图。我们可以这样想：我们在处理第一次答案时，我们先不考虑被修改过的边，把其它所有边先加进去，用并查集维护。那么等到询问时再把这些边补上。直接搞好像会 T，但是我们看，对答案有影响的只有：$G(x), x|d$ 或者 $G(x), x|d'$，d,d' 分别是新旧边权。那么我们可以只预处理出所有可能被修改的子图 G'，这些图都是“残缺的”，即被修改边没有被加入。那么询问时只需要对可能更改的 G' 重新把被修改边加进去统计答案就好了。还有一个小问题是空间问题，我们没办法把一个图的所有状态存下来。但是因为重新加入的边只有 100 条，我们只需要存下这些边的端点所属的联通块的信息即可。

17.4 算法流程

- 预处理出 μ。
- 预处理所有 $G(x)$

- 预处理所有有可能被修改的 $G'(x)$
- 对于每个询问，把对答案有影响的 $G'(x)$ 补全，统计答案。

17.5 复杂度

$O(C + K(N + Q^2))$，C 是权值上限，K 是最大的因子数。

18 Codechef QRECT

18.1 题意

给出 N 个操作。操作有以下三种

- 插入一个矩形，用左下角坐标 $(x1, y1)$ 和右上角坐标 $(x2, y2)$ 表示。
- 删除第 i 号矩形，保证第 i 号矩形已经被插入且没有被删除
- 询问当前有多少个与矩形 $(x1, y1)$，$(x2, y2)$ 有至少一个共同点。

范围是 $N \leq 10^5$，坐标范围在 $[1, 10^9]$。

18.2 关键词

二维数据结构，CDQ 分治，容斥原理

18.3 题解

先看一维的问题：插入和询问均是一条数轴上的线段，用左端点 L 和右端点 R 表示。询问时，直接统计线段相交是不好搞的。正难则反，我们考虑用总数减去不相交的线段。线段 A 和 B 不相交的充要条件是 $R_A < L_B$ 或 $R_B < L_A$。所以我们直接统计有多少条线段的右端点在询问线段的左端点的左边，有多少条线段的左端点在询问线段的右端点的右边，用总数减去即可。这样我们用一些简单的一维数据结构，比如树状数组就可以解决了。

回到二维上的问题。我们发现两个矩形相交的充要条件是他们投影到 X 轴和 Y 轴上的线段均相交。那我们就可以把二维问题转化成两个一维问题求解啦。用总数减去在 X 轴上不相交的，和 Y 轴上不相交的。但是

注意，这里有个问题是假如某矩形与询问矩形在两个轴上都不相交，它会被减去两次，我们要再把它们的数量加回去。要统计这类矩形，我们要统计四个方向的类似下面这样的矩形：左下角在询问矩形的右上角的右上方（其它三个方向类似）。这部分就是个在二维平面上插入，删除点，再询问某个矩形范围内点数的问题。这个问题常规上可以用二维数据结构，比如说树套树或者是分块加数据结构之类的实现。然而这些方法的空间上都非常吃紧（常常需要 1G 以上的内存），而且常数巨大。那么我们考虑其他方法。

用 CDQ 分治是比较优秀的一种做法。CDQ 分治的前提条件是只有插入和询问，且插入操作之间是独立的。核心是把题转化成先全部插入，后询问的问题。那么这题里面的删去一个矩形，相当于插入一个负权的矩形，那么就可以用 CDQ 分治了。我们按时间分治，设当前处理区间 $[l,r]$，设区间中点是 mid。我们只考虑区间 $[l,mid]$ 的插入对区间 $[mid+1,r]$ 的询问的贡献。在这题里面，我们可以把询问变成前缀询问，然后用扫描线从左到右扫一遍，遇到插入点就在它的 y 坐标处打上标记，遇到询问点就询问 $[1,y]$ 区间上的标记数即可（因为保证了插入的点都在它的左边）。处理完当前区间，就分治下去处理 $[l,mid]$ 和 $[mid+1,r]$ 的问题。所以我们可以对四个方向都做一次分治就可以把答案求出来了。

18.4 算法流程

- 读入，离散化坐标
- 先处理掉两个方向上的一维问题
- 再对四个方向上的二维问题 CDQ 分治
- 统计答案

18.5 复杂度

$O(Nlog^2N)$

19 Codechef RNG

19.1 题意

$A_i = (C_1A_{i-1}+C_2A_{i-2}+C_3A_{i-3}+\cdots+C_kA_{i-k})$，给出 $A_1, A_2, \ldots, A_k$，求 A_N。范围是 $N \leq 10^{18}$，$k \leq 30000$。

19.2 关键词

数学推导，多项式各种运算

19.3 题解

这是经典的 k 项线性齐次线性递推数列问题。

19.3.1 如何构造一个 k 项线性递推数列的生成函数

令 $A(x) = A_0 + A_1x + A_2x^2 + A_3x^3 + \ldots$

$$
\begin{array}{rcccccccccccc}
A(x) & = & A_0 & + & A_1x & + & A_2x^2 & +\cdots & + & A_kx^k & + & A_{k+1}x^{k+1} & +\cdots \\
C_1xA(x) & = & & & C_1A_0x & + & C_1A_1x^2 & +\cdots & + & C_1A_{k-1}x^k & + & C_1A_kx^{k+1} & +\cdots \\
C_2x^2A(x) & = & & & & & C_2A_0x^2 & +\cdots & + & C_2A_{k-2}x^k & + & C_2A_{k-1}x^{k+1} & +\cdots \\
C_3x^3A(x) & = & & & & & & \cdots & + & C_3A_{k-3}x^k & + & C_3A_{k-2}x^{k+1} & +\cdots \\
\cdots & & & & & & & & & \cdots & & \cdots & \\
C_kx^kA(x) & = & & & & & & & & C_kA_0x^k & + & C_kA_1x^{k+1} & +\cdots
\end{array}
$$

我们用第一条式子减去别的。可以得到

$$(1 - C_1x - C_2x^2 - C_3x^3 - \cdots - C_kx^k)A(x) = P(x)$$

$P(x)$ 是某个次数小于 k 的多项式。定义 $Q(x)$：

$$Q(x) = 1 - C_1x - C_2x^2 - C_3x^3 - \cdots - C_kx^k$$

那么可得

$$A(x) = \frac{P(x)}{Q(x)}$$

然后我们尝试理解这个式子的意义。它告诉我们，任意一个 k 项线性递推数列都有一个有理生成函数 $\frac{P(x)}{Q(x)}$，且有以下性质：

- $Q(x)$ 的次数为 k，而且包含了所有的递推系数。而且常数项为 1。
- $P(x)$ 的次数小于 k。

反过来，我们可以发现任意一个这样的生成函数 $\frac{P(x)}{Q(x)}$，满足 $\deg P < \deg Q$，以及 $Q(0) = 1$，一定是一个 k 项线性递推数列的生成函数。

19.3.2 关于特征多项式的一些证明

说在前面，本着让大家看懂的原则，这里参照的是 RNG 的官方题解，因为它没有使用线性代数的知识，而是用了更简单的语言叙述。遗憾的是它的证明是错的。所以大家假如想找正确的证明请参见郭晓旭的论文《线性递推关系与矩阵乘法》。如果对错误的证明不感兴趣可以直接跳到结论部分。

我们先把 $Q(x) = 1 - C_1x - C_2x^2 - \cdots - C_kx^k$ 分解因式：$Q(x) = (1 - r_1x)(1 - r_2x)\cdots(1 - r_kx)$。简单起见，我们先考虑 r_i 是互不相同的，而且均不为零（其他情况证明要更复杂）。然后我们可以对 $\frac{P(x)}{Q(x)}$ 进行有理分式分解，如下：。

$$\frac{P(x)}{Q(x)} = \frac{P(x)}{(1 - r_1x)(1 - r_2x)\cdots(1 - r_kx)} = \frac{c_1}{1 - r_1x} + \frac{c_2}{1 - r_2x} + \cdots + \frac{c_k}{1 - r_kx}$$

这里有些常数 c_i，我们不用管他们的值，只需要知道这个式子可以这样分解就可以了。接下来，考虑每一个简单的分式 $\frac{c_t}{1-r_tx}$。因为

$$\frac{1}{1 - y} = 1 + y + y^2 + y^3 + \cdots$$

因此我们有

$$\frac{c_t}{1 - r_tx} = c_t + c_tr_tx + c_tr_t^2x^2 + c_tr_t^3x^3 + \cdots$$

这样我们可以得到

$$\frac{P(x)}{Q(x)} = \frac{c_1}{1 - r_1x} + \frac{c_2}{1 - r_2x} + \cdots + \frac{c_k}{1 - r_kx}$$

然后看它的第 i 项系数，即 A_i，等于每个分式的第 i 项系数相加：

$$A_i = c_1r_1^i + c_2r_2^i + c_3r_3^i + \cdots + c_kr_k^i$$

然后考虑特征多项式 $M(x)$，定义为：

$$M(x) = x^k - C_1x^{k-1} - C_2x^{k-2} - \cdots - C_k$$

注意 $M(x)=x^kQ(1/x)$，类似于把 Q 的系数反过来就是 M 了。

我们发现 $M(x)$ 这个多项式的零点同样是 $r_1,r_2,\dots r_k$。

这个有相同零点显然是错的 (来自 RNG 的题解，就是说这篇题解给的证明是错的。正确的证明请参见线代)

考虑任意一个能被 $M(x)$ 整除的多项式。那么 $r_1,r_2,\dots r_k$ 也一定是它的零点。我们令 $M(x)$ 表示为：

$$M'(x)=m_lx^l+m_{l-1}x^{l-1}+m_{l-2}x^{l-2}+\cdots+m_0$$

因为 r_t 是一个零点，我们有：

$$0=m_lr_t^l+m_{l-1}r_t^{l-1}+m_{l-2}r_t^{l-2}+\cdots+m_0$$

乘上 c_t 得：

$$0=m_lc_tr_t^l+m_{l-1}c_tr_t^{l-1}+m_{l-2}c_tr_t^{l-2}+\cdots+m_0c_t$$

把 t 取 1 到 k 的式子加起来，可由上面 A_i 写成 c_t 和 r_t^i 的和的式子得

$$0=m_lA_l+m_{l-1}A_{l-1}+m_{l-2}A_{l-2}+\cdots+m_0A_0$$

我们得到了我们的结论。

19.3.3 结论

如果多项式

$$m_lx^l+m_{l-1}x^{l-1}+m_{l-2}x^{l-2}+\cdots+m_0$$

能被

$$M(x)=x^k-C_1x^{k-1}-C_2x^{k-2}-\cdots-C_k$$

整除，那我们有

$$m_lA_l+m_{l-1}A_{l-1}+m_{l-2}A_{l-2}+\cdots+m_0=0$$

它的确是对的。

19.3.4 算法

考虑多项式 $S(x) = x^N \bmod M(x)$ 。因为 $\deg M = k$，所以 $S(x)$ 的次数小于 k。我们展开 $S(x)$：

$$S(x) = s_{k-1}x^{k-1} + s_{k-2}x^{k-2} + \cdots + s_1x + s_0$$

我们知道 $x^N \equiv S(x) \pmod{M(x)}$，换句话说 $M(x)$ 可以整除

$$x^N - S(x) = x^N - s_{k-1}x^{k-1} - s_{k-2}x^{k-2} - \cdots - s_1x - s_0$$

利用上面的结论，我们有

$$A_N - s_{k-1}A_{k-1} - s_{k-2}A_{k-2} - \cdots - s_1A_1 - s_0A_0 = 0$$

或者说

$$A_N = s_{k-1}A_{k-1} + s_{k-2}A_{k-2} + \cdots + s_1A_1 + s_0A_0$$

这个式子直接告诉我们怎么算 A_n 了！然后我们只需要把 $S(x) = x^N \bmod M(x)$ 算出来就好了。用快速幂就可以了。

注意，计算 $S(x)$ 时会有多项式乘法以及多项式除法（取模）。暴力做是 $O(K^2)$ 的，我们可以使用 FFT 优化，那么复杂度是 $O(KlogK)$ 的。暴力取模的话可以类似于长除法那样做，或者说类似于高精度除低精度的除法。

总时间复杂度是 $O(K^2logn)$ 或者是 $O(KlogKlogn)$ 的。

19.4 算法流程

- 得到特征多项式 $M(x)$
- 计算 $x^N \bmod M(x)$
- 最后与 $A_1, A_2, \ldots, A_k$ 对应相乘得到答案。

19.5 复杂度

$O(KlogKlogN)$

19.6 多项式逆元与多项式整除

前置知识：多项式乘法，FFT。下面的思路来自 Picks 的博客。

19.6.1 定义

常规的多项式相除 $P(x)/Q(x)$ 可能除出来有无数个负次幂项。我们这里叙述的多项式是没有负次幂项的有限项的多项式。且这里定义它的所有系数都是在模某个质数 P 意义下的。（可以类比实数除法和整数的高精度除法来配合理解）

