IOI2015国家集训队第一次作业试题泛做

长沙市一中 张天扬

Contents

1 D9164 Codeforces 235C Cyclical Quest

1.1 题目大意

给定一个字符串s，再给n个字符串x。对每个x，求在s中有多少子串与它循环同构。

循环同构定义为，存在一种方法，将x的前若干字符依次放到末尾，使两个字符串相同。

1.2 题目解法

对s构造后缀自动机。将x倍长后在后缀自动机上运行，并维护当前长度。

如果失配，则沿fa指针回退。

如果当前长度达到x的原长度，那么就将当前节点的right计入答案，并将长度减一。如果减一后不在这个节点的范围内，就沿fa指针回退。

1.3 时空复杂度

时间复杂度：$O(len)$。空间复杂度：$O(len)$。

2 D9293 Codeforces 235D Graph Game

2.1 题目大意

给一个n个点n条边的图。我们每次将当前图的大小计入答案，然后随机选择一个点，删去它，再递归处理每个连通分量。求答案的期望值。

2.2 题目解法

我们不妨先考虑树的情况。

考虑$f(a,b)$表示当我们删去点a的时候，点b仍然与点a连通的概率。显然，如果这种情况发生，那么答案会增加1。所以我们只需要求$\sum f(a,b)$即可。

考虑(a,b)这条链上如果一共有x个点，那么对于这x个点，我们必须第一个随机到a。而其他点对此是没有影响的。因此，$f(a,b)=\frac{1}{x}$。

之后考虑环套树的情况。显然，如果某两点的路径不会经过环，那么$f(a,b)$的值与之前一样。而如果经过了环，不妨设a,b两点到环上一共经过了x个点，在环上两条路径分别经过y个点和z个点。那么我们要么对于$x+y$个点第一个随机到a，要么对于$x+z$个点第一个随机到a。根据容斥原理，$f(a,b)=\frac{1}{x+y}+\frac{1}{x+z}-\frac{1}{x+y+z}$。

2.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(n)$。

3 D8809 Codeforces 235E Number Challenge

3.1 题目大意

定义$d(n)$表示n的约数个数。

给定a,b,c，求$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}d(ijk)$.

3.2 题目解法

首先我们证明一个结论：

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}d(ijk)=\sum_{gcd(i,j)==1\&\&gcd(j,k)==1\&\&gcd(k,i)==1}\lfloor\frac{a}{i}\rfloor\lfloor\frac{b}{j}\rfloor\lfloor\frac{c}{k}\rfloor$$

不妨令$f(a,b,c)=\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}d(ijk)$，令$g(a,b,c)=\sum_{gcd(i,j)==1\&\&gcd(j,k)==1\&\&gcd(k,i)==1}\lfloor\frac{a}{i}\rfloor\lfloor\frac{b}{j}\rfloor\lfloor\frac{c}{k}\rfloor$。

为了证明$f(a,b,c)=g(a,b,c)$，事实上只需要证明：

$$\begin{aligned}&f(a,b,c)-f(a-1,b,c)-f(a,b-1,c)-f(a,b,c-1)+f(a-1,b-1,c)+f(a-1,b,c-1)+\\&f(a,b-1,c-1)-f(a-1,b-1,c-1)\\=&g(a,b,c)-g(a-1,b,c)-g(a,b-1,c)-g(a,b,c-1)+g(a-1,b-1,c)+g(a-1,b,c-1)+\\&g(a,b-1,c-1)-g(a-1,b-1,c-1)\end{aligned}$$

证明了这一结论，就可以通过类似数学归纳法的方法证明$f(a,b,c)=g(a,b,c)$。

我们观察上式的左侧，可以发现它等于$d(ijk)$。

我们观察上式的右侧，适当变换后可以发现它等于

$$\sum_{gcd(i,j)==1\&\&gcd(j,k)==1\&\&gcd(k,i)==1}(\lfloor\frac{a}{i}\rfloor-\lfloor\frac{a-1}{i}\rfloor)(\lfloor\frac{b}{j}\rfloor-\lfloor\frac{b-1}{j}\rfloor)(\lfloor\frac{c}{k}\rfloor-\lfloor\frac{c-1}{k}\rfloor)$$

而上式事实上求的是三元组(i,j,k)的个数，要求满足：$gcd(i,j)==1\&\&gcd(j,k)==1\&\&gcd(k,i)==1$ 且$a\%i=b\%j=c\%k=0$。

考虑某一个质数p。我们令x为最大的整数使得p^x为a的因子。同理我们定义y 和z。

对于左侧的$d(i,j,k)$，容易知道质数p 对其贡献为$x+y+z+1$，$d(i,j,k)$为所有质数贡献之积。这是一个经典问题，在此不赘述。

对于右侧所求的三元组$d(i,j,k)$的数量，同样可以发现质数p 对其贡献为$x+y+z+1$。这是因为我们考虑i,j,k中质数p 的指数，可以发现共有4 种方法，分别为$(0,0,0)$，$(n,0,0)$，$(0,n,0)$，$(0,0,n)$。（n表示某一正整数）。那么一共就有$x+y+z+1$种方法。

于是我们就证明了$f(a,b,c)=g(a,b,c)$。

我们考虑如何求解$g(a,b,c)$。首先我们枚举i，那么我们要求的就是

$$\sum_{gcd(i,j)==1\&\&gcd(j,k)==1\&\&gcd(k,i)==1} \lfloor\frac{b}{j}\rfloor\lfloor\frac{c}{k}\rfloor$$

考虑使用容斥原理计算这一式子。我们枚举j,k 的公因子s，要求s 中不含有平方因子且与i 互质。由于我们要求的是所有最大公因数为1 的j,k，所以我们应该用所有s 中含有偶数个因子的计算值减去所有s中含有奇数个因子的计算值。

枚举s之后，我们只需要统计在$[\frac{b}{s}]$ 和在$[\frac{c}{s}]$ 中分别有多少个和i 互质的数，对于这一点我们直接暴力计算即可。

这样做的时间复杂度大约是介于$O(ab)$ 和$O(ab\log b)$之间的，可以通过本题。如果哪位能算出准确复杂度，不胜感激。

3.3 时空复杂度

时间复杂度：$O(ab\log b)$。空间复杂度：$O(1)$。

4 D8807 Codeforces 238D Tape Programming

4.1 题目大意

给定一个程序，由数字和′ >′,′ <′组成。我们每次询问一个子串。

一开始，指针指向子串的最左侧字符，方向为向右。

如果当前指向的是一个数字，那么我们输出它，将这个数字减一。如果它原本是0，就删除这个数字。

如果当前指向的是′ >′或′ <′，就按箭头指向改变方向，如果下一个位置仍然不是数字，就把这一个箭头删掉。

求这一个子串中每个数字会输出多少次。

4.2 题目解法

考虑运行整个程序。

我们发现，我们曾经到过的位置总是程序的一个前缀。

那么我们对每个位置，记录下第一次到达它且方向是向右的时候，以及第一次到达它且方向是向左的时候，每个数字分别输出了多少次。那么对于一个子串，我们在这一段中每个数字输出的次数，总是等于我们第一次运行到这个子串外的时候的次数减去第一次运行到这个子串内的次数。

于是这个问题就得到了解决。需要注意的是，我们从头开始执行整个程序时，可能只能经过这个程序的一个前缀，那么我们需要从第一个没有到过的字符开始再次执行，知道不存在没有到过的字符为止。

4.3 时空复杂度

时间复杂度：$O(nD+qD)$。空间复杂度：$O(nD)$。

（D指数字种数，这里为10）

5 D9233 Codeforces 241D Numbers

5.1 题目大意

给定一行n个数，是$1-n$的一个排列，要求选出一些数，满足它们的异或和为0，且将它们从左到右写成一个数后是p的倍数。

5.2 题目解法

可以证明：对于$n \geqslant 25$我们只需要在$1-25$内做选择，就能满足任意p都有解。

这是因为，对于$1-25$的数，有超过10^6个子集满足异或和为0。而将它们从左到右写成一个数后对p取模，实际上近似的可以看作一个$[0,p)$的随机整数。

那么因为p的最大值只有5×10^4，期望意义上，我们就有20个选择方法能够满足题目要求。

5.3 时空复杂度

时间复杂度：$O(2^{25})$。空间复杂度：$O(1)$。

6 D9227 Codeforces 241E Flights

6.1 题目大意

给定一个DAG，要求给每条边定长度为1 或2，使所有从1到n 的路径长度均相同。

6.2 题目解法

首先我们去掉所有无用点，也就是无法到达n或者无法从1到达的点。

设$d[i]$为1到i的最长路。对一条边(u,v)，我们有：$d[v] = d[u] + 1 \ \ or \ \ d[v] = d[u] + 2$。

考虑改造这一式子，我们得到：$d[v] \leqslant d[u] + 2, d[u] \leqslant d[v] - 1$。

那么我们就得到一个差分约束系统。使用Bellman-Ford 求解即可。

6.3 时空复杂度

时间复杂度：$O(nm)$。空间复杂度：$O(m)$。

7 D9204 Codeforces 241F Race

7.1 题目大意

给一个地图，包括建筑物、路口和道路。建筑物是不能经过的，路口经过需要1 分钟，道路会给出经过所需时间。我们会给定一个经过路口的顺序，以及开始和结束的位置。求经过k分钟后所处的位置。

7.2 题目解法

直接按照题目意思模拟就可以了。

不明意义的一题。

7.3 时空复杂度

时间复杂度：$O(nm+k)$。空间复杂度：$O(nm)$。

8 D8745 Codeforces 243C Colorado Potato Beetle

8.1 题目大意

给一个坐标系，我们从原点开始沿某路径移动，问被路径围起来的面积有多大。

8.2 题目解法

将所有涉及到的坐标离散化后，从无穷远处开始BFS，没有BFS到的就是被围起来的地方。

8.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(n^2)$。

9 D8744 Codeforces 248E Piglet's Birthday

9.1 题目大意

Winnie的家里有n个架子，每个架子上都有一些（或没有）蜜罐。初始时所有蜜罐都是装满蜜的。Winnie 一共去了q次架子；第i 次Winnie 会先去第u_i 个架子，拿走k_i 个蜜罐，把这些蜜罐中的蜜吃掉，然后把这些蜜罐都放到第v_i 个架子上。当Winnie拿走蜜罐时，他会在第u_i 个架子上所有k_i 个蜜罐的集合中等概率选择一个集合，然后把集合中的k_i 个蜜罐拿走。

Winnie想知道，每次操作后，架子上所有蜜罐都被吃完的架子的期望个数是多少。

9.2 题目解法

设$dp[i][j]$表示当前第i个架子上有j 个没吃的罐子的概率。假设我们有一次在第x个架子上拿k 个罐子，那么我们可以考虑：一个一个的拿罐子。那么就有：

$$dp[x][j] = dp[x][j] \times \frac{a[x]-j}{a[x]} + dp[x][j+1] \times \frac{j+1}{a[x]}$$

之后将$a[x]$减一，重复此过程k次即可。

9.3 时空复杂度

时间复杂度：$O(nak)$。空间复杂度：$O(na)$。

10 D9208 Codeforces 251D Two Sets

10.1 题目大意

给定n个数，要求把它们划分成两个集合，使两个集合异或和的和尽量大。在此前提下，要求第一个集合的异或和尽量小。

10.2 题目解法

不妨设所有数的异或和为t。

对所有元素，设x_i，为1表示它在第一个集合中，为0表示它在第二个集合中。

如果t的某一位是1，那么这一位对异或和的和没有影响。

如果t的某一位是0，我们希望在第一个集合中这一位的异或和是1，因为这样在第二个集合中这一位的异或和也是1。能使和更大。

所以：我们从高到低考虑t每个为0的位。如果这一位是1，那么必定有$xor\ x_i = 1$，要求a_i的这一位上是1。那么我们就可以用高斯消元的方法判定这个方程是否与之前的方程矛盾。如果不矛盾就保留。

之后，我们还需要让第一个集合的异或和尽量小。此时只需要考虑t中为1的位，我们希望第一个集合中这一位的异或值为0，那么和上面一样使用高斯消元判定即可。

10.3 时空复杂度

时间复杂度：$O(n\log^2 v)$。空间复杂度：$O(n+n\log^2 v)$。

11 D9238 Codeforces 253E Printer

11.1 题目大意

有n个任务，每个任务在t_i时间接到，要求打印s_i张纸，并有着p_i的优先级。其中有一个任务的优先级未知。每秒钟我们会在当前有的任务中选择一个优先级最大的，并打印一张纸。我们要求优先级未知的任务的完成时间是T，求它的优先级并求出所有任务的完成时间。

11.2 题目解法

我们可以二分未知的优先级。

之后，我们使用一个堆来维护当前的所有任务就可以了。

注意优先级必须互不相同，所以我们需要做一些处理来避免二分到与其他某个任务相同的优先级。

11.3 时空复杂度

时间复杂度：$O(n\log^2 n)$。空间复杂度：$O(n)$。

12 D8765 Codeforces 254D Rats

12.1 题目大意

有一个二维地图，有一些地方有障碍。我们可以选择其中的两个空格子，每个格子可以覆盖和它的距离不超过d的所有格子，有一些给定的格子要求全部被覆盖到，求一种方案。

12.2 题目解法

随便选一个要求覆盖到的格子，找出所有与它距离不超过d的格子，这些中的一个需要是选择的第一个格子。我们不妨枚举一个。

然后再随便选一个要求覆盖到而第一个格子没有覆盖到的格子，找出所有与它距离不超过d 的格子，这些中的一个需要是选择的第二个格子。我们同样枚举一个。

以上两步的枚举总量不会超过$4d^4$。最后使用BFS判断所有要求的格子是否都被覆盖即可。

12.3 时空复杂度

时间复杂度：$O(d^6 + d^2nm)$。空间复杂度：$O(nm)$。

13 D9223 Codeforces 256D Liars and Serge

13.1 题目大意

有n个人坐成一排，每个人要么说真话，要么说假话。现在问他们：你们里面有多少个人说真话？说真话的人会给出正确答案，说假话的人会从$1-n$里随机一个不正确的答案。问：有多少种答案序列满足可以看出至少有k个人说谎？

13.2 题目解法

如果说i的正好有i个人，那么这i个人并不一定说谎。否则，说i的人就全部都是说谎的。

那么我们考虑DP。$dp[i][j][k]$表示考虑说$1-i$的一共有j个人，其中有k个一定是说谎的。考虑枚举有几个说了i的。如果有y个说了i的，如果y不等于i，那么从$dp[i-1][j-y][k-y]$转移。否则从$dp[i-1][j-y][k]$转移。转移的时候要乘以C_j^y。

但是这个是$O(n^4)$的。注意到题目里要求n是2的整数幂次，而n又不超过256，且k不超过n。那么我们直接把$n=256$和$n=128$打两个表，剩下的暴力跑就可以了。

13.3 时空复杂度

时间复杂度：$O(n^4)$。空间复杂度：$O(n^2)$。

14 D9195 Codeforces 258D Little Elephant and Broken Sorting

14.1 题目大意

给定一个$1-n$的排列，有m次操作，每次给定两个不同的位置a,b，有$\frac{1}{2}$的概率交换两个位置上的数。求所有操作之后序列逆序对数的期望。

14.2 题目解法

我们设$p[i][j]$表示第i个位置上的数大于第j个位置上的数的概率。那么当我们操作a,b时，对于所有i，有$p[a][i]=p[b][i]=\frac{p[a][i]+p[b][i]}{2}$。而$p[a][b]=p[b][a]=\frac{1}{2}$。

那么就容易统计答案了。

14.3 时空复杂度

时间复杂度：$O(nm)$。空间复杂度：$O(n^2)$。

15 D9279 Codeforces 261D Maxim and Increasing Subsequence

15.1 题目大意

给定一个长为$n \times t$的序列a，其中前n 项已给出，后面是前n项的不断重复。求序列的最长上升子序列。

15.2 题目解法

令$f[i]$表示最后一项不超过i的最长上升子序列长度。那么对于一个位置i，有$f[a[i]] = f[a[i]-1]+1$，并用它更新所有$f[k](k > i)$。

注意一些小剪枝即可。其实还有另一种时间复杂度更好的做法，但是因为常数问题比这种比较暴力的做法更慢……

15.3 时空复杂度

时间复杂度：$O(n \times maxb)$。空间复杂度：$O(n)$。

16 D8791 Codeforces 261E Maxim and Calculator

16.1 题目大意

有两个数字a,b，一开始$a=1,b=0$。我们每次操作可以将a变为$a\times b$或者将b变为$b+1$。求有多少个$l\leqslant x\leqslant r$满足我们可以进行不超过p次操作将a变为x。

16.2 题目解法

显然我们得到的任意一个a都不会含有超过p的质因子。实践得出，当p取最大值100时，我们的a的个数只有大约3×10^6个。我们首先把所有这些数提取出来。

之后，我们使用一个DP：$dp[i][j]$表示当b的值为j时，让a的值为i最少需要多少次操作。那么$dp[i][j]=\min(dp[\frac{i}{j}][j],dp[i][j-1])+1(i \bmod j=0)$。

