A. 沙币

$f(x)$是xorshift随机数生成器的迭代函数，意味着$f(x)$的取值随着x递增是到处乱飞的，因此要找到一个前缀是特定y的$f(x)$，只需要从小到大尝试每个x，总共只会尝试$O(2^w)$次

B. 组队

设序列是有序的，$a_1 \geq a_2 \geq \cdots\cdots \geq a_{3N}$，我们可以证明答案是 $a_2 + a_4 + a_6 + \cdots\cdots + a_{2N}$。

设第 i 强的队伍的第二强的成员是 p，则前面 $i-1$ 个的队的实力都大于等于 a_p，即前面的队伍的至少有 $2(i-1)$ 个实力强于 a_p 的成员，加上第 i 强的队中的实力最强的成员，共有 $2i-1$ 个成员的实力大于等于 a_p，故 $a_p \leq a_{2i}$，即第 i 强的队的实力小于等于 a_i，故所有队的实力和最大不会超过 $a_2 + a_4 + a_6 + \cdots\cdots + a_{2N}$。

一种组队方案是 (a_1, a_2, a_{3N})，(a_3, a_4, a_{3N-1})，$\cdots\cdots$。

复杂度为排序的复杂度，$O(Nlog_2^N)$。

C. 染色

最后每个点的颜色应为，最后一次覆盖到该的操作的染色，所以我们考虑倒着处理操作。

倒着处理操作，则每次操作要做的就变成了给能覆盖的到的还没有染色的点进行染色。但每次操作时找能覆盖的到的点的时候还是要经过染过染色的点，复杂度仍没有降低。给每个染过染色的点现在我们打一个标记，设 i 号点的标记为 f_i，标记的意义为 i 号点的半径 f_i 以内的点都已经染过色了，有了这个标记我们在 dfs 进行染色时，到点 i 时，我们比较一下我们现在要染色的半径 d，如果 $d \leq f_i$，因为我们现在是要给没染色过的点进行染色，但 $d \leq f_i$，当前要染色的半径内的点都已经染过色了，所以就不用继续 dfs 下去染色了，如果 $d \geq f_i$，则继续 dfs 下去，然后将 f_i 的值更新成 d。我们考虑一下这样做的复杂度，每个点每次 dfs 到没有停下来，就代表该点的标记会增大，至少增加 1，故每个点最多会被 dfs 到 d_{max} 次，故复杂度是 $O((N+M) * d_{max})$。

D. DDL

理想情况下，所有作业都在DDL那一天完成，这样显然时悲痛值最小的（等于0）。

但是如果某两个作业的DDL时同一天，这个设想就不能实现了。

更一般的说，若有多个作业的DDL是同一天，那么我们只能、被迫地只保留一个作业的DDL不变，而把其他作业的DDL都提前一天，但是这还没有结束——

一来，被提前的这些作业的DDL还是同一天。

二来，被提前的这些作业还可能和其他作业撞DDL。

此外，我们应该保留谁的DDL不变，其他的DDL提前呢？

所以让我们来整理一下思路：

将所有作业按照DDL从后到前排序一次处理。

每次抓取DDL相同的所有作业，保留起始时间最靠后的那个作业的DDL不变——它就在这一天完成，然后把所有的其他作业的DDL提前一天——它们将提前一天或者更多天完成，重复此过程，直到所有作业的的完成时间被确定。若有作业的完成还没被确定，就已经超过了其布置时间，那就输出-1.

用一个大顶堆来实现上述过程，按照DDL的倒序扫描时间轴，堆中维护DDL（被修改后目前）是当前时间的所有作业，关键字为作业的起始时间。

E. 栽树

每次需要交换的是两个矩形，如果暴力交换显然会超时。

首先考虑如果只有一行，那么问题变成在一个序列上交换两个子段，可以用链表维护。

二维的情况下，可以使用十字链表来维护这个结构，每个格子有两个指针，分别指向自己的右边和下边。

这样每次修改的时候，只需要修改矩形左边界和右边界的右指针，以及上边界和下边界的下指针就好。

整个思路和普通的线性链表交换两段区间是一致的，一次修改 $O(n)$。

F. 排列

从上往下依次看，每个横线相当于交换两个相邻的数字。

因此第一个问题，只需要模拟这个交换的过程，每次swap一下就好。

第二个问题可以描述成，如果要通过交换两个相邻的数字，从1-n生成指定的排列，至少需要多少次交换。

先说结论：需要的数量是排列的逆序对数。

证明：从指定的排列生成1-n，和从1-n生成指定的排列是等价的，我们考虑前者。只要不是升序的1-n排列，可以保证每次交换都可以减少一对逆序对，也最多减少一对逆序对。而当逆序对全部消失了，就说明排列变成了1-n。

G. 修复

给定一个长度为N的序列，有M位置为损坏位置，小A的目标是把每个损坏位置精确定位，他精确定位损坏位置的手段是：询问一个位置x，系统会告诉他下标小于等于x的位置是否还有没有被精确定位的损坏位置： 回答yes,说明前面还有（一个或多个）未精确定位的损坏位置。 回答no，说明前面没有未精确定位的损坏位置。

帮小A确定一种询问方式，使得他的询问次数在最差情况最少。

显然，若i,j两个位置都是未精确定位的损坏位置，且i<j那么一定不可能在i被找出来前先找出j，所以找出这些损坏位置的过程本质上一定是先找靠前的错误再找靠后的错误，先找最靠前的错误再找其他错误。

这样我们可以考虑每次询问获得的信息，围绕依次检查错误这个逻辑来写状态转移方程：

$$f[i][j][k] = MIN_{l=1}^{k-1} max(f[i][j][l], f[i-l][j][k-l]) + 1$$

$$f[i][j][1] = f[i-1][j-1][i-1]$$

解释：

$f[i][j][k]$表示一共有i个待检测的位置，有j个错误，第一个错误只会出现在前k个位置中，这之后需要的最少检测次数。

当我们询问一次L位置时，有两个可能的结果：

no，前面没有损坏位置，那么L及他前面的位置不用被检测了，待检测位置数量减少。

yes,前面有损坏位置，这样我们唯一得到的信息是，第一个错误只会出现在前L个位置。

当第一个错误只会出现在前1个位置中时，我们显然已经定位出了这个第一个错误了。

有了上述状态转移方程，我们就获得了原始的做法$O(mn^3)$。

进一步地，我们通过讨论f的含义，得到了如下性质：

$f[i][j][l]$随l的增大而不降。

$f[i-l][j][k-l]$随l的增大而不升。

$f[i-l][j][k-l]$随k的增大而不降。

设$G(l)=max(f[i][j][l],f[i-l][j][k-l])$：是一个有单谷函数。

由于随着k的增大$G(l)$的最小值右移，且$f[i][j][k]=MIN_{l=1}^{k-1}G(l)$其取值范围不断向右延伸，这也就意味着对与$f[i][j][k]$而言，随k的增大他对应的l有决策单调性，利用到决策的单调性，可以得到最终做法$O(mn^2)$。

此外，打表可以发现当输入的n与m的大小满足，m>[n/3]时，答案始终是n-1，利用这个性质，以及$O(mn^3)$的原始做法，可以通过暴力打表的方式通过此题，所以仅就能否被A掉这件事而言，这个题并不难。

H. 区间

$abs(A-B)\leq 1$ 则有解否则无解。

有一个特殊情况，$A=B=0$，是无解的。