定义多项式的整除：我们有四个多项式 $A(x), B(x), D(x), R(x)$。

- $\deg A = n(n > 0)$
- $\deg B = m(0 < m \leq n)$
- $\deg D = n - m$
- $\deg R < m$
- $A(x) = B(x)D(x) + R(x)$

可以证明，当 $A(x)$ 和 $B(x)$ 给定时，$D(x), R(x)$ 存在且唯一（类比高精度除法）。我们称 $D(x)$ 是 $A(x)$ 整除 $B(x)$ 的商，$R(x)$ 是 $A(x)$ 整除 $B(x)$ 的余多项式。即：

$$A(x) \operatorname{div} B(x) = D(x)$$

$$A(x) \bmod B(x) = R(x)$$

$$A(x) \equiv R(x) \pmod{B(x)}$$

19.6.2 求多项式逆元

定义：若有 $A(x)B(x) \equiv 1 \pmod{x^N}$，我们称多项式 $B(x)$ 是 $A(x)$ 在模 x^N 意义下的逆元。注意如果把 x^N 替换成一个一般的多项式逆元不一定存在。

下面我们讲怎么求给定多项式 $A(x)$ 的逆元 $B(x)$。

如果是在 $\bmod x^1$ 意义下，那么 $B(x)$ 只有一个常数项，且它是 $A(x)$ 常数项在模 P 意义下的逆元。可以用快速幂或扩展欧几里得算法求。

设我们当前已经得到 $\bmod x^t$ 意义下的答案 $B_0(x)$。要求 $\bmod x^2t$ 意义下我们有：

$$A(x)B_0(x) \equiv 1(\bmod x^t)$$

$$A(x)B(x) \equiv 1(\bmod x^t)$$

可得到

$$B(x) - B_0(x) \equiv 0(\bmod x^t)$$

两边平方得

$$B^2(x) - 2 * B(x)B_0(x) + B_0^2(x) \equiv 0(\bmod x^{2t})$$

注意后面的 x^t 变成了 x^{2t}。这是因为平方之后次数会翻倍。

接下来移项得

$$B^2(x) \equiv 2 * B(x)B_0(x) - B_0^2(x)(\bmod x^{2t})$$

$$A(x)B^2(x) \equiv 2 * A(x)B(x)B_0(x) - A(x)B_0^2(x)(\bmod x^{2t})$$

注意 $A(x)B(x) \equiv 1(\bmod x^{2t})$，消掉之后得

$$B(x) \equiv 2B_0(x) - A(x)B_0^2(x)$$

我们发现右边的所有多项式都是已知的，右边是我们要求的答案 $B(x)$。那么我们不停倍增直到次数比 N 大为止。然后取次数比 N 小的项即可。因为如果次数比 N 大的时候的逆元同样满足在模 x^N 时候逆元的条件。这里多项式乘法用 FFT 实现即可。那么时间复杂度是 $O(NlogN)$ 的。

19.6.3 求多项式整除的商和余多项式

有了逆元我们就可以求多项式整除的商和余多项式了。

我们同样用定义中的四个多项式 $A(x), B(x), D(x), R(x)$，已知 $A(x)$，$B(x)$，求 $D(x)$,$R(x)$。且它们的次数都和定义里面一样。

一种 naive 的想法是 $D(x) = A(x) * B^{-1}(x)$，然后 $R(x) = A(x) - D(x)B(x)$。但是问题在于 $B^{-1}(x)$ 根本没有定义！（因为这里我们并没有把它们放在模系中。）我们需要更靠谱的方法。

如果一个多项式 $S(x)$ 的次数是 N，那么我们看看 $x^N S(1/x)$ 这个多项式。直观上，后者的系数可由前者系数“前后翻转”得到。我们令翻转式 $S^r(x) = x^N S(1/x)$。

开始推导：

$$A(x) = B(x)D(x) + R(x)$$
$$x^n A(1/x) = x^m B(1/x)x^{n-m} D(1/x) + x^n R(1/x)$$
$$A^r(x) = B^r(x)D^r(x) + x^n R(1/x)$$

我们把上面的式子放到模 x^{n-m+1} 的模系中，那么 $x^n R(1/x) \equiv 0(\bmod x^{n-m+1})$。

$$A^r(x) \equiv B^r(x)D^r(x)(\bmod x^{n-m+1})$$
$$D^r(x) \equiv A^r(x)[B^r(x)]^{-1}(\bmod x^{n-m+1})$$

注意，这里 $[B^r(x)]^{-1}$ 是有定义的！因为我们已经将它放在模系当中了。然后由前面的方法，我们可以求出 $[B^r(x)]^{-1}$。观察式子，右边的多项式式全部是已知多项式的翻转式或是可通过求逆元求得，左边的是我们要求的多项式的翻转式。我们直接计算这条式子，就可以求得 $D(x)$ 了！而且因为 $\deg D = n - m$，它在模 x^{n-m+1} 下不变！

求得 $D(x)$ 之后，直接代进原始的式子可得 $R(x) = A(x) - B(x)D(x)$。

这样我们用几次翻转和多项式乘法，以及一次求逆元的过程就可以求出多项式整除的商和余多项式了。总时间复杂度 $O(nlogn)$。常数非常巨大。

至于算法实现请大家参照本题的各种程序，这题里面的程序都加上了各种常数优化（然而还是要跑 10s）。

20 Codechef SIGFIB

20.1 题意

求 $\sum 6xyz \cdot fib[x] \cdot fib[y] \cdot fib[z]$，$x + y + z = n$。对 m 取模。范围：不超过 500000 组数据，$n \leq 10^{18}$，$\sum m \leq 10^6$。

20.2 关键词

生成函数，数学推导

20.3 题解

官方题解比较奇怪没有看懂，下面介绍的是生成函数的方法（感谢策爷）。

我们设数列 $F[i] = i \cdot fib[i]$。它的生成函数 $F(x)$ 可以通过 fib 数列的生成函数求个导再乘 x 得到。

$$F(x) = \frac{x(1+x^2)}{(1-x-x^2)^2}$$

那么答案显然是 F 这个多项式的 3 次方的第 n 项系数乘 6。

$$ans = 6[x^n]\frac{x^3(1+x^2)^3}{(1-x-x^2)^6}$$

我们着重看后面那个分式。其实它也是某一个数列的生成函数。它下面展开胡是个 12 次的式子，那么它其实代表了一个 12 项线性递推数列，所以答案就是这个数列的第 N 项。我们就可以把它的特征多项式搞出来，用经典的 $O(k^2logn)$ 的方法做。因为模数之和小于 10^6，而数据组数又很多，我们可以预处理处理模数小于 200 的循环节，因为模数小，循环节不长，直接全部存下来就好了。

20.4 算法流程

- 预处理处理模数小于 200 的。
- 读入，分模数大的和模数小的做。

20.5 时间复杂度

$O(Tk^2logn)$

21 Codechef DEVLOCK

21.1 题意

求 N 位的十位数（可有前导 0）中，被 P 整除，且各数位加起来等于 M 的有多少个。询问对于 M 取 0 到 MM 的答案。范围是 $N \leq 10^9, P \leq 50, MM \leq 15000$.

21.2 关键词

分治，DP，FFT

21.3 题解

显然可以考虑数位 DP。设 $F[i][j][k]$ 表示 i 位的十位数中，数字模 P 得 j，各数位加起来是 k 的个数。然后 naive 的想法是直接枚举最高位转移。但是 N 很大。

所以我们考虑分治。假如我们知道 $F[n]$ 了，那么我们就可以推 $F[2n]$ 了。因为前一半数字和后一半数字是类似的，只是后一半数字对模 P 那一维的贡献要乘上 10^n 后再模 P。实际上调整一下第二维即可。设调整之后的是 $G[n]$。转移方程可以写成：

$$F[2n][i][j] = \sum F[n][a][b] * G[n][c][d], a + c \equiv i (\mathrm{mod} P), b + d = j$$

假如我们忽略掉 $\mathrm{mod} P$ 的影响，那么方程可写成：

$$F[2n][i][j] = \sum F[n][a][b] * G[n][c][d], a + c = i, b + d = j$$

直接暴力当然是不行的。我们发现这个式子是“二维”卷积的形式，那么我们可以枚举第二维（因为第二维很小），然后把所有的 $F[n][i]$ 和 $G[n][i]$ 用 DFT 弄出点值，枚举第二维，第三维直接对应相乘即可。然后得到 $F[2n]$ 之后在 IDFT 回去即可。

有了这种倍增的方法我们就可以分治的做了（类似于快速幂）。当 n 是奇数时暴力转移最后一位即可。

21.4 算法流程

- 当前要求 $F[n]$
- 先求 $F[n/2]$
- 倍增 $F[n/2]$ 得到 $F[n]$，若 n 是奇数则再暴力转移一位

21.5 复杂度

$O(P^2 MlogN + PMlogMlogN)$

22 Codechef PRIMEDST

22.1 题意

给一棵树，求长度为质数的路径数。点数小于 50000。

22.2 关键词

分治，FFT

22.3 题解

用点分治。然后每次统计经过一个点的路径数时，暴力合并是 $O(n^2)$ 的，因为它是卷积的形式，用 FFT 就好了。注意在做一个点为根的子树时，最好用总的 cnt 数组平方后，减去这个点的下面的子树的 cnt 的平方。这样能保证复杂度。还有就是注意算重的问题。

22.4 算法流程

- 点分治
- 对每个点，先把它的子树里的所有深度保存为 cnt，然后卷积（相当于枚举两个深度算答案）
- 对这个点的每个子树，把这棵子树的所有深度保存为 cnt'，卷积后从答案减掉。
- 分治到每个子树。

22.5 复杂度

$O(nlog^2n)$

23 Codechef GERALD08

23.1 题意

有一棵 n 个节点的树，树上每一条边的颜色都为红色或者蓝色。现在两个人轮流进行操作，第一个人每次选择一条红色边删除，第二个人每次

选择一条蓝色边删除，删除后和树根（1 号点）不连通的部分将被删除，若干轮之后不能操作的人算输。

如果两个人都使用最优策略，问第一个人先手时、第二个人先手时分别是谁赢得游戏。

范围是 $n \leq 10^5$

23.2 关键词

博弈，函数，高精度，启发式合并

23.3 题解

这题用常规的 SG 函数不好做。我们定义一种新的函数 $F(x)$ 表示以 x 的子树的这个局面对第一个人（选红边）的有利程度。假如 $F(x)$ 是负数，那么它就表示这个局面对第二个人更有利。

首先看 $F(x)$ 的意义。当 $F(x) > 0$ 时，第一个人赢。当 $F(x) = 0$ 时，后手一定赢。当 $F(x) < 0$ 时，第二个人赢。而且 $F(x)$ 的意思其实是获胜哪一方至多可以“走错”多少步，相当于他的优势有多大。

首先看一些简单局面的 $F(x)$。一条长度为 x 红色的链，它的 F 值是 $+x$。如果是蓝色则是 $-x$。

然后我们如何计算一个更复杂的局面呢？我们发现，如果是第一个人操作之后，权值一定会变小（局面变得对他更不利），第二个人同理。而且他们显然会选择操作之后不利程度最小的那个操作。设第一个人操作之后得到的权值是 P，第二个人操作之后得到的权值是 Q。显然有 $P \leq Q$。所以我们衡量当前局面的权值应该 P, Q 之间。所以我们的暴力方法应该是：首先暴力把当前局面的所有子局面的权值计算出来，然后从中找出 P 和 Q。然后我们选择 P 和 Q 之间的“最简单的”数，让 $F(x)$ 等于它。这里最简单的数的定义是 Surreal Number。详细的内容请参考论文。当然，除了暴力之外，我们还可以找规律（当然这个规律是可以证明的）！

通过找规律，我们得到一种计算方法：对一个节点 x，枚举它的所有子节点 y，如果如果连接他们的是红边，令 p 为最小的正整数满足 $F(y) + p > 1$，那么 y 对 $F(x)$ 的贡献就是 $\frac{F(y)+p}{2^{p-1}}$。如果连接他们的是蓝边，令 p 为最小的正整数满足 $F(y) - p < -1$，那么 y 对 $F(x)$ 的贡献就是 $\frac{F(y)-p}{2^{p-1}}$。

然后我们算出 $F(1)$ 即可。

然而我们发现这里的 F 值可能要用到高精度。我们这样实现：将一个实数 R 表示成整数部分 I，一个整数 w，一个数组 A_i。那么 $R = I + \sum 2^{A_i+w}$。我们主要应用到两个操作：加法，除以 2^x。

