A ZJH and Monkeys

对一对点(u, v)，显然树上任意一点到u和到v的距离的差的绝对值的奇偶性是一定的，而且和u到v的距离的奇偶性相同。

于是根据k的奇偶性，问题变成了求有多少对点距离为奇/偶。

用DP可以解决，点分治也可以通过。

B Beautiful Trees Cutting

被5整除的整数必然末位为0或5。考虑k=1时,遍历字符串，当在第i个字符出现0或者5，考虑以这个0或者5作为结尾时，方案数为2i-1个，答案就是对所有出现的0或者5求这个方案数的和。

当重复K次时，第i个字符上的0或5在第j次重复中的方案数为ans(i,sj)=2i-1+j个，将重复后的串写为s1s2s3s4…sk，可以观察到ans(i,s1), ans(i,s2), ans(i,s3), …，ans(i,sk), (i=1,2,3…) 为等比数列，用等比数列求和公式加上快速幂和逆元可以求得答案。

C Professional Manager

并查集维护集合大小以及合并集合。

对于删除操作只需要为删除的元素开一个新点即可。

D Trees Transplanting

对于任意三个有球的盒子，必定能在有限步操作里将一个盒子变空。

考虑以下做法：

假设三个非空盒子为i,j,k，令ui<=uj<=uk，

现在有 uj=ui\*⌊uj/ui⌋+uj%ui

令t=⌊uj/ui⌋

考虑t奇偶性：

若t为奇数，即t-1=⌊(uj-ui)/ui⌋是偶数，那么执行j->i，之后t’=⌊(uj-ui)/(ui\*2)⌋=(t-1)/2；

若t为偶数，那么执行k->i，之后t’=⌊uj/(ui\*2)⌋=t/2；

可见，这样循环下去t在的时间内迟早会变成0。

那么t=0意味着什么？

uj=uj%ui，少了的球被分配到了i与k里。而且明显uj%ui<ui，不断递减。那么在个这样的过程之后，会有一个盒子变成了0。那么，去掉这个盒子，加入另外一个盒子，不断处理到只剩两个盒子就完成了。可见步数和复杂度是少于的。

E Seeds

当你取到物品i时，是否放回取决于放回后的状态求得期望与当前物品价值的大小。所以采用DFS来求期望。

对于物品i若其放回次数为0，当取到该物品时，取它的价值，若其放回次数不为0，取到该物品时，取它的价值与放回后期望的较大值作为它的价值，然后求加权平均即为答案。

搜索过程为：

for(int i=0;i<n;i++)

if(p[i]==0) ans+=v[i]/n;

else {

p[i]--;

ans+=max(dfs(),v[i])/n;

p[i]++;

}

然而这样估计要跑个几百年。

由于直接搜索复杂度较大，而可放回次数总数不超过20种，所以我们可以对物品的放回次数取状态记忆化，进行记忆化搜索。

假设可放回次数为a1,a2,a3…0,0…则总共(a1+1)\*(a2+1)\*(a3+1)种状态，然后我们根据a1将总

状态分为a1+1份，然后将每一份分为a2+1小份，接着将a2+1小份每份分为a3+1小份……，这里提供一种具体方案：

假设a1=3,a2=2:

则总共12中状态；

a1等于0时取base=0；

a1等于1时取base=3；

a1等于2时取base=6；

a1等于3时取base=9；

a1等于0时取base+0；

a1等于1时取base+1；

a1等于2时取base+2；

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 00 | 01 | 02 | 10 | 11 | 12 | 20 | 21 | 22 | 30 | 31 | 32 |

则对应的状态如表所示（第二行为a1与a2的取值）。

由于pi的和小于20，我们的状态数最多为220。

然而这样估计还要跑个几百年。

由于有许多物品的pi为零，所以我们可以对这些物品进行合并，假设我们有k个物品pi为零，那么我们可以把这些物品删掉，然后创造一个新的物品，它的取到概率为k/n，价值为删掉物品的价值总和，那么我们的物品总数最多为21个。这样我们的时间复杂度可以降为O(state\*m\*m)其中m为处理完pi为零之后的物品数量,state为状态数。

（由于标程是每次根据pi现有的状态求得，需要耗费O(m)的时间，但其实将状态每次根据p--直接改变可以优化掉一个m使得时间复杂度更优，达到O(state\*m)，有兴趣自己去写一写）。

F Sorting Trees

对于写成k的冒泡排序，相当于每隔k个元素形成的子序列各自排序：

a1,ak+1,a2k+1,…

a2,ak+2,a3k+2,…

然后你可以用别的更快的排序方法排好这些子序列之后再还原，就能直接比较得出答案了。

G Triangular garden

这道题的数据里附有详尽的出题人的心理活动。出题人本来想卡掉double，但是很悲伤失败了。

要想达到完全精度，必须用整数来做计算。

首先是垂心，判断AB与CO垂直，只需求它们的内积是否为0即可

重心，利用公式

进行判断即可。

外心，求|AO| |BO| |CO|是否相等即可，为避免开根号，直接对它们的平方进行比较。

内心最为复杂，需要判断O点到三条边的距离是否相等。利用外积公式求解：

同样，为了避免开根号，直接比较h2即可。此外，该式中有除法，如果比较h12与h22将两侧的分母调换到对侧的分子，所得式的结果会超过long long的范围，需要使用高精度，或在模意义下进行。

H Magnificent Tree

题意大致是给WxW的矩阵，初始时全为0，进行两种操作：

（1）将坐标为(x,y)的元素加上一个数

（2）询问矩阵(x1,y1,x2,y2)内的元素的和

操作的次数N<=105矩阵维度W<=5x104

题目限制空间32MB

很容易考虑到二维树状数组来解，但O(W2)的空间肯定是不行的。动态开点的线段树，空间是O(N\*log2W)，空间会达到100MB左右也不行。

考虑用CDQ分治求解，按照时间对操作分治。

大体思路是对于x轴，把一个区间(l,r)分成两个区间(l,mid)和(mid+1,r)，处理(l,mid)对(mid+1,r)的影响，然后递归处理(l,mid)和(mid+1,r)。对于y轴再用一维数据结构线段树或树状数组去维护

赛中有人反映本题空间过小，std占空间是13m，所以只要用CDQ分治开两倍标程空间也是可以的，毕竟题目本意就是想要卡掉二维数据结构。

I Neat Tree

题意大致是给定n个元素的数组，求其所有子数组的最大值和最小值的差的和。

关键在于考虑到分开算贡献, 可以用单调栈维护最值的延伸区域。

另外，也可以用分治的思想，考虑维护L和R数组，分别表示ai作为最大值左边可以延伸到的位置以及右边可以延伸到的位置。显然可以O(n)维护。最小值同理。

J Various Tree

裸BFS。

K Walking in the Forest

二分答案，判定是否可以分成和不超过mid的少于k组。

L Fresh Air

倒着DFS/BFS。