IOI2015中国国家集训队第一次作业

浙江省镇海中学 张煜皓

Contents

1 Volume.1

1.1 235E. Number Challenge

Description

计算$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}d[i*j*k])$的值。其中$1 \le a,b,c \le 2000$，$d[n]$表示n的约数个数。

Analysis

首先我们可以线性筛出$d[n]$。接着我们明确几个定义：$p[n]$代表能整除n的最小质数；$cnt[n]$代表$p[n]$在n的质因数分解中的指数项；令$M[n] = p[n]^{cnt[n]}$；再令$Rm[n] = n/M[n]$；线性筛法的具体做法不再赘述，以上的定义都能线性求出。

如果我们枚举$i*j$，不妨设$X = i*j$。我们设$F[X][c] = \sum_{k=1}^{c}d[X*k]$。如果能够利用筛出的信息快速求出$F[X][c]$，那么这道题目就迎刃而解了。

首先是边界情况：

1、$F[X][0] = 0$

2、$F[X][1] = d[X]$

3、$F[1][c] = \sum_{k=1}^{c}d[k]$

接着，我们来用$M[X]$不断减小X的大小。如果$gcd(\prod_{k=1}^{c}, p[X]) = 1$，那么很简单$F[X][c] = F[Rm[X]][c]*(cnt[X]+1)$。如果$gcd(\prod_{k=1}^{c}, p[X]) \neq 1$，不能像上面那样转移的原因是因为$1\ldots c$之中有数能够被$p[X]$整除。

接下来，我们对转移进行“矫正”。对于那些在$1\ldots c$之中的能被$p[X]$整除的数，他们对答案的贡献是$\sum_{i=1}^{\lfloor c/p[X]\rfloor}d[i*p[X]*X]$，令$y_i$为$i*p[X]$的质因数分解中$p[X]$的幂次，有$A_i = d[Rm[X]*\frac{i*p[X]}{p[X]^{y_i}}]$。那么上述对答案的贡献可以化为$\sum_{i=1}^{\lfloor c/p[X]\rfloor}(cnt[X]+y_i+1)*A_i$。如果我们用$F[X][c] = F[Rm[X]][c]*(cnt[X]+1)$进行转移，则该部分答案被错误地计算为$\sum_{i=1}^{\lfloor c/p[X]\rfloor}(cnt[X]+1)*(y_i+1)*A_i$，两者相差$\sum_{i=1}^{\lfloor c/p[X]\rfloor}cnt[X]*y_i*A_i$，也就是$cnt[X]*F[Rm[X]][c/p[X]]$。那么有：

4、$F[X][c] = F[Rm[X]][c]*(cnt[X]+1) - F[Rm[X]][c/p[X]]*cnt[X]$

其中F[][]数组可以记忆化搜索得出。由于数组较大，X是$O(n^2)$级别，c的取值是$\lfloor c/i\rfloor$，所以级别为$O(n^{0.5})$。但是实际占用的位置大小不多，我们可以进行hash，然后放到数组中。

时间复杂度$O(n^{2.5})$

空间复杂度$O(n^{2.5})$（hash之后远远不达）

1.2 277D. Google Code Jam

Description

有n道GCJ的题目，比赛持续时间t。每道题目有easy和hard两部分，各有得分和消耗时间记为$scoreSmall_i, scoreLarge_i, timeSmall_i, timeLarge_i$，其中easy部分写出来就可以AC，hard部分还有$probFail_i$的概率FST，并且只能先做某题的easy部分，后做hard。在比赛中，每道题目的每个部分只能提交一次。求一个最优的做题策略，使得最大化期望得分，同时最小化期望罚时。GCJ 的罚时指的是最后一次提交正确解的时间。

Analysis

我们先简化问题，如果只最大化期望得分，那么我们可以将每道题目看成一个物品，放入背包中有三种情况：1、不放入；2、放入easy部分；3、两者都放入。Dp的转移非常简单，不不再赘述，并且这也不是本题所求。然而我们知道了最大的期望得分，能不能同时记录最小期望罚时呢？我们仔细考虑罚时的构成：如果是存在最优解中的easy部分，那么肯定在所有hard做之前就已经完成，否则期望罚时会更高；但是hard部分还需要调整顺序，如果在原来Dp的基础上记录hard的顺序，那么复杂度就会达到非多项式级别。

我们注意到，在原来放置物品的Dp中，物品的顺序是不会影响到Dp的结果的。我们是不是可以先排列hard的顺序，然后再进行Dp呢？考虑如下问题，如果第i个问题和第j个问题的hard 部分都要被选入，那么如果i在前j在后比j在前i在后更优，考虑四种情况：1、i pass,j pass；i pass,j fail；3、i fail,j pass；4、i fail,j fail。有不等式($time$ for $timeLarge$,$fail$ for $profail$)：
$(time_i + time_j) * (1 - fail_i) * (1 - fail_j) + time_i * (1 - fail_i) * fail_j + (time_i + time_j) * fail_i * (1 - fail_j) + 0 \leq (time_j + time_i) * (1 - fail_j) * (1 - fail_i) + time_j * (1 - fail_j) * fail_i + (time_j + time_i) * fail_j * (1 - fail_i) + 0$。
化简得：
$time_i * fail_i * (1 - fail_j) \leq time_j * fail_j * (1 - fail_i)$

现在我们有了一个比较函数，只要将这些题目排序即可，然后进行Dp。我们令$f[i][j]$表示考虑到第i个问题，现在用去时间j的(最大期望得分Es，在此情况下的最小期望罚时Ep)。 $f[i-1][j]$可以去更新：

$(Es, Ep) => f[i][j]$;

$(Es + scoreSmall_i, Ep + timeSmall_i) => f[i][j + timeSmall_i]$

$(Es + scoreSmall_i + scoreLarge_i * (1 - proFail_i), (Ep + timeSmall_i) * proFail_i + (j + timeSmall_i + timeLarge_i) * (1 - proFail_i)) => f[i][j + timeSmall_i + timeSmall_j]$

但是这道题目还有一个小问题，就是期望得分最大可达10^{12}级别，还需要保留6位小数，double的精度可能不够，导致期望时间出现错误。解决的办法就是用long long存储$*10^6$之后的期望得分。

时间复杂度$O(nt)$

空间复杂度$O(nt)$

1.3 USACO Open07.Connect

Description

给定一个R*C($1 \leq R \leq 2, 1 \leq C \leq 15000$)的网格图，和m($1 \leq m \leq 50000$)的询问和操作。每次删去或者加入一条边（仅相邻的点之间存在边），询问两点之间的“连通性”，这里“连通性”的定义是是否有一条路径可以从$(r1, c1)$到达$(r2, c2)$，并且经过的路径不能超出c1列和c2列之间的范围，即路径所经过的所有点的列编号cx满足$min(c1, c2) \leq cx \leq max(c1, c2)$。强制在线。

Analysis

如果是一般图的一般连通性，我们有lct，分治+并查集等离线算法。但是这道题目强制在线，并且不是一般图也不是一般的连通性。我们发现这个“连通性”事实上是弱化的，并且这个网格图的行数最多是2，这就给我们在线解决问题带来了可能。

我们首先考虑简单的情况：$R = 1$的时候。我们要解决的问题就是询问l到r的线段是否联通。这个可以用线段树方便地解决，我们记录这个节点所表示的线段是否联通，显然可以轻松地合并：如果左孩子和右孩子都联通，那么这个节点就是联通的。

事实上，线段树的做法可以推广到$R = 2$的情况。对于每一个节点所表示的线段，我们用$L_{x,0}$和$L_{x,1}$分别表示x边的上面、下面能够到达右边的哪里（用二进制表示）。

设该节点选段的左右中间端点分别为l, r, mid，我们这里的节点信息是不考虑l和r处“竖边”（即(1,x)到(2,x)的边）的影响的。考虑合并，我们知道了l能够到达mid何处，mid能够到达r 的何处。

考虑mid处竖边的情况，如果有竖边，那么l到mid的信息就会有所改变：$L_{l,0}$如果能够到达右边，那么$L_{l,0} = 3$，即上下都能到达，$L_{l,1}$的处理同理。mid到r的信息也有所改变，$L_{mid,0}$和$L_{mid,1}$共用了彼此的信息，$L_{mid,0} = L_{mid,1} = L_{mid,0}|L_{mid,1}$。然后进行常规的转移即可。

在最后统计答案时，再考虑最左边和最右边的竖边即可。

时间复杂度$O(m \log C)$

空间复杂度$O(RC)$

1.4 351D. Jeff and Removing Periods

Description

一个序列$b_1, b_2 \dots b_m$的消除方式：1、选择三个整数$v, t, k (1 \le v, t \le m; 0 \le k; v + tk \le m)$使得$b_v = b_{v+t}; b_{v+t} = b_{v+2t} \dots b_{v+(k-1)t} = b_{v+kt}$;，并且从原序列中删除;2、将剩下的$n - k - 1$个数按你的意愿重新排列，重复1直至删完为止。

再给你一个长度为n序列a和m组询问(l_i, r_i)。对于每组询问，输出序列$a[l_i], a[l_i + 1] \dots a[r_i]$的最少消除次数。

Analysis

可以发现，当进行第1次消除之后，可以将序列从小到大排序，然后当前的最小消除次数是当前的不同数字个数。如果将消除前的序列中不同数字个数记为diff。那么如果第一次消除能够将一个数字x全部消除（即x的出现位置成等差数列），那么最小消除次数就是diff；反之就是diff+1。

对于diff，我们可以用莫队算法求出。判断一个序列中是否存在一个数x，它的位置成等差数列，也可以用莫队算法求出。我们用$cnt[x]$表示x的出现次数，用双端队列$q[x]$ 记录x 的出现位置，再令$dlt[x] = \sum_{i=1}^{k-2} q[x][i] - q[x][i+1]! = q[x][i+1] - q[x][i+2]$。当加入一个数之前，如果$cnt[x] = 0$，则diff+=1；删除一个数之后，如果$cnt[x] = 0$，则diff-=1。如果当前时刻，存在一个$cnt[x] \neq 0$并且$dlt[x] = 0$则存在等差数列，只要在加入删除的时候维护$dlt[x]$即可。

时间复杂度：$O(n\sqrt{n})$

空间复杂度：$O(n)$

1.5 321D. Ciel and Flipboard

Description

给你一个n*n的矩阵$a[][]$，n为奇数。再令$x = (n+1)/2$。每次可以将一个$x * x$的子矩阵中的数字乘-1。你要最大化矩阵中数的和。

Analysis

如果我们把子矩阵乘-1，记为在$n * n$的$Xor[][]$矩阵中的子矩阵异或1。然后就有一个有趣的性质：$1 \le i, j \le x, Xor[i][j] \oplus Xor[i][x] \oplus Xor[i][j + x] = 0$，记为性质1，同理$1 \le i, j \le x, Xor[i][j] \oplus Xor[x][j] \oplus Xor[i+x][j] = 0$，记为性质2。

我们发现只要确定$Xor[][]$左上角的$x * x$的数字，就可以没有冲突地确定$Xor[][]$的整个矩阵了。并且这样的$2^{(x^x)}$个解是涵盖了所有的翻转方式的。

现在我们枚举$Xor[1][x], Xor[2][x] \ldots Xor[x][x]$的状态，由性质2可以得出第x列的状态。假设b[][]是经过变换后的a矩阵。观察第j$(1 \leq j < x)$列，由性质1 可以得出$1 \leq i < x, b[i][j]$ 和$b[i][j+x]$是同号还是异号。因为$1 \leq i < x$的状态不相互影响的，所以我们再枚举$Xor[x][j]$的状态，从性质2知，可以由$1 \leq i < x$推出$x < i \leq n$，然后贪心确定每一列$(1 \leq j < x$，列于列之间也相互独立）即可。

时间复杂度：$O(n^2 2^x)$

空间复杂度：$O(n^2)$

1.6 305E. Playing with String

Description

给定一个长度为n的字符串s。初始集合中只有s。两人轮流操作，没人每次可以从集合中拿出一个t，满足$\exists 1 < i < |t|, t[i-1] = t[i+1]$，并且选择一个满足条件的i，将t 分成$t[1 \ldots i-1], t[i \ldots i], t[i+1 \ldots |t|]$三个部分，并放入集合中。第一次不能操作的那个人输。

输出是先手获胜还是后手，如果是先手，那么输出第一次选择可的最小的i。

Analysis

可以发现，如果我们将s中连续的极长的形如t$\forall 1 < i < |t|, t[i-1] = t[i+1]$的串提取出来。那么两个串$t_i$和$t_j$只会在开头一个或者结尾一个字符处出现粘连的情况。也就是说两个串t_i和t_j的游戏是相互独立的。并且游戏的SG值只和t串的长度有关。

于是我们可以事先计算长度为i的sg值$sg[i]$。$sg[1] = 0, sg[2] = 0, sg[i] = mex(sg[j-1] \oplus sg[i-j]), 1 < j < i$。然后将各个独立的游戏异或起来就是最后的sg值。如果是先手获胜的话，我们枚举第一手的分裂位置i的位置，并判断分裂后的状态的sg值是否为0（是否当前先手必败）即可。

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

1.7 USACO DEC10.Threatening Letter

Description

给定一个串S和串T，你要选出尽可能少的S的子串，不重叠不遗漏地覆盖T串。

Analysis

可以发现，我们可以贪心地选出一个尽可能长S的子串匹配T的前缀，然后将T的前缀“删除”，再往下匹配，直到结束为止。可以证明，如果有一步没有尽可能地匹配T的前缀，那么这个解肯定是不优的。

那么只要将S和T一起放入后缀数组中，假设当前匹配到T的i位置，找出离rk[i]最近的两个rk[j]（前后各一个）,满足j存在于S串中，然后将i指针进一步移动到i+lcp(rk[i],rk[j])的地方。这样i 的移动次数就是要求的答案了。

时间复杂度：$O((n+m)\log(n+m))$

空间复杂度：$O((n+m)\log(n+m))$

1.8 333C. Lucky Tickets

Description

给定一个k,m。求出m个8位数字串，使得每个数字串通过添加"+", "-", "*","(",")"，之后能够变成k。

Analysis

打表即可发现，所有4位数字串能够组成的数字大约是600000个。我们存下所有产生数字的方案，然后暴力枚举前后两段的组合情况即可。计算答案的时间复杂度是$O(k^2+m)$，不过远远达不到上界。事实上只要前后两段进行加减就可以构造出所有情况了。那么我们枚举前面一段的方案，然后就可以计算后面的方案是什么，这样计算答案的时间复杂度就减小为$O(k+m)$了。

时间复杂度：$O(\sum_{i=0}^{9999} cnt[i])$$cnt[i]$表示形如i的四位数字能构成多少不同的数字。

空间复杂度：$O(\sum_{i=0}^{9999} cnt[i])$

1.9 263E. Rhombus

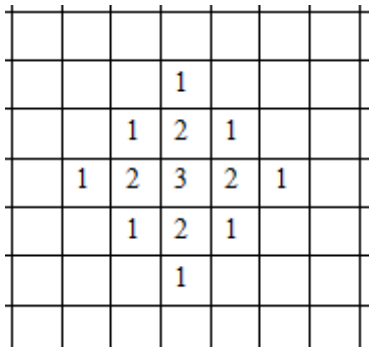
Description

给定一个n*m的矩阵a，和一个正整数k。

定义$k \le x \le n-k+1, \le y \le m-k+1, f[x][y] = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{i,j} * max(0, k-|i-x|-|j-y|); maxval = max(f[x][y])$。求任意一对$(a,b)$，使得$f[a][b] = maxval$。

Analysis

稍加观察即可发现，事实上$f[x][y]$是以x为中心的一个菱形的和。如下图：



我们暴力枚举x和y，然后计算$f[x][y]$。事实上利用前缀和就可以O(1)计算$f[x][y]$的值了。

定义

$sr[i][j] = \sum_{l=1}^{j} a[i][j]$

那么有$sr[i][j] = sr[i][j-1] + a[i][j]$

$s1[i][j] = \sum_{l=0} a[i-l][j-l]$

那么有$s1[i][j] = s1[i-1][j-1] + a[i][j]$

$s2[i][j] = \sum_{l=0} a[i-l][j+l]$

那么有$s2[i][j] = s2[i-1][j+1] + a[i][j]$

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

那么有$tl[i][j] = tl[i][j-1] + a[i][j] * k - sr[i][j-1] + sr[i][j-k-1]$

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

那么有$tr[i][j] = tr[i][j+1] + a[i][j] * k + sr[i][j] - sr[i][j+k]$

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

那么有$th[i][j] = th[i-1][j] + sr[i][j+k-1] - sr[i][j-k] - s1[i-1][j+k-1] - s2[i-1][j-k+1] + s1[i-k-1][j-1] + s2[i-k-1][j+1] + a[i-k][j]$

$Up[i][j] = \sum_{l=0}^{k-1} \sum_{p=j-(k-l)+1}^{j+(k-l)-1} a[i-l][p] * (k-l-|p-j|)$即菱形上半部分的和

那么有$Up[i][j] = Up[i-1][j] + tl[i][j] + tr[i][j] - a[i][j] * k - th[i-1][j]$

$Down[i][j] = \sum_{l=0}^{k-1} \sum_{p=j-(k-l)+1}^{j+(k-l)-1} a[i+l][p] * (k-l-|p-j|)$即菱形下半部分的和

和Up的转移同理。

$f[i][j] = Up[i][j] + Down[i][j] - tl[i][j] - tr[i][j] + a[i][j] * k$

转移的时候要注意边界情况。

时间复杂度：$O(nm)$

空间复杂度：$O(nm)$

2 Volume.2

2.1 293B. Distinct Paths

Description

给一个n*m的棋盘，你要给每个格子染上k种颜色中的一种（某个格子可能事先已经有颜色），使得从(1,1)到(n,m)的任何一条路径上，没有两个格子同色。求方案总数。