对j这一维使用滚动数组即可。

16.3 时空复杂度

时间复杂度：$O(np)$。空间复杂度：$O(n)$。

（n为最大质因子不超过p的数的个数，本题中约为3×10^6）

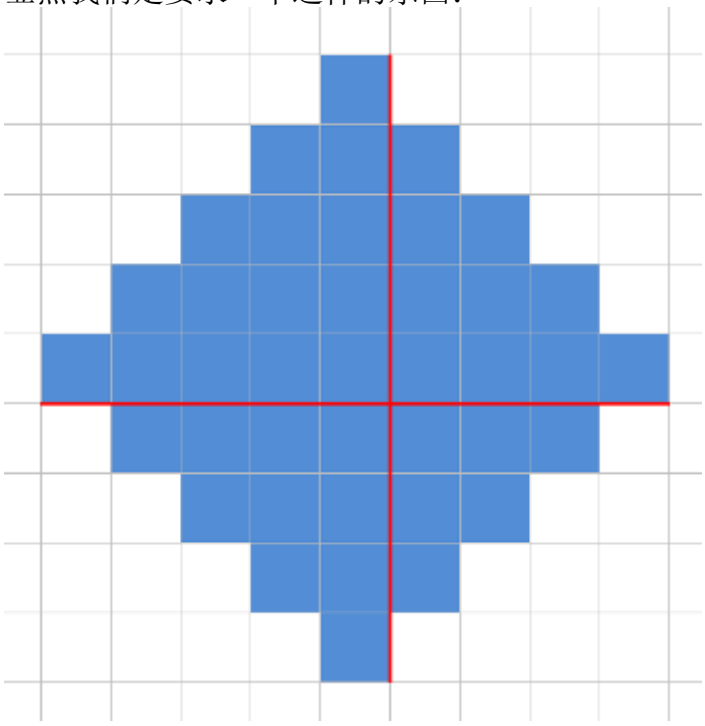
17 D8738 Codeforces 263E Rhombus

17.1 题目大意

给定一个$n \times m$的数字表格，要求找到一个位置(a,b)，满足$k \leqslant a \leqslant n-k+1$且$k \leqslant b \leqslant m-k+1$，且最大化$\sum_{i=1}^{n}\sum_{j=1}^{m} a_{i,j} \times \max(0, k-|i-x|-|j-y|)$。

17.2 题目解法

显然我们是要求一个这样的东西：



如图，我们可以把它划分成四个三角形，然后分别求解。

当我们求解某一个三角形的时候，记录一些状态就可以做到在$O(nm)$的时间内求出整个表格中每个位置的值。

具体来说，我们记录关于一列的值，以及一个三角形内部和。这样$ans_{i,j} = ans_{i,j-1} - sumtri_{i,j-1} + anscol_{i,j}$。

求四个三角形的过程事实上可以看作将整个表格旋转后再次求解。

这样就可以在$O(nm)$的复杂度内解决本题。常数比较大。

17.3 时空复杂度

时间复杂度：$O(nm)$。空间复杂度：$O(nm)$。

18 D8793 Codeforces 264D Colorful Stones

18.1 题目大意

有两排石头，每块石头有红蓝绿三种颜色。两个人一开始分别站在两排石头的第一个上。

每次我们可以选择一个颜色，每个人如果站在我们选择的颜色的石头上，就往后走一块石头。

求有多少种状态是可以达到的。

18.2 题目解法

我们猜测：对于第一个序列中的某一个位置，第二个序列中可以到达的位置是一个区间。

如果这个猜测是对的，那么我们就可以轻松的维护当前区间的范围，并统计答案了。

但是很遗憾，这个猜测并不正确。有且只有一种反例：

当第一个序列的末尾两个字符是xy，而第二个序列某位置的末尾两个字符是yx，且x和y 不为相同字符时，这一个位置不可达。

这种情况无法达到是容易证明的。我们更希望证明：其他情况一定可以达到。

不妨考虑两个序列的第一个字符。如果它们相同那么显然没有选择，只能同时前进。我们考虑它们不同的情况。设为c_1, c_2。

如果去掉c_1之后不是第二个序列向后扩展一位得到的序列的子序列，那么我们就可以去掉c_1，显然可以达成条件。去掉c_2 也是同样的。

那么接下来的问题只剩：去掉c_1和c_2 都不行。那么仔细思考可以发现：已经只剩下前面的那一种反例了！

因此我们就可以在线性的时间内推出第一个序列中每个位置对应的区间，然后减掉区间内的反例就可以了。

18.3 时空复杂度

时间复杂度：$O(n)$。空间复杂度：$O(n)$。

19 D8746 Codeforces 266D BerDonalds

19.1 题目大意

给一个无向带权图，求一个点（可以在边上），使它到最远的点距离最近。

19.2 题目解法

首先用floyd搞出两两的最短路。

然后我们枚举一条边(u,v)，考虑当我们求的点在这条边上时，最近距离是多少。我们把所有点按照到u的最短路长度来排序，然后按最短路长度从大到小枚举。考虑我们当前枚举的这一个点，以及所有到u的最短路长度大于它的点中到v的最短路长度最大的一个点。显然，当我们求的点在这条边上时，到这两个点的距离有可能是最大的。而为了使到最远的点最近，我们只需要让所求点尽量靠近这两个点的中点即可。

如果我们将所有与u相连的边一起考虑，那么它们就只需要排序一遍。这样复杂度就是$O(n^3 + n^2 \log n)$。

19.3 时空复杂度

时间复杂度：$O(n^3 \log n)$。空间复杂度：$O(n^2)$。

20 D8750 Codeforces 266E More Queries to Array...

20.1 题目大意

给定一个序列，支持两个操作：将一段区间赋为某个值，以及询问$\sum_{i=l}^{r} a_i(i-l+1)^k$。

注意k不超过5。

20.2 题目解法

$$\begin{aligned} & \sum_{i=l}^{r} a_i(i-l+1)^k \\ = & \sum_{i=l}^{r} a_i \sum_{j=0}^{k} C_k^j i^j (-l+1)^{k-j} \\ = & \sum_{j=0}^{k} C_k^j (-l+1)^{k-j} \sum_{i=l}^{r} a_i i^j \end{aligned}$$

于是我们用线段树维护对每个j的$\sum_{i=l}^{r} a_i i^j$。对每个询问，我们枚举j后按上式求解即可。

20.3 时空复杂度

时间复杂度：$O(km\log n)$。空间复杂度：$O(kn)$。

21 D8756 Codeforces 268D Wall Bars

21.1 题目大意

有一个管子，在$1-n$的每个整数高度，可以向四个方向中的一个伸出一根杆子。我们从地面上可以爬到任意一个$1-h$高度的杆子上，从一根杆子可以爬到和它方向相同的比它高不超过h 的杆子上，要求至少能到达一个高度在$n-h+1...n$的杆子，求有多少种方法。

21.2 题目解法

令$dp[i][a][b][c][d]$表示已经放了高度为$1...i$的杆子，a,b,c,d 分别表示四个方向最靠上的一根杆子离高度i 的距离。

注意到，a,b,c,d的顺序是没有影响的，而如果某一个方向离i 距离超过h，那么可以视作它离i距离为$h+1$，因为这个方向上始终无法达成要求了。那么我们的状态数量就可以减小到nC_{h+1}^4。而转移是$O(1)$的，这样就可以通过本题了。

21.3 时空复杂度

时间复杂度：$O(nC_h^4)$。空间复杂度：$O(C_h^4)$。

22 D8778 Codeforces 269D Maximum Waterfall

22.1 题目大意

有一张图，上面有N条水平线段，任意一条线段的高度在$(0,t)$范围内。在高度为0 和高度为t 的位置各有一条可以视为直线的线段。

两条线段i与j之间有流量定义为：$h_i > h_j, max(l_i,l_j) < min(r_i,r_j)$，且不存在线段$k$满足$(i,k)$ 与(k,j) 均满足上面两个条件。此时i与j 之间的流量大小为$min(r_i,r_j) - max(l_i,l_j)$。

求一条从最高线段到最低线段的路径，使得路径上每两条相邻的线段间的流量值的最小值尽量大。

$n \leqslant 10^5$

22.2 题目解法

显然在连完边之后只需要拓扑排序DP一遍即可得出答案。

将所有线段按高度排序，依次处理。我们维护一棵线段树，保存当前x的某一区间是哪一条线段。当我们处理到当前的线段时，我们首先对它的x 坐标内的每一段连续相同的线段，查询它们之间是否可以连边。之后我们将这条线段x坐标的区间覆盖。

查询的时候只需要查询它们的x坐标的交在当前线段树中是不是都是高度较低的那一条线段。

简单的分析可以得出，这样做的复杂度是$O(n \log n)$ 的。可以通过本题。

22.3 时空复杂度

时间复杂度：$O(n \log n)$。空间复杂度：$O(n \log n)$。

23 D8796 Codeforces 273E Dima and Game

23.1 题目大意

我们有n对数(l,r)。每次可以选择一对数，要求满足$r-l>2$，将其变为$(l,r-floor((r-l)/3))$或者$(l+floor((r-l)/3),l+2*floor((r-l)/3))$，其中floor表示向下取整。

现在要求你找出这样n对数，满足两人轮流操作，先手胜。且要求$1\leqslant l<r\leqslant p$。求有多少种方案。

23.2 题目解法

这是一个博弈类问题。我们考虑一对数(l,r)的SG函数值，容易发现，当$r-l$这个值固定时，所有数对的SG 函数值都相同。

那么我们可以得到SG函数递推式：$SG[i]=mex(SG[i/3],SG[i-i/3]$。初值$SG[1]=SG[2]=0$。

容易看出SG值只可能有$0,1,2$三种情况。

而对于一个$SG[i]$，事实上SG函数值为它的有$p-i$种情况。

如果我们能统计出SG函数值为$0,1,2$的分别有多少种情况，我们就容易使用DP或者直接计算得到答案了。

容易发现，我们的SG函数值总是一段连续的相同的值。那么我们可以把每一个SG函数值相等的段预(da) 处(ge)理(biao)出来，这样就可以统计了。

事实上，这样的段最多只有102个。

23.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(1)$。

24 D8741 Codeforces 274C The Last Hole!

24.1 题目大意

给一些点，以每个点为圆心有一个逐渐扩大的圆。求一个时刻，使得这个时刻之后，不再有任何的洞。洞的定义是一个封闭的没有被圆覆盖的图形。

24.2 题目解法

我们只需要考虑两类点：

1.每个锐角三角形的外心。

2.每个矩形的中心。

枚举每个锐角三角形和每个矩形并计算即可。但是要注意，当我们考虑某一个锐角三角形的外心时，它第一次被覆盖必须是被这个三角形的顶点覆盖，而不能被其它点覆盖。否则可能出现并没有洞的情况。矩形也是一样。

24.3 时空复杂度

时间复杂度：$O(n^4)$。空间复杂度：$O(n)$。

25 D8804 Codeforces 277D Google Code Jam

25.1 题目大意

有n个题目，每个题目有small和large 两部分。做small需要$timesmall$的时间，能得到$scoresmall$的分数。做large需要完成对应的small，需要$timelarge$ 的时间，能得到$scorelarge$ 的分数，但有$probfail$ 的概率无法得到分数。求在t 时间内最多期望得到多少分，并在期望得到最多分的前提下，让最后一次正确提交的时间期望尽量早。

25.2 题目解法

容易发现，对每个题目，我们有两种转移：只做small 和做large 与small。

对应方程为：$f[i] = f[i - timesmall] + scoresmall$

以及：$f[i] = f[i-timesmall-timelarge]+scoresmall+scorelarge*(1-probfail)$

然后考虑最后一次正确提交的期望时间，显然我们应该先做掉所有small，因为它们一定会正确。之后考虑所有large，容易发现，$\frac{probfail*timelarge}{1-probfail}$ 越大的题目应该放在越后面，可以使最后一次正确题交的时间尽量早。

那么将题目排序后，期望时间的转移为：

$g[i] = g[i - timesmall] + timesmall$

及$g[i] = i*(1 - probfail) + probfail*(g[i - timesmall - timelarge] + timesmall)$

25.3 时空复杂度

时间复杂度：$O(nt)$。空间复杂度：$O(t + n)$。

26 D8767 Codeforces 283E Cow Tennis Tournament

26.1 题目大意

有n头奶牛，每头奶牛有一个能力值。一开始任意两只奶牛的比赛会是能力值高的获胜。

之后有k次操作。每次会给定一个能力值区间$[l,r]$，对于这个区间中的任意两只奶牛，会将它们的比赛结果反过来。

求所有操作之后，三元组(p,q,r)满足p 胜q，q胜r，r胜p 的个数。

26.2 题目解法

直接计算不方便。我们考虑一个三元组(p,q,r)，如果它不满足题目的要求，那么一定有一个能够战胜其他两个。那么我们考虑计算这种情况，只需要计算出每头牛能够击败的数量d_i，答案就是$C_n^3-\sum C_{d_i}^2$。

那么我们只需要计算出d_i就可以了。考虑两头能力值接近的牛i和$i+1$，它们之间的差距只是那些影响到i而没有影响到$i+1$的操作，以及影响到$i+1$而没有影响到i的操作。因此，我们将所有奶牛排序后依次考虑，对每一只奶牛记录它对当前奶牛的胜负情况有没有被改变。每次对当前奶牛，我们首先考虑所有以它为左端点的区间，将区间内的元素取反。然后就可以统计它的d_i 了，此时的d_i是i左边的0的个数加上i 右边的1 的个数。接下来，我们再将所有以它为右端点的区间内元素取反，因为这样的区间不会对后面的d_i造成影响了。

将所有操作用线段树维护即可。

26.3 时空复杂度

时间复杂度：$O(n\log n)$。空间复杂度：$O(n)$。

27 D9173 Codeforces 285E Positions in Permutations

27.1 题目大意

给定n和k，求$1-n$的所有排列中，满足$|p[i]-i|=1$的i有k个的排列有多少个。

27.2 题目解法

设$dp[i][j][p][q]$表示：当前有j个这样的位置，上一个这样的位置在i，i位置的使用情况为$p(0/1)$，$i+1$位置的使用情况为$q(0/1)$，那么容易得到转移。

之后，我们将所有j相同的$dp[i][j][p][q]$加起来，不妨设为$f[j]$。由于还有$n-j$个元素没有摆放位置，那么$f[j]$应该乘以$(n-j)!$。根据容斥原理，答案为$\sum_{i=k}^{n}(-1)^{i-k}C_i^k f[i]$。

27.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(n^2)$。

28 D8783 Codeforces 293E Close Vertices

28.1 题目大意

给一棵树，边上有权。求点对(x,y)的数量，满足(x,y)之间的边数不超过L条且权值和不超过W。

28.2 题目解法

考虑使用点分治。

对于一棵树，我们希望统计其中经过根的满足要求的点对有多少个。我们不妨先不考虑在同一子树内部的情况。那么我们可以求出每个点到根的边数以及权值和，然后将它们按权值和排序。这时，对于一个点，我们就能够知道它与哪一些点的权值和不超过W。这些点必定是排序后的一个前缀。然后用一个树状数组维护边数即可。

之后，我们再把每一个子树内部用同样的方法减去不应计入答案的值。这样就能够得到对当前分治的子树的答案了。

28.3 时空复杂度

时间复杂度：$O(n\log^2 n)$。空间复杂度：$O(n)$。

29 D9157 Codeforces 294D Shaass and Painter Robot

29.1 题目大意

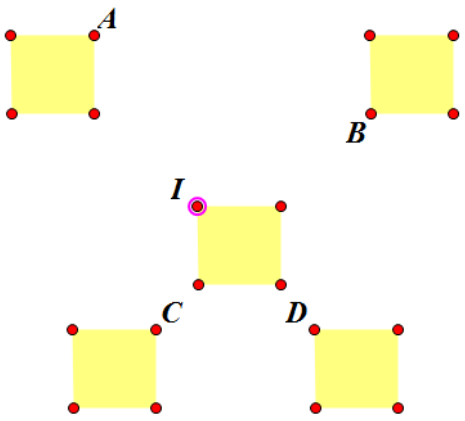
给定一个$n \times m$的白色网格，有一个机器人站在(x, y) 上。它会沿某个斜着的方向一直走，并把沿途的格子涂成黑色。当它碰到边界时，会类似于光反射来改变方向。直到整个网格变成黑白相间的时候才会停止。求多久以后会停止或者指出永远都不会停下来。

29.2 题目解法

注意到一个性质：如果我们能把整个网格染成黑白相间的，那么最后一个染的一定是边界上的。换句话说，如果边界已经变成黑白相间的了，那么整个网格都是黑白相间的。

有了这个性质，我们就可以直接暴力求解了。直接模拟机器人的行动即可。由于我们只关心边界上的格子，这些格子的数量是$O(n + m)$的。

而这个性质的证明是显然的：如果我们最后一个染的不是边界上的，考虑这个格子沿四个方向走到边界的地方：



考虑这四个格子：如果有某一个边界上的格子不是从I的方向过去的，那么它一定会染色到I。那么那一个边界上的格子就必须是边界上的格子中最后一个被染色的。也就是说，这四个格子中只能有一个这样的。那么剩下三个必须都是从I 的方向过去的，而这显然不成立。

因此，最后一个染色的一定是边界上的格子。

29.3 时空复杂度

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

30 D9174 Codeforces 295D Greg and Caves

30.1 题目大意

给一个$n \times m$的表格，要求选择一个行号区间$[L,R]$，使区间中每行有两个黑点，且存在一行$L \leqslant t \leqslant R$，满足对于任意$L \leqslant i \leqslant j \leqslant t$，$black\_left[i] \geqslant black\_left[j], black\_right[i] \leqslant black\_right[j]$，且对于任意$t \leqslant i \leqslant j \leqslant R$，$black\_left[i] \leqslant black\_left[j], black\_right[i] \geqslant black\_right[j]$。求有多少种方案。

30.2 题目解法

考虑枚举第t行，那么我们考虑求出：$dp[i][j]$表示以第i行长度为j的一段为底的图案有多少种方案。假设我们枚举了第i行，长度为j的一段作为第t行，那么我们的答案为：$dp[i][j] * (dp[n-i+1][j] - dp[n-i][j]) * (m-j+1)$。

