Contest 5

A

Abs(a[i] - b[j]) * (i - j)，当 a[i] >= b[j]时，有 a[i] * i - j * a[i] - i * b[j] + b[j] * j

枚举每一个 a[i]，用数组记录 b[j] * j，还有 j，以及 b[i]，同时记录小于 a[i]的 b 有多少个，这样可以得到 a[i] * i 出现了多少次，a[i] * j 可以是 a[i] *（比 a[i]小的 j 之和) i * b[j]同理，b[j] * j 同理

B http://115.28.76.232/topic/2983 大数据版题解

C.用 dp[i][j][k][r]表示前 i 个字符，修改了 j 次，有 k 个 A，r 个 AC 的“ACM”数目最大值这样做的用意是，当下一个字符是 A 的时候，ACM 是得不到的，得到的是 A 数目增加，是 C 的时候，AC 数目增加。当下一次字符是 M 的时候，ACM 的数目会增加前面 AC 的数量，即 k，所以一切需要用到的信息都被保存了起来，由于 C(20,2) < 200，所以开一个 dp[21][21][21][20 * 20]的数组即可

D.因为棋盘是 N * N 的，所以旋转 90 度或者 180 度或者 270 度，都是对称的，因此 abs(x1 - x2) < abs(y1 - y2)的数量和 abs(x1 - x2) > abs(y1 - y2)数量是等价的！！因此，我们只需要计算 abs(x1 - x2) = abs(y1 - y2)的数量，有且仅有每条对角线上的满足这个性质，枚举对角线，然后任意取对角线上任意两个点即可，类似于一个 sigma(c(n,2))求和,n 为对角线长度，这个公式的计算，c(n,2) = n * (n - 1) / 2 = 1/ 2 * (n^2 - n)，sigma(n^2) = n * (n + 1) * (2 * n + 1) / 2,sigma(n) = n * (n + 1) / 2，当然，多加一个 c(x,1)可以凑成 c(k,3)的形式。再次提醒一下逆元，(a / b) % c = a * inv(b) % c,其中 inv(b) * b = 1 % c，inv(b)为 b 的乘法逆元。

E.后缀数组排序之后，对于每个查询 L,R，直接 rank[L]得到 s[L...STRLEN(S)]的起始位置。对于排在 rank[L]之前的那些后缀，用 rmq 查询出最左边的与 s[L...STRLEN(S)]的 LCP >= R - L + 1 的位置 x，那么在 x 之前的所有后缀，字典序都是< S[L...R]，对于 x 到 rank[x]这一段，由于 LCP >= R - L + 1,所以它们贡献的长度只是 R - L（因为 R - L + 1 是相等的，要严格小于)，对于 x 之前的后缀什么求呢？用一个前缀和数组即可，sum[i] = sum[i - 1] + (strlen(S0 - sa[i])。之前的步骤恨简单，估计都能想到。 难点是后面的，即字典序>L 的处理。

我们也可以二分出在 rank(L)之后的最后边位置 y，满足 LCP(L,y) >= R - L + 1。那么[L,Y]这一段统一贡献 R - L。在 y 之后的那些，贡献的是其与 s[L,R]的 LCP ！！！！！！

比如

aa

aab

ac

ab

abc

假设 s[L,R] = “aa’，那么 ac,ab,abc 贡献的都是 a，即 lcp。

所以这个可以用一个后缀和来表示 dp[i]表示后缀数组排序后 i 与后面的所有后缀(i + 1,... Strlen(s))的 lcp 之和. 那么 ans += dp[y + 1]即可

Dp 数组是怎么计算的呢？Dp[i] = height[i + 1] * (pos - i) + dp[pos]，pos 是 i 的最右边的满足 lcp(i + 1,pos) >= height[i + 1]的

F.每堆石子最后的数目是石子总个数/总堆数。用 need 表示前 i 组石子已经摆好了，还多出

或者需要从右边拿来 need 个的最小代价。当 need 大于 0 的时候，就是处理完左边前 i 组的时候，多出了 need 个，且已经全部堆放在第 i 组的最右边（注意）。 当 need < 0 的时候，就是处理完左边前 i 组之后，还需要从 i + 1 组的最左边取来 need。
那么对于 need > 0 的处理，如果 i + 1 组的数目 >=avg（平均值)，那么新代价相当于将 need 个石子全部移到第 i + 1 组的最右边(need * 距离) + 将 i + 1 堆的每个石子移动到最右边，类似于 sigma(1 + 2.. + dis)的求和。如果 i + 1 组的数目 < avg，那么就需要将 need 移动补充，补充的规则是，肯定先满足 i + 1 组中第一堆的需求，然后再满足第二堆，又是一个数列求和。当 need 不够的时候，需要从 i + 2 组的最左边拿来（满足 need < 0 的定义），这个先给最左边不符合条件的，然后位置+1，一直到最右边，还是 sigma(i)的求和公式