Analysis

事实上，如果有$n+m-1>K$那么一定是无解的。所以n和m的大小就和k差不多级别。另外，我们发现原题的表述可以转换成，以(i,j)为右下角的$i*j$大小的子矩形中，不能有和(i,j)同色的格子。

我们考虑是否可以暴力枚举。但是这样符合条件的情况还是很多。但是我们发现在众多的状态中，有很多状态是“同构”的。即对方案A中的颜色建立一个一一映射，如果能成为方案B，那么就称这两个方案“同构”。事实上这样的本质不同的方案是很少的。我们规定对于每一条路径，颜色小的总是比颜色大的率先出现，这样就可以求出本质不同的方案了。如果存在本来就被染色的格子，我们只要最后判断一下是否满足就可以了。对于每个合法的方案，乘上分配颜色的方案并统计到最后的答案中即可。

时间复杂度：$O(k^{k^2})$（远远达不到）

空间复杂度：$O(nm)$

2.2 325D. Reclamation

Description

有一个r*c的棋盘，这个棋盘的最左边和最右边是联通的。给出n个有顺序操作(x_i,y_i)。依次在(x_i,y_i)的格子中放上障碍，使得任意时刻从最上面可以有一条路径到达最下面。如果在(x_i,y_i)的格子中放上障碍后不满足上述情况，那么跳过这个操作。求最后被放上了多少个障碍格子。

Analysis

我们可以用并查集来解决这个问题。首先我们将棋盘复制一遍放到右边，这样我们就可以解决最左边和最右边的联通情况。放上一个障碍之后，它会和自己周围3*3区域内的8个格子中的障碍形成一个大的障碍，我们用并查集来模拟这个合并的过程。

我们只要事先判断(x_i, y_i)周围的障碍和$(x_i, y_i + c)$的周围的障碍是否联通就可以判断是否存在一条从上到下的路径了。还是要注意第一列的左边应该是第2c列，第2c列的右边是第1列。

时间复杂度：$O(n\alpha(n))$

空间复杂度：$O(nm)$

2.3 325C. Monsters and Diamonds

Description

有n个不同的妖怪和m条变换规则。一个妖怪可以有若干条变换规则（至少1条），每次可以选择他进行哪个变换。一次变换可以将一个妖怪变成若干个妖怪（可以没有）和若干个钻石（至少1个）。你要对于各个妖怪计算，在最后妖怪全部消失的时候最多和最少能得到多少钻石。

每个妖怪有三种情况：1、无论如何选取变换方式，都不会出现没有妖怪的情况，输出"-1,-1"；2、如果存在最小的情况，但是可以得到任意多的宝石，那么输出"$min(Min_i, 314000000), -2$"；3、如果最大最小都存在，那么输出"$min(Min_i, 314000000), min(Max_i, 314000000)$"

Analysis

对于求每个妖怪最小的情况，我们可以用类似于dijkstra的算法解决。先将用不剩妖怪的策略去更新每个妖怪能够得到的最小钻石数。接着取出钻石最少的那个妖怪u，去更新那些能够产生u 的规则。接着重复上述过程，直到不存在妖怪，或者最少的钻石数为inf为止。那么现在为inf的妖怪就是"-1,-1"的状态了，我们称之为"死循环"。

求最大的情况，我们采用记忆化搜索。如果该妖怪是"死循环状态"，那么直接退出。如果要计算这个妖怪，那么我们先去除所有能够变换出"死循环"妖怪的规则。接着，如果有一个规则能够到达产生任意钻石（简称"任意"）的妖怪，那么这个妖怪就是"任意"妖怪。如果在记忆化过程中访问了还在栈中的x妖怪，那么这个妖怪也是"任意"妖怪。对于每个妖怪记忆化搜索即可。

时间复杂度：$O(m \log m)$

空间复杂度：$O(m)$

2.4 325E. The Red Button

Description

有n个点，标号为$0, 1 \ldots n-1$。从编号为i的点可以到达2i mod n和(2i+1) mod n的点。你要从0号点开始，找一条汉密尔顿回路。

Analysis

首先n为奇数时肯定无解。简略证明：如果要进入0号点，我们只能从(n-1)/2号点进入；如果要进入n-1号点，我们也只能从(n-1)/2号点进入。出现矛盾，所以命题的证。

当n为偶数的时候，观察i($i < n/2$)号点，可以进入2i和2i+1的点；再观察i+n/2号点，也可以进入2i和2i+1的点。我们先将i连向2i,i+n/2连向2i+1，这样就形成了若干个简单环。如果i 和i+n/2号点不在同一个环中，那么只要将i重新连向2i+1，i+n/2连向2i，这样调整直至并成一个简单环即可。可以用并查集来解决判断是否在一个环内的过程。

时间复杂度：$O(n\alpha(n))$

空间复杂度：$O(n)$

2.5 241E. Flights

Description

给出n个点，m条有向边，保证1到n号点联通，并且保证不出现环和重边。初始每条边的长度是1，你可以将某些边的长度变为2，使得从1到n的所有路径的长度一致。

Analysis

如果一个点x可以在1号点到n号点的路径上，我们称这个点是"可经过"的。反证法可以证得，题目中的条件的充要条件就是：对于所有可经过点x，从1号点到x号点的路径长度相等。

我们用$dis[u]$表示1号点到u的距离。考虑所有可经过的点$(u, v, (u, v) \in E)$，可列出方程$(dis[v] - dis[u] \leq 2, dis[v] - dis[u] \geq 1)$。对于这些不等式建立差分约束系统，然后用spfa判断是否成负环的情况（无解），如果有解，那么边长也肯定为整数，只要输出结果就行了。

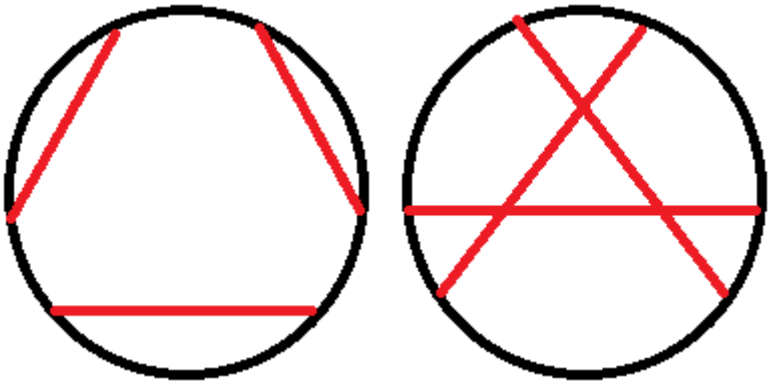
时间复杂度：$O(n * m)$

空间复杂度：$O(m)$

2.6 297E. Mystic Carvings

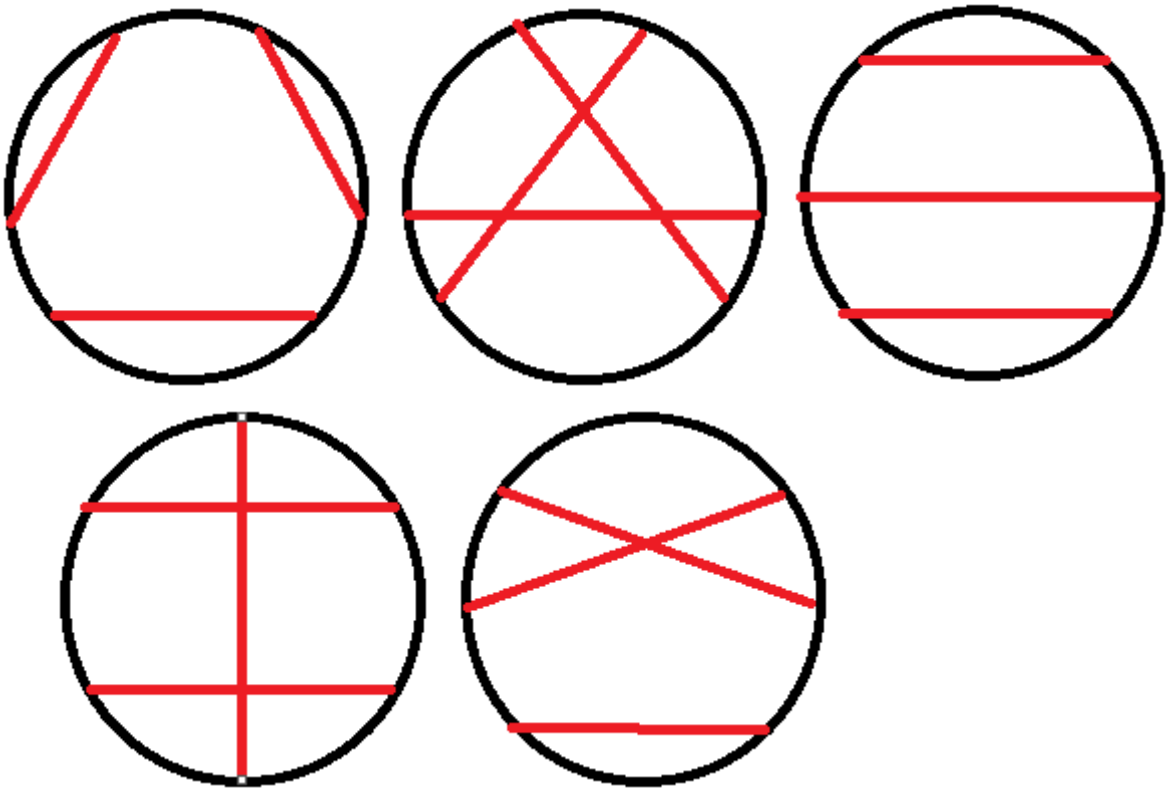
Description

2n个点按顺序分布在圆上，有n条线连接2n个点，保证每个点只有一条边与其相连，且一条线只连接2个点。求下图所示的线三元组个数：



Analysis

仔细分析我们发现总共有5种情况，三元组个数依次为s1,s2,s3,s4,s5：



我们的目标是求s1+s2。并且对于第i条线，我们可以求出$I[i]$，表示和i相交的有多少条线。并且我们把1到2n的点放在一条直线上，$In[i]$表示在第i条线内，且不和i相交的线有几条，$Out[i]$表示在第i条线外不和第i条线相交的条数。$I[i]$可以由树状数组方便地求出。$In[i]$和$Out[i]$也可以求出来了。

我们发现：1、$sum1 = s3 = \sum_i In[i] * Out[i]$；2、$sum2 = 3 * s1 + 2 * s3 + s5 = \sum_i (C^2_{In[i]} + C^2_{Out[i]})$；3、$sum3 = 2 * (s4 + s5) = \sum_i I[i] * (In[i] + Out[i])$；4、$sum4 = 3 * s3 + s4 = \sum_i C^2_{I[i]}$。

那么$s1 + s2 = (sum2 + sum4 - sum1 * 2 - sum3/2)/3$

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

2.7 241F. Race

Description

给定一个$n * m$的网格图，每个点有一个经过的时间或者是不能通行的障碍。你要按照一系列指令从一个点走到另一个点，并且保证相邻的两个指令点在同一行或者同一列上联通。求第k时刻你所处的位置。

Analysis

由于$n|s|$较小，直接模拟即可。

时间复杂度：$O(n|s|)$

空间复杂度：$O(nm + |s|)$

2.8 260E. Dividing Kingdom

Description

平面上有n个整点，你要用两条平行于x轴和两条平行于y轴的直线（直线不能穿过这n个点）将平面分成9份，将n个整点分成9个部分，每个部分的和为题中所给的值（题中没有规定顺序）。输出那四条直线。

Analysis

首先9!枚举每一块的大小。可以用前缀和在$\log n$的时间内，在前两行（列）满足条件的情况下，求出那四条直线。我们现在要解决的问题是求$x_i \leq X, y_i \leq Y$（X,Y为给定）的点有几个。这个问题可以通过可持久化线段树解决。

时间复杂度：$O((9! + n) \log n)$

空间复杂度：$O(n \log n)$

2.9 317C. Balance

Description

有n个容器和m条连接容器的双向通道。每个容器都有上限V，并且初始时刻水量为a_i，目标水量为b_i。你每步可以从一个容器向另一个容器输送一定量的水，前提是每个容器不超过它的上限。求步数小于$2*n^2$的方案或者判断无解。

Analysis

首先，如果有一个联通块的初始水量和不等于目标水量和，那么肯定无解。

反之一定有解，我们考虑下述的构造。如果当前水量等于目标水量，那么改点称为ready；如果当前大于目标，则为more；如果是小于，那么为less。我们找到一个more点s和一个less 点t。我们找到最小的水量d($d = min(a[s]-b[s], b[t]-a[t])$)使得其中一个满足条件。定义过程flow(s,t,d)表示从s向t推送d的流量：1、从t'即t的前一个点，向t推送d的流量，如果不够就尽可能推送，2、调用flow(s,t',d)，3、从t'向t推送剩下的流量。

简略证明，我们先考虑没有V上限的情况，那么每找到一对点，运行一个flow过程之后，会有一个不是ready的点变成ready，并且除了s和t之外的点流量平衡，那么ready的数量每次至少增加1。那么因为每次flow的步数最多为2n，最多调用在主程序调用flow过程n次，所以步数是不会出现问题的。如果有了V的限制，我们考虑每次flow的过程，考虑数学归纳法，如果flow(s,t',d)的过程能够成功，考虑flow(s,t,d)。1过程要不推送d的流量，那么t'至少有d的剩余空间，所以调用flow(s,t',d)之后没有问题；要不推不到d流量，那么t'必定当前为空，而($d \leq V$)所以调用flow(s,t',d)之后也没有问题。所以可以证明这样的构造是正确的。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

3 Volume.3

3.1 335D. Rectangles and Square

Description

平面上有n个不相交的矩形。要求一个集合使得矩形的并为一个正方形。输出解或者判断无解。

Analysis

因为我们要求的正方形必然满足，在正方形内不能有空点，并且边上的矩形不能够超出正方形。

我们只要枚举每个矩形成为右下角的情况。然后从小到大枚举正方形边长，如果发现有空点或者最下（右）边一行出现有矩形超出正方形的范围就break，接下来我们要做到的就是判断最上（左）边是否满足条件。我们将所有矩形的最左边染上红色，用$Le[i][j]$表示在坐标点(i,j)开始最多能向上延伸多少红色距离。同理将矩形最上边染上蓝色，定义$Up[i][j]$表示在坐标点(i,j)开始最多能向左延伸多少蓝色距离。这两个数组可以简单dp求出。每次只要判断左下角的$Le[][]$和右上角的$Up[][]$是不是大于等于枚举的边长即可。

时间复杂度：$O(nX)$（X为坐标轴范围）

空间复杂度：$O(X^2)$

3.2 USACO Dec06.Cow Patterns

Description

有n个数字a_i，你要选出连续的k个，使得离散化之后是给定的k个数b_i。求所有的方案。

Analysis

可以用kmp来解决这个问题。我们用$occur[x]$表示x在a数组中第一次出现的位置，用$low[x]$表示比x小但最大的东西第一次出现的位置，用$high[x]$表示比x大的最小的第一次出现的位置。只要对于所有$1 \le i \le k$满足1、$a[i] = a[occur[b[i]]]$;2、$a[i] > a[low[b[i]]]$;3、$a[i] < a[high[b[i]]]$。

我们发现1式确保了a和b之间的一一映射，而2、3两式确定了他们之间的大小关系。所以这样用广义的kmp就可以解决这个问题。

时间复杂度：$O(n+k)$

空间复杂度：$O(n)$

3.3 USACO Dec07.Best Cow Line Gold

Description

给定一个长度为n的字符串S，你每次可以将这个字符串的头或者尾拿出，放到新串的结尾，使得新串的字典序最小。

Analysis

考虑暴力，我们发现如果用i表示左指针，j表示右指针。那么如果以i为开始的串小于以j为开始的（倒过来的）串，那么我们将第i个放到新串，i指针右移；反之，第j个将被加入，j 指针左移。但是这样的复杂度是$O(n^2)$的。我们发现实际上就是比较两个子串的大小，这个可以用后缀数组解决。我们将S正着和倒着的串都放入后缀数组，求出rank数组之后，利用上述的暴力过程，将暴力比较换成rank[i]和rank[j]比较即可。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

3.4 306C. White, Black and White Again

Description

给定n天，和W个不同的白点，B个不同的黑点。你要给每个天放置若干个白点和黑点（不能不放），满足每天只有白点或者只有黑点。将放了白点的天称为“白天”，反之称为“黑天”，使得$1 \ldots i, i+1 \ldots j-1, j \ldots n, (1 \leq i, i+1 \leq j-1, j \leq n)$分别都是“白天”，“黑天”，“白天”。求所有的方案数。

Analysis

我们先将所有白点（黑点）看成没有区别的，如果这样求出来的答案是ans'，那么正确答案$ans = ans' * W!B!$。

我们枚举“白天”的天数i，黑天就是$j = n - i$。那么我们只要求出W个白点分配给i个“白天”的方案数$s1$，B个黑点分配给j个“黑天”的方案数$s2$，那么$s1 * s2 * (i-1)$就是分成三段的答案Si，$ans' = \sum_{i=2}^{n-1} Si$。$s1$和$s2$可以由挡板法求出，即$s1 = C_{W-1}^{i-1}, s2 = C_{B-1}^{j-1}$。那么我们只要用杨辉三角预处理组合数即可。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

