A.
递归的过程其实是对等差数列进行操作。
比如一开始公差为 1，然后为 2，然后为 4，因此只需要记录公差，首项，还有总数量，即可得到需要的信息

对于 1 操作，用 solve1(start,d,n,kth)表示找出执行完递归操作的第 kth 项
如果(n + 1) / 2 >= kth，那么被分到左边 solve1(start,d * 2,(n + 1) / 2,kth)
否则，被分到右边 solve1(start + d,d * 2,n / 2,kth - (n + 1) / 2(减掉左边有多少项))

对于 2 操作,solve2(start,d,n,kth)表示 kth 项最后的位置
if((kth - start) % d == 0)，同被分到左边 solve2(start,2 * d,(n + 1) / 2,kth)
否则，分到右边 solve2(start + d,2 * d,n / 2,kth) + (n + 1) / 2

B
问题 1，如何快速求 n / 1 + n / 2 + .. n / n (n <= 10^9)
问题 2，如果允许最暴力的容斥，你会怎么做？

1：对于 n / x，当 x * x <= n 的时候，我们可以暴力枚举 x = 1... sqrt(n)
当 n / x，x * x > n 的时候，n / x 相同的段是 sqrt 级别的。因为一个树 n = p * q,当 p >= q 的时候，q 的取值范围，即从 1 到 n / p <= sqrt(n)，所以对于 n / p = q(p > sqrt(n))的时候 q 进行枚举，就是 sqrt(n)的复杂度

2.暴力容斥，设 f(n,m,p)为 1 <= x <= n,1 <= y <= m 的时候，gcd 是 p 的倍数的有多少对不同的(x,y) f(n,m,p) = (n / p) * (m / p)（因为 gcd 为 p 的倍数，所以可以提取出来)
对数很简单 f[n,m,1) - f(n,m,2) - f(n,m,3) - + f(n,m,5) + f(n,m,6) + f(n,m,10)...

注意到 2，像 4，8，12 这种并没有在容斥中出现，我们定义一个莫比乌斯函数,mobius(n) = 0,如果包含了平方因子，即质数分解后，有某个质数因子出现次数>=2，如 4，8，12
否则 mobius(n) = (-1) ^ k，k 为不同质因子数目，比如 2 的时候，对应于(-1)
借助这个，将 2 的式子表示为统一的式子，sigma(f(n,m,i) * mobius(i)) = sigma((n / i) * (m / i) * mobius(i))
然后借助 1，我们知道, n / i 只有 sqrt 段不同的结果，同理 m / i 也是 sqrt 段，因此，它们的交集也是 sqrt 级别的，
sigma((n / i) * (m / i) * mobius(i))= (n / i) * (m / i) * sigma(mobius(k))，k 中任意一个数满足 n / i = n / k,m / i = m / k

还有求和操作，这里没写出来，定义函数 gi(n,m,i)表示 gcd 是 p 的倍数的时候，左边的和,gj(n,m,i)为右边的和
gi(n,m,i) = i * (1 + 2 + .. + n / i) * (m / i)
gj(n,m,i) = (n / i) * i * (1 + 2 + .. + m / i)，这两个一样和上面 f 一样优化，
sigma(i * (1 + 2 + ... + n / i) * (m / i) * mobius(i))，i * mobius(i)可以合并了，然后 1 + 2 + .. + n / i 的话，由于 n / i 全部相同，然后就等价于上面的式子

C

C 题，直接贪心。就是现实生活中的找零钱，将输入从大到小排序， for(int i = 1; i <= n; i ++) if(m >= a[i]) m -= a[i];

if(!m) YES; else NO

证明：不妨设存在合法方案 m = c1 + c2 + .. ck(c1 <= c2 <= ... <= ck)

假设你现在要用一个更大的数 X 去取代部分数

求出最小的 i，使得 c1 + c2 + ... + ci >= X,c1 + c2 + ... + c[i - 1] < X

尝试去掉 c1，如果 c2 + ... + ci >= X，最后剩下的串为 cj + ... + ci >= X，且 c[j + 1] + .. + ci < X，(否则继续去掉 cj)

如果 cj > X，那么 X 一定不能被替换了。

否则，因为 cj + 1 % cj = 0.. c[i] % c[i - 1] == 0，且 X >= c[j], X % c[j] = 0

设 cj + .. + ci = k1 * cj X = k2 * cj, cj + 1 + .. + ci = k3 * cj

k1 >= k2 > k3，

所以 k2 >= k3 + 1 因为 k1 - k3 == 1，只差一个 cj

所以 k2 = k1，因此可以直接用 X 替换掉 cj + .. + ci

D

DP

先不考虑时间因素，最简单的 DP dp[i][j]表示用了 dp[i][j]个数字，与第一个序列匹配位置为 i，第二个序列匹配位置为 j，枚举下一个数字，dp[i][j]->dp[next[i][k]][next[j][k]]

next[i]]k]表示 k 这个数字在 i 之后最近的出现位置。如果没有的话，设为 n + 1 或 m + 1，代表没有匹配的子序列了。

这样时间复杂度为 n * m * k

优化：

注意到一个事实，最短的没出现过的子序列，肯定长度<= 1 + max(min1(i),min2(i)) min1(i) 表示 i 这个数字在第一个序列中出现的次数，min2(i)同理

这样的话，如果有 2000 个数字，那么长度<=2，如果有 200 个，长度为 10，所以长度为 n / k

现在反向 DP,dp[len][i]表示你找出的序列长度为 len 的时候，在第一个串匹配位置为 i，最远能到达第 2 个序列的位置

这样世间复杂度为 max(n,m) / k * n * k,约为 n^2，next[i][k]需要预处理

思考题：考虑一个非常经典的问题，如果要求序列的 lcs，那么实间是 O(NM)，但是现在告诉你两者的最长子序列长度>=95% * MIN(N,M)，在 N,M <= 10000 的情况下，设置一个不超时的算法

E

这题求的是 sigma(cover(i),i) % 20140717 (i = 1.. n)

cover(i)表示有多少个不同的区间覆盖了 i 这个数

cover(i)很容易求，用树状数组搞一下.

现在问题是求 c(n,m) % 20140717

20140717 不是质数，我们将它分解成质数相乘的形式 41 * 107 * 4591
如果知道 c(n,m) % 41(107,4591)的结果，我们可以用中国剩余定理联合同余方程，求出 c(n,m) % 20140717 的结果
c(n,m) % prime(prime 不大的话)，这个可以用 lucas 定理 c(n,m) % prime = c(n % prime,m % prime) * c(n / prime,m / prime)，然后将 c[n][m]的范围压缩到 n,m < prime 内了。
这题因为 prime 最大为 4591，开个 short int 数组，像杨辉三角那样递推就行了

F
增加一条线段，删除一条线段，给出一条线段，判断有多少条线段与这条线段相交
非常经典的题目了，校门外的树 1-3，有兴趣可以百度搜索下
考虑反面，不相交的情况，线段左端点 ＞R，或者线段右端点 ＜L，两个树状数组维护反面