加法的实现是：先整数位相加，然后我们把数组 A_i 合并起来。如果有两个相同的 A_i，我们就做类似加法的进位即可，注意 w 的影响。为了保证复杂度，我们可以使用启发式合并，然后用 set 实现数组 A_i。

除以 2^x 的实现是：我们先把 w 减去 x，这样就不用对数组内的数进行修改了。然后将整数位的后 x 位扔进去。整数位最后除以 2^x 即可。注意当整数位是负数时方法稍微不同。

这个题最好看看这个链接：关于 Hackenbush 游戏的详细说明

23.4 算法流程

- 代公式算 $F(1)$
- 用 $F(1)$ 判断答案。

23.5 复杂度

$O(nlog^2n)$

24 Codechef FIBTREE

24.1 题意

维护一颗 N 个节点的树，支持以下操作：

- 给 x 到 y 路径上的第一个点的权值加上 $fib[1]$，第二个点加上 $fib[2]$，以此类推。
- 询问 x 到 y 路径上的点权和。
- 询问以 x 为根时，y 为根的子树的点权和。
- 令树上的状态返回第 R 次操作之后的状态。

所有答案模 $10^9 + 9$。范围是 $N, Q \leq 10^5$。

24.2 关键词

树链剖分，可持久化数据结构

24.3 题解

由 fib 数列的通项公式可得：$fib[i]$ 可表示为 $fib[i] = C(A^i - B^i)$，A, B, C 均是常数。又由于在模 $10^9 + 9$ 下操作，A, B, C 均可以用整数表示。

然后问题就很好解决了：我们对树进行轻重链剖分。然后对树 DFS 一遍，保证先走重边，这样一条重链的 DFS 序是连续的，一棵子树的 DFS 序也是连续的。我们可以按 DFS 序维护线段树。

我们对不同的操作分开来看：

- 修改操作：我们分成从 x 到 LCA 的操作和从 LCA 到 y 的操作。操作可以简化成对一条链加上一个等比数列。那么我们线段树上一个区间维护的就是恰好覆盖该区间的等比数列的次数。注意一个加上大的 N 项的从 1 开始的等比数列，可以拆成加上前面 P 项的等比数列，和加上 k^P 次 $N - P$ 项从 1 开始的等比数列（k 是公比）。然后按树链向上跳，维护一下即可。
- 询问路径和：按树链向上跳即可。
- 询问子树和：我们不妨以 1 为根。按 x 是否属于 y 的子树分情况讨论下即可。然后直接询问区间和即可。
- 回滚操作：实现线段树时可持久化即可。注意，这样做后标记可以使用永久化标记，这样就不用下传了。

24.4 算法流程

- 轻重链剖分
- 按上述方法询问和修改

24.5 复杂度

$O(Nlog^2N)$

25 Codechef LMATRIX3

25.1 题意

现在有 m 个数组和一个整数 P，第 i 个数组用 B_i 表示，由四个整数 F_i, C_i, D_i, L_i 生成。所有数组由以下方式生成：

- 令 $B_{i,j}$ 表示第 i 个数组的第 j 项。
- $B_{i,1} = F_i$
- 对于每个 $1 < j \leq L_i$，有 $B_{i,j} = (B_{i,j-1} * C_i + D_i) \bmod P$

然后我们将 B_i 数组按编号从小到大排在一起得到数组 A。设串联起来的数组长度是 N。然后可以执行若干操作，每个操作是这样的形式：

- 一个操作由三元组 s, t, k 表示。$1 \leq s \leq t \leq N, 0 \leq k \leq P$
- 然后对于所有 $s \leq i \leq t$，令 $A_i = (A_i + k) \bmod P$。

问最少需要多少次操作，可以使整个数组 A 变成 0。

范围是 $N \leq 10^{17}$,$m \leq 100$,$1 \leq P \leq 10$。

25.2 关键词

找规律，搜索，DP

25.3 题解

25.3.1 转换模型

我们首先要对数组差分。令 $A_0 = A_{N+1} = 0$。对于 $1 \leq i \leq N+1$，$B_i = (A_i - A_{i-1})$。我们可以将问题转化成：求 B 数组最多能分成多少个子序列（不一定连续），每个子序列中的数加起来模 P 为 0。

可以这样理解：

首先，B 数组全为 0 等价于 A 数组也全为 0。

不妨设已经分好了子序列。我们把同一个子序列放到一起看，设这个子序列在 B 中的位置编号为 $C_1, C_2, C_3, \cdots, C_k$。然后我们对于 $1 \leq i < k$，我们可以用一次操作 $(C_i, C_{i+1} - 1, -B_{C_i} \bmod P)$ 将 B_{C_i} 变成 0。然后由于 B 是原数组的差分，除了 B_{C_i} 变成 0 外，还有 $B_{C_{i+1}} = B_{C_{i+1}} + B_{C_i}$。一

直做到最后，由于一个子序列中的数加起来模 P 是 0，最后的 B_{C_k} 一定能和 $B_{C_{k-1}}$ 一起变成 0。

这样一个子序列只需要用 $k-1$ 次操作即可将它全部变成 0。可以类似于分出一个子序列，就可以“减少”一次不必要的操作。那么总的操作数就是 $N+1$ 减去分出子序列的个数。显然子序列个数越多答案越优。

25.3.2 计算答案

首先，由于 P 非常小，用循环节的性质可以轻松得到每个子数组。然后由于我们找的是子序列，数字的具体位置就没那么重要了。所以我们可以用桶把差分数组中每个数出现次数统计出来。

不妨设当前的 P 是 10。其他情况更加简单。

然后，0 显然是不用考虑的。而 19,28,37,46,55 这样的组合是最优的。我们直接把它们合起来。最后会剩下一些数字，不会超过 5 个，且有一个只可能剩下 1 个。不妨设剩下 12345。我们发现最优的子序列不会很长，因为假如一个子序列能拆分成两个和均为 0 的子序列它显然不会是最优的。所以我们直接把所有可能最优的子序列搜出来。

接下来，我们就把 B 数组拆成很多个可能的子序列组合起来即可。我们可以弄出一个可行解，然后对它进行优化。优化只有可能是几个子序列组合起来，然后拆成更多个小的子序列。比如

$$(1,3,4,4,4,4)+(1,2,2,2,3)->(2,4,4)+(1,2,3,4)+(1,2,3,4)$$

找规律可得最简的优化方法里面每个数出现的次数不会超过 12。

我们可以对于所有可优化数字组合，用 DP 求出该数字组合最优能拆成的子序列。然后我们的方法就是暴力枚举当前答案中的某个子集，然后看看它的数字组合是否已是最优。若不是最优则用最优方案替代即可。而且一次我们可以同时优化很多个相同的数字组合。若已经没有可以优化的数字组合，证明我们已经找到了一个最优解。

综上我们解决了这个问题。

25.4 算法流程

- 用找循环节的方法将差分数组统计出来，然后统计它每个数字出现次数。

- 将所有可能出现在最优解中的子序列搜索出来。
- DP 求出所有数字组合最优的拆分方案。
- 随意构造一组可行解。
- 搜索当前答案集的子集，寻找可以优化的数字组合。
- 当前答案不可继续优化时，即找到了一组最优解。

25.5 复杂度

$O(???)$

26 Codechef GNUM

26.1 题意

给出两个长度为 N 的数组 A, B。每一轮我们要进行一个操作：选取两个数对 $(i, j), (p, q)$，满足 $A_i < B_j$，$A_p > B_q$，$gcd(A_i, B_j, A_p, B_q) > 1$。一个数对不能被多次操作选中。问最多能有多少轮操作。

范围是 $N \leq 400$，$A_i, B_i \leq 10^9$。

26.2 关键词

网络流

26.3 题解

最简单的想法就是把所有合法数对分成两类之后，枚举两边的数对，然后构出类似二分图匹配的模型。但是这样边数会达到 $O(N^4)$ 级别。

我们发现如果对于两个 gcd 大于 1 的数，它们至少拥有 1 个相同的质数因子。所以我们对所有出现过的数质因数分解后，将所有出现过的质数新增一个点，将两部分与它能被它整除的数连起来。这样构出的图与暴力构图等价。这样就能过了。

26.4 算法流程

- 对所有数质因数分解
- 将所有数对找出来并分类
- 对于每个出现过的质数，找到能被它整除的数对，连边
- 连好源点汇点的边
- 网络流

26.5 复杂度

$O(maxflow(N^2, N^2logN))$

27 Codechef RIN

27.1 题意

有 N 个课程，M 个学期。每个课程都要用一个学期学习，且第 i 个课程在第 j 个学期学习会有 $C_{i,j}$ 的收益。还有 K 个限制，每个限制 (x, y) 表示第 y 个课程必须要在第 x 个课程后学习。求最大的收益。

范围是 $N, M, K \leq 100$。

27.2 关键词

网络流

27.3 题解

经典的最小割模型题。

对每个课程 i，我们建 $M+1$ 个点。然后第 j 个点连向第 $j+1$ 个点，容量为 $W - C_{i,j}$，割掉它表示我们第 i 个课程在第 j 个学期选的时候会损失多少收益（可以事先对每个课程加上它可能的最大收益）。对于每个限制 (x, y)，我们对课程 x 的第 i 个点，都连条容量无穷大的边到课程 y 的第 $i+1$ 个点。这样连边的意义是如果出现了非法的情况，那么这条边就会有流量，使得当前方案不合法（画出连边就很明显了）。这样跑最小割就可以了。

27.4 算法流程

- 按上述方法建图
- 跑最小割

27.5 复杂度

$O(maxflow(NM, NM + NK))$

28 Codechef SEAEQ

28.1 题意

定义两个长度均为 N 的序列 A, B 相似：对于 $1 \leq i \leq n$，都有 $CA(A[i]) = CB(B[i])$，$CX(Y) = X$ 序列中小于 Y 的数的个数。定义函数 $F(A, B)$ 等于数对 $[l, r], l \leq r$ 的个数，满足序列 $A[l, r]$ 和序列 $B[l, r]$ 相似，且序列 $A[l, r]$ 的逆序对数不超过 E。

问题要求 $\sum F(P1, P2)$。$P1, P2$ 是任意一个 1 到 N 的排列。

范围是 $N \leq 500, E \leq N * (N - 1)/2$

28.2 关键词

DP

28.3 题解

设 $F[i][j]$ 表示 1 到 i 的排列中，逆序对数不超过 j 的数量。

转移方程：$F[i][j] = F[i - 1][j] - F[i - 1][j - i] + F[i][j - 1]$。

我们用 $O(N^3)$ 的时间预处理出 F 即可。

然后枚举数对的长度（子序列的长度），设为 l。我们看看这样的数对在所有可能情况里面会有多少个。

首先这样的数对的位置有 $N - l + 1$ 个。

$P1$ 序列能选的数有 $C(n, l)$ 种方案，$P2$ 同理。

选择了数之后，在这个序列里的逆序对数不超过 E 的排列个数是 $F[l][E]$。

由相似的定义，$P1$ 这个子序列定下来之后，$P2$ 的子序列的排列也定下来了。

最后两个序列均剩下 $N-l$ 个位置可以随便排，都要乘上 $(N-l)!$。

累计入答案即可。

28.4 算法流程

- 预处理 F
- 枚举数对长度 l，累计答案

28.5 复杂度

$O(N^3)$

29 Codechef TWOCOMP

29.1 题意

有一棵 N 个点的树。现在有两类简单路径，都用 (x,y,c) 表示这条路径是从 x 到 y 的简单路径，价值为 c。现在要从两类简单路径中选出若干条路径，使得选手来路径的价值和最大且不存在一条一类路径和二类路径相交（有共同点）。求这个价值和。

范围是 $N \leq 10^5$，同一类简单路径不会超过 700 条。

29.2 关键词

DFS 序，最小割

29.3 题解

假如我们已经知道两类路径之间的是否相交，直接用最小割建图即可。

判断两类路径相交，为了方便，我们可以将一条路径拆成两条从祖先到后代的路径，然后用 DFS 序判断即可。

判断两条（拆分后的）路径相交：设两条路径是 $(x1,y1),(x2,y2)$。不妨设 $x1$ 是 $x1,y1,x2,y2$ 中深度最小的点。两条路径相交等价于 $x2$ 在 $(x1,y1)$ 路径上。这个用 DFS 序判断即可。

29.4 算法流程

- DFS 整棵树，搞出 DFS 序。
- 枚举两类路径判相交
- 建图跑最小割

29.5 复杂度

$O(N + M^2 + maxflow(M, M^2))$

30 Codechef QUERY

30.1 题意

维护一颗 N 个节点的树，支持以下操作：

- 给 x 到 y 路径上的第一个点的权值加上 A，第二个点加上 $A + B$，第三个点加上 $A + 2B$，以此类推。
- 询问 x 到 y 路径上的点权和。
- 令树上的状态返回第 R 次操作之后的状态。