3.5 GCJ Final09 A.Year of More Code Jam

Description

有n天，m场系列赛，每场系列赛有T_i场比赛，分别在系列赛开始的第$d_{i,j}$（$1 \leq j \leq T$）天举行。每场系列赛在每天举行的概率是一样的，对于每一种举办方式，定义$sum = \sum_{i=1}^{n} (X[i])^2$，其中$X[i]$是在第i天比赛的场数。你要求sum的期望。

Analysis

不难发现$E(sum) = \sum_{i=1}^{n} E(X[i])$。而$E(X[i])$都是独立的，所以可以独立计算，并且$E(X[i]) = E((\sum_{j=1}^{m} Y[j][i])^2) = E(\sum_{j=1}^{m} Y[j][i]^2 + 2 * \sum_{j=1}^{m} \sum_{k=j+1}^{m} Y[j][i] * Y[k][i])$，其中$Y[j][i]$表示第j场系列赛在第i天有没有比赛。可以发现$Y[][]$非0即1，那么上述式子可以写成$E(\sum_{j=1}^{m} Y[j][i] + 2 * \sum_{j=1}^{m} \sum_{k=j+1}^{m} Y[j][i] * Y[k][i])$，不难发现这个式子可以在$O(m^2T^2)$的时间内求出。

时间复杂度：$O(m^2T^2)$

空间复杂度：$O(mT)$

3.6 339E. Three Swaps

Description

给出一个$1 \ldots n$的排列，已知这个排列是将序列$1 \ldots n$选出不超过三个区间依次翻转之后的结果，求依次翻转了那些区间，保证有解。

Analysis

$O(n^6)$暴力搜索即可。我们可以倒着考虑这个问题，倒着枚举翻转的区间，只要最后成为序列$1 \ldots n$就可以了。我们考虑剪枝，我们将相邻的数只相差1或者-1的称为一段，那么x次翻转最多将$2 * x + 1$段减少为1段，最后目标就是1段。然后我们发现如果某次翻转增加了段数，那么肯定是不优的。加上这些可行性剪枝即可通过本题。

时间复杂度：$O(n^6)$（远远达不到）

空间复杂度：$O(n)$

3.7 286E. Ladies' Shop

Description

给出n个背包，每个背包有重量a_i，每个背包只能装严格等于a_i的。你要选出一个物品集合，使得从集合中任意选出若干个物品（每种物品有无限多个）都能放入背包中（前提是物品的总重不超过m），并且每个背包都被用到。最小化集合中数字个数，或判断无解。

Analysis

我们从最终的状态出发，发现最后的背包组合起来也一定能被放到背包里面。事实上，如果每两个背包组合起来能被放在背包中那么就可以证明是符合条件的，这个通过数学归纳法可以证明。

考虑下面构造：将背包两两组合起来的集合称为U，将在U且不是n个背包重量的数放在答案集合中。我们发现这个集合是极小的，并且它是必要的。

所以只要用fft将背包两两组合起来即可。

时间复杂度：$O(m \log m)$

空间复杂度：$O(m)$

3.8 360D. Levko and Sets

Description

给定n个数$a_1, a_2...a_n$，m个数$b_1, b_2...b_m$。定义集合U_i，有$(a_i^{\sum_{j=1}^{m} b_j * c_j}) mod P \in U_i$，其中$c_j$为任意自然数，$P$为素数。求这n个集合的并的大小。

Analysis

因为P为素数，那么肯定能找到一个P的原根g，满足$g^{r_i} = a_i$。那么就变成了$g^{r_i * \sum_{j=1}^{m} b_j * c_j}$。因为费马定理，有$g^{(r_i * \sum_{j=1}^{m} b_j * c_j) mod (P-1)}$。那么$\sum_{j=1}^{m} b_j * c_j$，就可以等同于$k * t$，$k$为自然数，$t = gcd(b_1, b_2...b_m, P-1)$。

如果我们知道了r_i，那么U_i就是$0, T_i, 2T_i...P-1$，其中$T_i = gcd(t * r_i, P-1)$。那么我们就只要求出在$0..P-1$之中有几个数可以至少被其中一个T_i整除。这个问题可以容斥解决，$f[i]$表示能被T_i整除，但是不能被T_i的倍数整除的有几个。$f[i] = (P-1)/T_i - \sum_{T_j mod T_i = 0} f[j]$。

现在我们考虑怎么求r_i，如果我们找到一个最小的d_i，使得$a_i^{d_i} \equiv 1$，那么因为$a_i^{P-1} \equiv 1$，所以$r_i = (P-1)/d_i$。

时间复杂度：$O(m \log m)$

空间复杂度：$O(m)$

3.9 286D. Tourists

Description

有m堵墙，每堵墙的范围是坐标$[l_i, r_i]$，从t_i时刻起出现（不消失）。有n对人，每对人从q_i时刻，坐标0开始，在墙的两侧向右（坐标轴正方向），以每时刻1单位距离的速度前进，问每对人看不到对方的时刻有多长。视线与墙有交点（包括墙的边界），就视为看不到。

Analysis

我们可以先将墙的坐标离散化，然后按出现顺序排序，用并查集即可处理出：对于每一段有墙出现的线段，最早的墙是什么时刻出现。

我们考虑每一条线段$[l_i, r_i]$，考虑每一对人q_j，一共有三种情况：1、$q_j + l_i \geq t_i$，即$q_j \geq t_i - l_i$，那么整个线段都会对答案造成贡献；2、$q_j + r_i \leq t_i$，即$q_j \leq t_i - r_i$，整个线段都不会对答案造成贡献；3、其他情况下造成的贡献是$q_j - (t_i - r_i)$。

我们先将每一条线段重新更改为$[t_i - r_i, t_i - l_i]$，然后差分。接着按人出发的先后顺序，即q_j 单调递增，进行扫描线，即可求出每对人的答案。

时间复杂度：$O(m(\log m + \alpha m))$

空间复杂度：$O(m)$

4 Volume.4

4.1 332D. Theft of Blueprints

Description

给定一个n个点的特殊的无向图，满足任意选出k($k < n$)个点，这k个点都有且仅有一个相邻的点A。每次选k个点的代价为这k个点到A的距离和，求代价的期望，保留整数（不用四舍五入）。

Analysis

我们发现，对于$k = 1$时，我们可以枚举选出的那个点，然后统计。当$k = 2$时，我们可以枚举每个i点作为A的情况，如果i的所有相连点为$v_1..v_j$，一共有C_j^2种情况，每条边被选中$(j-1)$次，贡献就是$(j-1)$倍的边权。

当$k \geq 3$时，我们发现这个图就是$k+1$个点的完全图。如果这个结论成立，那么就可以在$O(n^2)$的复杂度下，方便地计算出答案了。下面证明这个结论：

首先我们任取两个点S,T，并且列出那些和S,T两个点都有边相连的其他点$v_1, v_2..v_l$。我们先证明$l = k - 1$，称为引理1。若$l > k - 1$，那么我们在v_i中可以选出k个点，这k个点的公共点有两个:S,T，这与条件不符。若$l < k - 1$，那么我们这样选取k个点：$v_1, v_2..v_l, S, T$，那么有了l+2个点，如果不足k个再任取其他点。现在我们有了k个点，那个相邻点A一定不在v_i之中，也不与S,T相等，那么我们就又找到了一个点可以加入v集合之中，这样也与假设不符。

下面证明这个图中存在$k+1$个点组成的完全图，引理2。我们考虑如下的构造：设点集$v_1, v_2..v_k, v_{k+1}$。其中前k个点任取，第k+1个点是前k个点对应的A点。那么有v_{k+1} 和其他k个点有边相连。我们假设$v_{i+1}..v_{k+1}$，都分别与其他k个点相连，那么考虑v_i，如果$v_1..v_{i-1}, v_{i+1}..v_{k+1}$都有边和$v_i$相连，那么得证；如果没有，那么肯定有一个在v集合之外的点u，成为这k个点的A点，那么将v_i用u替换即可。如此类推可证明引理2。

如果$n > k+1$，由引理2，我们设集合S：$v_1, v_2..v_{k+1}$是一个完全图，并且必定有一个点u不属于S。我们取出v_1, u，由引理1，我们发现有k-1个与他们都相邻的点集T。T中肯定有点存在于S中，因为u肯定与S中的一个点有一条边相连（否则就不在一个联通块中，大于1个联通块是不可能的）。如果T 的点有不少于2个在S中，不妨取出两个设为(p,q)。我们由引理1，(p,q)相邻的点除了同在S集合的k-1个点，还有u点，那么有k个点与(p,q)都相邻，矛盾。如果只有1个，不妨设为x，从S中取出另一个点y（不与x相同），那么(x, y, u)肯定有一个公共点z，由题目的性质得。如果z不在S中，那么(x,y)就有k个公共点，如果z在S中，那么与T中只有1个在S中不符。所以$n \leq k+1$，即$n = k+1$。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

4.2 332E. Binary Key

Description

给定一个字典字符串p，和一个目标串s，和K。要求一个长度为K的字典序最小的01串，将其复制$\lceil |p|/K \rceil$份后形成一个长为$|p|$的01串T（超出的删去），记T中1的位置为$q_0, q_1..q_{t-1}$。使得其满足$t = |s|$并且$s[i] = p[q_i]$。

Analysis

我们可以先枚举K中1的个数i，然后用f[j][k]表示，在T的后j位，放置k个1，可不可行。那么$f[j][k] = f[j-1][k]$ or $(f[j-1][k-1]$ and $g[j][|s|-k+1])$，其中$g[j][k]$表示，T 中第j 位放置的是从左往右第k个1，总共有i个1，是否可行。g数组可以在均摊$O(K|s|)$的复杂度求出，对于$g[j][k]$，每次搞两个指针$P = j, Q = k-1$，一直暴力检查p[P]和s[Q] 是否相等即可，然后$P+ = K, Q+ = i$，直到超出范围为止。f 数组也可以在$O(K|s|)$的复杂度求出。如果$f[1][i]$ 为真，那么就有解，如果当前状态是$f[j][k]$，如果能放0，即$f[j+1][k]$ 为真；如果不能就放1，然后转移到相应的位置。

时间复杂度：$O(K|s|^2)$

空间复杂度：$O(K|s|)$

4.3 261D. Maxim and Increasing Subsequence

Description

给定一个长度为n的数组b_i，其中最大的数不超过maxb，要求将b_i复制t份，成为新的数组a_i。求a_i的最长上升子序列长度。

Analysis

我们可以发现，t达到一定程度，答案就不会再改变。因为最长上升子序列最多只需要maxb个。那么我们可以令新的$t = min(t, maxb)$。然后我们考虑dp，令$f[i]$表示以i为结尾的最长上升子序列的最长长度。如果我们当前考虑到a_j，我们可以用$f[a_j]+1$去更新$f[a_j+1], f[a_j+2]..f[maxb+1]$。又因为f数组是单调不降的，我们更新到$f[k] \geq f[a_j]+1$ 时，就不需要继续更新下去了。所以我们每次更新都能使得f数组的某一位增大，又因为$\sum_{i=1}^{maxb} f[i] \leq min(n, maxb) * maxb$，所以均摊的复杂度为$O(n * maxb + n\sqrt{n})$。

时间复杂度：$O(n * maxb + n\sqrt{n})$

空间复杂度：$O(n)$

4.4 261E. Maxim and Calculator

Description

给定一个二元计算器，每次可以对数对(a, b)进行操作，一开始你的数对为$(1, 0)$，每次可以选择一个操作：1、$(a, b) => (a * b, b)$；2、$(a, b) => (a, b + 1)$。在数对中的第一位记为你能得到的数。求[l,r]之间，P次操作之内，你能得到多少个不同的数。

Analysis

打表可以发现，在不超过r的数字中，质因数不超过P的数约有3000000个。这些数包含了所有能得到的数字，因为仅仅将b加到P，然后乘到a，就已经超过了P步。对于一个数字x，如果将其表示为$x = \prod_{i=1}^{k} q_i$，其中q_i不一定是质数，那么步数就是$k + max(q_i)$。

我们从小到大枚举$i = max(q_i)$，用f[j]表示第j个数当前i下的最小的k。每次只需要用$f[j]+1$更新$f[k](a[j] * i = a[k])$。然后实时统计$f[k] + i$是否小于等于P。

时间复杂度：$O(K \log K)$（K为在范围内的所有质因数不超过P的个数，约为3000000）

空间复杂度：$O(K)$

4.5 USACO Open08.Cow Neighborhoods

Description

平面上n个点，将两两曼哈顿距离小于等于C的点对之间连边。问最后形成多少联通块，以及最大的联通块的大小。

Analysis

我们假象出$n*(n-1)/2$条边，然后删去那些长度大于C的边，剩下的边都小于等于C。由最小生成树的环切性质，得出只需连接最小生成树中小于等于C的边就可以得到最后联通块的情况。然后用并查集统计即可。

平面曼哈顿距离最小生成树有一个性质，即可能在最小生成树中的边可以简化到$O(n)$级别。具体就是每个点只向，以它为中心的8个象限中最短的那个点连边。简略证明：如果点x为中心，在某个象限离他最近的是y，在这个象限中有点z（z不等于y），可以证明y和z的曼哈顿距离小于等于x到z的距离，即$|xz|$如果出现在最小生成树中，可以被$|yz|$替换。

时间复杂度：$O(n(\log n+\alpha(n)))$

空间复杂度：$O(n)$

4.6 USACO Mar08.Land Acquisition

Description

给定n块矩形土地，大小分别为(L_i,W_i)。你要将这n块地打包购买，每个包的花费是包中最大的长度乘最大宽度，即$max(L_i)*max(R_i)$。求买下这n块地的最少花费。

Analysis

首先对于两块地(L_i,W_i)和(L_j,W_j)，若有$L_i\geq L_j$且$W_i\geq W_j$，那么j的地就没有存在的意义，我们将其删去。删去所有的没有意义的地之后，我们将其按L从小到大排序之后，W就是严格递减的。

这样我们就可以进行dp了，因为每个包只会包含连续的一段。如果有个包不止包含一段，那么在这分散的几段中间的地，也可以被扔进包里，这样答案不会变差。朴素的方程就是$f[i]=min(f[j]+W[j+1]*L[i]),j<i$。我们将$(W[j+1],f[j])$看成一个点，$f[i]$的转移就是用斜率为$-L[i]$的直线去截这些点，直线在y轴的截距就是$f[i]$。那么因为$L[i]$是不降的，$W[j+1]$是递减的，我们用单调队列就可以维护左下角的凸壳，然后就能做到均摊线性的转移。

时间复杂度：$O(n\log n)$

空间复杂度：$O(n)$

4.7 USACO Dec12.First!

Description

给出n个字符串，问那些字符串有可能成为“字典序”最小的那个。“字典序”的顺序可以任意给定。

Analysis

我们先将这些字符串放到Trie里面。我们考虑类似于dfs的方法，首先如果到了这个点u，表示从根到u的字符串是当前最小的，同时我们维护的字典序中字母的大小关系没有出现矛盾。如果我们走到了某个叶子节点v，那么我们就能够说从根到v是一个合法的字符串，如果v还有孩子，那么就不需要接着往下，因为这些字符串肯定比v大。

接着，我们考虑如何维护字典序中字母的大小关系。用$B[i]$表示比字母i小的是什么，二进制第j位上是1就表示比i大，反之还未确定。如果我们到了一个点u，发现有若干个孩子，记为$v_1, v_2..v_k$。如果我们选择进入v_i的子树，那么v中其他的东西都比v_i大，所以$B[v_i] = B[v_1]|..B[v_k]|(1 << v_1)|..(1 << v_{i-1})|(1 << v_{i+1})|..(1 << v_k)$。并且如果对于一个j，有$B[j]$的第i位为1，那么我们要转移大小关系，即$B[j] = B[j]|B[i]$。如果有一个x，满足$B[x]$的第x位上为1，那么就出现了矛盾。B的值只需在一棵子树递归完成后回溯即可。

时间复杂度：$O(L * A)$（L为字符串总长，A为字符集大小）

空间复杂度：$O(L * A)$

4.8 USACO Open09.Tower of hay

Description

有n堆稻草，每堆稻草的面积为$1 * w_i$，你要将它们依次堆叠，从下到上从左到右，使得下面的稻草宽度大于等于上面的稻草宽度。求最大能堆叠的高度。

Analysis

这道题目有一个性质，即如果我们最小化了最下面一堆的宽度，就可以最大化堆叠的高度。如果这个性质正确，那么我们从后往前dp，$f[i]$表示以i为最底下一层第一块时，最小化的最底下一层稻草宽度，有$f[i] = min\{s[j-1] - s[i-1]\}$满足$s[j-1] - s[i-1] \geq f[j]$。这个方程可以使用单调队列来使得均摊的复杂度达到$O(n)$。

下面证明这个性质，设Max_i表示以i为最底下一层第一块时，最高的稻草高度。可以发现Max_i是单调不升的。因为假如有$Max_i < Max_{i+1}$，那么我们模仿Max_{i+1}的所有堆放方法，最后一层加一个w_i就可以使$Max_i = Max_{i+1}$。如果对于$w_i...w_n$的放置方法，最优的情况并没有最小化底层的宽度，即如果这种方法的最后一层的分界点在j的位置，必定有一个k($k < j$)，也可以作为分界点，又因为Max_i是单调不升的，那么$Max_k \geq Max_j$，即从k转移肯定不差于从j转移。

时间复杂度：$O(n)$

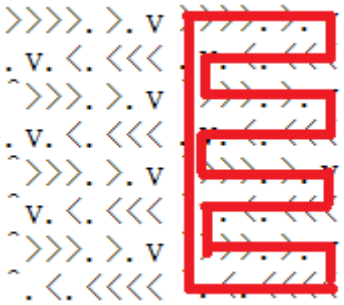
空间复杂度：$O(n)$

4.9 329D. The Evil Temple and the Moving Rocks

Description

你要在在n*n($n = 2k$)的棋盘上放置一些石头。石头可以有4个朝向，可以在某些格子不放置石头。放置完成后，你可以激活一块石头，被激活的石头将会一直朝着它的方向移动，直到撞到了其他石块或者撞到了房间四周的围墙。如果撞到了围墙，那么游戏结束；如果撞到了其他石头，那块石头会被激活，并重复上述动作。如果某块石头运动了至少一个距离才停下来（撞到石头或者围墙），那么就会发出一记响声，你要使响声次数x不少于$k^3 - k^2$。