而$dp[i][j] = 1 + \sum_k dp[i-1][k](j-k+1)$。

那么我们就可以计算答案了。

30.3 时空复杂度

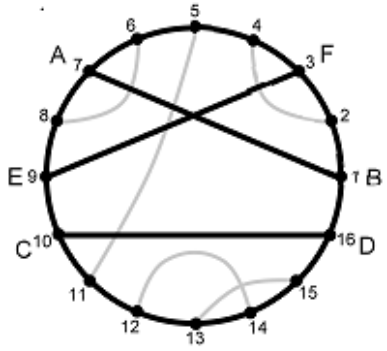
时间复杂度：$O(nm)$。空间复杂度：$O(nm)$。

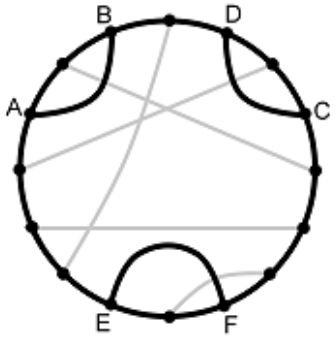
31 D9182 Codeforces 297E Mystic Carvings

31.1 题目大意

一个环上面有$2n$个点，我们把它们分为n对。要求选出3对点染为红色，满足任意一对红色点之间，最少经过的红色点数都相同（只能沿边缘）。求有多少种方案。

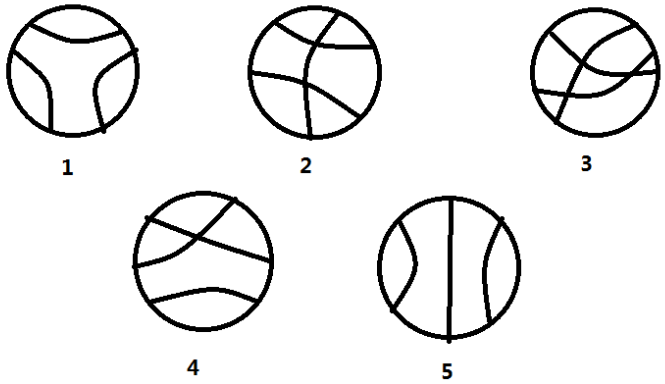
如下图，左图为不可行的方案，右图为可行的方案。





31.2 题目解法

任意三对点的情况实际上只有下面五种：



注意到情况1和情况3是我们需要计入答案的。但是它们不好计算。我们不妨考虑计算情况$2,4,5$的和，然后用c_n^3减去它们。

我们设c_i表示与第i条线段相交的线段个数，l_i表示第i条线段左侧有多少个点，r_i表示第i条线段右侧有多少个点。（l_i和r_i交换也没有关系）

那么情况5是比较好计算的。我们只需要考虑枚举中间一条线段，然后用它左边和它不相交的线段个数和右边与它不相交的线段个数乘起来就可以了。容易看出值为$\sum \frac{l_i-c_i}{2} \times \frac{r_i-c_i}{2}$。

情况2和情况4并不方便计算，但我们可以计算它们的和。对一条线段i，我们考虑一条与它相交的线段和一条与它不相交的线段。显然，如果这两条线段相交那么就是情况2，否则是情况4。但是，每一种情况被计算了两次。因此这部分的和为$\frac{\sum c_i(n-c_i-1)}{2}$。

那么我们就可以计算出答案了。最后，求c_i的过程可以使用树状数组维护来得出。

31.3 时空复杂度

时间复杂度：$O(n \log n)$。空间复杂度：$O(n)$。

32 D8810 Codeforces 301C Yaroslav and Algorithm

32.1 题目大意

给定n个数，我们要求给出一个程序，它以每个数字的字符串作为输入，能够将每个数+1。程序要求如下：

程序由一系列指令组成，每个指令形如$s[i] >> w[i]$或$s[i] <> w[i]$。其中$s[i]$和$w[i]$为长度不超过7 的字符串，且其中只含有数字和?。

当我们执行一个程序时，找到第一个i，满足$s[i]$ 是输入字符串的子串，将字符串中第一个$s[i]$替换为$w[i]$。如果这个命令中间是>>，那么继续此步。否则程序结束。

32.2 题目解法

我们可以这样构造：

考虑将?看作一个指针。一开始我们令这一个指针指向数字的开始，然后依次将它移动到最后。

接下来我们将?变为??这样一个新的指针。然后我们将?? 开始向前移动，如果当前?? 前的数字不为9，那么就把它+1 并结束程序。如果数字为9，那么将它变成0 并将指针前移。如果??前没有数字了，就给出一个1并结束程序。

使用以上算法就可以得到一个对任意数字都可行的程序。事实上本题的输入是没有用的……

32.3 时空复杂度

时间复杂度：$O(1)$。空间复杂度：$O(1)$。

33 D8739 Codeforces 301E Yaroslav and Arrangements

33.1 题目大意

我们定义一个长为r的序列为良好的，当：$|a_i - a_{i+1}| = 1, |a_r - a_1| = 1$ 且$a_1 = \min_{i=1}^{r} a_i$。

我们定义一个长为r的序列为优秀的，当：$a_i \leqslant a_{i+1}, 1 \leqslant r \leqslant n, 1 \leqslant a_i \leqslant m$，且重排整个序列可以得到至少一个至多$k$个不同的良好序列。

给定n, m, k，求有多少个不同的优秀的序列。

33.2 题目解法

考虑一个良好序列，其中最大的数为x。我们考虑放一些$x+1$ 进去。我们放的方法很简单：每次将一个x 变为$x\ x+1\ x$。如果我们原来有h 个x，那么我们现在就有了$h+1$个x。因此，我们如果放了t 个$x+1$进去，我们就能得到$\frac{(h+t-1)!}{(h-1)!}$种良好序列。但是，由于这t 个$x+1$ 没有区别，还要除去一个排列数$t!$。因此共有C_{h+t-1}^{h-1} 种不同的良好序列。

那么我们就容易得出dp方程了。设$dp[i][j][p][t]$表示：当前序列中有i 个不同的数，最大的数有j 个，总共有p个数，有t 种重排得到的不同良好序列。

考虑转移。我们枚举加入几个比当前最大的数大1的数进入到序列。假设加入x 个，我们贡献的就是$dp[i+1][x][p+2\times x][t\times C_{j+x-1}^{j-1}]$。

最后，由于我们可以自由改变最小值，而最大值只要不超过m 即可。因此当序列中有i个不同的数时，对答案的贡献是$m-i+1$。

复杂度理论为$O(n^5)$。常数非常小。

33.3 时空复杂度

时间复杂度：$O(n^5)$。空间复杂度：$O(n^3)$。

34 D8808 Codeforces 303D Rotatable Number

34.1 题目大意

定义“可旋转数”为一个b进制的n位数（允许前导0）,且满足它的1到n倍是它本身不断将最后一个数字放到最前面得到的数中的一个。

给定n, x，求最大的b满足$1 < b < x$，且在b进制下存在一个n 位可旋转数。

34.2 题目解法

这种数的名字叫做“Cyclic number”。有关它的信息见维基百科：https://en.wikipedia.org/wiki/Cyclic_number

从维基百科中，我们知道了一个b进制的n位可旋转数，一定满足$n+1$是一个质数，且这个数可以表示成$\frac{b^n-1}{n+1}$。

但是并不是每个形如$\frac{b^n-1}{n+1}$ 的数都是一个可旋转数。它有可能是一个可旋转数重复若干次。因此，我们判断一下这个数本身是否有循环节即可。

34.3 时空复杂度

时间复杂度：$O(\sqrt{n}\log n)$。空间复杂度：$O(\sqrt{n})$。

35 D9186 Codeforces 303E Random Ranking

35.1 题目大意

有n个人，每个人的权值是$[l_i, r_i]$ 中的一个随机实数，对每个人，求他的权值排名第j 小的概率。

35.2 题目解法

首先，我们将所有输入的权值放在一根数轴上，那么上面最多只有$2n$段区间。

我们考虑计算出一个rem数组，$rem[i][j][k]$ 表示：对于第i 个人，他左边的所有区间中有j个人，还有k 个人和他处在同一区间内的概率。

由于所有处在同一区间内部的人的排名是完全随机的，那么$rem[i][j][k]$对$ans[i][j+q](q \leqslant k)$ 的贡献为$\frac{rem[i][j][k]}{k+1}$。

接下来我们考虑如何求出rem数组。我们不妨枚举一段区间$[left, right]$，取出所有包含这一段区间的人，不妨设有cnt 个，每个人落在这一段区间的概率为pm_i，落在左边的区间的概率为pl_i，落在右边的区间的概率为pr_i。

那么有：$pl_i = \frac{left-l_i}{r_i-l_i}, pm_i = \frac{right-left}{r_i-l_i}, pr_i = \frac{r_i-right}{r_i-l_i}$

再设一定落在左边区间的人有out个。

考虑枚举这cnt个人中的某一个，不妨设为x。我们希望计算出，当x落在这一段中时，他左边的段有j 个人，他这一段还有k个人的概率。

那么我们用一个$dp[num][j][k]$表示当前考虑了num 个人，有j 个人在这一段左边，有k 个人在这一段中的概率。

那么对一个与x不同的人y，有$dp[num + 1][j + 1][k] += dp[num][j][k] * pl_y, dp[num+1][j][k+1] += dp[num][j][k] * pm_y, dp[num+1][j][k] += dp[num][j][k] * pr_y$。

那么我们就可以得出，这一个区间对$rem[x][j + out][k]$的贡献为$dp[cnt - 1][j][k] * pm_x$。

直接按以上做法做就可以通过本题了，复杂度为$O(n^5)$。

实际上我们可以做得更好。考虑两个不同的x，不妨设为x_1 与x_2。更新他们的dp数组时，只有一个人不一样。也就是说，我们会用x_1来更新x_2的dp，反之亦然，而其他人是完全一样的。那么，我们考虑使用分治：

当我们分治$[l,r]$区间时，我们记录下当前的dp数组。然后，我们将$[l,mid]$的人更新入dp数组，然后分治$[mid+1,r]$。接着，我们将dp数组还原，将$[mid+1,r]$的人更新进去，然后分治$[l,mid]$。

这样就可以在$O(n^4 \log n)$的时间内解决本题。

35.3 时空复杂度

时间复杂度：$O(n^4 \log n)$。空间复杂度：$O(n^3)$。

36 D9214 Codeforces 305D Olya and Graph

36.1 题目大意

有一个有向图，保证每条边(u,v)满足$u<v$。现在要求添加一些边，使得满足：

1.从点i出发，可以到达点$i+1,i+2,...,n$。

2.任意从u到v的有向边满足不等式：$u<v$。

3.两点之间最多有一条边。

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

5.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。

求有多少种方案。

36.2 题目解法

题目要求的实际上是这样一个图：

每个点u连出的边，要么连到$u+1$，要么连到$u+k+1$。且对于两个点u,v，满足$u+k+1\leqslant v$，不能同时连到$u+k+1$ 和$v+k+1$。

那么做法就非常显然了。我们首先处理输入的所有边，如果有不满足条件的就直接输出0.之后，我们枚举最左边一个连到$u+k+1$ 的点，判断是否有一条输入的边与它不满足条件，之后统计答案即可。

如果实现的足够精细，可以避免中间的枚举操作，从而做到$O(\log n)$的复杂度。

36.3 时空复杂度

时间复杂度：$O(n\log n)$。空间复杂度：$O(n)$。

37 D8802 Codeforces 305E Playing with String

37.1 题目大意

给定一个字符串，两人轮流操作，每次选择一个位置，要求它的左右两个字符一样，把整个字符串从这里分成两段，中间那一个舍去。不能操作者负。

求谁胜，如果先手必胜求第一步应该选哪一个。有多解选最靠前的一个。

37.2 题目解法

我们令$sg[n]$表示长度为n的字符串，且每一个位置都可以切开的SG值。（注意，原题中不能切边界上的，而这里我们允许）

那么$sg[n]$可以简单的求出。

而最后的答案事实上就是：取每一段连续的可以切的位置的SG 值，然后求异或和。这是因为我们从某一个位置切开时，事实上不会影响到其他段。

而第二问只需要枚举第一次切的位置，然后求新局面的SG值即可。

37.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(n)$。

38 D8784 Codeforces 306C White, Black and White Again

38.1 题目大意

有n个格子，w个白球，b个黑球。要求将n 个格子中前一段放白球，之后一段放黑球，最后一段放白球。所有球都必须放进去，不能有空格子。同一个格子内球的顺序也需要考虑。求一共有多少放球的方法。

38.2 题目解法

枚举有x个格子放黑球。那么答案乘上$n-x-1$，表示放黑球的段的方法数。

之后我们对黑球和白球都需要考虑这样一个问题：

将a个球放进b个格子中，不能有空格子，格子内考虑球的顺序，求方案数。

对于这个问题，我们考虑a个球的每一个排列，之后按排列顺序一个一个放进去。那么我们容易得到答案为：$a! * f(a,b)$，其中$f(a,b) = f(a-1,b) + f(a-1,b-1)$，$f(0,0) = 1$。

那么我们就解决了本题。

38.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(n^2)$。

39 D9235 Codeforces 306D Polygon

39.1 题目大意

求构造一个n边形，使每个角度数都一样，每条边长度都不一样。

39.2 题目解法

利用精度的一道题。我们让每条边的基础长度为一个比较大的数（比如100），然后每条边比它前一条边长很少的一点点（比如0.002），那么我们的角度变化就会在要求的精度范围内。

39.3 时空复杂度

时间复杂度：$O(n)$。空间复杂度：$O(1)$。

40 D9201 Codeforces 309B Context Advertising

40.1 题目大意

有n个单词，我们要从中选出连续的一段，满足可以把它们写在不超过r行中，每行不超过c个字母。同一行中相邻的单词间要用一个空格隔开。要求选出的单词数尽量多。

40.2 题目解法

我们考虑以每一个地方为结束，这一行最开头的单词是哪一个。注意到这个是单调递增的，那么我们可以在线性的时间内得出。

然后，不妨设以i为结束的一行最开头的是$f[i]$，那么我们从$f[i]-1$向i连一条边，就能够得到一个树形的结构（有可能有多棵树）。之后只要对每个点求它的r级祖先即可。这个可以直接用dfs解决。

40.3 时空复杂度

时间复杂度：$O(len)$。空间复杂度：$O(len)$。

41 D8769 Codeforces 309D Tennis Rackets

41.1 题目大意

一个等边三角形，每边上作$n+1$等分，共有n个等分点。每条边上靠近角落的m个点不能用。

现在要求在每条边上的可用点中各找出一个，使其构成钝角三角形。求有多少种方法。

41.2 题目解法

枚举钝角顶点，此时当一个顶点在其边上移动时，另一个顶点的可取范围也会随之移动。

我们可以使用余弦定理算出可取的范围，这样就可以统计答案了。

然后卡好常数……

41.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(1)$。

42 D9225 Codeforces 309E Sheep

42.1 题目大意

给定n个区间，求它们的一个排列，使得一对有交的区间在排列中的距离的最大值尽量小。

42.2 题目解法

考虑二分答案。

接下来我们就需要考虑一个判定性问题：是否存在一个排列，满足任意一对有交的区间在排列中的距离都不超过d。

我们不妨依次考虑每一个位置放哪一个区间。我们对每一个还没有放进排列的区间保存一个值，表示它当前最靠后能够放在哪里，如果再靠后就会与某个和它有交的区间距离大于d。不妨设这个值是$last_i$。假设当前考虑的位置为p，那么我们考虑最小的一个j，满足$last_i \leqslant j$的i恰好有$j-p$个。显然，在$p+1$到j的这些位置上只能放$last_i \leqslant j$的这些区间，否则一定无法满足条件。

那么，我们下一个位置应该放哪一个区间呢？事实上，我们应该放右端点尽量小的区间。因为如果放右端点最小的区间仍然无解的话，考虑放另一个右端点大于它的区间，显然$last$值会改变的区间会变多（或者至少不会变少）。因此，我们每次贪心的放右端点尽量靠左的区间就可以了。

最后，如果某个时候存在j使$last_i \leqslant j$的i多于了$j-p$个，那么显然就无解了。

42.3 时空复杂度

时间复杂度：$O(n^2 \log n)$。空间复杂度：$O(n^2)$。

43 D9200 Codeforces 311C Fetch the Treasure

43.1 题目大意

有n间宝藏密室，每间的位置为a_i，其中财宝的价值为c_i。我们一开始只有技能k。我们可能会增加一个技能x，或者减少第x间宝藏密室的价值y元，或者询问能到达的价值最大的宝藏密室并删除它。

一个宝藏密室能到达，当且仅当它的位置可以表示成$1+\sum v_i x_i$，其中x_i表示我拥有的技能，v_i为任意非负整数。

增加技能的操作不超过20次。$k \leqslant 10^4$。

43.2 题目解法

对于一个位置a，如果它是k的倍数+1，那么就一定能到达。我们不妨设lim_i表示第一个能到达的$\bmod k = i$的位置。

那么每一次增加技能的时候，我们暴力重算lim_i即可。

对于一间宝藏密室，如果$a_i \geqslant lim_{a_i \bmod k}$，则它是可到达的。

用一个堆维护所有可到达的宝藏密室即可。

43.3 时空复杂度

时间复杂度：$O((n+m)\log n + 20^2 k \log k)$。空间复杂度：$O(n+k)$。

44 D9205 Codeforces 311E Biologist

44.1 题目大意

有n条狗，每条狗有一个初始性别，改变它的性别需要v_i元钱。现在有m个要求，要求某k_i条狗都是某一性别，如果满足能够获得w_i 元钱。这些要求中有一些是朋友提出的，如果朋友的要求没有满足需要倒付g元钱。求最大盈利（有可能为负）。

44.2 题目解法

考虑网络流：

对每条狗建一个点，如果它是雌性那么从它向汇连边权为v_i 的边，否则从源向它连边权为v_i 的边。

对每个要求建一个点。如果它要求为雌性，则从源向它连边权为w_i的边（如果是朋友再加g）。否则从它向汇连边权为w_i 的边（如果是朋友再加g）。然后对每条它要求的狗，如果要求是雌性从它向要求的狗连边权为无穷的边，否则从要求的狗向它连边权为无穷的边。