范围是 $N, Q \leq 10^5$。

30.2 关键词

轻重链剖分，可持久化线段树

30.3 题解

这个题基本和 FIBTREE 是一样的。链剖之后维护可持久化线段树即可。

30.4 算法流程

- 轻重链剖分
- 按 FIBTREE 类似方法询问和修改

30.5 复杂度

$O(Qlog^2N)$

31 Codechef ANUDTQ

31.1 题意

维护一颗 N 个节点的带权树，支持以下操作：

- 新增一个新点，令他的父亲是 x，权值是 v。
- 给以 x 为根的子树的每个点的权值加上 v。
- 删除已 x 为根的子树。
- 询问以 x 为根的子树的权值和。

范围是 $N, Q \leq 10^5$。

31.2 关键词

LCT

31.3 题解

这题是动态树的子树问题，一般都用 ETT 或是 Top Tree 之类的高级算法。但是这题并不用这么麻烦。

我的做法是 LCT。

设 $cov[x]$ 表示对于以 x 为根的加权操作的权值和。$sum[x]$ 表示以 x 为根的子树和（不考虑以 x 的祖先为根的加权操作），$size[x]$ 表示 x 为根的子树大小。

那么要询问一个点时，它子树真正的权值和是 $sum[x] + size[x] * \sum_y cov[y]$，$y$ 是 x 的祖先。那么维护这几个值就可以只用单点修改查询或是单链修改查询即可。那么用 LCT 维护这几个操作是比较方便的，而且所有的操作均可以用这通过查询或修改这几个值实现。

这个做法存在着一个问题：修改了子树 x 的大小后，询问 x 会有问题。因为我么你直接查询 $\sum_y cov[y]$，然而打上 cov 标记的时候子树大小并不是当前的大小。那么这样计算就会多算新增的节点的部分或删去子树的部分。

举个例子：只有 1 个点的树，然后给它来个子树加 v，$cov[1] = v$，之后加个新点 2，然后 $size[1]$ 变成 2，询问点 1 的时候那个 v 就会算两次。实际上它打标记时的子树大小是 1。

解决办法是：我们在改 $size[x]$ 的时候，把答案多出来的部分 $\Delta size[x] * \sum_y cov[y]$ 在 $sum[x]$ 中减掉。这样多出来的部分就没有影响了。

又出现了新的问题：我们不能每次都查询一次 $\sum_y cov[y]$！解决办法是由于我们使用的是 LCT，一个点的左儿子就是它的祖先，那么我们利用这一点把 $\Delta size[x] * \sum_y cov[y]$ 维护出来。我们在给一个点打标记把就可以把它左子树的 $cov[y]$ 的和搞出来。下传标记时把左儿子的 $cov[y]$ 的和以标记的形式加到右子树去。这样就能在打标记的时候把一个点所有的祖先的影响都维护出来。

综上我们解决了这个问题。

(后记：这题的正确姿势似乎是用 Splay 维护 DFS 序？)

31.4 算法流程

- 遍历初始的树
- 询问和修改用 LCT 维护

31.5 复杂度

$O(NlogN + QlogN)$

32 Codechef SEAARC

32.1 题意

给定长度为 N 的序列 A，统计 A 中满足一下条件的四元组 (i, j, k, l) 的数量：$i < j < k < l$,$A_i = A_k, A_j = A_l, A_i \neq A_j$。范围是 $N \leq 10^5$。

32.2 关键词

分类讨论，容斥原理

32.3 题解

题目要我们求的是形如 $abab$ 的四元组。那么直接统计不好搞。我们考虑容斥原理。

设 $c[x]$ 表示 x 的出现次数。按各数对的排列情况可以分成三种：$aabb, abba, abab$，我们用总数减去 $aabb, abba$ 的情况即可。首先我们可以算出三种情况的总数。

接下来 $aabb$ 的情况扫一遍维护一下即可。

对于 $abba$ 的情况，我们考虑根号算法。

然后我们分成三类情况讨论：

对于 $c[a] > \sqrt{N}$ 的，这样的 a 不会超过 $\sqrt{N}$ 个。考虑维护数组 $pre[i]$ 表示 i 位置之前出现了多少个 a。那么对于 $A_l = A_r$ 的数对，它们作为 bb 的方案数有 $pre[l] * (c[a] - pre[r])$ 种。这里我们可以用 $O(N)$ 的方法维护出 pre，同样 $O(N)$ 维护下前缀和就可以计算了。计算这部分的复杂度是 $O(N\sqrt{N})$

对于 $c[a] \leq \sqrt{N}, c[b] > \sqrt{N}$ 的，这样的 b 不会超过 $\sqrt{N}$ 个。考虑维护数组 $pre[i]$ 表示 i 位置之前出现了多少个 b。那么对于 $A_l = A_r$ 的数对，它们作为 aa 的方案数有 $C(pre[r] - pre[l], 2)$ 种。把这个式子拆开来，同样 $O(N)$ 维护下前缀和就可以计算了。计算这部分的复杂度是 $O(N\sqrt{N})$。

对于 $c[a] \leq \sqrt{N}, c[b] \leq \sqrt{N}$ 的，我们考虑枚举 $A_l = A_r$ 作为 aa，然后统计 (l, r) 中 bb 的数量。为了方便，我们可以先枚举 r，再枚举 l，统计答案之后再把当前的 A_l 和 A_r 数对插入数据结构中。统计答案时，由于我们是枚举了右端点，那么之前的数对的右端点一定在当前 aa 的右端点的左边。所以我们只需要统计左端点在当前 aa 的左端点的右边的数对即可。这样我们把一个数对插入数据结构时以它的左端点做关键字即可。这部分的复杂度是 $O(N\sqrt{N}logN)$。

最后注意去掉 $aaaa$ 的情况。

32.4 算法流程

- 算总数
- 算 $aabb$ 的情况
- 算 $abba$ 的情况，分三种情况分别求解

- 用总数减去上两种情况得到答案

32.5 复杂度

$O(N\sqrt{N}logN)$

33 Codechef GRAPHCNT

33.1 题意

给出 N 个点的无向图的 M 条边，有 Q 个询问，每次询问只保留标号 $[l,r]$ 之间的边时图中有多少个联通块。

范围是 $N,M,Q \leq 200000$。

33.2 关键词

莫队算法，并查集

33.3 题解

暴力的话每次用并查集搞搞即可。

由于离线，我们可以用莫队实现高级暴力。唯一的问题是并查集不支持删除操作。那么莫队算法左端点可能会遇到删除操作的情况。我们直接暴力的把左端点所在莫队分出来的 $\sqrt{N}$ 长度的区间的边对并查集的影响全部记下来，每次重新计算左端点部分即可。

33.4 算法流程

- 分 $\sqrt{N}$ 区间
- 莫队算法

33.5 复杂度

$O(N\sqrt{N})$

34 Codechef CARDSHUF

34.1 题意

有一个从 1 到 N 拍好的牌堆。有 M 次洗牌操作。每次操作给出 A_i, B_i, C_i，洗牌如下：

- 从牌堆顶端拿走 A 张牌。
- 再从牌堆顶端拿走 B 张牌。
- 将第一步拿走的 A 张牌放回到剩下的牌堆上面。
- 从牌堆顶拿走 C 张牌。
- 将第二步你拿起的 B 张牌一张一张放到牌堆顶。
- 最后，将剩下的 C 张牌放回到牌堆顶。

要求最终牌堆的排列顺序。范围是 $N, M \leq 10^5$。

34.2 关键词

Splay

34.3 题解

用 Splay 维护下 rev 标记就可以了。（这是全部作业里最水的吧）

34.4 算法流程

不写了

34.5 复杂度

$O(NlogN)$

35 Sereja and Subsegment Increasings

35.1 编号

Codechef SEINC

35.2 题意

给出两个长度为 N 的序列 A 和 B，每次操作可以选择 l, r，使 $[l, r]$ 区间中的 $A_i = (A_i + 1) \bmod 4$。问最少需要多少次操作使得 A 变成 B。

范围是 $0 \le A_i, B_i \le 3, N \le 10^5$。

35.3 关键词

贪心

35.4 题解

令 $C_i = (B_i - A_i) \bmod 4$。问题转化成每次使 C 中的一段数减 1，最少需要多少次操作使 C 全部为 0。

令 $D_i = C_{i+1} - C_i$。不考虑模的情况，问题转换成每次操作使 $D_l - 1, D_{r+1} + 1$，最少需要多少次操作使 D 全部为 0。不考虑模的话这个问题的解是 $\sum_{i=0}^{N} max(D_i, 0)$，即所有正数和。考虑模意义后，我们优化解的方法是：让 C 的某些段落减多 4 的倍数次。反映在 D 上就是可以对于某些 $i < j$，令 $D_i += 4, D_j -= 4$，这样看上去不会优化答案，然而实际上我们考虑答案即为所有的正数和，那么我们确实可以通过这样的调整使得正数和变小。

首先我们有 $-3 \le D_i \le 3$，那么我们发现对于 $-1, 0, 1$ 的位置不需要调整。

然后对于 $-2, -3, 2, 3$，我们可以调整的数对是 $(-3, 3), (-3, 2), (-2, 3)$。那么我们可以扫一遍，保存前面出现过的 -2 和 -3 的次数。对于当前位置是 2 或 3 的我们就尝试调整即可。注意当前 2 调整过后会出现 $2 - 4 = -2$，要累计入 -2 的次数中。

最后统计答案即可。

35.5 算法流程

- 由 A, B 计算得到 D
- 从前到后扫描 D，按上述方法调整 D
- 统计答案

35.6 复杂度

$O(N)$

36 The Street

36.1 编号

STREETTA

36.2 题意

街道上有 N 个店铺，一开始没有货物。有 M 个操作。操作有如下三种：

- 给 $[l,r]$ 的店铺 i 加上 $(i-l)*a+b$ 的税费。
- 给 $[l,r]$ 的店铺 i 进价格为 $(i-l)*a+b$ 的货物。
- 询问在编号 i 的店铺中购买最贵的货物加税费的价格。

范围是 $N \leq 10^9, M \leq 3*10^5$。

36.3 关键词

线段树

36.4 题解

我们考虑使用动态开节点的线段树解决这个问题。第一个操作很简单。考虑如何维护第二个操作的最大值。

若区间 $[l,r]$ 要打上一个标记 (a,b)，而之前这个区间已经有一个标记 (c,d)，那么我们要比较两个标记。不妨假设 $a \geq c$。若一个标记在该区间内始终比另一个标记大，那么直接保留较优的即可。否则，我们可以计算得到 k，使得在 $[l,k]$ 区间中 (c,d) 标记更优，在 $[k+1,r]$ 区间中 (a,b) 标记更优。那么我们考虑在该区间保留优势区间较长的标记，将优势区间小于原区间一半的标记下传，这样每次都只用下传一个节点，所以打一个标记最多只会下传 $O(logN)$ 次。

这样询问第一个操作的总和很简单，询问第二个操作的话就在线段树向下走，每次用当前区间标记对 i 的影响统计答案即可。

36.5 算法流程

按上述方法修改和询问即可。

36.6 复杂度

$O(Nlog^2N)$

37 Count on a Treap

37.1 编号

COT5

37.2 题目大意

要维护一棵大根堆 Treap，有 N 个操作有如下三种：

- 插入一个关键字为 key，权值为 w 的节点。
- 删除关键字为 key 的节点。
- 询问关键字 $k1$ 的节点与关键字 $k2$ 的节点在 Treap 上的距离。

范围是 $N \leq 10^5$

37.3 关键词

线段树

37.4 题解

直接维护 Treap 显然会被卡。用动态树等方法似乎太过麻烦。

我们发现 Treap 有以下性质（注意以下用节点的关键字代表节点编号）：

- Treap 按中序遍历得到的 DFS 序等价于按关键字排序。
- 若 x 是 y 的祖先，当且仅当在 DFS 序区间 $[x,y]$ 或 $[y,x]$ 中 x 点权值最大。
- 两点 x 和 y 的 LCA 是 DFS 序区间 $[x,y]$ 中权值最大的节点。

那么我们考虑按关键字（即按 DFS 序）维护一棵线段树。我们可以维护下区间最大值来得到 LCA。剩下的问题就是求一个点的深度了。一个点的深度等于在它两边的祖先的个数。我们考虑如何求一个节点 x 左边的祖先数。由第二条性质，假如我们把区间 $[1,x]$ 的权值求后缀最大值，这个数组会是“阶梯形”。那么节点 x 左边的祖先数就是“阶梯”的级数。