Analysis

考虑下面的构造：  。形如这样的过程，会持续x次，x为一行中连续放置的相同朝向石头数，每次会撞击$y * (n + 1)$次，y为一行中"."的个数。我们要最大化$x*y*(n+1)$，而$x+2y = n$，由基本不等式得到，$x = 2y$的时候取到最大值，并且这样可以使响声次数x不少于$k^3 - k^2$。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

5 Volume.5

5.1 USACO Dec12.Gangs of Instanbull

Description

有m个帮派，每个帮派有a_i头奶牛，总共n头。每个时刻会有一头来自i帮派的奶牛来到牧场，如果牧场上没有奶牛或者存在的奶牛是属于自己帮派的，那么这头奶牛就会留在牧场上；反之，这头奶牛就会带着牧场上的一头奶牛出去单挑，并且永远不会回来。你要使1号奶牛在时刻n之后留在牧场上的牛数最多，如果有解，输出字典序最小的奶牛到来的序列（奶牛的号码就是帮派的号码）。

Analysis

首先判断是否有解，如果$n-a_1$为奇数，那么至少需要一头1帮派的奶牛；如果其他帮派的牛数最大值Max，满足$Max > n - Max - a_1$，那么需要$2Max - n$头1号奶牛才能使牧场上最后不存在牛。综上1号奶牛的损失数$lose = max((n-a_1)\&1, 2Max - n)$。如果$lose \geq a_1$，那么无解；反之，最后留下的1号牛数为$a_1 - lose$。

如果有解，并且只有两种帮派的奶牛，那么肯定是先派出所有1号奶牛，再派出所有2号奶牛。如果多于两个帮派，那么肯定是先派出lose头1号，最后派出$a_1 - lose$号奶牛占领牧场。中间的奶牛先按字典序派出，如果有一个帮派Maxp的奶牛头数大于其他还存在的奶牛数（存在指的是没有去单挑），那么就停止这个过程，用Maxp先干光所有在场上的奶牛。接着派光编号最小的奶牛，用Maxp去干，重复以上过程，直到Maxp为编号最小的，最后将所有帮派的奶牛按编号大小依次派出即可。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

5.2 USACO Nov08.Toys

Description

有n天，每天需要有T_i个新（消毒）玩具，每天可以在消毒处获得已经消毒完成的玩具，或者去购买，购买的新玩具单价为C。每天用剩下的玩具可以去消毒，有两种消毒方式，分别需要T1,T2天，单价分别为C1,C2。求满足每天需求的情况下，花的最少的钱数。

Analysis

如果我们知道了购买的玩具数x，那么我们可以使用贪心求出相对应的最小钱数$f(x)$。一开始的若干天都是用买来的玩具，如果出现买来的玩具用完的情况，我们优先使用廉价（花费天数多）的消毒方法，如果天数不够，那么我们使用昂贵（花费天数少）的消毒方法，如果还是不行，那么判断这种情况无解。在选择使用哪一天的玩具来进行消毒时，我们优先选择时间最近的玩具，因为我们要使时间久远的玩具尽量使用廉价（花费天数多）的方式去消毒。

事实上$f(x)$是一个凹的函数，如果这个性质成立，那么我们可以利用三分在$O(n \log n)$的时间内求出最小值。我们用$g(x)$表示买了x个玩具时，最优情况下使用消毒所花费的代价，即$f(x) = g(x) + C * x$。$C * x$是单调递增的，而$g(x)$是单调不升的，因为可用的玩具变多不会使花费变多。如果我们证明了$g(x+1) - g(x) \leq g(x) - g(x-1)$，那么$f(x)$就是一个凹函数。对于$g(x)$，事实上每加一个玩具的提升不会高于上一次加玩具的提升。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

5.3 USACO Dec08.Largest fence

Description

平面上n个点，保证不会有三点共线，求一个凸多边形，使得凸多边形上的点最多。

Analysis

很容易想到一个$O(n^4)$的做法，我们先枚举这个凸多边形左下角的点，然后$f[i][j]$表示当前凸多边形所用的最后一个点是j，倒数第二个是i，所能拥有的最大点数。因为枚举左下角是$O(n)$的，状态是$O(n^2)$的，转移是$O(n)$的，所以复杂度为$O(n^4)$。

实际上我们可以减少在转移的时候的复杂度。我们不用暴力地去转移，我们使用记忆化搜索，来试图将转移达到$O(1)$。我们重新定义状态，$f[i][j]$表示当前最后一个点为i，我们考虑要放的笑一个点是j，那么有两种转移方式$f[i][j] = max\{f[j][k] + 1\}$或者$f[i][j] = max\{f[i][l]\}$。前者是放置j点，并且下一个考虑的点是k；后者是考虑下一个要放的点l。为了防止重复等情况，所以我们要选择这样一个顺序，k满足$\overrightarrow{p_ip_j}$在$\overrightarrow{p_jp_k}$的逆时针方向上的第一个点，l则为$\overrightarrow{p_ip_j}$在$\overrightarrow{p_ip_l}$逆时针方向上的第一个点。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

5.4 USACO Jan09.Travel

Description

一张包含n个点，m条边的无向图，保证没有重边和自环。保证起点为1的最短路径生成树唯一。n-1头奶牛分别要从1号点去$2..n$号点，如果每头牛都不能经过原先最短路中的最后一条边的情况下，求现在每条牛的最短路（后面称为“次短路”），如果没有输出-1。

Analysis

我们先用dijkstra求出最短路径生成树。我们发现现在要求的“次短路”坑定是从1号点经过最短路到v1点，再经过一条边到v2点，接着用最短路到达目标点u（v2可以等于u）。考虑树上的点u，用$D[1][u]$表示从起点到u的最短路长，用$Dt[u]$表示我们要求的“次短路”长。$Dt[u]$能够通过节点v的$D[1][v]$转移，当v不是u的后代并且v不是u的父亲；此外（v为父亲时，是不可以转移的，因为没有重边），对于v的后代x_i我们可以用堆来维

护每个点的$D[1][v1]+e[v1][v2]+D[v2][x_i]$，从孩子向父亲转移的时候，我们给每个$x_i$的堆加上$e[x_i][u]$，然后将堆合并，取最小的那个满足条件的即可（这里如果不满足条件，那么对于u的祖先也不满足条件，可以直接删去）。堆的合并可以使用左偏树。

时间复杂度：$O(m\log m)$

空间复杂度：$O(n\log n)$

5.5 323C. Two permutations

Description

有两个包含n个数的排列。有m个询问$(l1,r1,l2,r2)$，每次求出在第一个排列中的位置在$[l1,r1]$，在第二个排列中的位置在$[l2,r2]$的数有几个，强制在线。

Analysis

事实上我们把在第一个排列中的位置看成x，把第二个排列的位置看成y，就相当于在平面中数在相应矩形中的点有几个，这个可以用可持久化线段树来解决。

时间复杂度：$O((n+m)\log n)$

空间复杂度：$O(n\log n)$

5.6 USACO Mar09.Clean Up

Description

有n个范围在$[1,m]$之间的数，你要把这n个数分成若干段，每段的花费是这一段中不同的数字个数d_i的平方，最小化这个花费。

Analysis

我们可以很快地写出一个转移方程$f[i]=max\{f[j]+(diff[j+1][i])^2\}$，其中对于diff的处理只需记录结尾为i，不同个数为k的最小的开头$p[j]$。那么我们就可以直接通过$p[j]$转移，因为f数组是单调不降的。p数组在i到i+1的改变也可以$O(n)$维护。但是这样的时间复杂度是$O(n^2)$的。

我们发现，最差情况下就是分成n段，这样的花费为n，也就是说每段中不同数字的个数不超过$\sqrt{n}$。那么我们只需记录p的前$\sqrt{n}$，所以时间复杂度就降到了$O(n\sqrt{n})$。

时间复杂度：$O(n\sqrt{n})$

空间复杂度：$O(n)$

5.7 USACO Mar10.Star Cow Craft

Description

星际争霸中有三种军队战斗力分别为x,y,z，我们不知道x,y,z具体的值，但是我们知道他们没有一个的战斗力超过另一个的100倍，并且我们知道n次战斗的信息，即知道$a_1 * x + b_1 * y + c_1 * z$和$a_2 * x + b_2 * y + c_2 * z$之间的大小关系。现在我们有n次战役$(a_1, b_1, c_1, a_2, b_2, c_2)$需要预测，输出那一方可能会赢，或者不能确定。

Analysis

我们可以将$a * x + b * y + c * z$转变为$a * (x/z) + b * (y/z) + c$，令$X = x/z$,$Y = y/z$，那么我们就可以将n个大小关系看成一条直线。并且每次用这条直线去截现在的合法多边形，即半平面交。初始的多边形是$(0.01, 0.01), (0.01, 1), (1, 100), (100, 100), (100, 1), (1, 0.01)$。

预测战役胜负的时候，我们只需要判断这个合法多边形是否整个在直线的某一边，如果是，就能预测胜负；如果不是，就不能确定。

时间复杂度：$O(n^2 + m)$

空间复杂度：$O(m)$

5.8 314E. Sereja and Squares

Description

给定一个长度为n的字符串包含小写字母和"?"。你要将问号替换成小写字母或者大写字母，不包含"x"和"X"，要求以相对应的小写字母和大写字母为左右括号，满足括号序列的性质。问方案总数。

Analysis

我们可以想到一个$O(n^2)$的dp，$f[i][j]$表示到第i位，当前没有闭合的左括号有j个，有多少方案。则第i位是小写字母时，$f[i][j] = f[i-1][j-1]$；如果是"?"，那么$f[i][j] = f[i-1][j+1] + f[i-1][j-1] * 25$。

事实上我们只要进行一些常数优化就可以通过这题了。首先，在中途对于第i位是"?"的转移时候，如果第i位之后还剩下偶数个"?"，那么$f[i][j]$的j也应该是偶数，反之应该是奇数，这个优化能使常数变为原来的一半。并且，我们知道了最后有几个"?"变成了左括号，事实上，我们只要知道有几个不同的选择位置x，然后答案就是25^x，这样也可以减少常数。

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

5.9 USACO Open10.Triangle Counting

Description

平面上有n个点，保证没有一个点和原点重合，并且没有两个点和原点共线。要求从n个点中选出3个点，使得这三个点构成的三角形包含原点。求选择的方案数。

Analysis

我们考虑正难则反，我们在所有的选择数C_n^3中减去不合格的选择数就是答案了。我们考虑一个点x和$\overrightarrow{Oy}$在$\overrightarrow{Ox}$的顺时针方向上的所有点y，我们选择x为三角形中的一个点，从y中任意选择2个点，这样就是以x为逆时针方向上最远点的不合格三角形个数。我们将n个点一一考虑过来就行了。我们可以先将这n个点极角排序，然后就可以$O(n)$计算了。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

6 Volume.6

6.1 USACO Jan11.Bottleneck

Description

一棵有n个点的有根树（以1为根），每个点上有C_i个奶牛，每个点i到他的父亲有一条单向道路，道路的流量上限为Lim_i。一头奶牛可以在一个单位时间内走到任意一个节点（祖先），但是道路的流量不能超过上限。有m个询问T_i，你要求出在T_i时刻，1号节点最多能有多少奶牛。

Analysis

首先很显然，每一时刻都尽可能地从每个点向1号点走，这样肯定是最优的。考虑暴力，我们可以每一时刻都从叶子节点开始，向它父亲推送尽可能多的流量（要不到达道路上限，要不到达这个点拥有的奶牛数），接着一直推到1号点。

然后我们发现，每条道路上的流量是随着时间不升的。接着我们考虑每个点，我们先假设每条边都是满流，我们对于每个点u可以列出$(in[u] - out[u])t + C_u$的式子，如果流入小于流出，那么在某个时刻这个点的流出就会有变动。我们用堆来维护这些事件的发生，如果在某次事件发生之后这条边的流量为0，那么我们可以忽略这个点，因为这条边再也不会达到上限了。忽略点的操作可以用并查集来维护。

时间复杂度：$O(n(\log n + \alpha(n)))$

空间复杂度：$O(n)$

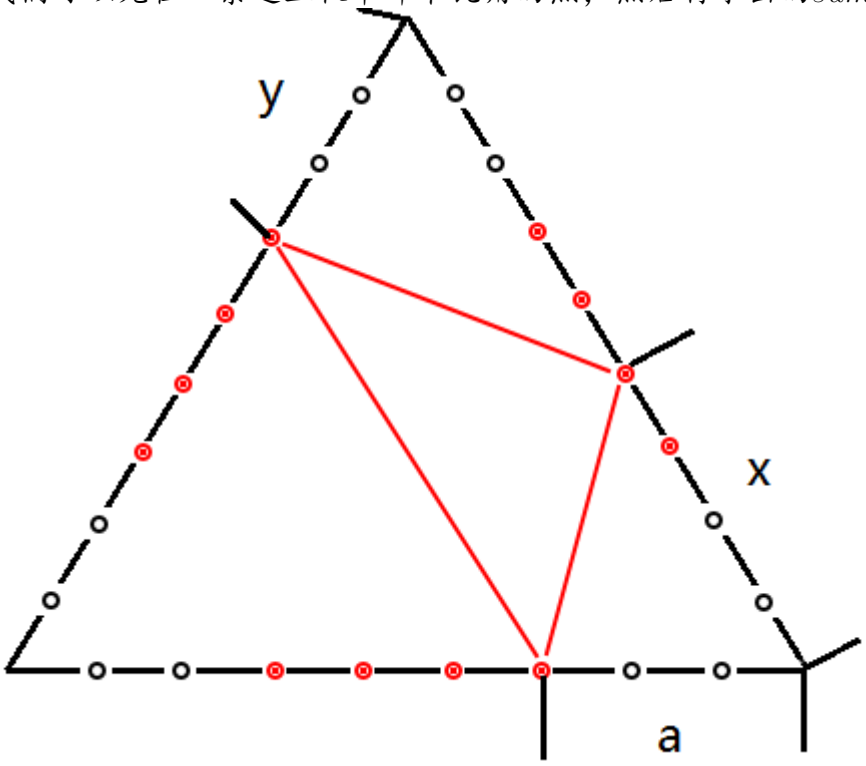
6.2 309D. Tennis Rackets

Description

有一个等边三角形，每条边上被均分为(n+1)等份。每条边上的n个点中，编号为$1..m, n-m+1..n$的点是不能选择的。现在你要在每条边上选择1个点，使得这3个点构成的三角形为钝角三角形，求选择的方案数。

Analysis

我们可以先在一条边上枚举那个钝角的点，然后将求出的sum'乘3即可。



我们固定左边那个点，它离左下角的距离为x，然后我们从小到大枚举a，我们发现y是随着a而递减的。并且y满足$x^2+a^2-ax+(A-x)^2+y^2-y(A-x) < (A-a)^2+(A-y)^2-(A-a)(A-y)$，其中A是边长，即$A=n+1$。这样的时间复杂度是$O(n^2)$的，我们进行常数优化，比如x的对称性，即$x \le A/2$和$x \ge A/2$的情况是一样的，就可以将常数变为原来的一半。

时间复杂度：$O(n^2)$

空间复杂度：$O(1)$

6.3 USACO Mar12.Cows in a Skyscraper

Description

有n个物品，每个物品的重量是w_i。你要将它们分成尽可能少的份数，使得每份的重量不超过m。

Analysis

我们用二进制来表示状态，即每个物品是否已经取到。我们发现，对于每一个状态，我们用$f[st]$记录当前状态下最少能分成几份，如果份数相同，我们再用$g[st]$记录最小块中的大小。转移的时候只需要看看最小块中放不放的下，或者是否要开辟新的一块即可。

时间复杂度：$O(n2^n)$

空间复杂度：$O(2^n)$

6.4 311E. Biologist

Description

有n条不同性别的狗，你可以将其中的若干条狗改变性别，每条狗改变性别所需要v_i的花费。同时你有m个赌局，每个赌局都是如下表述：如果第$t_1, t_2..t_{k_i}$的狗为y性别，那么将得到w_i的钱，否则如果赌局特殊，将会失去g的金钱。

Analysis

我们可以用最小割解决这个问题，我们首先加上所有能够得到的金钱Sum，然后用最小割求出：变性别的花费+不能获胜的赌局所失去的钱，从Sum减去就是我们的答案了。

我们将狗看成$1..n$个点，那么如果在最小割之中，某个点和S相连，我们认为它最后是0性别，若和T相连，那么就是1性别。所以如果原来是0性别，那么我们从连边(S, Dog_i, v_i)，否则我们连边(Dog_i, T, v_i)。

如果某个赌局要求若干条狗为0性别，就连边$(S, Bet_i, w_i+g*[Bet_i isspecial])$，并且连边$(Bet_i, Dog_j, inf)$。如果要求性别为1，那么连边$(Bet_i, T, w_i + g * [Bet_i isspecial])$，并且连边$(Dog_j, Bet_i, inf)$。

时间复杂度：$O(V * E^2)(V = n + m, E = n + \sum k_i)$

空间复杂度：$O(E)$

6.5 309B. Context Advertising

Description

有n个单词构成的句子。要选择一段连续的单词，使得将其按次序排放成r行之后，每行的字符数不超过c。并且每个单词之间需要有一个空格。选出单词个数最多的一段，并且输出方案。

