哪吒闹海

题解

不难发现，如果要消除所有的卡牌，必然要满足第i+1大的数和第i大的数之间，a[i]-a[i+1]=cs[a[i]]（a表示数字大小，cs表示某个大小的数出现的次数）。否则，如果cs[a[i]]<a[i]-a[i+1]，则将会需要修改a[i]-a[i+1]-cs[a[i]]的次数。

我们换一下思路。如果将差值转化为覆盖的话，那么a[i]显然是能覆盖从a[i]（包括a[i]）开始的cs[a[i]]个数的。那么我只需要知道，每次修改后，删除的那个数所覆盖的最前一个数的被覆盖次数是否是1，和增加的那个数所覆盖的最前一个数的前一个数的被覆盖次数是否是1。那么就可以O(1)修改了。加上预处理时的差分优化，总的复杂度是O(N+M)

重排题

题解

一个数是11的倍数的充要条件是奇数位的和跟偶数位的和模11同余。所以，我们先对所有数字做一个背包，用f[i][j]表示选出i个数，并且这i个数的和模11余数为j的方案数。然后用逐位确定的方法：从高位到低位依次确定每一位最大可以是多少。对于每一位，先尝试能否填9，再尝试能否填8，再尝试能否填7……，以此类推。那么如何快速知道能否填这个数呢？一种简单的方法是对去掉

这个数后剩下的数重新做一次背包，但是这样会很慢。那怎么办呢？我们只需要用反推的方法，把这个数从背包的可选数中去掉即可。

比方说，增加一个数的时候，代码如下： for (int i=mx; i>=0; i--) for(int j=0; j<=10; j++) (f[i+1][(j+x)%11]+=f[i][j])%=P; 那么，去掉一个数的代码就是这样的： for (int i=0; i<=mx; i++) for(int j=0; j<=10; j++) (f[i+1][(j+x)%11]+=P-f[i][j])%=P;

邮箱

题意

题意就不过多说明了，总体就是可以理解为有两块屏幕，一块邮件夹的，一块邮件的，要把邮件放到对应文件夹中。

分析

首先我们可以发现这个邮件的屏幕头出尾进，但中间又有一些删除，所以可以用的 list（不用 queue，有删除操作）。还有屏幕上方有可能还有些未归类的邮件，越在后的越早落下，所以可以使用 stack。

实现

主要是要会用 list 和 stack 思想是模拟+一点点贪心。贪心体现在如果一个文件夹并没有归这类的文件了，就下移文件夹屏幕。因为文件夹一旦弹出屏幕，就回不来了。还有下移时先判断是否能归，不能归再压入 list。

总体分三步走：

1. 能归就归。
2. 不能归下移。
3. 移到底，栈中文件下落。

脱单计划

题解

直接两两连边显然不行

考虑这样一个转化

$|x1-x2|=max(x1-x2,x2-x1)$

$|x1-x2|+|y1-y2|=max(x1-x2+y1-y2,x2-x1+y1-y2,x1-x2+y2-y1,x2-x1+y2-y1)$

我们额外建4个中转点表示上面的四种情况，红球和蓝球通过中转点连边，这样边数降到了O(N)

边权就按照上面四种情况的符号连，容量为1，跑最大费用最大流。

由于最大费用最大流的性质，保证了每个匹配都是最大的，因此恰好就是曼哈顿距离取了绝对值符号后的结果。

时间复杂度O(maxflow(N))