最后将所有w_i加起来，减掉最小割就是答案。

44.3 时空复杂度

时间复杂度：$O(maxflow(n, n+km))$。空间复杂度：$O(n+km)$。

注：$maxflow(n,m)$表示n个点m条边的网络流复杂度。

45 D8805 Codeforces 314E Sereja and Squares

45.1 题目大意

给一个长为n的序列，保证n为偶数。现在把n个位置两两配对，把每一对用一个字母（除了“X”）表示，靠前的一个为小写，靠后的一个为大写。

并且要求满足一个条件：任意两对不能相交。也就是不能出现类似abAB这样的序列。

现在去掉了部分小写字母和全部的大写字母，求有多少种补全的方法。

45.2 题目解法

我们把题目的序列看成括号序列，事实上就是有25种不同的括号。那么我们用一个DP：

$dp[i][j]$表示到序列第i项，左括号还剩j个的方法数量。

那么转移显然：$dp[i][j] = dp[i-1][j-1] \times 25 + dp[i-1][j+1]$

对于已知是左括号的位置，$dp[i][j] = dp[i-1][j-1]$

对这个DP，我们卡一卡常数就可以在4 秒内过掉……

45.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(n)$。

46 D9231 Codeforces 316D PE lesson

46.1 题目大意

有n个人，每个人可以进行一次或两次交换。每个人一开始有一个和他的编号相同的数，我们可以进行若干次交换，但是每个人的交换次数不得超过他的限制。求可以交换出多少种不同的序列。

46.2 题目解法

设有a个人是1，b个人是2，那么答案为$\frac{(a+b)!I(a)}{a!}$，其中$I(a) = I(a-1) + (a-1)I(a-2), I(1) = I(0) = 1$。

证明如下：

考虑某一个置换，我们把它们拆成对换。那么就容易发现每个置换里至多有两个1。所以我们就是要求每个循环中不超过两个1的排列的个数。

设$f_a(x)$是方案数关于a的指数型母函数。

$$f_a(x) = f_{a-1}(x)\frac{1}{1-x} + f_{a-2}(x)(a-1)\frac{1}{(1-x)^2}$$

化简一下就是：

$$f_a(x)(1-x)^a = f_{a-1}(x)(1-x)^{a-1} + f_{a-2}(x)(1-x)^{a-2}(a-1)$$

注意到这个可以变成：$I(a) = I(a-1) + (m-1)I(a-2)$。又$f_0(x) = \frac{1}{1-x}$，所以$I(0) = 1$。

因此就有答案是$I(a)C_{a+b}^{b}b!$。

46.3 时空复杂度

时间复杂度：$O(n)$。空间复杂度：$O(1)$。

47 D9180 Codeforces 316E Summer Homework

47.1 题目大意

给一个序列，要求支持：

1.将某个位置上的数修改为某值。

2.给一个区间$[l,r]$，求$\sum_{i=0}^{r-l} f_i a_{i+l}$，其中$f_i$表示第$i$个fibonacci数。

3.将一个区间的所有数加上某值。

47.2 题目解法

用一棵线段树来维护整个序列。

对于线段树的一个节点，如果它维护的是$[l,r]$这一段区间，我们维护一个长度为2的向量$(\sum_{i=0}^{r-l} f_i a_{i+l}, \sum_{i=0}^{r-l} f_{i+1} a_{i+l})$。

那么，我们就容易使用矩阵乘法来合并两个子节点的信息得到当前节点的信息了。

对于区间加的操作，我们维护一个标记即可。注意当我们打上一个区间加的标记的时候，这个区间的值增加的是fib序列的一个前缀和的若干倍。这个使用$\sum_{i=1}^{n} f_i = f_{n+2} - 1$即可。

47.3 时空复杂度

时间复杂度：$O(n + m \log n)$。空间复杂度：$O(n)$。

48 D8759 Codeforces 316G Good Substrings

48.1 题目大意

给一个字符串s，还有n个限制条件，每个条件给出一个字符串p，以及两个参数l, r。问s 有多少个子串满足：对每个限制条件，在p 中的出现次数在$[l, r]$中。

48.2 题目解法

对于s的每个位置，计两个值lm, rm，分别表示在满足所有限制条件的前提下，以这一位置为结束，最远可以向前延伸多远，最近必须向前延伸多远。

对于一个限制条件，我们用它来更新所有的lm, rm。这是一个后缀自动机的经典问题，只需要对串p构建后缀自动机，并将s 在后缀自动机上运行一遍即可。

之后，我们还需要去掉重复的子串。这一步同样可以使用后缀自动机来计算。只需要对后缀自动机中的每个节点，记录两个值lx, rx，分别表示这一个节点代表的串中，长度在$[lx, rx]$ 范围内的需要被计入答案。之后将每个点的$(rx - lx + 1) \times right$ 计入答案即可。

48.3 时空复杂度

时间复杂度：$O(nL)$。空间复杂度：$O(Ls)$。

（L表示字符串长度，s表示字符集大小）

49 D8806 Codeforces 317E Balance

49.1 题目大意

有n个容器，每个容器容量为v升。每个容器有初始水量a_i 和目标水量b_i。有e条管道将它们相连。

要求找到一种方法使每个容器都变为目标水量，且输水步数不超过$2 \times n^2$。

49.2 题目解法

我们每次找到一对点x, y，其中x的水量多于目标水量，y 的水量少于目标水量。我们尝试将x的水输向y，且不改变其他容器的水量。不妨设x 和y中变成目标水量所应该输出/输入的水量中较小的一个为d。显然我们应该从x输出d 升水到y。

任意找一条x到y的路径。不妨设这条路径上的点为q_i，其中$q_1 = x$，$q_L = y$。

设$f(q_i)$表示从q_i这个点输出d升水到q_L的过程，设$pour(q_i, q_{i+1}, d)$ 表示从q_i直接输出d升水到q_{i+1}。

考虑$f(q_i)$。分情况讨论：

1.如果$a_{q_{i+1}} + d \leqslant v$。这种情况下：$f(q_i) = pour(q_i, q_{i+1}, d) + f(q_{i+1})$。

2.如果$d \leqslant a_{q_{i+1}}$。这种情况下：$f(q_i) = f(q_{i+1}) + pour(q_i, q_{i+1}, d)$。

3.如果以上两种情况都不满足，我们就需要两次操作：设$t = v - a_{q_{i+1}}$，$f(q_i) = pour(q_i, q_{i+1}, t) + f(q_{i+1}) + pour(q_i, q_{i+1}, d - t)$。

重复这一操作，直到找不到这样的点对x, y即可。显然，每次我们一定会将x, y 中的一个变为目标水量。那么一共只会找n 个点对。每次最多进行$2n$次操作，那么操作数不超过$2 \times n^2$。

49.3 时空复杂度

时间复杂度：$O(n^3 + ne)$。空间复杂度：$O(n^2)$。

50 D8787 Codeforces 321D Ciel and Flipboard

50.1 题目大意

给定一个$n \times n$的矩阵，令$x = \frac{n+1}{2}$。每次可以将一个$x \times x$的子矩阵乘以-1，并可以执行任意多次。求矩阵的最大和。

50.2 题目解法

设$c[i][j]$表示位置(i, j)最终是否被乘以了-1。事实上，我们最后只需要满足：

$$c[i][j] \ xor \ c[i][x] \ xor \ c[i][j+x] \ = \ 0$$

且

$$c[i][j] \ xor \ c[x][j] \ xor \ c[i+x][j] \ = \ 0$$

必要性是显然的。每次取子矩阵时，以上每式必定包含0个或2个元素。因此取异或一定为0。

考虑充分性。对于一个给定的c矩阵，我们每次依次选择最小的i, j且$c[i][j] = 1$，以其为左上角取矩形。容易发现最后的矩形一定全为0。充分性得证。

那么我们只需要枚举$c[1][x]...c[x][x]$，之后就可以算出$c[x+1][x]...c[n][x]$。然后对$1...x-1$的每一列就可以单独贪心了。

50.3 时空复杂度

时间复杂度：$O(2^x x^2)$。空间复杂度：$O(n^2)$。

51 D8800 Codeforces 323B Tournament-graph

51.1 题目大意

给定一张N个点的无向完全图，要求将每条边定向，使得以任意两个节点为起点和终点的最短路长度不超过2。

51.2 题目解法

考虑构造：

如果N为奇数，我们可以这样构造：每个点向自己后面$n/2$个点连边。显然的，这样就可以保证每个点只需要经过两条边就能到达任何一个点。

如果N为偶数，那么先将前n-1个点按以上方法构造，然后将前n-1 个点中奇数编号的向n号点连边，偶数编号的从n号点连边。这样对$6 \leqslant n$的所有偶数n，容易证明任意两点之间最短路最多为2。

最后，当且仅当n为4时无解。

51.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(n^2)$。

52 D8763 Codeforces 323C Two permutations

52.1 题目大意

给定两个$1-n$的排列，有m个询问，每次询问有多少个数在第一个排列中位于$[l_1, r_1]$中，在第二个排列中位于$[l_2, r_2]$中。

强制在线。

52.2 题目解法

计$v[i]$表示第一个排列中的第i个数在第二个排列中的位置。

那么我们就是要求l_1到r_1间有多少个元素的v值在$[l_2, r_2]$中。

这个可以用主席树简单的维护。

52.3 时空复杂度

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

53 D8740 Codeforces 325D Reclamation

53.1 题目大意

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

53.2 题目解法

我们把地图复制一遍放在右边。

注意到从上到下四联通等价于被删掉的格子从左到右八连通。那么我们就用并查集判断一下从左到右是否已经八连通了就可以了。

53.3 时空复杂度

时间复杂度：$O(n\alpha(n) + r \times c)$。空间复杂度：$O(r \times c)$。

54 D9224 Codeforces 325E The Red Button

54.1 题目大意

给定一个n个点的有向图，标号为0到$n-1$。点i有两条出边，分别指向$2i \bmod n$和$2i+1 \bmod n$。求一条从0 开始，回到0的路径，满足除了0以外的点均只经过一次。

54.2 题目解法

本题算法很简单：如果n是奇数则无解，如果n 是偶数，那么我们倒过来考虑，到点i的前一个点必然是$\lfloor \frac{i}{2} \rfloor$或$\lfloor \frac{i+n}{2} \rfloor$。我们贪心的选择后者，如果后者已经到过则选择前者，直到回到点1为止。这样一定能构造出一个合法解。

接下来是证明：

n是奇数时，为了最后回到0，很显然我们需要到达点$\frac{n-1}{2}$。但是为了到达点$n-1$，我们同样必须到达点$\frac{n-1}{2}$。而这个点只能经过一次，因此显然无解。

对于n是偶数的情况，我们考虑：假设先考虑$\lfloor \frac{i+n}{2} \rfloor$是不可行的，那么总是下两种情况之一：

到某一个点i时，发现$\lfloor \frac{i}{2} \rfloor$和$\lfloor \frac{i+n}{2} \rfloor$ 均已经到过。但这显然不可能，因为$\lfloor \frac{i}{2} \rfloor$和$\lfloor \frac{i+n}{2} \rfloor$到的点都是$i$ 和i *xor* 1，如果这两个点都到过那么说明i和i *xor* 1 均已经到过，而这与假设矛盾。

有某一个点i没有到过。这也不可能，因为如果没有到过点i，那么说明我们一定没有到过点$\lfloor \frac{i}{2} \rfloor$，那么说明我们也没有到过点$\lfloor \frac{i}{4} \rfloor$，……点1，而这与我们的算法结束了矛盾。

综上，我们的算法一定能构造出一个合法解。

54.3 时空复杂度

时间复杂度：$O(n)$。空间复杂度：$O(n)$。

55 D8775 Codeforces 329D The Evil Temple and the Moving Rocks

55.1 题目大意

给定一个$n \times n$的空地，我们可以在其中放入一些石头，每块石头有一个朝向。一开始我们激活一块石头，它会沿它的朝向方向移动。如果它撞到了另一块石头或者墙壁，那么它会停下来，撞到的那一块石头会开始移动。如果在撞击前至少移动了一个格子的距离，那么就会发出一声响声。

我们要求至少发出x声响声。

55.2 题目解法

构造出类似这样的东西：

>>>> . > . > .v

^ ^. < . < . <<<

^>>> . > . > .v

^ ^. < . < . <<<

就可以了。

55.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(1)$。

56 D9212 Codeforces 332D Theft of Blueprints

56.1 题目大意

给出一个n个点的带权无向图，满足对于任意一个大小为k的顶点集合S，恰好有一个点与S 每一个点都有边。令这个点为$v(S)$，并且对S 进行操作的代价是S中每个点与$v(S)$的边权之和。现在求对于一个大小为k 的子集操作代价的期望。

56.2 题目解法

由于题目保证对任意大小为k 的集合都满足条件，那么我们就只需要求出所有大小为k的集合的权值和，最后除以C_n^k即可。

考虑枚举点v，那么点v的贡献为$\sum_i w[v][i] \times C_{p[v]-1}^{k-1}$，其中$w$ 表示边权，p 表示度数。

那么我们就可以根据这个暴力做了。可以用long double 实现，要注意精度问题。

56.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(n)$。

57 D9161 Codeforces 332E Binary Key

57.1 题目大意

给定两个串p，s，要求求出一个字典序最小的长度为k的只包含'0' 和'1'的串q，使得：我们将p 每k个分成一段，取出q对应位置上为'1'的字符，连接起来得到的是串s。

57.2 题目解法

我们枚举q中有几个字符'1'。不妨设当前有x 个'1'。

那么，如果我们将p中模k相同的位置一同考虑，s中模x 相同的位置一同考虑，那么我们每次对于一个模k的值j，考虑p 中模k 为j的位置的字符连接起来是否与s 中模x 为y 的位置的字符连接起来相同？如果相同，那么j这个位置上就应该放上一个'1'，否则就只能放上一个'0'。

为了满足字典序最小，我们从后往前枚举j，一开始令y为$x-1$，如果放了一个'1'，就将y左移一位。如果y达到了负数，那么说明我们找到了字典序最小的有x 个'1'的所求串。

最后，对于所有x求出的串，取字典序最小的一个即可。

57.3 时空复杂度

时间复杂度：$O(k^2 + ks\log k + p)$。空间复杂度：$O(p + s)$。

58 D9199 Codeforces 333C Lucky Tickets

58.1 题目大意

给定一个k，要求找m个八位数（允许前导零），使这些数每一个都存在一种在中间加入$'+','-','\times'$的方法，使得其运算结果为k。

58.2 题目解法

我们不妨求出：使用每一个四位数可以运算出哪些数。经实验，一个四位数平均可以运算出60 个数左右。

之后，我们枚举一个四位数，考虑它能求出的数：比如一个四位数x，能够运算出y，那么我们在最后的八位数的前四位放上$k-y$，在后四位放上x，就可以运算出k了。

这样做可以得到将近6×10^5个解。

58.3 时空复杂度

时间复杂度：$O(\sqrt{10^8})$。空间复杂度：$O(\sqrt{10^8})$。

59 D9198 Codeforces 335D Rectangle And Square

59.1 题目大意

一个平面上有n个矩形，矩形之间可能接触但不会相互覆盖。现在问是否存在一个矩形的子集构成一个正方形。

所有坐标都是正整数，且不超过3000。

59.2 题目解法

枚举正方形的左上角和左下角。注意到坐标不超过3000，因此这一步枚举的复杂度是nL的。

然后考虑怎么$O(1)$判断是否可行。由于矩形之间不会相互覆盖，我们只需要判断正方形的内部是否全部被覆盖，以及正方形的边界是否全部是矩形的边界就可以了。这个可以用前缀和预处理做到每次询问$O(1)$。

59.3 时空复杂度

时间复杂度：$O(L^2 + nL)$。空间复杂度：$O(L^2 + n)$。

（L为地图边长）

60 D8797 Codeforces 335E Counting Skyscrapers

60.1 题目大意

一条直线上有一些楼房（数量是2到314!中的一个随机数），每栋楼房有一个高度，每栋楼房高度为i的概率是2^{-i}。楼层编号为0到$i-1$。

有一些楼房间有滑索。楼房i和楼房j之间有一条高度为h的滑索，当且仅当两栋楼房的高度都不小于h，且两栋楼房之间的所有楼房高度都小于h。

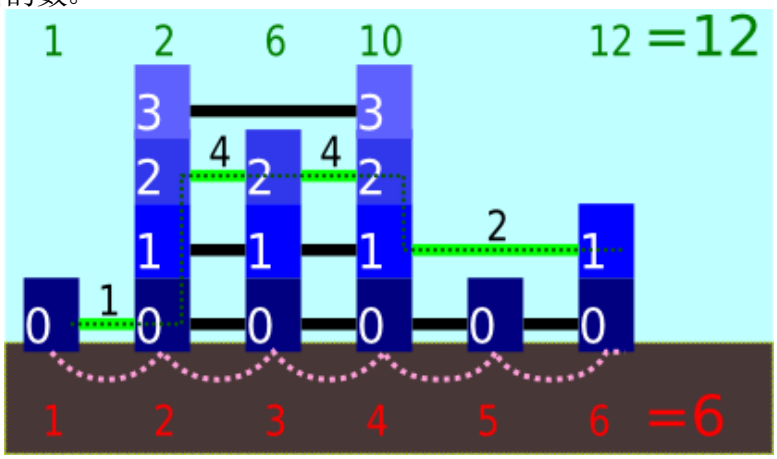
现在Alice和Bob想数一数有多少楼房。

Alice非常严谨，她会沿着地面一步一步走过去，数出有多少楼房。

Bob比较没耐心，他从最左边的楼房开始，每次选择一条高度不超过H且尽可能高的滑索，假设高度是t，他会沿着滑索移动并计入2^t个楼房。

现在可能会给你Alice或者Bob数出的楼房数量为n，求另一人数出的数量的期望。

下面的图表达了某一种情况。红色的代表Alice数出的数。绿色的代表Bob数出的数。



60.2 题目解法

先考虑已知Bob求Alice的情况。

由于楼房个数上限是314!，这个数非常大以至于我们可以把它视作无穷大。那么对*Bob*来说，他的每根滑索经过的楼房数量是互相独立的。

不妨设一条滑索经过的楼房个数是L，高度是h。中间$L-1$个楼的高度必须小于h。我们可以算出概率是$\frac{(1-2^{-h})^{L-1}}{\sum_{k\geqslant 1}(1-2^{-h})^{k-1}}$。这个化简以后就是$2^{-h}(1-2^{-h})^{L-1}$。