Analysis

首先，如果我们知道该从第i个单词开始，那么我们肯定放入单词，如果一行超过c个字符就换行。事实上，我们可以通过倍增来加速这个过程，这样我们暴力枚举该从哪个点开始，然后用倍增就可以将查找的过程变为$O(\log n)$。

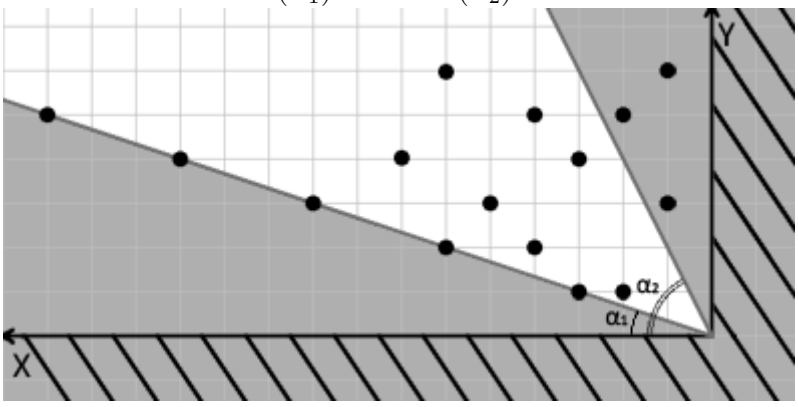
时间复杂度：$O(n \log n)$

空间复杂度：$O(n \log n)$

6.6 249D. Donkey and Stars

Description

在平面上有n个点，每个点在(α_1, α_2)之间有一个可见范围。可以从每个点，跳到它的可见点上，你要从(0,0)开始跳，直到不能跳为止，求最大的经过的点数。$\frac{a}{b} = \frac{sin(\alpha_1)}{cos(\alpha_1)}, \frac{c}{d} = \frac{sin(\alpha_2)}{cos(\alpha_2)}$



Analysis

如果我们当前在点i，那么点j在其视野中的情况就是：$a(x_j - x_i) - b(y_j - y_i) < 0$且$-c(x_j - x_i) + d(y_j - y_i) < 0$，化简后就是$ax_j - by_j < ax_i - by_i$且$-cx_j + dy_j < -cx_i + dy_i$。我们只需要将第一维$ax_i - by_i$从小到大排序，然后使用第二维$-cx_i + dy_i$在树状数组里查询即可。

时间复杂度：$O(n \log n)$
空间复杂度：$O(n)$

6.7 249E. Endless Matrix

Description

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

求形如上图的子矩形(x_1, y_1, x_2, y_2)的和。

Analysis

我们可以将子矩形差分，即分别求$(1,1,x_1-1,y_1-1),(1,1,x_2,y_2),(1,1,x_1-1,y_2),(1,1,x_2,y_1-1)$的和。那么问题就变成了求$(1,1,x,y)$的和。如果$x=y$，那么答案就是$(x^2+1)x^2/2$；如果$x>y$那么我们先求$(1,1,y,y)$的和，剩下部分就是$y*\sum_{i=y+1}^{x} i^2-(y-x)*y*(y-1)/2$；如果$x<y$也同理，只不过剩下部分的和变为$x*\sum_{i=x}^{y-1} i^2+(x-y)*x*(x+1)/2$。注意要使用高精度。

时间复杂度：$O(n \log n)$
空间复杂度：$O(n)$

6.8 USACO Jan12.Cow Run

Description

有n组卡牌，每组8张，从上到下依次排列，每张牌上有一个数字。第i个回合，John会拿出第i组卡牌，并选择上面4张或者下面4张给Bessie，而Bessie会从这4张牌中再选择上面2张或者下面2张。记这两张牌中上面那张为a，下面那张为b，他们会命令奶牛绕长为m的操场跑$a*dis+b$的距离，其中dis是奶牛已经跑的距离（初始情况下奶牛在起点）。如果n个回合结束后，奶牛离起点的距离不超过K，则John获胜，反之Bessie获胜。保证无论Bessie如何选择John都有必胜策略，现在给出Bessie的策略，要

使John的策略必胜，且字典序最小。John在决策时并不知道Bessie接下来的决策。

Analysis

在每次John决策时，我们尽可能地选择字典序最小的那个决策，然后暴力枚举Bessie和John接下来的决策，看是否必胜，如果是，那么就可以用那个字典序小的，反之只能用大的。每次这样判断，我们发现最后的复杂度是$O(4^n)$的。接下来，我们有一个小优化，就是John的选择可以看成是or，Bessie的可以看成是and。如果John的某个选择为True，那么不用进行下一次枚举，直接返回True；如果Bessie的某个选择为False，那么直接范围False。这个优化对复杂度没有太大的影响，因为可以构造数据来卡掉它。但是如果我们随机搜索选择，那么复杂度就可以证明了。

用$f(i)$表示在第i层的or决策时，如果返回False的时间，用$t(i)$表示在第i层的or决策时，返回True的时间。由德摩根定律，not(A or B)=(not A) and (not B)，$f(i) = 2t(i-1)$。那么如果返回True，$t(i) = 1/2(2 * f(i-1) + t(i-1))$。化简之后就是$t(i) = 2 * t(i-2) + 1/2 * t(i-1)$，即$t(i) = (\frac{1+\sqrt{33}}{4})^i$。

时间复杂度：$O(2.84307033^n)$

空间复杂度：$O(n)$

6.9 311C. Fetch the Treasure

Description

有n个宝藏，每个宝藏在数轴的a_i处，并且价值为c_i。你一开始拥有一个行走方式k。如果你当前的行走方式为$w_1, w_2..w_{op_1}$，那么你能走到的格子就是$1 + \sum_{i=1}^{op_1} v_i * w_i$，其中$v_i$为任意自然数。每次有3种操作：1、增加一种行走方式；2、将第i个宝藏的价值减少y；3、将所能走到的格子上最大的宝藏拿走，拿走后这个宝藏消失。

Analysis

如果我们能够知道任意时刻我们能够到达那些格子，由于到达的格子数是不降的，我们可以用一个堆来存储所能达到的宝藏，然后取最大值即可。并且，如果我们将数轴上的点按mod k分类。如果在第i堆中能够达到v格子，那么$v + k, v + 2k..$都能达到，那么我们只要维护$dis[i]$表示第i堆中最小的能到达的点。为了维护dis，我们只要将每一类看成一个点，然后连边，最短路维护即可。具体就是，对于第i类，我们向$(i + w_j) mod\ k$类，连长度为w_j的边。

时间复杂度：$O(op_1 * (k \log k + n) + m \log n)$$op_1$为操作1的个数。

空间复杂度：$O(n)$

7 Volume.7

7.1 USACO Open13.Photo

Description

有n个点和m个区间$[l_i, r_i]$，每个区间内有且只有一个点是特殊的。求特殊点最多可能有多少，或者判断无解。

Analysis

我们可以用动态规划来解决这个问题，用$f[i]$表示第i个点是特殊点，并且满足每个区间有且仅有一个特殊点的条件下最多的点数。考虑转移，如果从$f[j]$转移而来，那么i和j不能在同一个区间内，并且在i和j之间不能有完整的区间，即$max\{r_k|l_k \leq j\} < i$且$min\{r_k|l_k > j\} \geq i$。记$L[j] = max\{r_k|l_k \leq j\} + 1, R[j] = min\{r_k|l_k > j\}$，即状态j 可以转移给$[L[j], R[j]]$之间的状态，我们用线段树维护这个过程即可。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

7.2 USACO Open13.Figure Eight

Description

有一个$n * n$的棋盘，上面有些点是坏点。我们要在这个棋盘上划两个矩形框，满足这两个矩形框一个在上一个在下，并且上面的底边和上面的顶边相互粘连，且上面的底边所包含的列是下面的顶边所包含的列的子集（不严格），你要划两个框，使得两个框相乘的面积之积最大。

Analysis

我们用$Top[i][j][k]$表示左右底边分别在i,j列k行的最大矩形框的面积，因为底边的长度已知，所以只需最小化顶边所在行即可。这个可以在$O(n^3)$的时间复杂度内求出，同理我们也可以求$Bot[i][j][k]$，即左右顶边分别在i,j列k行的最大矩形框的面积。最后我们用$Top'[i][j][k]$表示左右边在第i,j列之间，底边在第i行的最大面积，即$Top'[i][j][k] = max(Top[i][j][k], Top'[i+1][j][k], Top'[i][j-1][k])$。最后我们只要枚举下面矩形的左右端点$(k, i), (k, j)$即可，$ans = Top'[i][j][k] * Bot[i][j][k]$。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^3)$

7.3 USACO Mar13.Hill walk

Description

平面上有n条线段$(x1, y1, x2, y2)$，满足$(x2 > x1, y2 > y1)$，并且线段两两不相交。这几条线段可以看成一条山路，保证有一条山路的起点在$(0, 0)$。你要从$(0, 0)$开始沿着山路向上走，走到$(x2, y2)$（这个端点可以看成是空的），就会笔直下落，如果落到地上，那么就结束，如果落到另一条山路上，就继续前行。求最后能到达几条山路。

Analysis

我们发现，只要知道从某条线段的右端点下落会落到哪里，就可以解决这个问题。我们使用扫描线，如果维护当前经过扫描线的山路的上下关系，那么我们只需维护最高的那条就够了，并且上下关系只会在加一条线段和删除一条线段会进行改变，那么就可以用堆来维护了。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

7.4 295D. Greg and Caves

Description

一个n*m的棋盘，每个格子可以染成黑白两种颜色。一个棋盘称为好的，当且仅当有$(1 \leq l \leq r \leq n)$，在$[l, r]$的行中有两个黑色格子，并且其他的格子都是白色。并且有一个t，使得对于$(l \leq i < j \leq t)$，第j行的黑格子所在列为(c_1, c_2)，第i行为(c_3, c_4)，满足$c_1 \leq c_3, c_2 \geq c_4$。且对于$(t \leq j < i \leq r)$，第j行的黑格子所在列为$(c_1, c_2)$，第i行为$(c_3, c_4)$，满足$c_1 \leq c_3, c_2 \geq c_4$。求所有好的棋盘的方案数。

Analysis

通过观察，我们发现可以将有黑色格子的行分为上下两个部分。上下部分的计算基本一样，我们先计算上部分的方案数。用$f[i][j]$表示到第i行，该行的黑格子长度，即$c_2 - c_1 + 1$为j的方案数，那么$f[i][j] = 1 + \sum_{k=2}^{j} f[i-1][k] * (j - k + 1)$，这个可以用前缀和在$O(nm)$的时间复杂度内求出。我们通过枚举枚举最长的黑格子长度的行（如果有多个相同，就选择最下面的），来计算答案。上半部分的方案数已经求出，如果上半部分占有i行j列，那么下半部分就是$part2 = \sum_{k=2}^{j-1} f[n-i][k] * (j - k)$，那么对ans的贡献就是$f[i][j] * part1 * (m - j + 1)$。

时间复杂度：$O(nm)$

空间复杂度：$O(nm)$

7.5 283E. Cow Tennis Tournament

Description

有n头奶牛，每头奶牛有两两不同的能力值a_i。这n头奶牛两两间进行了比赛，正常情况下能力值大的将战胜能力值小的。现在有m个时间，每个事件j都会将能力值$a_i \in [l_j, r_j]$的所有奶牛之间的比赛胜负反转。求有多少个三元集合(p,q,r)，满足p战胜q，q战胜r，r战胜p。

Analysis

我们可以正难则反，答案就是C_n^3减去那些不满足的三元集合，不满足条件的三元集合中一定有一个元素，不妨设为p，被其他两个元素战胜。如果p是其中最小的，那么r,q和p之间被反转了偶数次；如果p是最大的，那么r,q和p之间的比赛被反转了奇数次；如果$r < p < q$，那么r和p之间被反转了奇数次，q和p之间被反转了偶数次。我们记$Big[i]$表示比i的能力值大，并且和i一起被反转偶数次的点的个数，$Small[i]$表示比i的能力值小，并且和i被反转了奇数次的点的个数。$ans = \sum_{i=1}^{n} C_{Big[i]}^2 + \sum_{i=1}^{n} C_{Small[i]}^2 + \sum_{i=1}^{n} Big[i] * Small[i]$。而Big和Small数组可以用线段树在$O(n \log n)$的时间复杂度内求出。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

7.6 264D. Colorful Stones

Description

有两个字符串S和T，其中只有"R","G","B"三种字母。我们用(x,y)分别对应S的第x个字符和T的第y个字符，你一开始在(1,1)的位置，每一次你可以叫出一个字母c，如果$S[x] = c$，那么x=x+1，如果$T[y] = c$，那么y=y+1，如果操作后$x > |S|$或者$y > |T|$，那么这次操作无效。一个状态(x,y)称为可以到达的，仅当你可以从(1,1)出发，用一个叫字母的顺序到达(x,y)。求所有能到达的状态数。

Analysis

我们可以发现，如果(a, b)可到达，设我们的操作序列为I，那么充要条件就是S[1..a-1]和T[1..b-1]是I的子序列，而S[1..a]和T[1..b]则不是。其中，一个必要条件就是：T[1..b]不是S[1..a-1]的子序列，S[1..a]不是T[1..b-1]的子序列，不妨称这个条件为P。

我们来看一下这个条件P是不是充分的。我们先假设在(c,d)的时候条件P是满足的，并且我们的目标点是(a,b)。如果$S[c] = T[d]$，那么我们转移到(c+1,d+1)，条件P任然满足；如果$S[c] \neq T[d]$且S[c+1..a]不是T[d..b-1]的

子序列，那么我们操作S[c]，即状态变为(c+1,d)；如果$S[c] \neq T[d]$且T[d+1..b]不是S[c..a-1]的子序列，那么我们操作T[d]，即状态变为(c,d+1)。另外一种情况就是，$S[1..a] = ...xy$而$T[1..b] = ...yx$的情况。

我们只需要求出不符合情况的条件，从$|S| * |T|$的总数中减去即可。

时间复杂度：$O(|S|)$

空间复杂度：$O(|S|)$

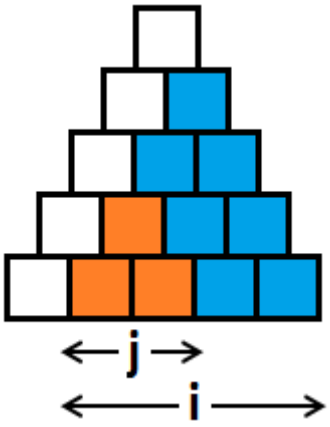
7.7 354D. Transferring Pyramid

Description

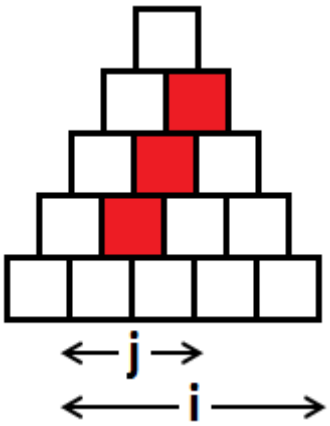
一个n行的平面金字塔，即从上往下数第i行有i个方块。这个金字塔的格子中有K个是特殊格子，你要使用2种操作将其覆盖：1、你可以选择一个格子覆盖，花费为3；2、你可以选择一个子金字塔，如果子金字塔为m行，那么花费为$2 + m(m+1)/2$。问每个特殊格子至少覆盖一次需要多少花费。

Analysis

我们设$f[i][j]$为覆盖了右下角，高为i的子金字塔的左下角少了高为j的子金字塔的最小花费。那么答案就是$f[n][0]$。



$g[i][j]$为下图红色格子中特殊点的个数。



那么$f[i][0] = min(f[i-1][k-1] + 2 + k*(k+1)/2 + g[i][k+1]*3)$。而$f[i][j] = min(f[i][j-1], f[i-1][j-1] + g[i][j+1]*3)(j > 0)$。这样的dp是$O(n^3)$的，但是我们发现如果我们所有的操作都是用操作1，那么我们至多只需要$3K$的花费。而我们dp中的$f[i][0]$中含有$k*(k+1)/2+2$，事实上我们枚举的时候不需要所有的k都没举，只需要使$k(k+1)/2+2 < K$即可，这样转移的复杂度就是$O(\sqrt{k})$的级别了。

时间复杂度：$O(n\sqrt{k})$

空间复杂度：$O(n)$

7.8 356E. Xenia and String Problem

Description

我们称一个字符串S是完美的，当且仅当：1、$|S|$为奇数；2、$S_{\frac{|S|+1}{2}}$在S中只出现了一次；3、$S[1..\frac{|S|+1}{2}-1]$和$S[\frac{|S|+1}{2}+1..|S|]$相等并且它们也是优美的。$|S| = 1$的字符串S都是优美的。我们称一个字符串S的优美程度为P，那么如果有$S[i..j]$为优美的，那么对P的贡献就是$(j-i+1)^2$。现在你有一个字符串T，你可以至多修改一个字符，最大化T的优美程度P。

Analysis

我们首先考虑不修改的时候该怎么求P。我们发现优美的字符串的长度依次为$1, 3, 7...2^m - 1$。那么只要枚举每个点作为优美串的中间点，然后用hash判断左右的串是否一样，然后因为长度只有$O(\log|T|)$种不同的取值，所以可以在$O(|T|\log|T|)$的时间复杂度内求出$L[i]$(表示以i为中心的最长的优美串，因为如果长度为$2l+1$优美，那么长度为l肯定也优美)。

事实上我们在求以每个点为中间点的优美串的时候，可以顺带求出修改某个字符后会有什么改变。如果以i为中心，长度为$2l+1$的串不为优美

串，有两种情况：1、中间的点出现了两次，且左右的串一样并且优美，那么枚举中间那个应该修改为哪个字符；2、左右的有一个不一样，我们可以修改左边或者右边，并且要满足中间不出现两次。