定义 $max[l,r]$ 表示区间 $[l,r]$ 中的权值最大值。函数 $calc(l,r,v)$ 表示在区间 $[l,r]$ 右边的最大值是 v 时，$[l,r]$ 中的“阶梯”数。设 mid 表示区间中点。当 $v>max[mid+1,r]$ 时，直接返回 $calc(l,mid,v)$ 即可（右区间不会有贡献）。否则，返回 $calc(mid+1,r,v)+calc(l,mid,max[mid+1,r])$。

考虑第二种情况，直接计算似乎与暴力无异。

但是我们发现 $calc(l,mid,max[mid+1,r])$ 与 v 无关。若我们能 $O(1)$ 得到这个值，那么函数 $calc(l,r,v)$ 的复杂度就是 $O(logN)$ 的。

所以我们优化的方法是将这个值作为标记保存在当前区间中。当一个区间的最大值改变时，我们修改它的标记与它在线段树上的“相邻区间”的标记即可。而且我们是递归的处理，所有子区间已经处理完毕，更改一个标记只需要调用 $calc$ 函数即可。更改一次的复杂度为 $O(logN)$。这样我们就可以在 $O(logN)$ 的时间内求出一个点的深度，在 $O(log^2N)$ 的时间内修改一个点权（插入和删除均可归约为修改点权）。

综上我们解决了这个问题。

37.5 算法流程

按上述方法修改与询问即可。

37.6 复杂度

$O(Nlog^2N)$

38 Counting on a directed graph

38.1 编号

GRAPHCNT

38.2 题目大意

给定一个 N 个点 M 条边的有向图。统计无序对 (X,Y) 的个数，其中 (X,Y) 满足存在一条从点 1 到点 X 的路径，和一条从点 1 到点 Y 的路径，且两条路径除了点 1 以外没有公共点。

38.3 关键词

Dominator Tree

38.4 题解

显然合法的点对除了 1 以外没有公共必经点。直接构出 Dominator Tree，对 1 的子节点的子树搞搞即可。

38.5 算法流程

建 Dominator Tree 后统计

38.6 复杂度

$O(N)$

39 Counting D-sets

39.1 编号

CNTDSETS

39.2 题目大意

统计 N 维空间中的点集，满足点集的直径等于 D（直径的定义是点集中切比雪夫距离最远的两点的距离），当两个点集可以平移重合时两个点集被认为是等价的。7 范围是 $N \leq 10^3, D \leq 10^9$

39.3 关键词

容斥

39.4 题解

先转换成直径小于等于 D。然后相当于在一个长为 D 的 N 维立方体中统计点集。为了避免重复，要保证点集中每一维都有至少有一个 0。反过来算有 N 维一定没有 0 的方案数，容斥一下就可以了。

39.5 算法流程

直接容斥即可。

39.6 复杂度

$O(N)$

40 Counting The Important Pairs

40.1 编号

TAPAIR

40.2 题目大意

一个 N 个点 M 条边的联通的无向图 (无重边，自环)，删去两条边之后使得图不联通，求可行的选边的方案。范围是 $N \leq 10^5, M \leq 3 * 10^5$。

40.3 关键词

随机，DFS 树

40.4 题解

搞出 DFS 树之后，首先我们给每条非树边分配一个 $[1, 2^{64})$ 的随机权值，然后每条树边的权值定为覆盖它的非树边的权值异或和。发现如果要删除两条边使得图不联通，要么是有一条边权值为 0（没有被覆盖过的树边），要么两条边权值相同。直接统计即可。出错的概率可以忽略不计。（该题有确定性算法。）

40.5 算法流程

- 构出 DFS 树
- 对非树边分配权值，对树边计算出覆盖它的非树边的权值异或和
- 排序后统计答案

40.6 复杂度

$O(NlogN)$

41 To Queue or not to Queue

41.1 编号

TMP01

41.2 题目大意

一个字符串 S。初始时这个串是空的。还有 N 操作，每个操作是下面两种之一：

- 在 S 的末尾插入一个字符
- 删除 S 的第一个字符

每个操作过后输出当前字符串不同的子串个数。范围是 $N \leq 10^6$。

41.3 关键词

后缀树

41.4 题解

这题是 Ukkonen 算法的应用。Ukkonen 算法是一种在线增量构造后缀树的方法。这题还要支持删除第一个字符的操作。

首先将后缀树一个叶子节点删除（表示原来最长的后缀）。然后将它祖先上度数为 1 的节点直接删掉（维护后缀树的性质）。如果删到了活跃节点，就类似于插入一样释放活跃节点即可。

理解 Ukkonen 算法可以参见下面链接：

stackoverflow 一个详细的解释

Codechef 上这题的官方题解

41.5 算法流程

使用 Ukkonen 算法在线增量或删除即可。

41.6 复杂度

$O(N)$

42 Fibonacci Number

42.1 编号

FN

42.2 题目大意

给出 P, C，求 $fib(n) = C \bmod P$ 的 n 的最小值。满足 P 是质数，且末位是 1 或 9。范围是 $P \leq 2 * 10^9$。

42.3 关键词

BSGS，Cipolla’s algorithm

42.4 题解

在题目给定条件中，2 有逆元，$\sqrt{5}$ 有对应的整数（注意有两个）。

然后推出通项公式，替换之后就是要解方程：$x^n - (\frac{-1}{x})^n = a$（$a$ 是常量，x 也是常量），求 n 的最小值。不妨假设 n 是偶数（奇数类似做法）。得到 $x^n - x^{-n} = a$。解得 $x^n = \frac{a \pm \sqrt{a^2+4}}{2}$。如果 $\sqrt{a^2+4}$ 能算出来，那么我们就可以直接得到 x^n，然后用大步小步算法即可。

对于求二次剩余（模系下的开根），使用Cipolla' s algorithm即可。

42.5 算法流程

- 求出 $\frac{1+\sqrt{5}}{2}$ 在模系下对应的值 x
- 对于 n 是偶数的情况，按上述方法求解方程
- 奇数同理，然后取 n 的最小值。

42.6 复杂度

$O(\sqrt{P})$

43 Two Roads

43.1 编号

TWOROADS

43.2 题目大意

在平面上有 N 个点，现在要求两条直线，使得所有点到这两条直线的最短距离的平均值最小。求这个最小的平均值。范围是 $N \leq 100$。

43.3 关键词

扫描线

43.4 题解

如果只有一条线的情况，我们可以通过维护点的各种信息（$\sum x, \sum x^2$ $\sum y, \sum y^2, \sum x * y$，点数）直接列方程求解即可。

回到两条线的情况。对于任意一个解，我们可以发现这两条直线夹角的角平分线把平面划分成 4 个区域。顶点相对的区域到其中的同一条直线的距离更短。所以我们现在可以枚举平面中两条相互垂直的角平分线，它们划分出的 4 个区域分别对应哪些点，直接算出来对应区域的最优答案即可。可以发现我们限制其中一条直线至少穿过两个点，另一条至少穿过一个点就可以囊括所有的情况。直接扫描线算法即可。

43.5 算法流程

- 枚举两个点的连线作为第一条线
- 将剩下的点按投影排序，然后枚举第二条线
- 维护出两个区域的信息直接算出答案，取最小值即可

43.6 复杂度

$O(N^3 logN)$

44 Fibonacci Number

44.1 编号

FN

44.2 题目大意

给出 P, C，求 $fib(n) = C \bmod P$ 的 n 的最小值。满足 P 是质数，且末位是 1 或 9。范围是 $P \leq 2 * 10^9$。

44.3 关键词

BSGS，Cipolla’s algorithm

44.4 题解

在题目给定条件中，2 有逆元，$\sqrt{5}$ 有对应的整数（注意有两个）。

然后推出通项公式，替换之后就是要解方程：$x^n - (\frac{-1}{x})^n = a$（$a$ 是常量，x 也是常量），求 n 的最小值。不妨假设 n 是偶数（奇数类似做法）。得到 $x^n - x^{-n} = a$。解得 $x^n = \frac{a \pm \sqrt{a^2+4}}{2}$。如果 $\sqrt{a^2+4}$ 能算出来，那么我们就可以直接得到 x^n，然后用大步小步算法即可。

对于求二次剩余（模系下的开根），使用Cipolla' s algorithm即可。

44.5 算法流程

- 求出 $\frac{1+\sqrt{5}}{2}$ 在模系下对应的值 x
- 对于 n 是偶数的情况，按上述方法求解方程
- 奇数同理，然后取 n 的最小值。

44.6 复杂度

$O(\sqrt{P})$

45 Two k-Convex Polygons

45.1 编号

TWOROADS

45.2 题目大意

给定 N 个棍子的长度和整数 K，求能否在其中选出 $2K$ 个棍子拼成两个凸多边形。使得两个凸多边形都恰好有 K 根棍子组成, 且任意相邻的边都不共线。范围是 $N \leq 10^3, K \leq 10$

45.3 关键词

搜索

45.4 题解

形成 K 多边形的充分必要条件是最长的边要小于其它边之和。我们将边长排序，显然两个多边形的取值要么是一个长度为 $2K$ 的区间，要么是

两个不相交的长度为 K 的区间。对于前一种情况，直接枚举这个区间暴力搜索即可（因为 N 比较大的时候一定有解，大概是 70 左右，直接暴力是没问题的）。后一种直接前缀后缀搞搞即可。

45.5 算法流程

- 维护前后缀信息判断区间不相邻的情况
- 暴力枚举区间然后搜索判断另一种情况

45.6 复杂度

$O(N * C(2K, K))$

46 Count Special Matrices

46.1 编号

SPMATRIX

46.2 题目大意

求满足下列条件的 $N * N$ 的矩阵个数：

- $A_{i,i} = 0$
- $1 \leq A_{i,j} < N - 1 (i \neq j)$
- 对于 $0 \leq k \leq N - 2$，都有某个 $A_{i,j} = k$
- $A_{i,j} \leq max(A_{i,k}, A_{k,j}) (1 \leq i, j, k \leq N)$

范围是 $N \leq 10^7$

46.3 关键词

数学推导

46.4 题解

不明觉厉的公式题。大概推导过程：$A_{i,j}$ 是一个超空间中的 N 个点的距离矩阵。然后就能推得答案等于

$$\frac{n!(n-1)!}{3*2^{n-1}}(\frac{3n}{2}-2-\sum_{i=1}^{n-1}\frac{1}{i})$$

预处理阶乘和逆元就可以 $O(1)$ 计算了。注意常数问题。

46.5 算法流程

预处理后代公式算即可。

46.6 复杂度

$O(N)$

47 Little Elephant and Colored Coins

47.1 编号

LECOINS

47.2 题目大意

给出一些硬币，每个硬币有价值 V_i 和颜色 C_i。有 Q 个询问，询问是否可能用这些组成价值 S，如果能再询问最多用多少种颜色的硬币。

范围是 $N \leq 30, V \leq 2*10^5, Q \leq 2*10^5, S \leq 10^18$

47.3 关键词

DP

47.4 题解

不考虑颜色的情况下，可以先找出价值最小的硬币，其价值为 v。可以 DP 出 $f[i]$ 表示组成的价值模 v 为 i 最小的价值是多少。然后对于询问

判断 $f[S \bmod v]$ 和 S 的大小关系即可。DP 时注意优化，对于每一个硬币 i，我们有 $f[(x+V_i) \bmod v] = min(f[(x+V_i) \bmod v], f[x]+V_i)$。把转移画出来发现实际上只有 $gcd(V_i, v)$ 个独立的环，环之间互不影响。环上的话，找一个最小的值作为开始转移就可以一次转移完，不用最短路算法了。

考虑上颜色，就加多一维进行 DP 即可。注意特判最小价值硬币的颜色造成的影响。

47.5 算法流程

- 预处理 DP 数组
- 对于每个询问直接查询即可

47.6 复杂度

$O(N * V_{min})$

48 A New Door

48.1 编号

ANDOOR

48.2 题目大意

一个 $W * H$ 的框里面挖 N 个圆形的洞，给出这些洞的圆心和半径。求挖出来的部分的周长。范围是 $N \leq 1000$。

48.3 关键词

计算几何

48.4 题解

直接计算每个圆对答案的贡献。对于每个圆，暴力扫描它和别的圆的交，那么有交的圆弧一定不会被计入答案。我们可以用一个弧度区间 $[l, r]$

表示被删掉的圆弧。再把它和四周的框的交算出来，同样用区间表示被删掉的圆弧。最后排序合并删掉的圆弧。统计答案即可。

48.5 算法流程

如上所述。

48.6 复杂度

$O(N^2logN)$

49 Sine Partition Function

49.1 编号

PARSIN

49.2 题目大意

求 $f(n,m,x)=\sum_{k_1+\cdots+k_m} sin(k_1x)sin(k_2x)\cdots sin(k_mx)$，范围是 $m\leq 50, n\leq 10^9$。