然后期望就是$\sum_{L>0} 2^{-h}(1-2^{-h})^{L-1}$。这个可以算出就是$2^h$。也就是说，Alice数出的数期望意义上和Bob 的是一样的。因此Alice 的期望答案就是n。

然后考虑已知Alice求Bob的情况。

考虑增量法。一开始我们不妨假设H是0。此时Bob的答案显然是n。

然后考虑从$H-1$变成了H。我们只需要考虑增量。

此时，我们会有一些高度是$H-1$的滑索高度变为H。假设这条绳索经过L个楼房，那么概率是$2^{-2h}(1-2^{-H})^{L-1}$。而每条这样的绳索的增量是2^H。

然后考虑损失。容易得出是$2^{H-1}(\frac{L-1}{2^H-1}+1)$。

把上面的式子加起来得到增量总共是：

$$\sum_{L=1}^{n-1}(n-L)2^{-H}(1-2^{-H})^{L-1}(\frac{1}{2}-\frac{1}{2}\times\frac{L-1}{2^H-1})$$

60.3 时空复杂度

时间复杂度：$O(nh)$。空间复杂度：$O(1)$。

61 D9166 Codeforces 338E Optimize!

61.1 题目大意

给一个长为n的串a以及一个长为len 的串b，求a的所有长为len 的子串中，有多少个子串能够使其中的元素与b 中的两两配对，且每一对的和均不小于h。

61.2 题目解法

考虑两个长为len的串，它们在什么情况下能够满足要求。

显然，我们把它们一个从小到大排序，另一个从大到小排序，然后一一配对即可。

但我们还有另一个方法：对a中的所有元素，我们找到b中与它的和大于等于h 的最小的一个元素与它配对。

考虑将a分为$\sqrt{n}$段。每一段中，一共有$len - \sqrt{n}$ 个元素没有变化，那么我们先将它们按照上面的第二个方法找到对应元素并删去。此时，我们只剩下$\sqrt{n}$个元素，那么直接暴力即可。

而如果len小于$\sqrt{n}$,就可以直接暴力了。

61.3 时空复杂度

时间复杂度：$O(n\sqrt{n})$。空间复杂度：$O(n)$。

62 D8768 Codeforces 339E Three Swaps

62.1 题目大意

给定一个$1-n$的排列，我们至多经过三次区间翻转操作将一个$1-n$的递增序列修改为它。求依次进行的区间翻转操作。

62.2 题目解法

我们不妨考虑如何使用至多三次区间翻转操作将给定排列变为有序。最后只要逆序输出即可。

如果只用两次操作，那么我们去掉前后已经匹配的部分后，容易证明：第一次翻转操作要么触及左端点，要么触及右端点。并把所触及端点还原。

但是，如果需要用三次翻转，那么这个结论是不正确的。

我们设：$b[a[i]] = i$。那么当我们让排列变为有序时，实际上会让一个$1-n$的有序序列变为b。于是我们也可以考虑如何让b变得有序。

我们可以得到另一个结论：

对于第一次翻转，必定触及左右端点中的一个，要么将其还原，要么将b中的对应端点还原。

我们可以这样来证明：如果两个条件均没有满足，那么也就是说我们的第一次翻转和最后一次翻转都不能还原左右端点。于是我们的第二次操作必须要同时还原左右两个端点。

但是我们可以调整这样的方案使得其中一个端点在第一次或第三次被还原。

于是我们只需要考虑有限的翻转情况数即可。事实上，情况数是$O(1)$的。

62.3 时空复杂度

时间复杂度：$O(n)$。空间复杂度：$O(n)$。

63 D9221 Codeforces 341E Candies Game

63.1 题目大意

有n个盒子，每个盒子中有a_i个糖，现在要求进行若干个操作，每次操作选择两个盒子i, j，不妨假设$a_i \leqslant a_j$，那么我们从盒子j中拿a_i个糖放入盒子i。要求构造一个操作序列，使得最后恰好只有两个盒子中有糖。

63.2 题目解法

一个重要性质：如果我们任意选择三个盒子A, B, C，我们总能够找到一种方法，使其中某一个变空。

不妨假设盒子中糖的数量为$A \leqslant B \leqslant C$。设$t$ 为$\lfloor \frac{B}{A} \rfloor$。我们这样构造：从低到高考虑$t$的每个二进制位，如果这一位为1，那么操作$(A, B)$，否则操作$(A, C)$。这样操作之后，我们就从$B$中减掉了$t$个$A$, 也就是把$B$变成了$B \bmod A$。那么我们重新将$A, B, C$排序后，新的$A'$至多是$A$的一半。反复这样的操作，就能使$A$ 变为0。

那么我们每次任选三个不空的盒子执行以上操作，最后一定能剩下两个有糖的盒子。

63.3 时空复杂度

时间复杂度：$O(n \log^2 v)$。空间复杂度：$O(n)$。

64 D8795 Codeforces 342D Xenia and Dominoes

64.1 题目大意

有一个$3 \times n$的格子，有一些地方是禁止块，还有一个圆圈标记。要求将整个图放满1×2 的多米诺骨牌，禁止块和圆圈标记不能被覆盖，且其它格子全部被覆盖。并且要求这个拼图能够被移动一次。移动是指将圆圈标记与它旁边的一个骨牌交换位置。

64.2 题目解法

用状态压缩来表示状态，然后按列来依次转移即可。

为了满足整个拼图能被移动，我们额外开一维来保存当前状态是否已经满足这个条件。

用位运算可以很好的简化代码。

64.3 时空复杂度

时间复杂度：$O(2^3n)$。空间复杂度：$O(2^3n)$。

65 D9222 Codeforces 346E Doodle Jump

65.1 题目大意

给定一个序列$a \times i \ mod \ p(i \leqslant n)$，求序列排序后相邻两项差值的最大值。

65.2 题目解法

我们发现：依次得到的总是若干个公差相同的等差数列。那么我们只需要考虑等差数列的第一项即可。这样就可以将问题的规模缩小（实际上，至少缩小一半）

注意对于最后一个等差数列，如果它的长度不够，那么这一个等差数列就不应该考虑到问题中。

65.3 时空复杂度

时间复杂度：$O(\log a)$。空间复杂度：$O(1)$。

66 D9187 USACO DEC 05 Gold Cow Patterns

66.1 题目大意

给定一个长为n的序列a，再给定一个长为k 的序列b，求序列a 有多少个连续子序列与序列b 相似。

相似的定义为两个序列离散化后相同。

66.2 题目解法

对两个序列，求出每个位置离前一个和他相同的数的距离。

那么我们就是要对a序列找出有多少个连续子序列和b一样。但我们需要对每一种数第一次出现的时候特殊判断一下它的前缀排名。

将KMP算法稍作修改即可。

66.3 时空复杂度

时间复杂度：$O(ns)$。空间复杂度：$O(ns)$。

67 D8790 USACO Open07 Gold Connect

67.1 题目大意

给定一个2*n的网格图，要求支持：删除一条边，加入一条边，询问两点连通性。

在询问$(r1,c1)$到$(r2,c2)$连通性时，只能访问列坐标在$[min(c1,c2),max(c1,c2)]$范围内的点。

强制在线。具体来说，我们有一个值xr,每次操作额外给出两个值$A0[i],A1[i]$，如果上一次询问的答案是Y，则$xr=(xr+A1[i])\&1048575$，否则$xr=(xr+A0[i])\&1048575$。之后将$r1,c1,r2,c2$均异或xr。

67.2 题目解法

本题有一个经典的线段树做法，在此略过不谈。下面讨论一个破解加密的方法：

注意本题要求$r1,r2$都只能是1或2，那么在$A0$和$A1$是随机生成的前提下，我们几乎可以确定上一问的答案。而在不能确定的时候（几率只有五十万分之一），我们直接暴力做就可以了。

67.3 时空复杂度

时间复杂度：$O(n)$。空间复杂度：$O(n)$。

68 D9167 USACO DEC 07 Gold Best Cow Line

68.1 题目大意

一个串，每次可以从左或从右取出一个字符，求组成的串中字典序最小的。

68.2 题目解法

每次贪心的选择左右中较小的一个。如果左右两个一样大，那么向里找，直到出现一个不同的，从较小的那边开始选。

优化：由于当一个区间两边向里找时，区间中点不变，因此我们可以用f[i]表示以i/2 为中点的区间当前最靠外的一个不同点。如果发现当前区间的f[i]在区间内，那么它必定就是找到的不同的地方，可以直接调用。而如果当前区间的f[i]在区间外，就从当前地点向内找，并更新f[i]。

68.3 时空复杂度

时间复杂度：$O(n)$。空间复杂度：$O(n)$。

69 D8801 USACO MAR 08 Gold Land Acquisition

69.1 题目大意

有n块土地，每次可以买一些土地，价格是这些土地中最长的宽乘以最长的长。

现在求买下所有土地的最少花费。

69.2 题目解法

我们去掉所有没有意义的土地，也就是存在一块土地宽与长均比它大的土地。

之后我们将土地排序后容易得到动态规划转移方程：$dp[i] = min(dp[j] + l[j+1] * w[i])$

使用经典的斜率优化求解即可。

69.3 时空复杂度

时间复杂度：$O(n)$。空间复杂度：$O(n)$。

70 D8742 USACO Open08 Gold Cow Neighborhoods

70.1 题目大意

一个平面上有一些点，如果两个点的曼哈顿距离不超过C就连一条边，问有多少个连通块以及最大的连通块的大小。

70.2 题目解法

考虑曼哈顿距离时，有一个比较厉害的想法。我们把每个点的坐标变为$(x+y, x-y)$，那么对于一个在原图中$|x_1-x_2|+|y_1-y_2| \leqslant C$的点，在新图中就有$|x_1-x_2| \leqslant C$且$|y_1-y_2| \leqslant C$。也就是对于一个点，它影响的范围从一个斜着的正方形变成了一个边平行于坐标轴的正方形。

后面的问题就比较简单了。我们将所有点按x排序后，从左向右扫一遍，对每个点，我们考虑它左边的x不小于它的x减C的点，然后取出最接近$y+C$和$y-C$的两个点，用并查集合并一下就行了。这个可以直接用set来维护。

70.3 时空复杂度

时间复杂度：$O(n \log n)$。空间复杂度：$O(n)$。

71 D8749 USACO Nov 08 Gold Toycar

71.1 题目大意

现在有$D(1 \leqslant D \leqslant 100000)$天，每天需要$T_i(1 \leqslant T_i \leqslant 50)$ 个玩具。买一个玩具需要$Tc(1 \leqslant Tc \leqslant 60)$元钱。用过的玩具消毒后才可以再用，有两种消毒方式，第一种需要c_1 元，耗时n_1个晚上，第二种需要c_2 元，耗时n_2 个晚上。$(1 \leqslant c_1, c_2 \leqslant 60, 1 \leqslant n_1, n_2 \leqslant D)$

现在求最少用多少元前可以满足每一天的需求。

71.2 题目解法

本题有一个经典的费用流做法，但是本题数据规模较大，是无法通过的。因此在此不赘述。（注：经测试，ZKW费用流可以得到90 分）

我们考虑一共需要多少个玩具，并且把它们在第一天就全部买入。

那么在靠前的几天中，我们全部使用买入的玩具。对于之后的每一天，我们优先考虑使用较为便宜的消毒方式，如果没有玩具可以使用较为便宜的消毒方式的时候则使用较贵的消毒方式。但是要注意，使用较贵的消毒方式时，我们需要尽量使用“上次使用时间更晚”的玩具，来让更多的玩具可以使用便宜的消毒方式。

经过以上方法我们就可以求出在确定购买玩具的数量的情况下，我们的最少花费。

仔细思考可以发现，答案随购买玩具的数量呈单峰关系。这是因为随着购买玩具的变多，每多购买一个玩具带来的收益在下降。

于是我们可以三分购买玩具的数量，并套用上述算法求出每次的花费。这样就可以通过本题。

71.3 时空复杂度

时间复杂度：$O(n \log(\sum T_i))$。空间复杂度：$O(n)$。

72 D9230 USACO MAR 09 Gold cleanup

72.1 题目大意

有一列数，每次我们选择靠前的若干个数，付出的代价是其中不同数个数的平方。求最少代价来选择所有数。

72.2 题目解法

$dp[i]$表示选择前i个数的最少代价。

$$dp[i] = min(dp[j] + count(j+1, i)^2)$$

其中$count(l, r)$表示l到r中不同的数的个数。

我们发现：如果所有数都单独选择，那么答案为n。所以显然，答案不会超过n，那么一次选择最多不超过 $\sqrt{n}$ 种不同的数。

那么我们可以维护 $\sqrt{n}$个指针，表示从何处开始到当前位置的数的种数不超过i。那么我们就可以做到$O(\sqrt{n})$的转移了。

72.3 时空复杂度

时间复杂度：$O(n\sqrt{n})$。空间复杂度：$O(n\sqrt{n})$。

73 D8779 USACO Open09 Gold tower

73.1 题目大意

给定N个数，要求把数列分为若干段，使每一段的和不小于后一段的和。求最多能分为多少段。

73.2 题目解法

考虑设$f[i]$表示以i为开始的数列最多能分为多少段，$g[i]$ 表示在以i为开始的数列在分为最多段的前提下，最前一段的最小和。那么我们容易得到DP方程。

但是，可以发现：我们在让$g[i]$最小的前提下，总是能让$f[i]$ 最大。这是由于$f[i]$总是从n 到1 依次递增的。

那么我们就可以用一个单调队列简单的解决这个问题了。

73.3 时空复杂度

时间复杂度：$O(n)$。空间复杂度：$O(n)$。

74 D8788 USACO DEC 10 Gold Threatening Letter

74.1 题目大意

我们有无数份上面写着一行有N个字母的文字的报纸。现在我希望拼出一个由M个字母构成的句子。每次我可以剪下报纸上文字的一个子串。

求最少需要剪多少次。

74.2 题目解法

考虑贪心策略：每次我们从希望拼出的句子的第一个字母开始向后，找到最长的一个在母串中出现过的子串，把它剪去。

这样贪心的正确性显然。因为每次剪得多一定不会比剪得少差。

于是我们构造母串的后缀自动机。然后将要拼的串在后缀自动机中检查，一旦失配就将答案加一并退回后缀自动机的初始节点。

74.3 时空复杂度

时间复杂度：$O(nS)$。空间复杂度：$O(nS)$。

（S为字符集大小，在本题中为26.）

75 D8762 USACO Dec 12 Gold First!

75.1 题目大意

给定n个串，问哪些串可以通过重排字符的字典序，使它成为字典序最小的串。

75.2 题目解法

构造一棵trie树。

考虑一个串：如果在根到它的路径上有一个结束节点，那么它显然不可能，因为存在一个它的前缀，而它的前缀的字典序总是小于它。

之后：对于根到它路径上的每一个点，不妨设我们下一步应该走的边为t，那么对于其它字母i，必须有t的字典序在i 之前。不妨抽象成一条t 到i 的有向边，那么我们只需要考虑最后的图中是否存在一个环，使用dfs 判断即可。

75.3 时空复杂度

时间复杂度：$O(slen + n \times 26^2)$。空间复杂度：$O(slen)$。

76 D9207 GCJ2008 Final E The Year of Code Jam

76.1 题目大意

有一个$n \times m$的表格，每个格子是白色或者蓝色或者还没有定颜色的。我们定义一个蓝色的格子的得分为它旁边白色格子的数量。要求将没定颜色的格子定为白色或者蓝色，使得总得分尽量大。

76.2 题目解法

实际上，每一条蓝色和白色格子之间的边会使得分增加1。而由于边的总数是固定的（我们认为整个格子外面包了一圈白色格子），那么我们为了最大化异色格子之间边的数量，只需要最小化同色格子之间边的数量。

我们将整个表格棋盘染色，也就是把$i+j$是奇数的位置的颜色反过来。那么本来的同色格子之间的边就变成了异色格子之间的边，异色格子之间的边变成了同色格子之间的边。这样，我们就需要最小化异色格子之间的边。

那么我们就可以进行网络流构图。对每个格子，如果它确定为白色（注意是棋盘染色后，下同），那么从源向它连无穷大的边，如果确定为蓝色，从它向汇连无穷大的边。然后将相邻的格子之间连容量为1 的双向边。这个图的最小割就是异色格子之间边的最小值。那么我们用总共的边数减去它就是原图中异色格子之间边的最大数量。

