A、倒水

source：瞎编的，个人感觉难度< 1800。

直觉告诉我，这个过程应该是这样的：

定义操作5为：操作（1,2,4）若干次，然后此时的水都倒进桶里。

那么应该是：操作5执行若干次，然后再执行若干次，把此时一部分水倒进桶里。

对拍之后发现是对的。

于是我们定义$cost[x]$表示最少几次，能让三个杯子恰好凑够x的水，且都倒进了桶里，$cost2[x]$表示最少几次，我能让三个杯子中一部分水恰好是x，且都倒进了桶里。

显然$cost$随时可以用，$cost2$只有最后一次可以用。

那么我们就可以开心`dp`了，没有细节。

复杂度$O(xyz+N(x+y+z))$。

B、让他们连通

部分分做法略。这题简单死了。

第一种做法就是重构树了。

我们根据操作时间建立一棵重构树，在重构树上求出i和$i+1$最早连通的时间a_i。

那么，对于每次询问，答案就是$\max([a_{l_i},a_{r_i-1}])$。

其实我想让大家练练整体二分的。

我们也可以通过整体二分来求a_i。

一个超级好写的方式是，我们动态开点建一棵线段树，每个节点存当前答案在$[l,r]$的询问有哪些。

然后直接跑log遍模拟，每一遍模拟的过程中，一旦合并到询问$[l,r]$对应的一半的时候，就把所有询问算一下，看看应该往左丢还是往右丢就行。

复杂度多一个log，但是不是每道题都会多这个log的。

需要注意的一个细节是，你不能递归建树，只能`bfs`的形式来建树，因为这样你编号的顺序才是按层数编号。

C、通信网络

对于一个部分分做法，我们只需要枚举每个点作为中继点z，然后二分答案，dfs一下每棵子树，看看子树内有多少个在范围内的点即可。

这个查询过程显然可以通过按`dfs`序建立的主席树来加速，可以做到两个log。

一个log的做法也很明显：我们直接把这个点当根的时候，对应的所有主席树都拿出来，然后从这些主席树的根节点处同时二分，这样复杂度就只有一个log了。

D、3SUM

SOURCE：arc180D

部分分做法略。

这题当T4其实有点简单的（但是去年CSP/NOIP的T4好像还不如这个难）。

我们考虑对于每组询问，区间内的最大值显然是不可避免的要被选到的。

除了最大值所在位置x以外（如果有多个，可以钦定最左/最右），一共就这么三种情况：

（1）选了$[L+1,R-1]$，以及L,R两个端点。

（2）选了$[L,t-1],[t,t],[t+1,R]$三个区间。

为啥只有这两种形式呢？

我们考虑假设$t<x$，那么，$[t+1,R]$的最大值一定是A_x，$[L,t]$需要划分成两段区间，无论第二段区间是什么，我都可以疯狂把它右端点归给$[t+1,R]$这一段，依旧不会改变$[t+1,R]$的最大值。

所以一定只有上述两种情况。

第一种情况非常容易处理，不说了。

第二种情况，我们可以分t在x左边还是右边分别做。其实只用做一边，另一边就是翻转序列/询问再做一遍。

以下用t在x右边来设计算法：

我们考虑建立一个笛卡尔树。

那么每一个位置i，都有一个管辖范围$[l_i,r_i]$。

我们对于询问$[L,R],x$，想找的分割位置，实际上就是满足：

- $x<i\le R$
- $r_i\ge R$

的所有i中，$A_i+min(l_i,i-1)$的最小值。

对于每一个i，我们可以在建笛卡尔树的过程中预处理出$r_i,A_i+min(l_i,i-1)$。

关键是，怎么对于每一组询问去求答案。

考虑扫描线。

我们按照i从n到1的顺序来枚举。

首先，把所有$r_j=i$的位置j，插入到线段树中。在线段树中的下标是j，值是$A_j+min(l_j,j-1)$。

然后，我们枚举所有$R=i$的询问。此时，线段树中存储的所有元素，都满足$r_j\ge R$，我们就只用关心$x<j\le R$一个限制了，直接在线段树中查询区间$[x+1,R]$的最小值即可。

时间复杂度$O((q+n)logn)$,常数有点大。