49.3 关键词

矩阵乘法，DP

49.4 题解

考虑 DP。

$$f_{n,m}=\sum_{k_1+\cdots+k_m} sin(k_1x)sin(k_2x)\cdots sin(k_mx)$$

$$g_{n,m}=\sum_{k_1+\cdots+k_m} sin(k_1x)sin(k_2x)\cdots cos(k_mx)$$

可得

$f_{n,m}=(f_{n-1,m-1}+g_{n-1,m})*sin(x)+f_{n-1,m}*cos(x)$

$g_{n,m}=(f_{n-1,m-1}+g_{n-1,m})*cos(x)-f_{n-1,m}*sin(x)$

直接矩阵乘法优化 DP 即可。

49.5 算法流程

如上所述。

49.6 复杂度

$O(m^3 logn)$

50 Hypertrees

50.1 编号

HYPER

50.2 题目大意

一个 3-超图类似与一个普通的图, 只不过其中的边都连接三个点。一个 3-超树是一个去掉任意一条边都以后都不连通的 3-超图。给定 N，问有几种含有 N 个带标号的点的本质不同的 3-超树。范围是 $N \leq 17$

50.3 关键词

搜索

50.4 题解

先考虑一个点双联通的 3-超树。由于每删一条边都能使图不联通，那么只能是每条边都连接至少一个叶子（类似于一个正常的点双联通图的每条边“长出”了一个额外的点）。所以我们直接暴力枚举正常的点双联通图即可。然后我们再考虑“拼接”这样的 3-超树形成更大的 3-超树 (不必满足点双联通)，记忆化搜索即可。然后本地搜然后打表交上去就好了。

50.5 算法流程

如上所述。

50.6 复杂度

$O(?)$

51 Short II

51.1 编号

SHORT2

51.2 题目大意

给定 p(一个质数)，问有多少对 $a,b(a>p,b>p)$ 满足 $a*b$ 被 $(a-p)*(b-p)$ 整除。范围是 $p\leq 10^{12}$

51.3 关键词

数论

51.4 题解

等价于求 $ab|(a+p)(a+p)$ 的对数。展开后可得 $ab|(a+b+p)p$。可以分情况讨论：

- $p|a,p|b$，发现 $(a/p,b/p)$ 只有 5 对可能情况 $(1,1),(1,2),(2,1),(2,3),(3,2)$。
- $p\nmid a,p\nmid b$ 这时条件转换为 $ab|a+b+p$。不妨假设 $a<b,kab=a+b+p$，那么由 $ab\leq a+b+p$ 可推得 $(a-1)(b-1)\leq p+1$，所以 $a\leq 1+\sqrt{p+1}$。又有 $b=\frac{a+p}{ka+1}>a$。令 $d=ka+1$，可得
  - $p\nmid a$
  - $d\mid a+p$
  - $a\mid d+1$
  - $b=\frac{a+p}{d}>a$。

  由第四个条件可知 b,d 中有一个小于等于 $\sqrt{a+p}$。

先做 $d \le \sqrt{a+p} \le \sqrt{1+\sqrt{p+1}+p}$ 的情况。枚举 d，由条件 3 得 $1 \le a \le d+1$，由条件 2 得 $a \equiv -p(\mathrm{mod}d)$。然后对于特定的 d，直接令 $a = d-p$，然后不停地加 d，满足条件的 a 最多只有两个。

再做 $b \le \sqrt{a+p}\sqrt{1+\sqrt{p+1}+p}$ 的情况。由 $bd = a+p$，有 $a \equiv -p(\mathrm{mod}b)$，类似的，令 $a = b-p$，满足情况的 a 最多只有一个。$

- $p \mid a, p \nmid b$ 或者反过来。那么这个条件下的每一对 (a,b) 都能由第二类情况的 (a,b) 转化过来。$(a,b) \to (a, \frac{p(a+p)}{b})$ 和 $(a,b) \to (\frac{p(b+p)}{a}, b)$。所以满足第三类情况的 (a,b) 的对数是满足第二类条件的两倍。

51.5 算法流程

直接计算第二类，分情况讨论 $d \le \sqrt{a+p}$ 和 $b \le \sqrt{a+p}$ 即可。

51.6 复杂度

$O(\sqrt{P})$

52 Little Elephant and Boxes

52.1 编号

LEBOXES

52.2 题目大意

有 N 个盒子，打开第 i 个盒子，有 p_i 的概率能获得 V_i 元，否则获得一个钻石。还有 M 个物品，购买第 j 个物品要 C_j 元和 D_j 个钻石。在打开了所有的盒子后买尽可能多物品的前提下，问期望买物品的数量。范围是 $N, M \le 40$

52.3 关键词

meet in middle

52.4 题解

首先处理出 $f_{i,j}$ 表示要买 i 个物品，且拥有恰好 j 个钻石时还要有 $f_{i,j}$ 元。

然后分前后暴力枚举 (meet in middle)。首先把前面大概 2/3 的部分箱子的所有情况暴力枚举出来，记录下每个情况出现的概率，总共获得的钱数和钻石数，按获得的钻石数为索引值建立桶，然后把另外两个值作为二元组扔到桶里面。

然后再暴力枚举后面的 1/3 的箱子的情况，类似的记录下三个值。然后再暴力枚举要购买的物品数量和前 2/3 的箱子获得多少个钻石。这样就可以计算出总钻石数和前面还需要的钱数。那么在对应的桶里边二分查找一下之后把概率取个前缀值加入答案即可。

52.5 算法流程

- 预处理 $f_{i,j}$
- 枚举前 2/3 的箱子的情况
- 枚举后 1/3 的箱子的情况，并且同时枚举购买物品数量和钻石数，再前面搞出来的桶里边二分并且求和即可

52.6 复杂度

$O(2^{2N/3} + 2^{N/3} * N^2)$

53 Selling Tickets

53.1 编号

TICKETS

53.2 题目大意

有 N 个物品，M 个人。第 i 人有他想要的两个物品 a_i 和 b_i，他至少需要拿到其中的一个。现在要保留 K 个人，且要保证任意选 K 个人，都能满足他们的需求。求 K 的最大值。范围是 $N \leq 200, M \leq 500$。

53.3 关键词

图论

53.4 题解

把每个物品看成是点，每个人看成是 a_i 和 b_i 之间的边，题目转换成了求这个无向图边数恰好比点数多 1 的联通子图的点数最小值减一。

有且仅有两种情况：

1. 两个点之间的三条路径。可以枚举一个点，以它为源，求出其他点的前三短的最短路的长度，要保证跑出来的路不可以重复走同样的点（做法：可以从与源点相邻的点开始求最短路）。然后枚举另外一个点更新答案即可。

2. 两个由一条简单路径链接起的环 (或是直接有一个共同点)。同样是随便枚举一个点，然后搞出 BFS 树。那么把一个环的权值设为它的点数加上它到根的距离。然后在 BFS 树上找两个权值最小的环更新答案即可 (无需考虑点是否重复，假如有重复证明当前不是最优解)。

53.5 算法流程

按上述方法讨论两种情况即可。

53.6 复杂度

$O(N^2M + N^3)$

54 Something About Divisors

54.1 编号

DIVISORS

54.2 题目大意

对于给定的正整数 b, x，求满足条件的正整数 n 的个数: 要求对于 n，至少存在一个数 $d, n < d \leq b$ 能整除 nx。范围是 $b \leq 10^{12}, x \leq 60$

54.3 关键词

数论, 容斥

54.4 题解

考虑统计对于不同的 $k=\frac{nx}{b}$，统计可能 n 的个数。为避免重复，要满足在不同的 k 里面，当前 k 是最大的能整除 nx 的 k，记合法的 n 的个数是 S_k。

对于当前的 k，由 $k|nx$ 得 $\frac{k}{gcd(k,x)}|n$，令 $C_k=\frac{k}{gcd(k,x)}$ 则有 $n=p*C_k(p\in N^*)$。推导：

$$k=\frac{nx}{d}\Rightarrow n=\frac{dk}{x}\Rightarrow n\leq\frac{bk}{x}\Rightarrow p\leq\frac{bk}{xC_k}$$

令 $\frac{bk}{xC_k}$ 为 p 的上界 lim。

再考虑一个比 k 大的 j, 满足

$$k<j<x,k|nx,j|nx\Rightarrow j|C_kpx\Rightarrow\frac{j}{gcd(j,C_kx)}|p$$

记 $\frac{j}{gcd(j,C_kx)}$ 为 $D_{k,j}$，即 $D_{k,j}|p$。

现在我们想要统计所有的 S_k。S_k 要计算合法的 N，等价于计算合法的 p。那么

$$S_k=lim-|\{p|p\leq lim,\exists D_{k,j}|p\}|$$

容斥一下即可得

$$S_k=\sum_{\{j_1,j_2,\ldots,j_t\}\subseteq\{k+1,k+2,\ldots,x-1\}}(-1)^t\frac{lim}{lcm(D_{k,j_1},D_{k,j_2},\ldots,D_{k,j_t})}$$

直接暴力计算肯定不行。加上一堆常数优化：

- 对于有倍数关系的 $D_i|D_j$ 我们去掉 D_i。
- 我们一个一个 D 值加入。然后我们可以将之前 D 的 lcm 值相同的项合并起来一起搞，对于 lcm 大于 lim 的状态直接扔掉。
- 对于相同的 x 的询问合并起来做。

54.5 算法流程

- 把 x 相同的询问一起搞
- 枚举 k，算 C_k 和 $D_{j,k}$
- 一个一个 $D_{j,k}$ 加入状态里，最后更新答案

54.6 复杂度

$O(?)$

55 The Baking Business

55.1 编号

BAKE

55.2 题目大意

给出 N 个操作。操作有订单或询问两种。

订单格式如下：

I 产品编号 [. 大小编号] 省编号 [. 城市编号 [. 地区编号]] 性别年龄出售数量

产品细节例如 6.2 表示第 6 号产品，2 号大小。下一个是位置例如 8.18.4 代表 8 号省的 18 号城市的 4 号区域。然后顾客的性别为 M 或者 F，年龄从 1 到 90。注意所有的编号都是从 0 开始。还需要注意方括号内的部分是可选的，因为某些出售的这些信息丢失了。出售数量不会超过 100。

询问的格式如下：

Q 产品编号 [. 大小编号] 省编号 [. 城市编号 [. 地区编号]] 性别起始年龄 [-结束年龄]

这询问在该范围下的出售总数。如果可选部分缺失的话，那么它意味着询问在该限制下的所有出售总数。对于年龄参数来说，如果结束年龄缺失，那么这个询问只询问年龄 = 起始年龄的顾客的购买总数，否则就是所有在此年龄范围内的顾客的购买总数。比较特殊的是如果产品编号为 -1，意思为所有的商品，类似的如果省份为 -1 那么就是所有的省份。除此之外其它参数不会为 -1.