76.3 时空复杂度

时间复杂度：$O(Tmaxflow(nm, 5nm))$。空间复杂度：$O(nm)$。

$maxflow(n,m)$表示n个点m条边的网络流复杂度。

77 D8786 GCJ2009 Final A Year of More Code Jam

77.1 题目大意

一年有N天，这一年有T轮比赛。每轮比赛有$m[i]$场，分别为第$d[1]...d[m[i]]$天。其中$d[1] = 1$。

每轮比赛的第一天在N天中等概率随机一天，其后的每天依次排布。（可能在第二年）。

对今年的每一天，如果这一天有s场比赛，那么就会获得s^2的愉悦值。

求总愉悦值的期望。

77.2 题目解法

对每一天，假设第i轮比赛在这一天有比赛的概率为$p[i]$。那么$p[i]$容易求出。

设Y_i表示第i轮比赛在这一天是否有比赛。那么我们要求的就是：$E((\sum Y_i)^2)$

根据期望的线性性，我们容易得到上式即：$\sum E(Y_i^2) + 2\sum_{1\leqslant i<j\leqslant T} E(Y_iY_j)$

容易得到：$E(Y_i^2) = p[i]$，$E(Y_iY_j) = p[i]p[j]$。

那么我们就容易计算答案了。实现时注意：为了防止溢出，可以用三个变量表示当前答案是$a + \frac{b}{N} + \frac{c}{N^2}$。

77.3 时空复杂度

时间复杂度：$O(GT^2)$。空间复杂度：$O(Tm)$。

（G指$d[m]$的上界）

78 D9294 GCJ2009 Final B Min Perimeter

78.1 题目大意

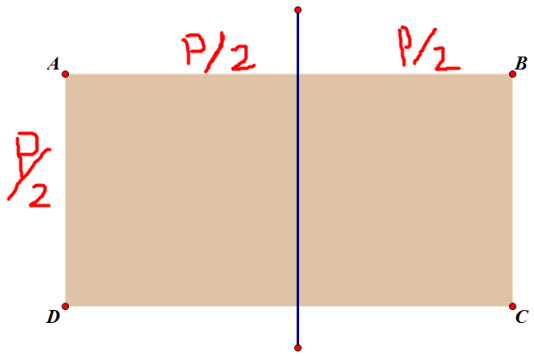
给出平面上n个点，求其中周长最小的三角形。允许退化为线的三角形。

78.2 题目解法

考虑分治。

每次，我们将当前的点集按照某一条垂直于x轴的线均分成两份，两边分别求解。不妨设两边求出的最小周长中较小的为p。显然，我们的答案一定不超过p。

接下来，我们只需要考虑所有跨越均分线的三角形。我们将当前点集按照y坐标排序，然后从下往上依次考虑每个点。显然我们只需要考虑到均分线的距离不超过$\frac{p}{2}$的点。如下图：



我们当前考虑的点在矩形的上边缘上，那么我们显然只需要考虑矩形内部的点。而容易证明，矩形内部的点不超过12个。我们直接暴力枚举即可。

复杂度为$O(n \log n)$。由于每次需要在矩形内枚举两个点，常数会很大。

78.3 时空复杂度

时间复杂度：$O(n \log n)$。空间复杂度：$O(n)$。

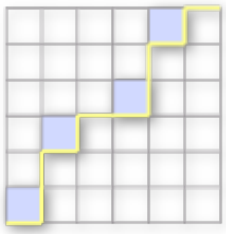
79 D8789 GCJ2009 Final C Doubly-sorted Grid

79.1 题目大意

有一个$n \times m$的网格，每个格子中需要填入一个小写英文字母。现在有一些格子中已经填入了字母，有的还没有填入。求有多少种方案填满网格，使得每一行每一列的字母从左到右（从上到下）都是单调不下降的。

79.2 题目解法

见下图：



我们实际上就是要找出26条像上图一样两两不交叉的折线（允许部分重合），这样每两条直线间填上对应的字母，就得到了一个满足条件的方法。

我们发现所有折线的数量是C_{n+m}^{m}条。考虑这样DP：

$dp[S][j][k]$表示当前折线为S，是第j个字母的折线，且第k列之后折线上方不存在字母j，有多少种方案。

考虑当前的一列：当且仅当它这里有一个拐点（类似右上图中蓝色格子），我们才可以在这里放一个字母j。

那么转移有两种：要么，这一列不放字母j，那么$dp[S][j][k]$ 从$dp[S][j][k+1]$ 转移过来。要么，这一列当前位置放字母j，要求这里允许放字母j，那么$dp[S][j][k]$ 从$dp[S'][j][k]$ 转移过来，其中S'表示将这个位置删去之后的折线。

另外，$DP[S][j][m+1] = DP[S][j+1][0]$。

然后需要注意常数：这个DP可以用一个一维数组来实现，可以大幅度减小常数。

79.3 时空复杂度

时间复杂度：$O(26TmC_{n+m}^{m})$。空间复杂度：$O(2^{n+m})$。

80 D9232 GCJ2009 Final D Wi-fi Towers

80.1 题目大意

在一个二维平面上有n个信号塔，每个塔有一个信号范围。现在我们可以改造一些塔，每一个塔改造有一个收益（有可能为负），如果改造了一个塔，那么所有在它范围内的塔必须都被改造。求最大收益。

80.2 题目解法

这是一个经典的最大权闭合图的模型。考虑最小割。

新建源、汇。对每个点，如果它的收益为正，那么从源向它连权值为收益的边。否则从它向汇连权值为收益的相反数的边。如果点j在点i范围内，从i向j连一条权值为无穷的边。

所有正收益的和减掉最小割就是答案。

80.3 时空复杂度

时间复杂度：$O(maxflow(n,n^2))$。空间复杂度：$O(n^2)$。

（$maxflow(n,m)$表示n个点m条边的网络流复杂度）

81 D8792 GCJ2009 Final E Marbles

81.1 题目大意

有n对珠子放在x轴上，我们要把每一对珠子之间连起来。每根线只能先沿y方向，再沿x方向，再沿y方向这样。问最高的线（y值最大的）减掉最低的线的最小值。

81.2 题目解法

实际上，我们考虑每条线放在x轴上方还是下方，要求上方的线之间不能相交，下方也一样。

那么我们考虑所有必须相交的珠对。它们一定放在x 轴的两侧。我们把这样的点之间连一条边，那么我们就能得到一个二分图。（如果不是二分图就显然无解）

接下来，我们考虑两个二分图之间的关系。考虑两个二分图A 和B中各一个珠对a 和b。如果存在a,b满足a 包含b，那么我们就说A包含B。否则我们说A和B相离。容易发现，任意两个二分图要么包含要么相离，不会出现相交的情况。

那么，如果对每个二分图，找到包含它的最小的二分图，那么整个会构成一个树形结构。显然父亲的答案会受到子节点答案的影响。

然后我们就可以DP了。对一个二分图，我们记录$dp[x][h]$表示对于二分图x，当x 轴上方的高度不超过h时，x轴下方的高度最小是多少。我们只需要将当前二分图所有珠对以及子二分图排好序，然后记录一下前缀的x 轴上方和下方的最小的最大高度即可。

81.3 时空复杂度

时间复杂度：$O(Tn^2)$。空间复杂度：$O(n^2)$。

82 D8771 GCJ2009 Final F Lights

82.1 题目大意

在坐标范围是[0, 100]的房间中，有2个光源，一个发出红光，一个发出绿光。还有n 个圆柱。没有被任何一色光到达的地方是黑色的，同时被红光和绿光照着的地方是黄色的。求黑色、红色、绿色、黄色的面积分别有多大。注意不包含圆柱的面积。

82.2 题目解法

首先，我们只要求被红光照着的面积，被绿光照着的面积，以及被红绿光同时照着的面积。这样就可以使用简单的容斥计算出四个区域的面积了。

注意到坐标范围很小，考虑使用simpson 积分。

由于simpson积分要求函数是连续可导的，但这个问题中有一些地方函数并不连续。我们可以对每一个$[i, i+1]$这样的坐标范围单独积分，由于所有坐标和半径都是整数，可以保证一小块内函数一定连续。

那么我们需要考虑的问题就是：对于$x = x_0(0 \leqslant y \leqslant 100)$ 的一条线段，它有多长一段是被照亮的？对于这个问题，我们枚举每一个圆柱，它一定是让线段的某一个区间无法被照亮，那么我们就求出所有区间后，将区间端点排序，就可以统计出照亮的长度了。

然后考虑如何求出某一个圆柱对某一个光源来说会挡住哪一个区间。可以分圆柱和线段有交以及无交两种情况讨论。当没有交的时候，求出光源对圆柱的切线后就容易得出区间了。当有交的时候需要判断一下，如果切点和光源不在同一侧，那么遮挡住的就不是切线和线段的交点，而是圆和线段的交点。

细节比较繁杂，仔细讨论后即可。

82.3 时空复杂度

时间复杂度：$O(100nsimpson)$。空间复杂度：$O(n)$。

($simpson$代表simpson积分的复杂度，一般不会太大）

83 D9177 GCJ2010 Final A Letter Stamper

83.1 题目大意

给定一个长度为n的，仅包含'A','B','C' 三个字母的字符串，我们有一个栈，每次可以：压一个字母进序列，弹出栈顶的字母，输出栈顶的字母。要求输出给定字符串，求最少操作次数。要求最后栈为空。

83.2 题目解法

显然我们有以下性质：

1.如果栈顶的字母和我们当前需要输出的字母相同，那么我们一定优先输出它。

2.我们总是不会压入一个和栈顶的字母相同的字母。

3.如果我们需要压入字母，那么我们总是压入当前需要输出的字母。

还有一条不是那么显然的性质：

4.如果当前栈中的第二个元素和我需要输出的字母相同，那么我们选择弹出栈顶元素而不是重新压入。

这是因为：我们总是可以修改压入的方案为弹出的方案，而使得答案不会变差。

那么我们栈中的元素总是三个字母某种排列的不断重复。这样我们就可以DP 了。

83.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(n^2)$。

84 D8794 GCJ2010 Final C Candy Store

84.1 题目大意

有k个人，每个人选一个不超过C的正整数。现在要求找出最少的数的集合（允许重复），要求不管k 个人选的什么数，都能给每个人一些数，使其相加等于他选的数，且每个数只给一个人。

84.2 题目解法

先介绍算法，随后是正确性的证明。

我们令当前所有数的和为s。每次，我们加入一个新数$\lfloor \frac{s}{k} \rfloor + 1$ 。直到s 大于$k \times C$ 为止。

首先，我们的s总是满足，没有一个数大于$x = \lfloor \frac{s}{k} \rfloor + 1$。当我们考虑，如果$k$个人的数之和不大于$s$，那么我们可以用这些数满足要求。我们满足要求的方法是：依次考虑每个人，尽量选择大的数给他。

我们使用归纳法证明以上结论。当我们加入一个新数x时，s 增大到了$s + x$。那么没有一个数大于$\lfloor \frac{s+x}{k} \rfloor + 1$。这是显然的。

接下来我们考虑：如果存在一个人的数大于等于x，那么我们把新加入的x这个数给他。这时，我们的问题就回到了加入x 之前的版本。根据归纳法，我们是可以满足所有人的要求的。

而如果没有任何一个人的数大于等于x，那么实际上我们不需要用到x 这个数，因为所有人的数之和最多$k(x-1) \leqslant \frac{ks}{k} = s$，我们无视$x$这个数之后，同样与前一个问题相同。根据归纳法，可以满足所有人要求。

这样我们就证明了这个算法是正确的，接下来我们证明这个算法是最优的。

不妨考虑另一个数的集合T。我们将它和我们以上算法构造出的集合从小到大一一比较。

考虑一个不同的数，不妨设T中的为y，我们构造的为x。考虑第一个$y > x$，那么考虑k 个人选择的数都是x，此时的和为$kx > s$，其中s 为我们得到x 之前的s。那么T 必定无法满足条件。

而如果不存在以上情况，那么所有$y < x$，容易发现T 中数的个数不可能少于我们构造的，因为所有数之和必须大于$k \times C$。

那么我们就证明了以上算法构造出的集合一定是最优的。

84.3 时空复杂度

时间复杂度：$O(ans)$。空间复杂度：$O(1)$。

85 D9229 GCJ2011 Final A Runs

85.1 题目大意

给定一个字符串，一个极大连续相同子序列称为一个run。

问有多少种原字符串的排列，使它与原字符串的run的数量相同。

85.2 题目解法

令$dp[i][j]$表示考虑前i种字母，有j个run 的排列个数。

我们考虑转移：设前i种字母共有S_i个，那么我们就有$j+1$个插入一段字符$i+1$会导致run的数量+1的位置，还有S_i-j个插入一段字符$i+1$会导致run 的数量+2 的位置。那么由此，我们得到转移方程：

$$dp[i+1][j+x+2y]+=dp[i][j]\times C_{j+1}^{x}\times C_{S_i-j}^{y}\times C_{num_{i+1}-1}^{x+y-1}$$

85.3 时空复杂度

时间复杂度：$O(nm)$。空间复杂度：$O(nm)$。

（m指run的数量）

86 D8747 GCJ2011 Final B Rains Over Atlantis

86.1 题目大意

有一个$n \times m$的地图，每个点有一个海拔高度。现在下起了大雨，有的点可能会有积水。我们把积水的高度叫做这个点的水平面（如果没有积水就是它的海拔高度）。认为地图外的水平面高度为0。

现在有的点会被侵蚀。具体来说，对某一个点S，如果它四周的点中水平面最低的是T，并且T的水平面低于S，那么S 的海拔会降低$min(h_S - h_T, M)$，M是定值。所有的点会同时被侵蚀。

现在求多少天之后，所有点的海拔高度全部为0？

86.2 题目解法

不妨假设我们已经计算出了所有点的水平面。

暴力的做法是：考虑每个没有积水的点，然后在它旁边找到一个水平面最低的点。

我们可以发现一个非常重要的结论：如果一个点在某时刻没有积水了，那么它以后再也不会积水了。

然后还有另一个非常重要的结论。我们一天一天考虑：第一天，最外面一圈的土地显然会以M的速度下降。第二天，第二圈的没有积水的土地中至少会有一个在以M 的速度下降。依此类推……最多$O(nm)$天之后，所有没有积水的土地都会以M的速度下降，直到有某一个土地的海拔变成0了或者有某一个积水了的土地不再积水。

接下来的问题就比较显然了。我们至多暴力模拟$O(nm)$天之后，所有没有积水的土地都会以M的速度下降。那么我们直接算出多少天之后会出现某一个土地的海拔变成0了或者某一个积水的土地不再积水。显然，这样的次数最多也就是$O(nm)$ 次。那么我们就只需要暴力模拟$O(n^2m^2)$次。

最后，计算所有点的水平面实际上就是找海洋到每个点的最小瓶颈路。这个可以在$O(nm)$的时间内解决。简单的话也可以直接用SPFA，因为图时刻在变化，SPFA的复杂度基本上是趋近$O(nm)$ 的。

86.3 时空复杂度

时间复杂度：$O(n^3m^3)$。空间复杂度：$O(nm)$。

87 D8780 GCJ2011 Final C Program within a Program

87.1 题目大意

有一个机器人站在数轴上，他需要往东走n步。现在数轴上每一个整点有一个电线杆，每根电线杆上有一个标记，初始均为0。机器人本身有一个状态，初始也为0。机器人有若干条行动规则，每条规则形如$<S><M>-><$操作$>$，这表示当机器人的状态为S且当前所在的电线杆上标记为M时机器人的行动。操作有两种，如果是一个'R'则表示程序结束，否则操作形如$<D><NS><NM>$，D表示下一步移动的方向（'W'为向西，'E'为向东），NS表示移动后机器人所处的状态，NM表示离开的这一根电线杆的新标记。也就是说机器人会先更改标记，然后移动，最后修改状态。

现在给定n，希望你给出一系列规则，使机器人在程序结束时位于数轴的n位置。注意规则数最多只能有30条，并且机器人的移动次数不能超过1.5×10^5。

87.2 题目解法

我们在数轴上用$\log n$个位置的电线杆上的标记表示当前的n。

程序的一开始，我们先把n的二进制表示写在前$\log n$个电线杆上。然后，我们模拟一次二进制减一的操作。如果发现我们当前的数不能减一了，那么我们就直接结束程序。否则，我们将整个二进制表示的n向右移动一位。

这样，我们每次减一都向右移动了一位。最后就可以停在n的位置上了。

注意由于0是默认的空状态，二进制应该用1和2来表示。

87.3 时空复杂度

时间复杂度：$O(\log n)$。空间复杂度：$O(1)$。

88 D9158 GCJ2011 Final D Ace in the Hole

88.1 题目大意

Amy有一个$1-n$的排列，其中满足没有任意一个长度为3的子序列（不一定连续）为递减的。现在Ben会按某种顺序询问序列中每个位置的值，我们要帮助Amy构造一个排列，满足Ben询问的最后一个位置的值为1。如果有多解求字典序最大的。

88.2 题目解法

算法是这样的：对于某个询问，我们设当前还没有询问过的位置为$p_1...p_m$，还没有出现过的值为$v_1...v_m$。那么对于一个询问p_k，如果它询问的不是p_m，那么p_k位置上的值为v_{k+1}。否则：如果$m \leqslant 2$或者存在一个已经询问过的i满足$i < p_m$且i上的值介于v_{m-1}和v_m之间，那么p_k位置上的值为v_m，否则为v_{m-1}和v_m都可以。为了字典序最大，我们应该选择v_{m-1}。

证明如下：

这个问题是一个博弈问题。每次Ben选定一个位置，然后Amy 把那个位置上放上一个值，要求不能放重复的值，不能放出长度为3 的递减子序列，以及在最后一轮之前不能放1。

首先我们证明：以上算法不会产生长度为3的递减子序列。

在过程中，我们称一个位置是“安全的”，当这个位置已经被Ben选过（也就是值已知），且所有大于它的位置都已经被选过或者大于它的值都已经被选过。

考虑所有不安全的位置$p_1...p_m$以及值$v_1...v_m$。

我们可以证明两点：

- 1.没有一种选定位置的方法能够形成一个包含“安全位置”的长度为3的递减子序列。
- 2.如果一个不安全的位置p_i已经被选定过了，那么它的值是v_{i+1}。

考虑使用归纳法：

一开始，所有的位置都是不安全的且没有被选定过，那么以上结论显然成立。

不妨设Ben这一次选择的是位置p_k。显然它当前是不安全的。分情况讨论：

- 1.$k < m-1$。显然位置p_m还没有被选定过（否则它就一定是安全的，因为所有大于它的位置都被选定过了），因此我们应该根据第一条策略。设有x个小于p_k的位置还没有放值，那么p_k就应该放第$x+1$小的还没有放的值。根据归纳假设，这个值显然就是v_{k+1}。由于$k+1 < m$，不会有不安全的位置变为安全的位置，那么以上结论仍然成立。