我们枚举哪个位置修改为哪个字符，所有修改所导致的变化最多为$O(\alpha|T|\log|T|)$，每次的变化就是将$L[i]$变为X，对于P的贡献就是$P = P - S[L[i]] + S[X](S[i] = \sum_{j=1}^{i}(2^j-1)^2)$。这样我们就可以直接维护了。

时间复杂度：$O(\alpha|T|\log|T|)$

空间复杂度：$O(\alpha|T|\log|T|)$

7.9 360E. Levko and Game

Description

一个包含n个点的有向图，有向图上的边分为两种，第一种有m条边，表示为(x,y,z)即从x到y，长度为z，第二种有k条边（称为特殊边），表示为(x,y,a,b)即从x到y，长度可以人为定为$z \in [a,b]$。现在你在起点s1，你的对手在起点s2，终点为f，求一种方案，使得你能够获胜，如果不能就追求平局，否则只能认输。

Analysis

我们可以发现，第二类的边长必定为$z = a$或者$z = b$。我们先不确定特殊边的边长，我们用dijkstra求出以s1,s2为起点的最短路，在这个过程中确定边长。如果一个点u，$dis[s1][u] < dis[s2][u]$则称其为优势点，反之称为劣势点。如果一条特殊边的入点x为优势点，那么我们将z设为a，反之设为b，因为这样不会使我们的最短路变差，也不会使对方的最短路变优。我们只要做一次最短路，然后看是否$dis[s1][f] < dis[s2][f]$，如果是的，那么直接输出，如果不是，我们只需要将优势点重新定义为$dis[s1][u] \leq dis[s2][u]$，然后再做一次最短路看是否$dis[s1][f] = dis[s2][f]$即可知道能否平局。

时间复杂度：$O((V+E)\log(V+E))$

空间复杂度：$O(V+E)$

8 Volume.8

8.1 323B. Tournament-graph

Description

给定一个n个点的竞赛图，你要给每条无向边定向，要求对于任意两个点$u,v,u \neq v$，u到v的距离不超过2。

Analysis

首先讨论n为奇数的情况，即$n = 2k + 1$。我们把形如$[2i, 2i + 1]$的点归为一组，并且连有向边$(2i, 2i + 1)$。对于一组点$[2i, 2i + 1]$，并且对于$j(j < 2i)$，我们连有向边$(j, 2i), (2i+1, j)$。对于一组点$[2i, 2i+1]$，$dis[2i][2i+1] = 1, dis[2i + 1][2i] = 2$，$(j < 2i)dis[j][2i] = 1, dis[j][2i + 1] = 2, dis[2i][j] = 2, dis[2i + 1][j] = 1$。

如果n为偶数，我们先将$n - 1$的情况构造出来，然后将最后一个点n加入图中，我们发现如果$n > 4$，那么只需连边$(n, n - 1), (n, 1), (j, n)$即可，但是$n = 4$就无解了。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

8.2 264E. Roadside Trees

Description

数轴上有n$(1 \sim n)$个点可以种树，每个月的开头会有一个操作：1、在y的位置种植一棵树高为$x(1 \leq x \leq 10)$的树；2、砍掉从左往右数第$p(1 \leq p \leq 10)$ 棵树。在每个操作之后，你需要输出这些树形成的最长上升子序列的长度，并且每个月种在地上的树会长高一个单位，数据保证每个时刻树的高度不同。

Analysis

我们发现如果把一棵树的位置x和高度y表示成平面上的一个坐标，那么最长上升子序列就是平面上的“上升”。我们发现种树的时候树高不高，也就是说当前最多有x棵树比新种的树矮，那么我们暴力重算这以x棵树为开头的最长上升子序列，这个过程我们可以用一棵以位置为关键字的线段树维护。并且我们发现砍树的时候树在较前面的位置，也就是说最多有y棵树在当前的树之前，我们只需要暴力重算以这y棵树为开头的最长上升子序列，这个过程我们可以用一棵以高度为关键字的线段树维护。

时间复杂度：$O((x + q)m \log n)$

空间复杂度：$O(n)$

8.3 319E. Ping-Pong

Description

有m个区间，对于区间(a, b)和(c, d)，如果有$a < c < b$或者$a < d < b$那么我们称区间(c, d)可以到达区间(a, b)。有n个操作，每次我们可以加入一个区间，或者询问第a次加入的区间能否经过若干个区间到达第b次加入的区间。保证每次加入的区间的长度严格长于之前存在的区间的长度。

Analysis

我们先用并查集将那些能够相互到达的集合并起来，即如果$a < c < b < d$或者$c < a < d < b$那么这两个区间就在一个集合里面。这个我们可以用线段树来维护，即对于区间(a,b)我们在线段树的$(a+1, b-1)$的范围内加上这个区间，然后对于新来的区间(c,d)，将包含c的和包含d的区间并起来就行了。因为保证每次加入的区间的长度严格长于之前存在的区间的长度，所以后来的区间(c,d)不会被之前的区间覆盖，那么一旦有$a < c < b$或者$a < d < b$，就保证了$a < c < b < d$或者$c < a < d < b$。

至于询问，如果第a个区间和第b个区间在一个集合里面，那么就是YES，如果第a个区间被第b个区间所在的集合包含了，那么也是YES，反之就是NO。

时间复杂度：$O(n \log n \alpha(n))$

空间复杂度：$O(n)$

8.4 258D. Little Elephant and Broken Sorting

Description

给你一个$1 \sim n$的排列，以及m个操作，每个操作(a,b)，表示交换第a个数和第b个数的位置，每个操作有1/2的概率被接受，问m次操作之后的期望逆序对数。

Analysis

我们用$f[i][j](i \neq j)$，表示第i个数比第j个数大的概率。最后的期望就是$\sum_{i=1}^{n} \sum_{j=i+1}^{n} f[i][j]$。每次交换(a,b)，我们就对相应的$f$进行修改。用$f'$表示操作前的$f$数组。则对于$(i \neq a, i \neq b)$，有$f[i][a] = f[i][b] = (f'[i][a] + f'[i][b])/2$，$f[a][i] = f[b][i] = (f'[a][i] + f'[b][i])/2$，$f[a][b] = f[b][a] = (f'[a][b] + f'[b][a])/2$。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

8.5 273D. Dima and Figure

Description

有一个n*m的棋盘，每个格子可以染上白色或者黑色，你要将棋盘染成规定的样子，并统计这种样子的方案数。1、至少有一个黑色格子；2、黑色格子形成一个联通块；3、对于任意两个黑色格子，只允许在黑色格子中通行，他们之间的最短路要等于他们之间的曼哈顿距离。

Analysis

可以发现第3个条件的意思就是要这个黑色联通块成为一个正交凸多边形。也就是说如果这个联通块覆盖了$u \sim d$行，每一行为$l_i \sim r_i$，那么l_i就要满足能够找到一个$t, u \le t \le d$，使得$l_i \ge l_j, i \le j \le t; l_i \le l_j, t \le i \le j$。同理，那么$r_i$就要满足能够找到一个$t, u \le t \le d$，使得$r_i \le r_j, i \le j \le t; r_i \ge r_j, t \le i \le j$。

我们用$f[i][j][k][jj][kk]$表示当前到第i行，覆盖了$j \sim k$列，且l_i是否处在下降，r_i是否处在上升。转移的时候用前缀和优化即可做到转移的复杂度为$O(1)$。

时间复杂度：$O(nm^2)$

空间复杂度：$O(nm^2)$

8.6 280D. k-Maximum Subsequence Sum

Description

给定一个n个数的序列，每次可以修改一个数，还可以查询一个区间$[l, r]$里面最大至多k个子段的和，即最大化$(x_1, y_1)..(x_t, y_t)(l \le x_1 \le y_1 < x_2 ... < x_t \le y_t \le r, t \le k)\ a_{x_1} + ...a_{y_1} + ... + a_{x_t} + ...a_{y_t}$。

Analysis

对于一个区间里面的最大子段和，我们可以用线段树来维护，然而如果是最大化至多k个子段的和，我们可以先求出区间内最大的子段和，然后将其中的数乘以-1，再接着求最大子段和，这样求k次，就是我们要求的答案了。

因为要区间取相反数，所以我们不仅要维护最大子段和的相关信息，还要维护最小子段和的相关信息。

时间复杂度：$O(mk \log n)$

空间复杂度：$O(nk)$

8.7 305D. Olya and Graph

Description

给定一个含n个点的有向图，保证所有有向边(u, v)满足$u < v$。你要加上若干条（也可能为0）有向边，使得：1、从第i个点可以到达点$i + 1..n$。2、每对点之间最多只有一条有向边。3、所有有向边(u, v)满足$u < v$。4、对于u,v$(u < v)$之间的最短路l，如果$v - u \le k$,则$l = v - u$；如果$v - u > k$,则$l = v - u$或者$l = v - u - k$，k是输入中给出的。问加边的方案数。

Analysis

我们发现如果原有边$(u,v), u < v+1$，那么u,v之间的最短路就是1，也就是说$v-u-k=1$，如果这个条件不成立，那么无解。接着如果对于点u,u+1，他们之间的最短路肯定为$1k \geq 1$，所以最后$(u,u+1)$这条边一定存在。

对于$u,v,(v-u>k)$，如果不存在有向边(a,b)满足$b-a=k+1, u \leq a < b \leq v$，那么$dis[u][v]=v-u$。如果存在有向边$(a,b)$满足$b-a=k+1, u \leq a < b \leq v$，并且不存在有向边$(a,b)$，$(c,d)$，满足$b-a=k+1, d-c=k+1, u \leq a < b \leq c < d \leq v$，那么$dis[u][v]=v-u-k$。如果存在有向边$(a,b)$，$(c,d)$，满足$b-a=k+1, d-c=k+1, u \leq a < b \leq c < d \leq v$，那么$dis[u][v]<v-u-k$，并且此时无解。

我们枚举第一条$(u,u+k+1)$出现的位置，然后就可以求出所有的方案数了。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

8.8 253E. Printer

Description

有n个任务，每个任务有t_i表示到达时间，s_i表示任务量，p_i表示优先级。在第T时刻初，会有$(t_i=T)$的若干个任务进入列表，在这个T时刻，机器会选择一个任务列表中优先级最大的任务，完成该任务1个单位的工作量。保证每个任务的优先级大小各不相同。

现在有一个任务的优先级不知道，但是我们知道这个任务的完成时间，你要确定这个任务的优先级，并且输出所有任务的完成时间。保证有解。

Analysis

我们发现，这个任务的优先级只需要在那些t_i+1, t_i-1之中选择即可（还要保证不和其他任务的t_i相等），并且优先级越高他的完成时间不会变得更迟。所以我们二分这个任务的优先级，然后用堆模拟执行任务的过程即可。

时间复杂度：$O(n \log^2 n)$

空间复杂度：$O(n)$

8.9 240F. TorCoder

Description

给定一个长度为n的包含小写字母的字符串，有m个操作，每次操作

为(l_i, r_i)，表示将$s_{l_i}...s_{r_i}$之间的字符子串重新排列成一个字典序最小的回文串，如果不行，则忽略这个操作。问m次操作之后的字符串。

Analysis

我们发现，进行修改后的那段子串一定是形如"$ABC..X..CBA$"这样的。如果我们用线段树记录每个叶子区间的每个字符的个数，然后只需要区间覆盖就可以维护每次操作了。

时间复杂度：$O(m\alpha \log n)$

空间复杂度：$O(n\alpha)$

9 Volume.9

9.1 306D. Polygon

Description

求一个凸n边形，使得这个n边形的内角都相等，但是n条边长两两不同。

Analysis

我们先搞出一个正n边形，然后随机调整，对于每一条边，我们随机向外或向里平移一段距离，这样可以保证角度不改变。我们重复这个过程，直到满足条件为止。

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

9.2 342D. Xenia and Dominoes

Description

有一个$3 * n$的棋盘，棋盘上有若干个障碍格子，和一个圈格子，其余的都是可以放置的格子。要在上面覆盖$1 * 2$或者$2 * 1$的骨牌，只能覆盖在可以放置的格子上，并且不能重叠。一个覆盖方案是好的还要满足所有的可以放置的格子都被覆盖，并且至少有一个骨牌可以移一格进入圈格子。

Analysis

我们用$f[i][st1][st2][0..1]$表示到第i列，当前列的状态为st2,上一列的状态为st1,当前是否有骨牌可以移一格进入圈格子。每次只需要枚举骨牌是

否覆盖$(1,i-2),(1,i-1)$;$(2,i-2),(2,i-1)$;$(3,i-2),(3,i-1)$;$(1,i-2),(2,i-2)$;$(2,i-2),(3,i-2)$即可。

时间复杂度：$O(n*2^{11})$

空间复杂度：$O(n)$

9.3 254D. Rats

Description

一个$n*m$的棋盘，上面有若干个格子上有老鼠，若干个格子上有障碍。你可以引爆2个手榴弹，每个手榴弹爆炸后会向四个方向蔓延爆炸波，总共蔓延d的距离，障碍格子不会被蔓延。你要炸死所有的老鼠，问应该将手榴弹放在哪里。保证至少有两个格子没有障碍，保证至少有一个格子有老鼠。

Analysis

我们发现，一个手榴弹最多炸死$d^2+(d+1)^2$个老鼠，也就是说如果老鼠个数$R>2*(d^2+(d+1)^2)$，那么就是无解。现在老鼠最多只有$O(d^2)$个，我们发现手榴弹的引爆点只可能在一个恰好炸死某只老鼠的点，这样的点最多有$O(d^3)$个，于是我们计算每个这样的点能够覆盖哪些老鼠，将其放入Bitset里面，然后$O(d^6)$枚举判断。

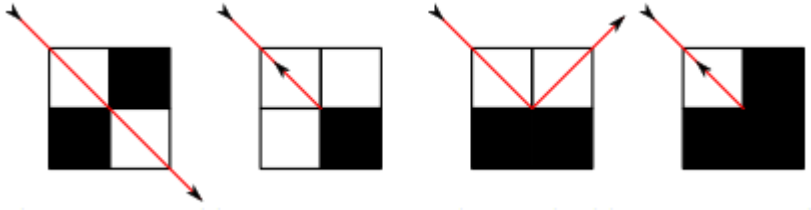
时间复杂度：$O(d^8/w)$

空间复杂度：$O(d^3)$

9.4 274E. Mirror Room

Description

有一个$n*m$的棋盘，k个格子上有障碍。我们从一个指定点(x,y)的中心向两条对角线的4个方向射出光线，碰到障碍就会反射，规则如下：



现在问最后有几个格子被光线经过过。

Analysis

从(x,y)射出的光线有两种情况，1、经过若干次反射之后沿着相同对角线的相反方向射入，这样的话就形成了一个封闭的多边形，可以证明每

个点都只经过一次，统计即可。2、经过若干次反射之后沿着相同对角线的相同方向射入，这种情况一定在最后进行了第2种和第4种反射，然后经过了之前经过的点总共2次，并且在到达(x,y)之后还要再计算向沿着相反方向射出的经过的格子，这个和上述的计算是一样的，并且只可能是再次沿相同方向再次射入(x,y)。

光线的传播用set模拟即可。

时间复杂度：$O(k\log k)$

空间复杂度：$O(k)$

9.5 266D. BerDonalds

Description

你有一张n个点的无向图，你要在其中选择一个点（可以在边上），使得n个点到这个点的最短路中的最大值最小。

Analysis

我们可以Floyd求出任意两点之间的最短路。如果要求的点在n个点之中，我们枚举那个点，然后求出那个最小的最大值。如果要求的点在某条边上，我们枚举这条边，设为(x,y,z)，事实上我们要求一个L，使得$Max=max\{min(dis[i][x]+L,dis[i][y]+z-L)\}$最小。我们发现最优情况下一定有某2个点$i,j$，使得$dis[i][x]+L=dis[j][x]+z-L=Max$。我们将某个点搞成一个二元组$(dis[i][x],dis[i][y])$，然后将其按第一第二关键字从小到大排序。我们枚举i，然后在i之后的一定是以$dis[j][x]+z-L$的形式出现，我们用$dis[i][x]$ 和$max\{dis[j][x]|j>i\}$来计算L，再进一步计算答案。

时间复杂度：$O(n^3\log n)$

空间复杂度：$O(n^2)$

9.6 266E. More Queries to Array...