有 10 种产品，每种都有 3 种不同的大小。有 10 个省份，每个省份可以被划分为 20 个城市，每个城市又可以被划分成 5 个地区。

范围是 $N \leq 10^5$。

55.3 关键词

乱搞

55.4 题解

直接用一个 6 维数组存下所有的订单（把年龄和性别用一维表示）。如果某一维的下标为 0 则表示该值为当前下标任意的所有订单之和。

对于订单操作就枚举那一维是 0 插入即可。

对于询问操作就到对应的位置，然后暴力扫描最后一维查询即可。

55.5 算法流程

如上述。

55.6 复杂度

$O(N)$

56 Dynamic GCD

56.1 编号

DGCD

56.2 题目大意

维护一棵大小为 N 带权树。有 Q 个操作，操作分两种：

- 计算从 u 到 v 路径上权值的 gcd。
- 将 u 到 v 路径上的权值加 d。

范围是 $N, Q \leq 50000$。

56.3 关键词

树链剖分

56.4 题解

考虑一条链上怎么做。由于 $gcd(a,b,c,\ldots,z)=gcd(a,b-a,c-b,\ldots,z-y)$。所以直接将权值差分一下。这样询问和修改都很简单了。

树上的话，用轻重链剖分之后重链优先的 DFS 序跟单链类似的搞就可以了。

56.5 算法流程

如上述。

56.6 复杂度

$O(Nlog^2N)$

57 Evil Book

57.1 编号

EVILBOOK

57.2 题目大意

有 N 个怪。打掉第 i 个怪消耗 C_i 点体力，得到 M_i 元。怪只能打掉一次。可以消耗 X 元使得任意一个怪的体力和奖励乘以 1/3(变成实数)，该操作不限次数。问至少需要多少体力能得到 666 元。

范围是 $N\leq 10$

57.3 关键词

搜索

57.4 题解

可以发现对于一个最优的打怪顺序，使用操作的次数单调不减。

直接搜索，维护状态：当前至少需要使用多少次操作，哪些怪没死，剩余的钱，已经付出的体力。直接爆搜，加些最优化剪枝和合法性剪枝即可。

57.5 算法流程

如上述。

57.6 复杂度

$O(?)$

58 Different Trips

58.1 编号

DIFTRIP

58.2 题目大意

一棵大小为 N 的有根树，每个点的“字符”设为它的度数。一条可行路径定义为一个点到它的某一个祖先的路径，一条可行路径能代表一个字符串。问可行路径中，有多少不同的字符串。

58.3 关键词

SAM

58.4 题解

直接类似 trie 树上建 SAM。然后用 $len[p] - len[fail[p]]$ 加起来即可。注意字符集大小，用 map 存转移边。复杂度不保证（?）。

58.5 算法流程

如上述。

58.6 复杂度

$O(?)$

59 Ciel and Earthquake

59.1 编号

CIELQUAK

59.2 题目大意

一个四联通的网格图，地震时每一条边都有独立的 p 的概率坏掉。求地震之后 (1,1) 点和 (R, C) 点仍然联通的概率。范围是 $R \leq 8, C \leq 10^{18}$

59.3 关键词

轮廓线 DP，马尔科夫链

59.4 题解

首先对 C 比较小的情况，直接用经典的轮廓线 dp，维护连通性即可。大概是当我们做到 (x, y) 点的时候，同时维护 $(1, y), (2, y), \ldots, (x, y), (x+1, y-1), (x+2, y-1), \ldots, (R, y-1)$ 这 R 个点的联通情况。然后枚举 $(x, y+1)$ 点和当前轮廓线相连的两条边的情况转移即可。注意 (R, y) 到 $(1, y+1)$ 的转移。

其实可用的联通状态不会很多。我们维护 R 个点每个点所属的联通块，用最小表示法表示联通状态。用 0 联通块表示和 $(1, 1)$ 联通的。注意联通块嵌套的时候状态不合法。比如说状态 abab，由于我们的图是平面图，a->a 的路径和 b->b 的路径必定相交，所以 a 和 b 应该同属一个联通块，状态不合法。这样搞出来的状态不会超过 4000 个。

令 $S(r,c)$ 表示 (r,c) 点和 $(1,1)$ 联通的概率。可以证明 $\frac{S(r,c+1)}{S(r,c)}$ 这个值收敛。所以我们只需要计算到大概 $c=50$，然后后面的直接计算即可。注意常数问题。

59.5 算法流程

- 搜出合法状态
- 枚举每个状态在每个不同的横坐标的转移
- 轮廓线 DP，当 $c>50$ 的情况就直接计算。

59.6 复杂度

$O(RS)$，S 表示状态数。带 50 的常数。

60 Substrings on a Tree

60.1 编号

TSUBSTR

60.2 题目大意

一棵大小为 N 的有根树，一条可行路径定义为一个点到它的子树中的某一个点的路径，一条可行路径能代表一个字符串。问可行路径中，有多少不同的字符串。然后给出 Q 个询问。每个询问先给出一个 26 个字母的排列，然后询问按照这个顺序排第 K 小的字符串是什么。范围是

60.3 关键词

SAM

60.4 题解

直接类似 trie 树上建 SAM。然后用 $len[p]-len[fail[p]]$ 加起来即可。对于第二类询问，我们先将 SAM 拓扑排序，然后求出一个节点后继能走

出多少个后缀。然后对于一个询问，按给定的顺序扫描当前节点的后继节点，然后判断后缀数量即可。找到靠谱的就继续往后走即可。

60.5 算法流程

如上述。

60.6 复杂度

$O(N)$

61 Annual Parade

61.1 编号

PARADE

61.2 题目大意

一张 N 个点，M 条有向边的图，现在可以走若干次。每一次从任意一个点开始，从任意一个点结束，且每经过一条路都要花费这条路的路费。如果一次行走。开始和结束的点不同，则额外要花费 K 元。如果一个点从来没有被经过，同样要额外花费 K 元。给出 Q 个不同的 K，求对于每一个询问的最小花费。

范围是 $N \leq 250, M \leq 30000, Q \leq 10000$。

61.3 关键词

费用流

61.4 题解

首先传递闭包。可以发现最优解的路径一定可以转换成传递闭包之后的图中一组不相交的路径。那么直接在这个新图中做即可。

接着考虑图中 K 的限制。实际上，如果我们把条件换成“在新图中一个点如果没有被走进过就要付出 K 的代价”和原来的条件等价。而且路径不相交，所以一个点最多斌走进一次。所以直接拆点，把一个点拆成出点

和入点，入点和源点连容量 1 费用 0 的边，出点和汇点连容量 1 费用 0 的边。对于新图中的有向边则相应连容量 1 费用即为路费的边。然后跑费用流。观察每一次流，本质上都是流经一条边。相当于我们每次用一条边的代价去走进一个点，减少 K 的费用。所以如果当前流所需费用大于 K，那么剩下的未被走进的点我们就花费 K 的费用来避免走一条费用更大的边。

由于有多个询问，我们发现询问之间互不影响。只需要对询问的 K 排序之后然后扫描即可。

61.5 算法流程

- Floyd 算法传递闭包，构出新图。
- 按上述方法建网络流图。
- 将询问排序，准备扫描。
- 跑费用流。若当前被扫到的询问的 K 比当前流一次的费用要小则可以回答该次询问。一直做费用流和扫描直到达到最大流或是没有询问为止。

61.6 复杂度

$O(maxflow(N, N^2))$

62 Knight Moving

62.1 编号

KNGHTMOV

62.2 题目大意

一个无限的棋盘上，一个棋子在 $(0,0)$ 处要走到 (X,Y) 处。它有两种走法：设当前所在点 (u,v)，第一种是走到 $(u+x1, v+y1)$，第二种是走到 $(u+x2, v+y2)$。然后还有 K 个障碍格不能经过。问行走的方案数。范围是 $K \leq 15$，坐标绝对值不超过 500。

62.3 关键词

容斥，线性方程组，DP

62.4 题解

相当于是给出了两个二维向量 $(x1, y1), (x2, y2)$，求一个非负系数的线性组合得到 (X, Y)。分情况讨论：

- 两个向量不在同一方向上。那么直接求解线性方程组即可。无解或无穷解的情况很好判断。至于障碍点的话直接容斥即可。
- 两个向量在同一方向上。那么问题转换成一维的模型。我们把有用的点映射到一维数轴上面，注意到有用的坐标范围不会超过 $500 * 500$，直接在这个数轴上暴力做，判断连通性来判断是否有解，判断有无环来判断是否无穷解，以及在上面 DP 来计算方案数。
- 注意两个向量相同或一个向量是 $(0, 0)$ 的情况，都很好处理。

62.5 算法流程

如上述分情况讨论即可。

62.6 复杂度

$O(2^K + X^2)$，X 是坐标范围。

63 Luckdays

63.1 编号

LUCKYDAY

63.2 题目大意

给定一个数列的前两项 $S_1 = a, S_2 = b$，再给出递推公式 $S_i = (X * S_{i-1} + Y * S_{i-2} + Z) \bmod P (i >= 3)$。再给出常数 C 和 Q 个询问，每个询问要求统计 $L_i \le k \le R_i$ 中有多少个 $S_k = C$。

范围是 $P \le 10007, Q \le 20000, L_i \le R_i \le 10^{18}$。满足 P 是质数。

63.3 关键词

矩阵乘法，BSGS

63.4 题解

首先我们考虑常规求这个线性递推数列的方法是矩阵乘法。在这题里面，我们令矩阵 $A = (S_i, S_{i-1}, 1)$，然后转移矩阵就很好写出来，不妨设为 B。考虑到虽然 L, R 很大，但是 P 很小，循环节长度不会很长，大概是 $O(P^2)$ 级别的。这启示我们使用大步小步算法来求出循环节。要求解数列的循环节长度，相当于求解矩阵方程 $B^x = I$。我们直接用矩阵的大步小步算法即可，这里的复杂度是 $O(P)$ 的。

有了循环节，我们考虑一个循环节里面出现了多少次 C。我们同样是使用矩阵的大步小步算法，求解 $AB^x = (C, i, 1)$，这里的 i 是我们枚举的 S_{k-1}。这里的复杂度是 $O(P^2)$ 的。

注意这里矩阵大步小步算法有挺多细节。平时我们是令 $A^{xw+y} = B$，我们令 A^{xw-y}，这样就可以不用对矩阵求逆。还有哈希的时候只哈希有用的位置来优化常数。

有了循环节长度和在一个循环节里面 C 出现的位置，我们只需将这些位置排好序，然后对每一个询问，先转化成前缀询问，然后去掉完整循环节的部分，然后剩下的部分二分即可。

注意 $Y = 0$ 的情况，找循环节随便搞搞即可。

63.5 算法流程

- 判断 $Y = 0$ 的情况
- 预处理出 $B^i (0 \le i \le P)$
- 用大步小步算法计算得到循环节和一个循环节内 C 的出现位置
- 对于一个询问，转成前缀询问之后二分查询

63.6 复杂度

$O(P^2 + QlogP)$，注意常数问题

64 A Game of Thrones

64.1 编号

GTHRONES

64.2 题目大意

给出 N 个数 a_i。然后两个人不停的选数。先手任选一个数开始游戏，然后轮流选。一个数最多被选 c_i 次。不妨假设上个人选的数是 a，当前选的数是 b，则 a, b 必须满足 $a = pb$ 或 $b = pa$，p 是任意一个质数。直到一个人不能选时游戏结束，不能选的人输。问先手是否能赢，如果能赢输出他一开始选的数最小是哪一个。

范围是 $N \leq 500, a_i \leq 10^{18}, c_i \leq 10^9$。

64.3 关键词

网络流，Miller-Rabin 算法

64.4 题解

考虑能连选的数连边。每个数按它质因子个数分类，那么一个数只会和它相邻的类连边，显然这个是二分图。那么我们暴力 $O(N^2)$ 连边就好了，连边的时候用 Miller-Rabin 判断质数。

连了边之后，发现如果图有完美匹配，那么先手一定输（后手一直沿着匹配边走即可），而当图没有完美匹配的时候，先手找一个没有匹配边的点开始。然后类似于上面后手的策略，一直沿着匹配边走就能赢了。所以问题在于找到一个最小的点它可以不在匹配集里面。我的方法是二分出这个点，然后判断从 1 到这个点所构成的子图是否有完美匹配，找到最小的没有完美匹配的子图即可。每次都跑一遍网络流即可。

64.5 算法流程

如上所述。

64.6 复杂度

$O(N^2 + maxflow(N, N^2) * logN)$

65 Find a Subsequence

65.1 编号

FINDSEQ

65.2 题目大意

给出长度为 N 的序列 A。再给出长度为 5 的一个排列 $B_1, B_2, \cdots, B_5$。要找到一个 A 的长度为 5 的子序列 $A_{i1}, A_{i2}, \cdots, A_{i5}$，这个子序列的 5 个数互相的大小关系和 B 要一样。输出一组解。范围是 $N \leq 1000$。

65.3 关键词

乱搞

65.4 题解

枚举 A_{i2}, A_{i4}。然后考虑贪心的取 A_{i1} 和 A_{i5}。大概是前后缀中取一个不大于某个数的最大值或是不小于某个数的最小值，这 4 个数组都可以在 $O(N^2)$ 的时间预处理出来。

这样就可以定下来 A_{i3} 的取值范围。那么我们的问题就是在某一段区间中询问是否有某个取值范围的数。这个同样可以转换成前缀询问用 $O(N^2)$ 预处理，$O(1)$ 询问。然后找到了一组解之后，暴力扫描一遍就可以得到这组解的位置。

65.5 算法流程

如上所述。

65.6 复杂度

$O(N^2)$

66 Short

66.1 编号

SHORT

66.2 题目大意

给你两个数 n,k，你需要找出所有的数对 (a,b)，满足 $n<a<k, n<b<k$，并且 $ab-n$ 可以被 $(a-n)(b-n)$ 整除。范围是 $N \leq 10^5$。

66.3 关键词

数论乱搞

66.4 题解

题目等价于找出所有 (c,d) 满足 $1 \leq c,d \leq n-k$，并且 $cd|(c+n)(d+n)-n$。

不妨设 $c<d, kcd=(c+n)(d+n)-n$。可得 $d=\frac{cn+n^2-n}{kc-c-n}>c$。再由于 $kc-c-n>1$，这两个条件联立起来可以得到 $c \leq 3n$。大概就是解个一元二次不等式就可以了。首先我们可以枚举 c，直接将 $cn+n^2-n$ 的所有因数全部检验一遍。当 c 比较大的时候，我们发现算出来的 k 很小。所以当我们枚举的 c 比较大的时候就不要枚举因数，直接枚举 k 判断即可。大概阈值设在 4000 左右就能过。