- 2.$k = m-1$。和第一种情况相同，位置p_k的值应该是v_m。但此时安全位置会有所增加。不妨设$p_u...p_{m-1}$已经被选定过，而p_{u-1}还没有被选定过。那么$p_u...p_{m-1}$都会变为安全位置，因为所有大于它们的值都已经出现过了。注意到，此时只有p_m一个大于它们的位置还不是安全位置，而根据归纳假设，$p_u...p_{m-1}$ 必定是一个递增序列，那么包含它们的下降子序列最长也只有2。因此，以上结论仍然成立。

- 3.$k = m$。显然这时v_m还没有出现过。如果v_{m-1} 已经出现过，那么出现的位置显然在p_m之前，根据我们的策略，p_m 位置只能放v_m。否则，我们既可以放v_m也可以放v_{m-1}。

  - 如果放v_{m-1}：显然p_m位置会变成安全位置，而所有其他位置都不会改变。而p_m位置并不会组成长度为3的下降子序列，因为至多只有一个大于它的值位置在它之前，而由于大于它的位置都是安全位置，根据归纳假设它们不可能被包含。
  - 如果放v_m：考虑$p_u...p_{m-2}$已经被选定过，它们也会变成安全位置，因为$v_{u+1}...v_m$ 都已经出现过。注意到前面不会再有比他们大的数了，而在它们之后只有p_{m-1}一个位置仍然是不安全的，那么不可能会有长度为3的下降子序列包含它们。

这样，我们就证明了：我们的算法不会产生一个长度为3的下降子序列，且只有最后一次询问才会出现1。

接下来我们证明：Amy必须使用以上算法，否则会被Ben提前找到1。

我们先不考虑已经被询问过的位置。只考虑m个空位置。那么我们不妨认为剩下的值就是1...m。假设Ben 这次选定的位置是k。

- $k < m$。此时Amy必须放上$k+1$。假设放的是$j \neq k+1$。那么此时，位置m不可能是1。这是因为，如果位置m放了1，那么前$m-1$个位置必须是有序的，否则会出现长度为3的下降子序列。那么容易知道位置k的值是$k+1$，而现在它的值是j，矛盾。因此位置m不能是1。这样，Ben就可以直接把小于m的位置都询问一遍，一定能找到1。这样1 就提前被找到了。因此Amy只能放上$k+1$。

- $k = m$。此时Amy必须放上$m-1$或m。假设放了$j < m-1$。如果$m = 3$，那么显然这一次就出现了1，是不行的。否则，我们有$m > 3$。注意到，如果我们把前面$m-4$个位置以及位置$m-2$询问一遍，根据上一种情况，得到的序列必然是：$2, 3, ..., j-1, j+1, j+2, ..., m-2, ???, m, ???, j$。剩下两个位置的值还剩1和$m-1$。显然，在$m$和$j$之间是不能放上$m-1$的，因为$m, m-1, j$构成了一个长度为3的下降子序列。因此那个位置只能放1，这样1就被提前找到了。因此Amy必须放上$m-1$或m。

那么我们就只剩下一种情况了：假设当前没有确定的位置是$p_1...p_m$，没有确定的值是$v_1...v_m$，当前Ben询问了位置p_m，而存在一个位置$i < p_m$且位置i上的值位于v_{m-1} 和v_m之间，此时Amy必须让p_m上的值是v_m。

不妨假设Amy在位置p_m上放了v_{m-1}。考虑所有的不安全位置$q_1...q_r$与值$w_1...w_r$。注意到一定有u使$v_{m-1} = w_u$。而我们考虑位置i，由于v_m还没有出现，且p_m还没有确定，那么i一定是一个不安全的位置。考虑它的值是w_t，而由上面的结论知$i = p_{t-1}$。另外由于$w_t > v_{m-1} = w_u$，因此$t > u$。

注意到：由于w_u被放在了位置$q_r = p_m$上，那么位置q_{u-1}就还没有确定（否则它上面的值必定是w_u，这与w_u出现在q_r上矛盾）。q_{r-1}也还没有确定，否则它的值会是w_r，这样它就变成了一个安全位置。接下来，Ben可以询问掉所有除了q_{u-1}和q_{r-1}之外的位置。剩下的值就会是1和w_{u+1}。但位置q_{r-1} 上不能是w_{u+1}，否则会构成下降子序列：w_t, w_{u+1}, w_u。那么位置q_{r-1} 上就只能是1。这样1就会被提前找到。

综上所述，我们证明了我们的算法是可行的而且是必须的。

88.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(n)$。

89 D9163 GCJ2011 Final E Google Royale

89.1 题目大意

我们在进行一场赌博。一开始我们有A 元钱，如果我们的钱小于等于0 那么就输了，如果我们的钱大于等于V那么就赢了。每次，假如我们当前有X 元钱，我们可以自由选择$[1, min(M, X)]$ 中的一个整数的钱来押注。不妨假设我们压了B元钱。我们每一次赌博都有50%的概率胜，有50%的概率败。如果我们胜了，那我们会获得B 元钱。如果我们败了，我们有两种选择：要么，我们可以直接支付B元钱。或者，我们也可以选择暂时欠着这B 元钱，把赌注翻倍变为$2B$ 进行下一次赌博。注意，如果赌注翻倍后会超过M那么就不能翻倍了。当我们胜利一次或者选择不再翻倍之后，再结算之前所有输掉的钱。现在问：在我们一开始有A 元钱的情况下，如果我们使用最佳策略，我们有多大的概率能够赢得整场赌博（最后钱大于等于V）？并求出在获胜概率最大的前提下，第一次下注最多可以压多少钱？

89.2 题目解法

首先有一个非常重要的结论：不论我们下注多少，是否翻倍，翻倍几次，我们最后期望的钱始终是A。不妨设我们下注B，翻倍了k次，那么我们有$\frac{1}{2^{k+1}}$ 的概率支付$(2^{k+1}-1)B$元，有$1-\frac{1}{2^{k+1}}$ 的概率获得B 元，因此总共的期望获得的钱数是0。也就是完成这一次赌博后手上的钱和赌博之前手上的钱始终是一样的。

然后我们来证明几个引理：

引理一：如果我们有X元钱，我们不可能会下注多于$V-X$元钱。

证明：如果我们下注多于$V-X$ 元钱，那么和我们下注$V-X$元钱相比，如果这一盘赢了，那么都会获得整场赌博的胜利。如果这一盘输了，那么会输的更多。因此显然我们下注$V-X$ 元会更优。

根据引理一，如果我们获胜那么我们总是恰好有V元钱而不会更多。

引理二：不妨设我们有A元钱时能够获得整场赌博的胜利的概率为P，如果我们输掉整场赌博，最后的钱数（其实就是还欠多少钱，一个非正数）期望为L。那么我们有$P=1-(V-A)/(V-L)$。由于V和A 是固定的，那么我们应该要最小化L，这样就能最大化P。

证明：由于我们无论如何下注，最后手上的钱期望总是A。因此我们能得到：$P \times V + (1-P) \times L = A$。化简得$P = 1 - (V-A)/(V-L)$。

那么我们的任务就是最小化L。我们考虑一个最优策略：

引理三：如果我们有x元钱，我们只有两种可能的策略：要么我们将x 元全部压上去，如果输掉就一直翻倍直到不能再翻倍。要么我们只压1 元钱，如果输掉我们不翻倍。

证明：不妨假设我们有一个最优策略是压y 元并翻倍k 次。我们希望证明我们能够找到一种满足引理三的策略使它和这一个最优策略同样优。

注意到：如果我们赢了这一轮赌博，那么我们手上的钱是$x+y$ 元。否则我们手上的钱是$x - y \times (2^{k+1}-1)$。不妨设$z = y \times (2^{k+1}-1)$。

分情况讨论：

1.$x - z \geqslant 0$。此时我们不断压1 元钱并不翻倍，直到钱数变成$x+y$或者$x-z$。从引理二可以看出：如果我们胜利时的钱数和失败时的钱数以及期望的钱数固定，那么无论我们采用什么策略，获得胜利的概率总是相同的。而我们一次压y元并翻倍k 次，以及不断压1元并不翻倍，两种策略期望的钱数是相同的（都是x 元），胜利的钱数也相同（都是$x+y$），失败的钱数也相同（都是$x-z$），因此获胜的概率是相同的。那么我们就可以把原策略变为一个满足引理三的策略。

2.$x - z < 0$。这种时候我们不断压1 元就不行了，因为钱数变为0的时候就会被迫停止。但我们可以使用这样的策略：我们先不断压1元，直到钱数变为$x+y$ 或者y 元。如果我们钱数为y 元了，那么我们就压y元并不断翻倍。如果我们最后输了，那么就输掉了整场赌博。否则，我们会变为$2y$ 元，再次不断压1 元直到钱数变为$x+y$ 或者y 元。容易发现，我们的胜利钱数仍然是$x+y$，期望钱数也仍然是x。但是我们的失败钱数至少会是$y-z$，这个值显然不会大于$x-z$。那么根据引理二，我们的策略比原策略不会劣，因为我们的失败钱数不大于原策略的失败钱数。

因此总是有一个最优策略满足引理三。我们只需要考虑压1元并不翻倍和压全部的x 元并不断翻倍两种情况。接下来我们考虑：在哪一些x 我们应该压全部的x 元，而在其他的x 我们应该压1 元。

引理四：对每个非负整数i，令M_i 是满足$M_i \times 2^i \leqslant M$ 的最大整数。我们只可能在某一个M_i 上压全部的x 元。

证明：不妨考虑某一个$M_i \leqslant x \leqslant M_{i-1}$。注意到压$x$ 元最多翻倍$i-1$ 次。我们考虑如果压$y = \lfloor \frac{x+1}{2} \rfloor$ 元，我们可以翻倍i 次。

对于压x元的策略，我们的胜利钱数是$2x$，我们的失败钱数是$-(2^i-2)x$。

考虑另一个策略：我们先不断压1元并不翻倍，直到钱数变为$2x$ 或者变为y为止。如果我们的钱数变为y了，我们就压y 元并不断翻倍。如果胜利了就继续上面的步骤，否则就输掉了整场赌博。注意到，我们的胜利钱数还是$2x$，但是我们的失败钱数变为了$-(2^{i+1}-2)y$。容易发现这个值是小于$-(2^i-2)x$ 的。那么根据引理二，我们胜利的概率大于原策略。

证明了引理四，我们就将压x元的情况缩小到了一个很小的范围。但是注意，并不是所有的M_i 都应该全部压上。由于我们的目标是最小化L，因此，对每个M_i计算出如果压x 元并不断翻倍，我们的L 是多少。如果存在一个$M_j < M_i$并且$L_j > L_i$，那么我们就不应该在M_i时就全部压上，取而代之的，我们应该不断压1 元直到我们达到了M_j或者达到了一个更大的M_k。

还需要注意一点：根据引理一，我们只应该考虑不超过$\frac{V}{2}$的M_i。

下面我们所提到的M_i都指应该全部压上的M_i。如果某个M_i 不应全部压上，我们直接把它删掉不予考虑。

下面来看一下：如果某一个M_i应该全部压上，我们如何**计算它的胜率**。考虑某个x，它并不是M_i，那我们应该压一元并不翻倍。因此$P_x = \frac{P_{x-1}+P_{x+1}}{2}$。也就是$P_x - P_{x-1} = P_{x+1} - P_x$。所以在相邻的两个全部压上的点之间，胜率是一个等差数列！根据这一点，我们可以得到这样的计算方法：

从大到小考虑每一个应该全部压上的点（我们令V也是一个这样的点，并且它的胜率为1）。不妨设当前考虑的为y，前一个考虑的（恰好比y 大的一个）为z。设我们压上y 并不断翻倍最后能获胜使钱数变为$2y$ 的概率为p。那么我们就有$P_y = p \times P_{2y}$。如果有$2y \geqslant z$，那么我们就可以根据之前算出的P_z 等直接算出P_{2y}。这一步直接利用等差数列来计算即可。如果$2y < z$。那么我们可以得到：$P_{2y} = \frac{P_z - P_y}{z-y} \times y + P_y$。联立$P_y = p \times P_{2y}$，就可以直接解出$P_y = \frac{ypP_z}{(1-p)z+(2p-1)y}$。

当我们得到所有的P_{M_i}之后，我们就可以回答第一问了。找出比A 大的最小的以及比A 小的最大的两个M_i，根据等差数列公式直接计算就可以了。

接下来我们考虑**第一次下注的最大值**。如果A恰好是一个M_i，那我们显然可以全部压上去。否则，考虑恰好比A 大的是M_i，恰好比A 小的是M_{i+1}。为了保证它始终是满足等差性质的，显然，我们最多只能取$min(M_i - A, A - M_{i+1})$。并且我们不进行翻倍。这样与我们不断取1的胜率是一样的。但如果再增加下注就会对答案造成影响了，因为它不再满足等差数列性质。

我们来分析一下复杂度。M_i显然最多$\log v$ 个。如果我们在后面计算时写

的暴力一点，复杂度大概是$O(T \log v \log \log v)$ 的。实现的精细的话也可以达到$O(T \log v)$，但是没有什么必要。

89.3 时空复杂度

时间复杂度：$O(T \log v \log \log v)$。空间复杂度：$O(\log v)$。

90 D9171 GCJ2012 Final C Xeno-archaeology

90.1 题目大意

一个无限延伸的图案，正中心是一个'.'，之后按一圈'#'一圈'.'一直延伸出去。现在给出一些位置的符号，求中心的位置。多解输出距离$(0,0)$曼哈顿距离最小的点，仍然多解按x最大、y最大输出。

90.2 题目解法

我们考虑一个点A(x,y)，假设中心是(x',y')。如果A是'#'（是'.'同理），那么必定有：$max(x-x',y-y')$为奇数。

那么我们讨论以下情况：

1.$x-x',y-y'$均为奇数。此时任意(x',y')均可作为答案。

2.$x-x',y-y'$均为偶数。此时任意(x',y')均不可作为答案。

3.$x-x'$为奇数，$y-y'$为偶数。此时要求$|x-x'| \geqslant |y-y'|$。

4.$x-x'$为偶数，$y-y'$为奇数。此时要求$|x-x'| \leqslant |y-y'|$。

可以发现，以上四种情况都是根据x'和y'的奇偶性来区分的。那么我们考虑某一种x'和y'的奇偶性，要求不存在一个点为情况2。那么此时我们的要求就是求出一个点满足所有情况3 和4的点，同时尽可能优。

我们发现：情况3和情况4事实上把整个图划分为$(x+1)(y+1)$个矩形。（x 表示情况3 的点数，y 表示情况4的点数，可能有的矩形延伸到无穷远处）

如果我们枚举每个矩形并判断，时间复杂度是$O(n^3)$ 的。

但我们可以这样做：枚举相邻的两条形如$x'+y'=x_i+y_i$的直线。那么所有形如$x'-y'=x_i-y_i$的直线必定构成一个区间，那么我们就可以直接得到唯一的可能矩形。这样做的时间复杂度是$O(n^2)$ 的，可以接受。

最后，当我们确定一个矩形内部某种奇偶性的点均可行的时候，我们需要求出这个矩形中题目中要求的最优的点。我们讨论以下这些点：

1.原点。（如果原点在矩形内部的话）

2.四个顶点。

3.四条边与坐标轴的交点。但是，由于要满足题目中要求的最优性，事实上，对于右边两条边只需要考虑和Y 轴交点，左边两条边只需要考虑和X 轴交点。

对于以上这些点，它们有可能不满足要求的奇偶性，那么我们对每个点扰动一下。而为了满足题目要求的最优性，每个点都有确定的扰动方向。

仔细地实现，即可通过本题。

90.3 时空复杂度

时间复杂度：$O(n^2)$。空间复杂度：$O(n)$。

91 D9217 GCJ2012 Final D Twirling Towards Freedom

91.1 题目大意

平面上有n个给定的点。一开始我们的人在原点上，现在每次我们可以选择一个点，绕着它顺时针旋转90°。现在最多能进行m次这样的操作，问最后离原点欧几里德距离最远能有多少。

91.2 题目解法

首先，绕一个点旋转4次之后就会回到原点，那么我们只需要考虑进行$m-3, m-2, m-1, m$次操作就可以了。接下来我们只考虑恰好进行m次操作。

我们考虑用形如$a+bi$的复数形式来表示一个点$P(a,b)$。

那么我们考虑P绕原点顺时针旋转90°。容易发现$P' = -i \times P$。那么考虑P绕一个点Q_1顺时针旋转90°，显然$P' = -i \times (P - Q_1) + Q_1 = -iP + (i+1)Q_1$。接下来再考虑$P'$绕另一个点$Q_2$顺时针旋转90°，那么$P'' = -iP' + (i+1)Q_2 = -P + (1-i)Q_1 + (i+1)Q_2$。依此类推。

那么我们旋转m次之后，得到的就是$P_m = a_0 P + \sum_{i=1}^{m} a_i Q_i$。注意到$P = (0,0)$，而$a_i$只有$(i+1), (i-1), (1-i), (-1-i)$四种取值。那么实际上我们就是要求：

$P_m = (i+1)(Q_1 + Q_5 + ...) + (1-i)(Q_2 + Q_4 + ...) + ...$

显然，我们要让最后的P_m离原点的欧几里德距离尽量大，必定有$Q_1 = Q_5 = ..., Q_2 = Q_4 = ...$，这是因为如果有不相等的，比如$Q_1 \neq Q_5$，那么要么把$Q_1$变为$Q_5$会更优，要么把$Q_5$变为$Q_1$会更优。这是显然的。

接下来问题变成了：选出四个向量，让它们乘以对应的系数后的和向量的模尽量大。容易发现这四个向量一定在凸包上，那么我们先求出凸包。

我们发现，这四个向量乘以对应的系数实际上就是把它旋转了90°，180°和270°。如果我们把原凸包旋转这么多次再把每个点的坐标乘以$\frac{m}{4}$（注意四个方向上有可能有1的偏差），那么我们要求的就是$S = \{a+b+c+d | a \in T_1, b \in T_2, c \in T_3, d \in T_4\}$中离原点最远的点。