Description

给你一个n个数的序列，你要支持操作：1、将$a_l,a_{l+1}...a_r$的数赋为x；询问：2、计算$\sum_{i=l}^{r}a_i*(i-l+1)^k$。

Analysis

我们用线段树来解决这个问题，如果线段树的节点（设为$[l,r]$）记录的是对于$\sum_{i=l}^{r}a_i*(i-l+1)^k$。对于操作1，我们只需要用懒标记延迟更新即可。对于询问，我们只需要支持将两个节点合并即可，假

设我们有$[l..mid],[mid+1..r]$的两个节点，我们知道了$\sum_{i=l}^{mid} a_i*(i-l+1)^k$和$\sum_{i=mid+1}^{r} a_i*(i-mid)^k$。我们只需要计算$\sum_{i=mid+1}^{r} a_i*(i-l+1)^k$，也就是$\sum_{i=mid+1}^{r} a_i*(i-mid+mid-l+1)^k=\sum_{i=mid+1}^{r} a_i*[(i-mid)^k*C_k^0+..+(mid-l+1)^k*C_k^k]$，要求这个东西，我们可以利用已有的信息，并且预处理组合数即可。

时间复杂度：$O(mk^2\log n)$

空间复杂度：$O(nk)$

9.7 338D. GCD Table

Description

一个n*m的棋盘，第i行第j列的格子上的数字是$gcd(i,j)$。有一个长为k的序列，要求这个序列又没有在这个棋盘中出现，即存在$i,j,(1\le i\le n,1\le j\le m-k+1)$满足$gcd(i,j+l-1)=a_l,(1\le l\le k)$。

Analysis

我们可以发现，如果有解，那么$i=lcm(a_1,a_2...a_k)$一定可行。首先i要满足$a_l|i$，那么i必定是$lcm(a_1,a_2...a_k)$的倍数，如果$I=i*x,x>1,i=lcm(a_1,a_2...a_k)$，那么$gcd(i,j)$和$gcd(i*x,j)$的区别就是对于某个j,$gcd(i*x,j)>gcd(i,j)$，而$a_l|gcd(i,j)$但是不一定有$a_l|gcd(i*x,j)$，也就是说$I=i$成立的$I=i*x$不一定成立。

接着，对于j，我们发现一个必要条件就是$j+l-1\equiv 0(mod\ a_l)$。我们可以用中国剩余定理解出$j\equiv j_0(mod\ lcm(a_1,a_2...a_k))$。并且那个目标点$(I,J)$的J一定是j的倍数，而因为$gcd(I,j+k*I)=gcd(I,j)$，那么只需要从$(i,j)$开始判断k个，即可知道是否有解。

时间复杂度：$O(k\log k)$

空间复杂度：$O(k)$

9.8 338E. Optimize!

Description

有一个长度为n的数组a_i和长度为len的数组$b_i,(len\le n)$。你要求有多少个i，使得能找到一个长度为len的排列p_i，对于$1\le j\le len$有$a_{i+j-1}+b_{p_j}\ge H$。

Analysis

事实上找到这么个排列的意思就是将长度为len的那个a子数组从小到大排序，b从大到小排序，使得所有$a_i+b_i\ge H$。由于b数组不会改变，我

们可以稍微改一下这个的形式，令$b_i = H - b_i$，将长度为len的那个a子数组从小到大排序，b从小到大排序，使得所有$a_i \geq b_i$。我们发现如果将这两个数组放在一起排序之后，如果将a_i视为-1，b_i视为1，那么只要前缀和的最小值不小于0，即说明满足条件。这个过程利用权值线段树即可维护。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

9.9 256D. Liars and Serge

Description

有n个人，他们要回答有几个人在说谎，他们的答案用一个$a_1..a_n$的数组表示。你要求有多少种a_i，使得能够看出恰好有K个人在说谎。

Analysis

如果我们用cnt_i表示说有i个人在说谎的人数，那么如果$i \neq cnt_i$则说明这i个人在说谎。我们用$f[i][j][k]$表示当前考虑到说有i个人说谎的人，当前已经有j个人，并且有k个人显然说谎，$f[i+1][j+x][k+x] += C_{j+x}^{x} * f[i][j][k], (x \neq i); f[i+1][j+x][k] += C_{j+x}^{x} * f[i][j][k], x = i$。但是这样是$O(n^4)$的，由于题目保证n只可能是2的幂次，我们只需要打表即可。

时间复杂度：$O(1)$

空间复杂度：$O(k \log n)$

10 Volume.10

10.1 268D. Wall Bars

Description

有1根杆子，在离地1到n的距离上分别有插口，你可以插入向某个方向（一共有4个方向）的杆子。一个插入方式是合法的，当且仅当某个方向的杆子两两间距离不超过h，并且第一根杆子的高度在$1..h$，最后一根为$n-h+1..n$。

Analysis

我们可以想到一个$O(n * h^4)$的dp，$f[i][p1][p2][p3][p4]$表示到离地第i个插口，之前第i个方向的距离当前的距离是pi。然后枚举插在哪个方向，然后更新即可。然后我们发现，事实上我们可以这样定义$f[i][p1][p2][p3][p4 = 0..1]$表示到离地第i个插口，之前第i个方向的距离当前的距离是pi，最近插的一个方向的杆子为p4，p4表示最近插的杆子是否在之前满足条件。

时间复杂度：$O(n*h^3)$
空间复杂度：$O(n*h^3)$

10.2 301C. Yaroslav and Algorithm

Description

有一个算法，它的输入是一个字符串S，算法由若干条命令m_i组成，每个m_i的形式是$s_i >> w_i$或者$s_i <> w_i$，s_i, w_i可以包含0..9,?。执行的时候：1、找到最小的那个下标i，满足s_i是S的子串，2、将那个子串替换为w_i。3、如果那个m_i的形式是$s_i >> w_i$，那么再次执行1，若m_i的形式是$s_i <> w_i$，则算法结束。你的算法要针对所有的数字串S，输出的是S的十进制表示加1后的结果。

Analysis

算法的大概思想就是在S的开头插入"??"，然后将其不停地移到结尾，然后结尾不停进位。如果是0..8，那么进位结束，如果是9，那么?指针继续向左移，如果移到最左边，那么在开头插入1算法结束。

时间复杂度：$O(1)$
空间复杂度：$O(1)$

10.3 238D. Tape Programming

Description

有一种利用数字和<,>实现的程序。程序运行时有一个指针。最开始指针的指向最左字符，移动方向为向右。重复以下操作直到指针指向串外：1、如果指针指的位置是一个数字，输出这个数字，然后将指针沿着原来移动方向移动，同时将原来的数字减一。如果原来的数字为0则删除这个数字，串的长度减一；2、如果指针指的位置是<或>，那么指针的移动方向对应得改为向左或向右（与符号的尖角方向相同），接着指针沿着新的移动方向移动。如果新的位置也是<或>，则删除原来的<或>字符。

有一个由n个由<,>和数字构成的串S，你需要回答q个询问。每个询问会给你两个数l,r，如果把看出一个单独的程序$s_l, s_{l+1}..s_r$，问你每个数字会被输出多少次。

Analysis

我们发现如果一直对整个串运行程序，然后直到没有剩下数字。我们记录每次到达的位置和上面的数字，对于某个询问l, r，真正运行的程序就

是第一次进入区间$[l,r]$，直到之后移出区间$[l,r]$。对于m个询问离线求解即可，然后同时用set维护。

时间复杂度：$O((n+q)\log n+nD)D=10$

空间复杂度：$O(nD)$

10.4 238E. Meeting Her

Description

有一个n个点的有向图，你要从起点A到终点B。有k辆公交车，每辆车的起点和终点为s_i,t_i，每辆车会随机选择s_i到t_i的最短路行驶。你可以在一个有公交车经过的点上下车，问最差情况下至少要乘多少辆不同的公交车，或者输出无解。

Analysis

我们先用Floyd求出两两之间的最短路，对于每辆车，我们预处理$Must[i][j]$ 和 $May[i][j]$，分别表示第i辆车是否必定和可能经过j号点。并且，我们发现你在途中的状态可以表示为一个二元组(u,x)，表示乘着x号车来到u号点。转移就是：1、如果u是终点，那么$f[u][x]=1$；2、如果继续乘x，设x接下去可能到达的点为$v_1..v_t$，那么$f[u][x]=max\{f[v_i][x]\}$；3、如果要换车，设有车$y_1..y_t$必定经过u，那么$f[u][x]=min\{f[u][y_i]+1\}$。我们从$f[B][x](Must[x][B]=1)$出发，不断拓展状态直到$f[A][]$为止。

时间复杂度：$O(n^2k^2)$

空间复杂度：$O(n^2)$

10.5 293E. Close Vertices

Description

有一棵n个点的树，树上的边有边权，两个点之间的距离表示为这两个点之间的边数，两个点之间的重量表示为这两个点之间的边长之和。问有多少个点对$(u,v),u<v$，满足距离小于等于L，重量小于等于W。

Analysis

我们用点剖解决这个问题，剩下的就是两条边之间的合并。我们将其按重量排序，然后距离就可以用树状数组统计了。

时间复杂度：$O(n\log^2 n)$

空间复杂度：$O(n)$

10.6 316E3. Summer Homework

Description

给你一个序列a_i，你需要支持两个操作:1、将a_x赋值为y；2、将$a_l, a_{l+1}..a_r$都加上d。询问：$\sum_{i=0}^{r-l} f_i * a_{l+i}$，其中$f_0 = f_1 = 1, f_i = f_{i-1} + f_{i-2}(i > 1)$。

Analysis

我们可以用线段树解决这个问题，如果能够将两个叶子节点合并，那么就可以这个问题就迎刃而解了。我们发现$f_i * a_{l+i}$事实上就是以$f_0 = f_1 = a_{l+i}$为开始的斐波那契数列，然而对于这种斐波那契数列，如果要求g_{l+r}，而当前知道了(g_l, g_{l-1})，通过f_r, f_{r-1}, f_{r-2}就可以推出(g_{l+r}, g_{l+r-1})。而对于这种斐波那契数列，只要知道了h_i, h_{i-1},g_i, g_{i-1}，如果要求$h_k + g_k$，我们可以将两个数列合并为$h_i + g_i, h_{i-1} + g_{i-1}$，然后通过f（斐波那契数列）转移即可。

时间复杂度：$O(m \log n)$

空间复杂度：$O(n)$

10.7 274C. The Last Hole!

Description

平面上有n个圆。最开始所有圆半径都为0，然后所有圆同时开始变大，在时刻$t(t > 0)$所有圆的半径都为t。我们可以想象成一些黑色的实心圆放在一个无穷大的白色平面上，每个时刻都会存在一些黑色和白色的联通快。我们定义一个白色的封闭区域为一个“洞”，问最后一个洞消失的时刻。

Analysis

可以发现，那些可能成为最后的点（洞）的点只可能是一个由3个圆心构成的锐角三角形的外心和一个由4个圆心构成的矩形的中心。因为所有的凸多边(> 4)形都可以分割成一些三角形。我们只需要求出所有可能成为答案的点，然后$O(n)$判断是否在最后被覆盖到即可。

时间复杂度：$O(n^4)$

空间复杂度：$O(n^3)$

10.8 GCJ Final10 C.Candy Store

Description

从$1..C$中选m个数字（允许重复），对于任意k个$1..C$中的数字（设

第i个为 a_i），均能将这m个数字划分成$k+1$份，第i份的和为a_i，第k+1份不做任何要求。求m的最小值。

Analysis

我们先贪心地构造一个可行解。假设我们现在有了若干个盒子，设总和为N，并且只要k个顾客的需求总和不超过N，这些盒子都能满足。我们再加上一个条件：所有的盒子的单个大小不超过$X = \lfloor \frac{N}{k} \rfloor + 1$。现在我们再加入一个盒子，大小为$X = \lfloor \frac{N}{k} \rfloor + 1$。对于所有的要求$Y(N < Y \leq N + X)$，必定有一个要求大于等于X，我们只需要在满足这个要求的时候使用那个大小为X的盒子，剩下的就可以解决了，因为$Y - X \leq N$。

下面证明这个贪心已经做到最优了。利用数学归纳法，首先我们必须选择k个大小为1的盒子，接着是一个大小为2的，这都和之前的贪心一样。然后我们假设总和为N时满足最优性，接下来如果我们选择的盒子大于$Y > X = \lfloor \frac{N}{k} \rfloor + 1$。如果所有的顾客的要求都是$X$，那么Y不可能被使用到，并且$X * k > N$，剩下的盒子不够用了，所以这不满足条件。如果选择的盒子$Y < X$，那么这样子肯定不会比选择X更优，所以下一步必定选择大小为X的盒子。

时间复杂度：$O(Tk \log C)$

空间复杂度：$O(1)$

10.9 267C. Berland Traffic

Description

一张有向图，每条有向边有一个上限c_i。每条边的实际流量C的范围为$[-c_i, c_i]$。并且这张图满足除了起点1和终点n之外的点流量平衡，并且对于两点x, y之间的任何一条路径的流量大小相等。求1到n的最大的流量，并输出方案。

Analysis

我们可以将整张图看成一张电路图，这是因为每个非起点非重点的节点的出入流相等，并且对于两点x, y之间的任何一条路径的流量大小相等，这就相当于把n个点看成了不同的电势，然后两点之间的不同路径的流量相等。我们只需要利用基尔霍夫方程组来解出每个点之间的电势差（如果把1到n的电势差设为1的话）。要使1到n的电势差最大，必定是中间的某条导线达到满流，我们只需要找到那条导线，然后用$R = U/I$求出单位的电阻，那么就可以求出每条导线的流量和总流量了。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

11 Volume.11

11.1 341E. Candies Game

Description

有n个编号从1到n的箱子，每个箱子有a_i的糖果在里面。这个游戏的目的是将所有的糖果放入恰好2个箱子里。每一次可以选择两个有糖果的箱子i和j。不妨假设$a_i \leq a_j$。然后从箱子j拿a_i颗糖果放入箱子i中。判断是否有解，如果有解输出方案。

Analysis

事实上，如果有糖果的箱子大于等于2个，那么必定有解。首先对于三个箱子(A, B, C)不妨设$(A > 0, B > 0, C > 0, A \leq B \leq C)$。我们总是可以在仅仅考虑这3个箱子的情况下构造使得B箱子为空。

令$q = \lfloor \frac{B}{A} \rfloor$。我们来从右到左看q的二进制位上的数，如果为1，那么就执行操作(A, B)，即$A = A * 2, B = B - A$；反之执行(A, C)，即$A = A * 2, C = C - A$。如果A能够一直增长下去，即(A, C)操作如果一直能够合法，那么B一定会变成0。因为我们有$C \geq B$，并且C所减少的数必定小于B所减少的，所以C一定不会变为0，即一定可以维持A的增长。所以这样的构造是可行的，我们只需要每次选择3个非空的箱子，直到不存在2个以上的非空箱子即可，这样的步数不超过$n \log a$。

时间复杂度：$O(n \log a)$

空间复杂度：$O(n)$

11.2 GCJ Final09 D.Wi-fi Towers

Description

平面上有n座塔。每个塔x可以将数据发送给附近的塔y，只要两座塔之间的距离小于或等于塔x的发射范围。每座塔可以选择升级，如果x升级了，那么所有接受x数据的塔y也必须升级。不同的塔升级会带来不同的收益或者亏损。选择一种升级的方案使总分最大。

Analysis

这道题目是经典的最小割求解最大权闭合子图问题。我们如果塔i升级后盈利x，我们让S向其连边x，反之让i向T连边x。对于塔i和j，如果选

了i必定要选j，那么让i向j连∞ 的边。我们的目的是用最小割求出最小的“损失”+“得不到的盈利”。

时间复杂度：$O(VE^2)$

空间复杂度：$O(E)$

11.3 269D. Maximum Waterfall

Description

一堵墙的高度是t，有n块水平板嵌在上面。每块水平板可以看作一条水平的线段，高度为h_i，从l_i到r_i。墙的顶端可以看作一块在$(-10^9,t)$到$(10^9,t)$间的水平板。墙的底端可以看作一块在$(-10^9,0)$到$(10^9,0)$间的水平板。不会有两块水平板之间有公共点。瀑布中的水可以从第i块板流向第j块板当且仅当：

1. $max(l_i,l_j)<min(r_i,r_j)$
2. $h_j<h_i$
3. 不存在$k(h_j<h_k<h_i)$，且(i,k)与(k,j)均满足以上两个条件。
4.从i到j的流量为$min(r_i,r_j)-max(l_i,l_j)$，

在瀑布中，水会沿着一条单向的路径从顶部流到底部。整个瀑布的水流量被定义为水流路径上每连续的两块水平板间的水平相交部分长度的最小值。请找到可能的最大水流。

Analysis

我们用$f[i]$表示以i为结尾的最大流量，如果j满足$h_j<h_i$且能够从j流到i，那么$f[i]=max(f[i],min(f[j],flow(j,i)))$，其中$flow(j,i)=min(r_i,r_j)-max(l_i,l_j)$。现在我们面临的问题就是判断那些j能够流到i。

我们用一条扫描线，到达h_i的时候我们记录从(h_i,x)向上看去的(h_j,x)中，视线第一个到达的水平板j。这条扫描线显然是若干条不同的线段。如果当前要处理线段I，对于在扫描线中的所有线段K，如果I和K有交集，我们就把K分成两份，使得刚好I和扫描线中的线段没有交集（只有一个点有交）。然后对于那些在线段I$[l_i,r_i]$ 中的线段J$[L,R]$，在4 种情况j 能够到i：1、I中只有J；2、J是完整的，即$L=l_j,R=r_j$；3、J在I 的最左端，并且J 的右端完整，即$L=l_i,R=r_j$；4、J在I 的最右端，并且J的左端完整，即$R=r_i,L=l_j$。

处理完之后我们将在I线段中的线段全部覆盖为线段I，并继续处理下一条线段。

时间复杂度：$O(n\log n)$

空间复杂度：$O(n)$

11.4 257E. Greedy Elevator

Description

有一幢m层的办公楼，有一个电梯控制系统，工作方式如下：所有办公楼层用1..m依次编号。在$t = 0$的时刻，电梯位于第1层。接着，在时刻$ti(ti > 0)$会有人来到某楼层等待电梯。对于每一个人我们知道三个参数：他开始等电梯的时刻，他初始位于哪个楼层以及他想去哪个楼层。

电梯的移动方式如下：在一个时刻$t(t \geq 0)$电梯肯定停在某一个楼层。电梯中所有想去这个楼层的人会出电梯，然后所有在这个楼层等待的人会进入电梯。然后，电梯需要决定向哪个方向开动：

1.如果电梯已经是空的而且整幢楼中都没有人在等电梯，那么电梯在$t+1$时刻仍然会保持在那一楼层

2.否则，如果电梯在楼层x，我们定义up表示电梯中要到编号比x大的楼层的人和当前时刻t在编号比x大的楼层等电梯的人的总数，$down$表示电梯中要到编号比x小的楼层的人和当前时刻t在编号比x小的楼层等电梯的人的总数。如果$up \geq down$，那么电梯在时刻$t + 1$会到楼层$x + 1$，否则电梯会到楼层$x - 1$。