66.5 算法流程

如上所述。

66.6 复杂度

$O(Nlim)$，lim 是阈值，实际上达不到理论上界。

67 Expected Maximum Matching

67.1 编号

MATCH

67.2 题目大意

给出一个左边 N 个点，右边 M 个点的二分图，再给出 $p[i][j]$ 表示左边 i 点和右边 j 点之间右边的概率。求这个图最大匹配的期望。范围是 $N \leq 5, M \leq 100$。

67.3 关键词

Hall 定理，状压 DP

67.4 题解

这题要用到 Hall 定理。它的表述是一个大小为 K 的任意一边子图的子集 S 有完备匹配的充要条件是它的任意一个子集，不妨设为 S'，则与之相邻的另一边的点集大小 $N \geq |S'|$。

那么由这个定理，我们就可以设 $f[i][S]$ 表示右边子图做到第 i 个点，左边子图的每一个子集是否满足 Hall 定理的条件，这样的概率是多少。那么转移的时候直接枚举当前点与左边的链接情况直接对 S 进行修改即可。注意，直接枚举状态的话状态会达到 2^{2^5} 这么多。其实合法的情况没有这么多，一个简单的剪枝就是如果一个集合满足条件，那么它所有的子集都必须满足条件，否则状态非法。剪枝之后状态就只有几千了。

67.5 算法流程

- 预处理出左边子集的各种状态，注意剪枝
- 对于右边子图进行转移
- 处理到最后直接由状态得到最大匹配，统计答案

67.6 复杂度

$O(M * 2^{2^N})$，实际上远远不到

68 Two Magicians

68.1 编号

MAGIC

68.2 题目大意

有两个人在一张 N 个点，M 条边的无向图的两个不同点开始游戏。轮流进行一个回合。若回合开始两人属于一个联通块则当前人获得胜利。然后他必须连接一条原来没有的边。最后可以他可以瞬间移动，每个人至多移动 P 次。问谁能赢。范围是 $N \leq 7777, M \leq 10000$，至多 100 组数据。

68.3 关键词

状压 DP，找规律

68.4 题解

考虑 DP。当前状态只需要存当前人在的联通块的奇偶性，另一个人所在联通块的奇偶性，两人剩下的传送次数，其它奇数块的数量和偶数块的数量。转移很简单。发现一些规律：首先一个人最多使用一次传送，然后当偶数块大于 10 的时候 DP 值不变，奇数块大于 10 的时候 DP 值以 4 为周期循环。这样状态只有常数个了。

所以对于每一个询问，用并查集搞搞，然后直接带入求值即可。

68.5 算法流程

- 预处理出 DP 值
- 用并查集求出所有联通块的状态，直接带入求值

68.6 复杂度

$O(N+M)$

69 Billboards

69.1 编号

BB

69.2 题目大意

要构造一个长度为 N 的 01 序列，要求它所有长度为 M 的连续子序列都有 K 个 1。求 1 的个数最少的构造方案有多少种。范围是 $M \leq 50, N \leq 10^9$。

69.3 关键词

杨氏矩阵，分情况讨论

69.4 题解

先考虑 $M|N$ 的情况。设分成了 $P = \frac{N}{M}$ 段。显然 1 最少的时候有 PK 个，每段里面都要放 K 个。然后我们构造一个 $K * P$ 的矩阵 A，$A_{i,j}$ 表示第 j 段的第 i 个 1 在这一段的位置。那么如果序列合法，必须满足矩阵每一列单增，每一行单调不增（因为满足每一个长度为 M 的连续子序列都要有 K 个 1）。而且一个矩阵恰好对应一种合法方案。

这是半标准杨氏矩阵的方案数计算。一个大小为 $N * M$ 的矩阵的方案数是 $ans = \prod_{i,j} \frac{r+j-i}{N-i+1+M-j+1}$。这个计算我们发现很多项都可以约掉。实际上有用的只有 KM 个，直接算。

对于 $M \nmid N$，设 T 表示 $M \bmod M$。如果 $T \leq M-K$，那么每一组前 T 个一定是 0. 如果 $T \geq M-K$，我们增加一组，然后每一组后 $M-T$ 个都是 1。这样同样是转成上面的情况直接做。

69.5 算法流程

如上所述。

69.6 复杂度

$O(KM)$

70 Children Trips

70.1 编号

TRIPS

70.2 题目大意

给出一棵带边权的大小为 N 的树，再给出 Q 个询问，每次询问给出 x, y, z 表示一个人从 x 出发，要走到 y，他一天最多走的长度是 z，而且每天结束的时候他必须停在某一个点上。即他一天有可能走不到他能走到的最长长度。询问这个人需要走多少天。

70.3 关键词

根号算法，预处理

70.4 题解

对于 z 大于 $\sqrt{N}$ 的询问，答案不会超过 $\sqrt{N}$。因此可以预处理出每个点向上走 2^i 的距离能走到哪里。然后从 LCA 处拆成两条链，直接暴力向上走即可。

然后对于 z 小于 $\sqrt{N}$ 的询问，我们把所有 z 相同的询问放到一起做。可以直接搞出一个点向上走 z 的距离能走到哪里，然后再处理出每个点向上走 2^i 天能走到哪里。然后对于每一个询问直接 $O(logn)$ 的倍增算法就可以处理了。

70.5 算法流程

如上所述。

70.6 复杂度

$O(N\sqrt{N}logN)$

71 To challenge or not

71.1 编号

CHAORNOT

71.2 题目大意

请你找这样一组整数，里面没有 3 个数构成等差数列。让问题更难些，你必须从给定的 M 个整数 $B_1, B_2, \ldots, B_m$ 中选取。不一定要最大数集，但是数集越大分数越高。范围是 $N, B_i \leq 10^5$。

71.3 关键词

乱搞

71.4 题解

考虑到最优解不会很大，我们可以考虑各种乱搞方法。我的方法是排序之后，将一个数直接插入，然后如果不合法就抛弃掉。一直做到最后。这个方法已经能得到不错的结果。优化就是我们对于一个合法的数字，我们以一个较小的概率抛弃它 (比如 0.02)，这样随机几次取最优解，这样做更好。

然后还有别的方法是三分这个概率然后多搞几次，看上去会更加优秀。

71.5 算法流程

如上所述。

71.6 复杂度

$O(Tm + Tn^2)$，n 是求出数集的大小

72 Stepping Average

72.1 编号

STEPAVG

72.2 题目大意

考虑以下的方法求出 N 个数的“迭代平均数”。每次拿出其中任意两个数，并用他们的平均数取代他们，重复 $N-1$ 次直到最后只剩一个数。我们叫这个剩下的数叫“迭代平均数”。值得注意的是，不同的合并顺序最后会导出不同的“迭代平均数”。

你要做的是：给你 N 个数。找到一个合并的顺序使得“迭代平均数”尽量接近给定的数 K。保证 $N = 1000$

72.3 关键词

乱搞

72.4 题解

合并的方法有很多，我选择的是将数字定下某一个顺序之后从前到后合并。发现这样合并，最后一个数字对最终平均值的贡献系数是 1/2，倒数第二个是 1/4，以此类推。然后可以发现贡献系数太过小的数字对最终的平均值几乎没有影响。因此我们只考虑最大的 30 个数字，其他的随便放即可。

而对于前 30 个数字，可以采用随机的办法。随机乱取是不可取的。我的办法是先随机出一组解，然后从贡献系数最大的位置开始往后做，每次对于一个位置，我们可以随机一个位置的数字与当前位置交换，判断是否更优。如果是则交换。然后对于一个位置重复 3000 次左右。这样子相当于不停的优化这组解。这样得出来的结果会很好。选取多次次初始解然后重复这个过程，这样就可以得到很不错的解了。

72.5 算法流程

如上所述。

72.6 复杂度

$O(NT)$，T 是随机次数

73 Deleting numbers

73.1 编号

DELNMS

73.2 题目大意

假设当前有 n 个数形成的数组 $a[n]$，每次可以选择 v,t 两个数，满足 $v+t \leq n+1$，且令 k 为最大的满足 $v+kt \leq n$ 的数，必须满足 $a[v] = a[v+t] = a[v+2t] = \cdots = a[v+kt]$。然后这些数将会删除 (若 $v+t = n+1$ 则只删除第 v 个数)，之后形成新的由 $n-k$ 个数形成的数组 $a[n-k]$，满足这个数组的元素是原来数组被删除之后剩下的元素，且相对位置保持不变。要求输出一组方案，使得操作次数尽可能少。范围是 $n \leq 10^5$。

73.3 关键词

乱搞

73.4 题解

最简单的就是直接倒序删除。稍微优化一点就是找出出现次数最多的数，然后删掉别的数字，当剩下一种数字（即出现最多的数字）时候一次删掉。也可以贪心的做。每次找到最长的可删除的序列删掉。贪心的解是比较优秀的。

然而直接做贪心的复杂度是 $O(n^2 logn)$，可能会超时。所以为了避免超时我们牺牲一点答案的最优性，考虑将整段数分成 k 个小段。然后我们从后往前每次删掉整一个小段（相当于一个较小的子问题）。这样做就能在避免超时的前提下得到不错的解。

73.5 算法流程

如上所述。

73.6 复杂度

$O(n^2logn/k^2)$

74 Closest Points

74.1 编号

CLOEST

74.2 题目大意

给出平面上 N 个固定点和 Q 个询问。询问会给出一个询问点，求离这个询问点最近的固定点是哪一个。在时限内要回答尽量多的询问 (非在线)。你回答正确的询问越多得分越高。保证 $N = Q = 50000$。

74.3 关键词

KD-Tree

74.4 题解

一开始我的想法是将询问点和固定点一起按照横纵坐标和排序，然后对于每一个询问点，前后暴力 2000 次左右。这样看上去能搞不少分数，然而实际效果不怎么样。我又尝试了好几种排序方式，但是没有较好的思路。

当然，我们可以使用 KD-Tree 的方法，我们将固定点构出一颗 KD-Tree，然后直接询问，在给定时限内是无法全部询问完的，所以加上卡时，在时限内回答尽量多的询问，剩下的随机输出。我的 KD-Tree 是要每次按照方差最大的一维来划分，这样能更优，然后别人讲了一个常数优化，我们可以每次将区间的开头作为这颗子树的根来优化内存访问。

然后有一个最大的优化，我们发现 KD-Tree 其实有点类似于一个带了最优性剪枝的暴力 (当做到某一个平面上的子矩阵范围，这个平面范围和询问点的距离已经大于当前的答案就无需扫描这部分固定点)。实际上我们可

以一开始对于每一个询问点，随机选取几百个固定点，得到一个不错初始答案，这样就能大大优化查找的时间，来得到更多的精确答案。这样做能得到很不错的效果。

74.5 算法流程

如上所述。

74.6 复杂度

$O(N\sqrt{N})$，这是 KD 树的期望复杂度

75 Similar Graphs

75.1 编号

SIMGRAPH

75.2 题目大意

给出两张无向图的相似度的定义: 相似度等于两张带标号图中的这样的点对 (a,b) 的数量：满足两张图中 a,b 两点之间都有边。现在给出两张大小都为 N 的无向图，现在要你对两张图重标号，使得两张图的相似度尽量大。范围是 $N \leq 75$，有一百组测试数据。

75.3 关键词

模拟退火 chagn'shi

75.4 题解

首先发现其实只需要对一张图重标号就够了。

一开始我的想法是直接随机交换两个标号，每次看看是不是能使相似度变大，如果能就一定交换，否则不交换，一直随机直到时限为止。然而这样的效果并不好。原因是很容易落到一个局部最优解就出不去了，而且这个题目的局部最优解非常多，而且他们实际上相似度都不是很高。

我又尝试以一个固定的概率 P，强制交换（即使它会使得解变差）。然后不停的调整 P 的大小，发现效果都很一般。原因是这样一直做下去，它很难停留在某一个较好的解，找到了一个不错的解没有在这个解的基础上进行微调，而是进行了“大刀阔斧”的更改。这样实际上很难找到一个非常好的解。

解决的办法就是模拟退火算法。实际上模拟退火算法非常好理解。它就是将上述方法的 P 在一开始的时候设定的很大，随着算法的进行 P 不停的变小，比如说可以在每次合法的交换过后令 $P* = 0.99$，这样在前期能够“广泛涉猎”，后期找到较优解的时候能“细致钻研”。这样搞出来的解就比较优秀了。

75.5 算法流程

如上所述。

75.6 复杂度

$O(NT)$,T 是交换次数。