这是一类经典问题，叫做闵可夫斯基和问题。解决办法是：我们把四个凸包中所有的向量求出来，按照极角排序后依次连接起来，得到的凸包就是四个凸包的闵可夫斯基和。但是这时我们只有这个凸包的形状，而我们需要它的具体位置。我们将原来四个凸包中x最小的点（相同取y最小）的坐标加起来，这

就是最后的凸包中x最小的点的坐标，然后就可以推出其他点的坐标了。

91.3 时空复杂度

时间复杂度：$O(Tn\log n)$。空间复杂度：$O(n)$。

92 D9175 GCJ2012 Final E Shifting Paths

92.1 题目大意

有n个点，我们一开始在1号点。我们的目标是到n号点去。当我们奇数次到达i 号点时我们只能往l_i号点走，当我们偶数次到达i号点时我们只能往r_i 号点走。求到达n 号点要走多少步。

92.2 题目解法

考虑折半法。

我们首先暴力搞出我们从1号点开始到达的前$\frac{n}{2}$个点。显然我们的步数不会超过$2^{\frac{n}{2}}n$步。如果这里面能到达n号点，那么就直接得出答案了。如果发现我们到达了重复的状态，那么显然无解。不妨设前$\frac{n}{2}$个点为A 集合，其余为B集合。

之后，我们再次模拟。但是我们这样考虑：如果我们当前到达的是A 集合中的点，那么我们可以用一个递推直接得出当我们在某一个点时，我们将会到达的第一个在B 集合中的点，以及这中间的步数和出去之后每个点的状态。显然这样的状态最多只有$2^{\frac{n}{2}}n$ 个，可以接受。那么当我们到达A集合中的点时，我们就直接取出上面递推出来的状态，否则我们暴力模拟。显然，这样的操作次数不会超过$2\times 2^{\frac{n}{2}}n$ 次，是可以接受的。

92.3 时空复杂度

时间复杂度：$O(2^{\frac{n}{2}}n)$。空间复杂度：$O(2^{\frac{n}{2}}n)$。

93 D9228 GCJ2013 Final A Graduation Requirements

93.1 题目大意

有一个环形道路一共有n个路口，有C 辆车在上面逆时针行驶，每辆车在t_i的时刻从路口s_i进入，行驶到路口e_i 后离开。每个时间间隔每辆车可以行驶到下一个路口，路口按逆时针标号。我们要求找到一个时刻进入道路，顺时针行驶最多的时间，且不和任意一辆车相撞。要求离开的时刻不能晚于X。注意，相撞的定义包含进入和离开道路。

93.2 题目解法

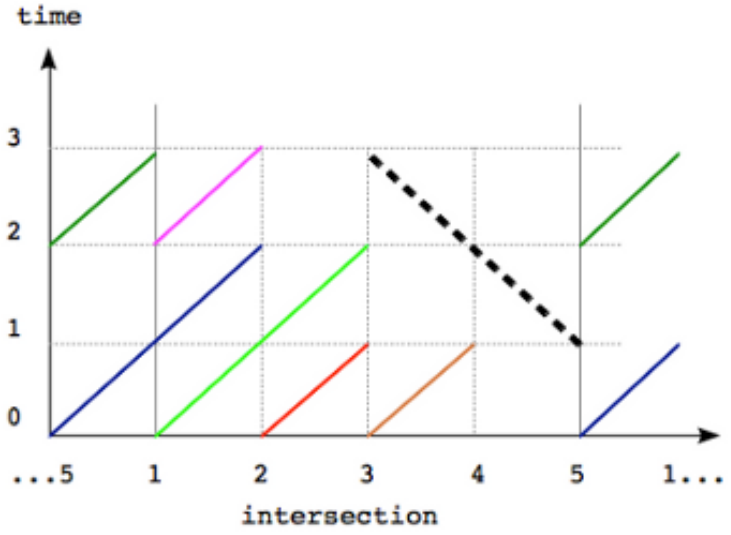
我们考虑将问题抽象在二维坐标系上。其中x坐标表示位置，y 坐标表示时间。

对于一辆车，我们实际上可以把它抽象成二维坐标系中的一条斜率为1 的线段。

但是要注意，把这条线段左右平移n 后，仍然成立。

而我们要求的是一条斜率为-1的线段，使它尽可能长且不与任何给定线段相交。

见下图：



我们发现，任意一个可行解，总是可以沿左下到右上的方向平移，直到遇到某一个线段的端点为止。

那么，我们就可以令解总是过某线段的两个端点之一的附近的点（再移动一下就会与该线段相交），然后暴力判断最长可以延伸多远即可。

93.3 时空复杂度

时间复杂度：$O(C^2)$。空间复杂度：$O(C)$。

94 D9216 GCJ2013 Final C X Marks the Spot

94.1 题目大意

一个平面上有$4 \times n$个点，我们要在平面上画两条互相垂直的直线，满足没有点在直线上，且两条直线划分出的四个区域中各有n个点。

保证所给点没有三点共线。

94.2 题目解法

考虑一条直线的倾斜角为α。那么另一条直线的倾斜角就是$\alpha + \frac{\pi}{2}$。显然，这两条直线各自都必须平分整个点集。那么它的截距显然就应该取整个点集中所有点的截距的中位数。

假设四个区域中的某一个有x个点，那么其他三个区域显然就分别有$2n - x, x, 2n - x$ 个点。那么我们只需要判断某一个区域中是否恰好有n个点就可以了。不妨设某一个区域中有$f(\alpha)$ 个点。

考虑到$f(0) + f(\frac{\pi}{2}) = 2n$。我们发现：如果$f(0) = f(\frac{\pi}{2}) = n$，那么我们直接取0就可以了。否则的话，不妨假设$f(0) < f(\frac{\pi}{2})$，那么中间就一定存在一个倾斜角$\beta$满足$f(\beta) = n$。我们就可以直接二分了。

注意：这个地方并不一定是单调的，但我们只要保证左右端点函数值一个大于n 一个小于n，就可以找到一个可行解了。

最后：如果我们发现当前的倾斜角的截距中位数会导致有点在直线上，我们稍微扰动倾斜角一点点就可以了。因为所有点中没有三点共线的，那么我们向前或向后扰动一点点，可以有一种满足平分整个点集。

实现过程中要注意精度的问题。

94.3 时空复杂度

时间复杂度：$O(Tn \log n)$。空间复杂度：$O(n)$。

95 D9206 GCJ2013 Final D Can't Stop

95.1 题目大意

有n个集合排成一行，每个集合有D个元素。我们要找出一个尽量长的区间，满足存在k 个数，区间内的每个集合至少包含这k 个数中的一个。

95.2 题目解法

考虑一个暴力的做法：我们枚举区间的开始位置，枚举它包含哪一个数，然后向右扩展直到不能再扩展，再枚举那一个集合包含哪一个数，依次类推直到枚举了k 个数。容易发现，这样做是$O(D^{k+1}n^2k)$ 的。

但是，我们只需要多加一个判断，就可以优化时间复杂度。不妨设区间的开始位置是s，当我们枚举某一个数时，如果$s-1$这一个集合包含了这一个数，那么我们就不枚举选这一个数的情况。因为这时我们一定能够将区间向左扩展一个位置来变得更优。

加上这一个判断之后，时间复杂度变成了$O(D^{k+1}nk)$的！我们考虑一个位置t，考虑它会被扩展到多少次。如果我们从s 扩展到了t，那么我们选的数一定不会在$s-1$ 这个集合中。反过来想，我们从t 向左扩展到s，接下来就无法再扩展了，需要新加入一个$s-1$ 中的数才能继续扩展。而这样的位置显然只有$\sum_{i=0}^{k} D^i$ 个，也就是$O(D^k)$ 次。那么我们的复杂度就是$O(D^{k+1}nk)$ 了。

95.3 时空复杂度

时间复杂度：$O(TD^{k+1}nk)$。空间复杂度：$O(Dn)$。

96 D8781 GCJ2013 Final E Let Me Tell You a Story

96.1 题目大意

给定一个长度为n的序列，我们按照某顺序删掉其中的数，直到剩下一个不上升序列为止，求这样的顺序有多少种。

96.2 题目解法

如果给定一个长度为k的序列，问有多少种方法得到它？

显然我们有$(n-k)!$种方法。但是：有一些方法是不可行的，因为我们一旦得到一个不上升序列就会马上停止。那么对于一个长度为$k+1$的不上升序列，且原来长度为k的序列是它的一个子序列，我们就应该减掉$(n-k-1)!$。

考虑一个长度为$k+1$的不上升序列，显然它会对$k+1$ 个长度为k 的序列有影响。那么我们就可以得到答案为：$\sum_k A_k(n-k)! - A_{k+1}(n-k-1)!(k+1)$。其中$A_k$表示长度为$k$ 的不上升序列的情况数。

我们容易使用DP求出A_k。在DP的过程中需要使用到树状数组来求前缀和。

96.3 时空复杂度

时间复杂度：$O(n^2 \log v)$。空间复杂度：$O(nv)$。

97 D9168 GCJ2014 Final C Symmetric Trees

97.1 题目大意

给一棵树，每个点有一个颜色（用大写字母表示）。我们希望把它放在一个二维平面上，使它关于y轴对称。对称的定义是：如果我们在(x,y)放了一个颜色为c的点，那么在$(-x,y)$也必须有一个颜色为c的点。（特别的，如果$x=0$，那么这两个点就是一个点，这种情况也是合法的）并且如果我们在(x_1,y_1)和(x_2,y_2)之间有一条边，那么我们在$(-x_1,y_1)$和$(-x_2,y_2)$间也必须有一条边。并且任意两条边不允许在非端点处相交。求这棵树是否能够关于y轴对称。

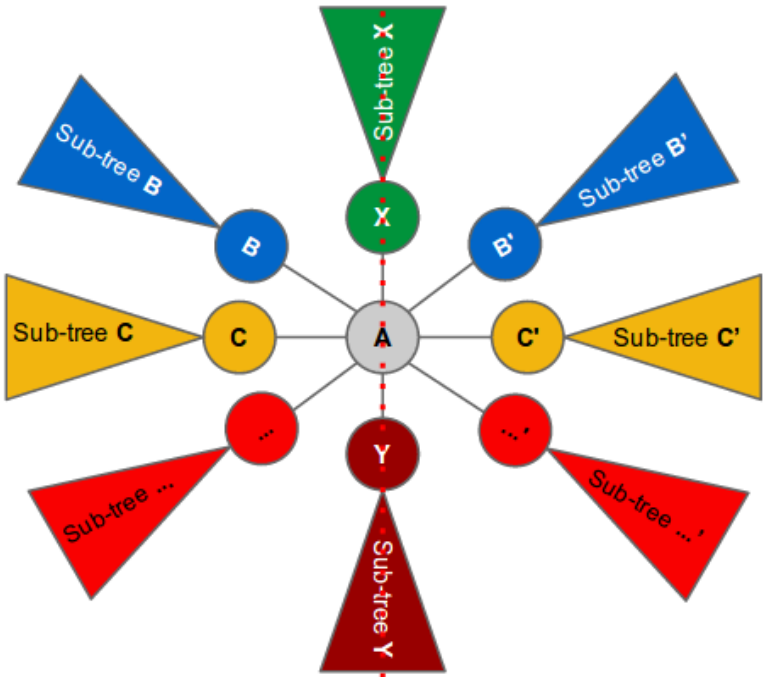
97.2 题目解法

首先我们定义两棵子树是“可对称”的，当且仅当它们能在y轴的两边完全对称地放置。考虑如何判断两棵子树是“可对称”的。对一棵子树，我们给它一个字符串：首先是一个左括号，然后是它的颜色，接下来是它所有子树的字符串按字典序从小到大接上去，最后是一个右括号。这样，我们判断两棵子树是“可对称”的，当且仅当它们的字符串完全相同。

我们考虑：y轴上是否有点。

如果y轴上没有点，那么一定有且仅有一条边穿过y轴。那么只需要判断这条边两端的子树是否是“可对称”的即可。

如果y轴上有点，那么就比较复杂了。观察下图：



注意到，如果我们选择了一个点A放在y轴上，那么它的所有子树要么也放在y轴上（如图中的X和Y，并且这样的子树最多两个），要么是“可对称”的子树两两配对后放在两侧。那么我们首先尽量将子树配对后放在y轴两侧，然后对于剩下的子树，最多只有两棵，再用相似的办法判断它们即可。

这样子做已经可以通过本题了。但是我感觉最坏情况应该是可以达到$O(n^2)$的。如果我们直接使用字符串的hash值来做的话就可以做到$O(n\log^2 n)$的复杂度。

97.3 时空复杂度

时间复杂度：$O(Tn\log^2 n)$。空间复杂度：$O(n)$。

98 D8753 GCJ2014 Final D Paradox Sort

98.1 题目大意

有n种不同的蜡烛，给出两两之间更喜欢哪一个的关系，求是否存在一个排列，满足我们从前往后依次取蜡烛与手上的比较，并留下更喜欢的一个，最后留下的是蜡烛A。如果存在这样的排列，求字典序最小的一个。

98.2 题目解法

我们不妨将原题看成一个有向图：如果x 比y 更好，那么我们连一条从x 向y 的边。

首先考虑是否存在这样的排列：如果我们从A开始DFS，能够访问到所有点，那么就存在一个这样的排列。这是因为，我们按DFS 树的后序遍历组成的排列总是合法的。

之后考虑如何求字典序最小的排列。我们不妨从前往后依次对每一位枚举它上面放的哪一个数字。那么我们的问题就是：给定排列的前几个数，求是否存在一个排列满足条件。

我们可以先求出前几个确定的数的结果，不妨设为B。如果A 已经被使用过，但A 与B不相等，那么显然不存在这样的排列。之后，我们去掉所有已经使用过的点（除了B），并同样进行DFS。但是，当我们访问到了B，我们不访问它的任何子节点。在DFS进行完后，我们再对所有没有访问过的点考虑：如果B 比它优，那么就算它为访问过。

为什么要这样做？因为我们已经知道当前手中的是B，假设存在一个C，只有B 向它有边，而又存在一个D，只有C向它有边，那么我们是无法构成一个排列的！因为：如果我们下一个放C，那么当我们放D 的时候，已经没有点向D有边了，那么最后留下的一定是D。而如果下一个放D，那么当我们到C 的时候同理。所以我们不能从B访问与它隔两条边或以上的点。

这样我们就可以通过本题了。

98.3 时空复杂度

时间复杂度：$O(Tn^4)$。空间复杂度：$O(n^2)$。

99 D8760 GCJ2014 Final E Allergy Testing

99.1 题目大意

有n种食物，对其中一种过敏。每次选一些试吃，A 天之后可以知道是否过敏，如果过敏还需$B-A$天。求最少多少天找出过敏的那一种食物。

99.2 题目解法

考虑$f[i]$表示用i天最多能在多少种食物中找出过敏的一种。那么有：$f[i]=f[i-A]+f[i-B]$，其中$f[i](i<A)=1$。不妨把i小于0 的$f[i]$计入到$f[i+B]$ 上。

假设答案为k，对于一个$f[i]$，考虑它被计入了答案多少次。显然我们有：$0\leqslant k-xA-yB<B$。化简后有：$k-B<xA+yB\leqslant k$。

由于我们每次将食物分半的策略可以达到$B\log n$的答案，而k 不会超过这个值，那么容易看出上式中$y\leqslant\log n$。考虑枚举y,我们容易算出x 的最大和最小值，不妨设为$xmin$ 和$xmax$，那么我们就要求：$\sum_{x=xmin}^{xmax}C_{x+y}^{x}$

进行一些组合推导后，上式即$C_{xmax+y+1}^{y+1}-C_{xmin+y}^{y+1}$。

最后，我们二分答案k，就可以求得答案了。

99.3 时空复杂度

时间复杂度：$O(T\log^3 n)$。空间复杂度：$O(1)$。

100 D9213 GCJ2014 Final F ARAM

100.1 题目大意

有一个游戏，每一盘我们从n个英雄中随机选择一个。每个英雄有一个胜率。

我们有一个货币叫RD，每一盘我们可以花费一个RD 来重新随机一个英雄。可以多次使用。

每玩一盘可以获得$\frac{1}{G}$个RD，但最多不能超过R个RD。一开始拥有R个RD。

我们一共玩10^{100}盘，求在最优策略下我们能达到的最高总胜率。

100.2 题目解法

首先，由于我们的局数非常大，可以认为游戏是永远进行下去的。

考虑二分答案。假设我们二分到的值为Q。我们希望求：是否存在一个策略答案可以达到Q。

我们设$A[i]$表示：我们当前有i个RD，在我们达到$i+\frac{1}{G}$个RD的时候，相比胜率为Q 的时候期望能够多胜利多少盘。对于$A[R]$，我们的定义是再次回到R个RD的时候，相比胜率为Q 的时候期望能够多胜利多少盘。

对于$0 \leqslant i < 1$，由于我们不能重新随机，因此$A[i] = \frac{1}{n}\sum_{j=1}^{n} p[j] - Q$。

对于$1 \leqslant i < R$，我们可以重新选择英雄。显然，如果我们对一个英雄的选择是重新随机，那么我们对所有胜率不如他的英雄都会重新随机。不妨设我们将胜率较小的k 个英雄选择重新随机。考虑重新随机之后我们还有$i-1$个RD，而我们需要达到$i+\frac{1}{G}$ 个RD，因此$A[i] = \frac{k}{n}\sum_{j=0}^{G} A[i-1+\frac{j}{G}] + \frac{1}{n}\sum_{j=k+1}^{n}(p[j]-Q)$。

对于$i = R$，与上一个情况类似，$A[i] = \frac{k}{n}\sum_{j=0}^{G-1} A[i-1+\frac{j}{G}] + \frac{1}{n}\sum_{j=k+1}^{n}(p[j]-Q)$。

如果$A[R]$大于等于0，那么就说明答案Q是可以达到的，否则说明答案Q是不可达到的。

100.3 时空复杂度

时间复杂度：$O(TnRG\log v)$。空间复杂度：$O(n+RG)$。