Contest6
A.费用流，取的不能少于某个值，那么他的反面就是某行某列最多取那么多个数，变成了普通下界为 0，上界为特定值的普通费用流问题了。 总数 - 最大费用最大流即可

B.算出不碰到渣渣的最大概率即可。最短路存两个值，距离&概率，修改一下最短路，在距离相同的前提下，取不碰到渣渣概率最大的更新即可

C 调和级数 1 / 1 + 1 / 2 + ... + 1 / n = ln(n) + 欧拉常数。
所以 n / 1 + n / 2 + ... + n / n = n * (ln(n) + 欧拉常数)
欧拉常数约为 0.5 - 0.6 之间，可以忽略，所以是 nln(n)(ln(n)的意思是自然对数)
对于(a * x + b) / x ，其中(0 <= b < x)，我们有(a * x + b) / x = a
所以对于每个数 x，我们枚举 a，那么[a * x,(a + 1) * x - 1]中的任意数除以 x 的结果都是 a，我们只需要计算出这个区间有多少个数即可，用 sum[x]表示 x 出现了多少次，那么求前缀和(sum[x] += sum[x - 1])表示 1..x 这些数出现的次数，sum[(a + 1) * x - 1] - sum[a * x - 1]即是[a * x,(a + 1) * x - 1]这个范围的数的个数

思考题：本来想这周出的，由于连续六场，不出了
将 a[i] / a[j]改成 a[i] % a[j]，怎么求？

D 六场中最难的一题（？），rej 后无人通过
这题本质是求 gcd(ad + bc,bd) = 1 的 a,c,d <= n 的数目
假设 a 与 b 不互质，那么 bc 与 bd 不互质，且具有 b 这个因子，所以 ad + bc,bd 不互质，这是不可能的，所以 gcd(a,b) = 1。这个直接容斥即可算
同理可以推出 gcd(d,c) = 1，且 gcd(b,d) = 1。
现在难点在于求 gcd(d,c) = 1 且 gcd(b,d) = 1 的数目
定义一个函数 fun(b,n,m)表示 gcd(d,c) = 1，且 gcd(b,d) = 1，(d <= n,c <= m)枚举 d,c 的最大公约数，最大公约数为 1 的时候，利用第 2 场的 GCD SUM 那题 sqrt 来算出。
但是(d,c) = 1 不保证(b,d) = 1，所以枚举 d 具有 b 的某个因子 factor，利用容斥，相当于求子问题 fun(factor,,m,n / factor)，表示现在因为 d 具有了 b

的因子 factor，那么 c 必须与 factor 互质

E.(i ^ j ^ k) = 0，且 i < j < k。最简单的方法，是数位 DP。详见题目加强版
http://acdream.info/onecontest/1080#problem-C
但是对于 3 个数异或方案数或者求和，都是不需要数位 DP 的。
由于异或只与二进制对应位有关，所以，这三个数中，每一位或者 3 个 0，或者 2 个 1（1 个 1 或者 3 个 1，异或为 1）。因为 0 < i < j < k，所以 j,k 二进制最高位是相同的(因为 j,k != 0，所以最高位必为 1，因为只能有 2 个 1，所以相同)。那么 j ^ k 的结果，必然是消掉了最高位（最高位 1 相同，抵消），即 (j ^ k) < j（少了最高位），也就是 i = (j ^ k)必然存在!!!!!!，且 i = (j ^ k) < j。所以，我们只需要枚举最高位，然后从中选出任意两个数就行。
也就是 c(1,2) + c(2,2) + c(4,2) + .. C(x,2) (1 + 2 + 4 + 8 + 16 + .. + x = n)

F.斐波那契数，因为对于 n 来说，要么 n 在原位置，即 f[n - 1]，要么 n 与 n - 1 交换位置 f[n - 2]，所以直接是 f[n] = f[n - 1] + f[n - 2]