求出每个人到达目标楼层的时间。

Analysis

我们用一个堆来维护事件的发生。假设当前时间为t，电梯在楼层f，我们先处理当前在电梯内要出去的人和在电梯外想进来的人，并且维护up和down的值。接下来就是模拟时间的流逝，在时间流逝的时候会有新事件发生，下一个发生的事件有两种可能：1、电梯向上或者向下的到达了有人想要进来或者有人想要出去的楼层，在这个过程维护up和down。2、电梯的移动中没有到达有事件的楼层，而在下个时刻有一个人进入等待的序列，那么也许需要维护up和down。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

11.5 235D. Graph Game

Description

如果点分治的过程变成这样：让我们定义变量totalCost，初始totalCost=0。然后调用solve(T)

1.$totalCost = totalCost + (sizeofT)$

2.在图T中随机选择一个结点x

3.从图T中删除结点x

4.然后T变成了一些联通块
5.分治处理所有的Solve(S)(S是剩下的连通块)

求出对于一个n个结点n条边的连通图，这个算法中变量totalCost的数学期望。

Analysis

我们先考虑是树的情况，我们可以计算两个点之间的期望贡献，然后将它们加起来就是期望了。两个无序点对(A,B)的期望可以表示为选择A为x点（删除点）此时B还与A联通的期望。如果A到B的路径上的点为X，总点数为n，这个期望就是$1/X$。在A和B一开始还联通的情况下，如果选择的点x在A和B的路径上，那么只有$1/n$的期望贡献，如果不在他们的路径上，那么贡献就是$\frac{n-x}{n}F$，综上就是$F = 1/n + \frac{n-X}{n}F$，解得$F = 1/X$。

那么我们回到环加外向树的情况，如果A到B的路径上没有环，那么就归约到树的情况，如果A到B的路径上经过环，设A到环上的第一个点C的路径上的点个数+B到环上的第一个点D的路径上点个数=X，C到D的第一条路径长度为Y，第二条路径上的长度为Z，总点数为n。如果删的点为A，那么贡献就是$1/n$；如果删的点在X个点之中不包括A，那么贡献为0；如果删的点在Y上不包括C,D点，那么就又形成了树，贡献为$\frac{Y-1}{n(X+Z-1)}$；同理，如果删的点在Z上不包括C,D，那么贡献就是$\frac{Z-1}{n(X+Y-1)}$；其他情况下，贡献就是$\frac{n-X-Y-Z+2}{n}F$，综上$\frac{1}{n}+\frac{Y-1}{n(X+Z-1)}+\frac{Z-1}{n(X+Y-1)}+\frac{n-X-Y-Z+2}{n}F = F$，解出F，发现其也与n无关。

我们只要预处理X,Y,Z的信息即可计算期望。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

11.6 285E. Positions in Permutations

Description

P_i是$1..n$的一个排列。我们称排列中的第i个位置是完美的，当且仅当$|P_i - i| = 1$。请求出长度为n的而且完美的位置数刚好为k的排列数是多少。

Analysis

我们用$f[i][j][st]$表示当前考虑到i这个数，已经有j个数在完美位置上，当前的第$i-1,i,i+1$位的状态为st的方案数。我们发现这样转移完之后，设$F[j] = (n-j)! * \sum f[n][j][st]$，$F[k]$并不是我们要求的答案，因为这求的是完美位置至少为k的排列数。我们还需要去重，首先$F[n]$必定是对的，

假设当前我们的$F[j], j > i$都已去重完成，$F[i]$ 中重复计算$F[j]$的次数就是C_j^i。只要将所有的F去重完成，$F[k]$就是所求了。

时间复杂度：$O(nk2^3)$

空间复杂度：$O(nk2^3)$

11.7 273E. Dima and Game

Description

你可以在纸上写下n对整数$(l[i], r[i])(1 \le l[i] < r[i] \le p)$。然后玩家轮流进行操作。

1.选择第i对数，满足$r[i] - l[i] > 2$;

2.将第i对数替换为$(l[i] + \lfloor \frac{r[i]-l[i]}{3} \rfloor, l[i] + 2 * \lfloor \frac{r[i]-l[i]}{3} \rfloor)$或者$(l[i], r[i] - \lfloor \frac{r[i]-l[i]}{3} \rfloor)$。

不能进行操作的玩家则输。计算有多少种写法使得先手必胜。

Analysis

首先我们发现一对数(L, R)的sg函数之和它的长度$R - L$有关，接着我们发现他的sg函数只有$0, 1, 2$三种，然后我们打表就可以发现10^9的sg函数分成了若干段。我们可以打表出所有的段数，然后记录在$L \le P, R \le P$的情况下的sg分别为$0, 1, 2$的单对写法，最后我们dp统计即可。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

11.8 251D. Two Sets

Description

有一个由n个非负整数a_i组成的集合。你可以将集合分成两份，允许有份为空。我们用x1表示第一个集合的xor值，用x2表示第二个集合的xor值。要使得x1+x2尽可能地大。假如有多种划分集合的方法使得集合满足上述条件，让x1尽可能地小。

Analysis

我们设$X = a_1 \oplus a_2 ... \oplus a_n$。对于X的二进制位上为1的情况，我们发现无论怎么分组，这一位i对$x1 + x2$的贡献必定为2^i；对于第i位为0的情况，它对答案的贡献可以为2^{i+1}，也可以为0。

我们暂且不考虑前一个情况，优先从大到小满足使得第i位（为0）对答案的贡献为2^{i+1}。用x_i表示第i个数是否划分到第二个集合。对于第i位

（X的第i位为0），设那些第i位为1的数为$a_{p_1}, a_{p_2}...a_{p_k}$，我们列出方程$x_{p_1} \oplus x_{p_2}... \oplus x_{p_k} = 1$。考虑完X的所有0所在的位置之后，我们从小到大考虑1，目的是尽可能地减小x1。对于第i位（X的第i位为1），设那些第i位为1的数为$a_{p_1}, a_{p_2}...a_{p_k}$，我们列出方程$x_{p_1} \oplus x_{p_2}... \oplus x_{p_k} = 1$。在求解异或方程组的时候可以用二进制压位来加速求解。

时间复杂度：$O(\frac{n \log^2 a}{w})$

空间复杂度：$O(n \log a)$

11.9 316D3. PE Lesson

Description

体育课上有n个同学，第i个人拿着编号为i的球。有些人的体力允许他最多相互扔1次球，其他人最多可以相互扔2次。问一共有多少种不同的持球方式。两种持球方式不同，当有同学在两种持球方式中拿着的球编号不同。

Analysis

我们先考虑所有球员都最多扔一次球的情况，用$f[i]$表示i个人之间最多有多少持球方式。有$f[i] = f[i-1] + f[i-2] * (i-1)$，因为第i个人可以自己不扔球，也可以从$i-1$个人中找出一个，剩下的$i-2$个人自成一派。

如果有n个人最多扔1次，m个人最多扔2次。我们先考虑那只能扔球1次的n个人，这n个人的总方案数是$f[n]$。如果来了一个最多能扔球2次的人A，我们回过头来看场上的情况，假设有N人，有若干个互相扔球所组成的简单环、单独的一人、相互扔球的两人：设某个简单环中点个数为k，那么A进入这个环中的不同方案数就是k；对于单独的一人，那么A和他扔球也是有1种情况；对于相互扔球的两人(x, y)，可以是先x-A、再y-A，先y-A、再x-A，也一共有两种情况。同理总共有N个人，一共的情况就是N 种；但还有一种情况就是不进入其他人的环中，自己自成一派。综上假设原来N个人的答案是ans_N那么来了一个最多能扔球2次的人之后的答案就是$ans_{N+1} = ans_N * (N+1)$。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

12 Volume.12

12.1 348E. Pilgrims

Description

有n个城镇，它们形成一棵树。这里有m个修道院坐落在m个不同的城镇。每个修道院有一个教徒。每个教徒会选择离他最远的一个修道院。如果有多个，他会把所有的都列入清单。你可以摧毁一个没有修道院所在的城镇，所有教徒都不能经过这个城镇，如果一个教徒的清单上没有修道院可去，那么他就会不开心。你需要求出最多能让几个教徒不开心，以及方案数。

Analysis

将有修道院的城镇称为黑点，反之成为白点。我们先删掉所有的能删除的白点，使得以一个黑点为根的情况下，所有的叶子节点都是黑点。我们求出当前树的直径，设左端点为L，右端点为R，中点为Mid。

对于一个黑点u，如果它“投射”到直径上的点w在Mid的左边，那么我们要找到一个在mid右边的点v，使得v投射到直径上的点尽可能靠左，并且v到w的距离等于R 到w的距离，那么u到v之间的白点删除，就可以让u不开心；同理，对于一个黑点u，如果它“投射”到直径上的点w在Mid的右边，那么我们要找到一个在mid左边的点v，使得v投射到直径上的点尽可能靠右，并且v到w的距离等于L到w的距离，那么u到v之间的白点删除，就可以让u不开心；如果u投射到直径上的点就是Mid，那么u到Mid的白点删除，就可以让u不开心。

我们只要计算出一个点u被线段覆盖了几次，就知道删除这个点能够使多少黑点不开心。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

12.2 USACO Nov12.Balanced Trees

Description

FJ的农场是一棵树，每个点都被标记为“(”或“)”。树上的任意两点之间的路径代表了一个括号串。如果这个括号串满足括号的匹配，称为平衡路径。FJ想知道所有的平衡路径所表示的括号串中，最大的递归深度。递归深度即如果把括号串中的“(”视为1，“)”视为-1，就是这个串的前缀最大值。

Analysis

我们用点剖来解决这个问题，首先设在分治到u这个点的时候，和u相连的孩子分别为$v_1, v_2..v_k$。对于所有u子树中的点i，我们记录两个值L_i和R_i。L_i表示以i为开始，u为结尾的括号串的前缀最小值Min，前缀最大值Max和括号之和sum。R_i表示以i所对应的那个祖先v_t开始，i为结尾的括号串的前缀最小值Min，前缀最大值Max和括号之和sum。我们考虑以i为结束的以j为开始的那个括号串，j需要满足以下条件，i和j所对应的祖先v_i, v_j不能相等，$L_j.Min \geq 0$，$L_j.sum + R_i.sum = 0$，并且要使$max(L_j.Max, -R_i.sum + R_i.Max)$最大。那么我们对于那些括号之和相同的$L_j$，只需保留$L_j.Max$大的就可以了。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

12.3 248E. Piglet's Birthday

Description

有n个架子，第i个架子上有a_i个蜜罐。初始时所有蜜罐都是装满蜜的。q次操作，第i次会将第u_i个架子上的任意k_i个蜜罐中的蜜吃完，并把这些蜜罐都放到第v_i个架子上。问每次操作后，架子上所有蜜罐都被吃完的架子的期望个数是多少。

Analysis

我们用$f[i][j]$表示在第i个架子上有j罐装有蜂蜜的概率。那么每次操作之后只要维护$\sum_{i=1}^{n} f[i][0]$的值就好了。假设有操作u, v, k，并且设我们更新后的数组为$f[u][i]$，更新之前的f数组为f'。用$f'[u][i]$去更新f数组，假设这次操作中拿了j个有蜜的罐子，那么$f[u][i] -= P, f[u][i-j] += P$，其中$P = C_i^j * C_{a_u-i}^{k-j} / C_{a_u}^k * f'[u][i]$。

时间复杂度：$O(qa_i{k_i}^2)$

空间复杂度：$O(qa_i)$

12.4 294D. Shaass and Painter Robot

Description

给你一个n*m的棋盘，一开始都是白色的。上面有一个机器人，一开始位于格子(x, y)上。然后机器人会一直顺着某个方向走下去，每当遇到边界时会遵循光的反射定律改变方向，然后继续走。每当机器人走到一个格子后，它会将这个格子染黑，用掉一个单位颜料。即便这个格子已经被染黑了，也需要耗费一个单位颜料。直到这个n*m的网格已经变成黑白相间

后停止行动。问机器人停下来的时候，耗费了多少颜料，或者指出永远不可能停下来。

Analysis

我们可以发现，这个n*m的网格已经变成黑白相间的时候，可以等价于这个网格的边界上要染黑的点已经被染黑。我们只关注边界上的点的情况，这样的点只有$n+m-2$个，我们模拟机器人的每次行走（从一个边界上的点到另一个边界上的点），直到所有的点都被经过过。并且一个点不可能被经过超过2次，如果走了超过$2(n+m-2)$次还没有停止，那么就说明永远不可能停下来。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

12.5 243D. Cubes

Description

有一座由积木搭成的城市，城市的底座为n*n的正方形，正方形的一组对角落在$(0,0)$和(n,n)。每个单位方格上有若干单位立方体。如果你到了一个几乎是无穷远的地方，以向量$(vx,vy,0)$的方向望向城市。问可以看到多少不同的立方体。我们认为这个立方体是可见的，当且仅当立方体上存在一个点p，从p以向量-v发出的射线上不存在属于其他立方体的点。

Analysis

不失一般性，我们假设$vx \geq 0, vy \geq 0$，对于其他情况我们可以翻转坐标系来归约到这个情况。然后我们发现对于占据$(x,y),(x+1,y),(x,y+1),(x+1,y+1)$的格子，只有$(x,y+1),(x+1,y)$这两个点沿-v方向的射线l,r需要被考虑。每次如果只要查询在l和r之间最矮的建筑高度min，然后如果这做建筑的高度为now，那么就能看到$max(0,now-min)$的立方体。整个操作就是查询一段区间内的最小值，并给一段区间执行$a_i = max(a_i,x)$的操作。这2种操作用线段树即可轻松完成。

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n^2)$

12.6 243C. Colorado Potato Beetle

Description

有一个可认为是无限大的棋盘，一个人一开始在某个格子中，他每次会沿着上下左右走若干格，并在沿途所经过的地方碰上杀虫剂。操作全部

完成后，害虫开始入侵，入侵过程从无限远处开始，每次可以感染四个方向上的没有杀虫剂的格子。问最后有多少格子没有被感染。

Analysis

对于他每次走的路径的起点和终点，设为(x_i, y_i)，我们只需要考虑直线$x = x_i, x = x_i + 1, y = y_i, y = y_i + 1$，就可以了。现在整个棋盘被分成了$O(n^2)$块，我们只需要bfs一遍就可以统计出没有感染的块的大小了。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

12.7 GCJ Final09 B.Min Perimeter

Description

有n个点，问这n个点中任选三个点组成的三角形的最小周长是多少，包括退化的三角形。

Analysis

我们先将n个点按x轴排序，然后分治。每次分治Solve(l,r)表示使用l到r之间的点，最小的周长为多少。在Solve(l,mid)和Solve(mid,r)之后，我们可以得到一个当前的最小值D，那么每次只需要考虑以(x, y)为中心的边长为D的正方形就可以了。可以证明这个正方形内的点数为常数级别的，因为这里面的点如果很多就不满足最小周长为D的条件了。那么我们只要暴力枚举就行了。

时间复杂度：$O(n \log^2 n)$

空间复杂度：$O(n)$

12.8 293D. Ksusha and Square

Description

有一个n个点的凸多边形，每次可以选择2个在凸多边形中的点（包含边界，这两个点不能相同），然后以这两个点的连线为对角线做一个正方形，求这个正方形的期望面积。

Analysis

这个正方形的期望面积可以等同于这两个点连线的长度的平方的期望E，$ans = E/2$。E就等于$E((x_i - x_j)^2 + (y_i + y_j)^2) = E((x_i - x_j)^2) + E((y_i - y_j)^2)$。我们将x和y分开考虑，如果知道了如何求Ex，那么Ey也就同理了。$Ex = E(x_i^2) + E(x_j^2) - 2E(x_i x_j)$。第一部分$E(x_i^2) + E(x_j^2) =$

$2cnt*\sum x_i^2$，其中cnt为这个凸多边形中点的个数。而第二部分$-2E(x_ix_j) = -2((\sum x_i)^2 - \sum x_i^2)$。而对于$\sum x_i^2, \sum x_i, cnt$，我们只需要枚举x，然后在凸多边形的上凸壳和下凸壳处找出这个x的最上面的点和最下面的点，就可以维护了。

时间复杂度：$O(|x_i| \log n)$

空间复杂度：$O(n)$

12.9 331C3. The Great Julya Calendar

Description

你得到一个数字x，你可以得到数字$x - d$，d为x某个数位上的数，重复以上操作，直至x为0。求最少的操作次数。

Analysis

可以发现，对于一个数字x，我们减去的一定是x的数位中最大的数，因为数字小的操作步数一定不差于数字大的，这个可以用数学归纳法证明。用$f[x][y]$表示对于数字x，在x的数位之外还有最大的数位y，在这种情况下的最小操作步数。$g[x][y]$表示在相同情况下，利用最小操作步数之后减完x之后，还要减多少。对于$x < 10$的情况，我们直接减就行了。如果$x \geq 10$，设$t = 10^{\lfloor \log_{10}^{x} \rfloor}$，我们只需要求出$[x\%t][max(x/t, y)]$和$[x - x\%t + g[x\%t][max(x/t, y)]][y]$就可以了，$f[x][y] = f[x\%t][max(x/t, y)] + f[x-x\%t+g[x\%t][max(x/t, y)]][y], g[x][y] = g[x-x\%t+g[x\%t][max(x/t, y)]][y]$。因为状态很少，我们只需要记忆化搜索即可。

时间复杂度：$O(10^3 * \log_{10} x)$

空间复杂度：$O(10^3 * \log_{10} x)$