1 机器人指令

不难发现，无论机器人执行一次命令序列之后方向如何旋转，它重复执行 4 次命令序列之后总能回到原来的方向，所以每 4 次命令序列使它产生的位移是相同的。

我们暴力模拟 4 次命令序列，就能知道 $4 \times \lfloor \frac{T}{4} \rfloor$ 次命令序列的执行使它产生的位移，同时它现在恢复了初始的朝向，我们对 $T \mod 4$ 的零散部分再模拟一遍即可得到它最终的的位置 (x, y)，答案即为 $|x| + |y|$。

时间复杂度 $O(n)$，空间复杂度 $O(1)$

2 逛餐馆

显然我们只会往一个方向走，不会回头（回头一定浪费时间）。那我们将所有餐馆排序。

那我们不妨枚举我们能走到最远的餐馆，假设我们枚举我们走到第 i 个餐馆（排完序之后）。

那么我们就是在 $1-i$ 餐馆中寻若干个餐馆，使他们和不超过 $m - x_i$，然后求最多的个数即可。那显然就是在 $1-i$ 餐馆中按照 t_i 从小到大排序，然后不断累加直至超过 $m-x_i$ 的餐馆个数。

这个经典题目用优先队列/堆来维护即可。

我们维护一个大根堆，这个堆里面维护 $1-i$ 所有餐馆中，最小的若干个餐馆，并且他们和 $\le m - x_i$。

那我们从 1 到 n 枚举 i，每次执行三个操作：

1. 把 t_i 加进堆里面。

2. 若堆中元素之和大于 $m - x_i$，则不断弹出堆顶元素直到总和小于等于 $m - x_i$。

3. 堆的大小就是我们能吃的餐馆的个数，用这个更新答案。

时间复杂度 $O(n \log n)$。

3 符文师

首先由于第一种操作的存在，我们把它转化成一个环上的问题，并删去第一种操作。

接下来我们证明一个结论：

设 S 为一个符卡的集合，则存在一个方案能够使用 S 中所有符卡的充要条件为 $\sum_{i \in S} level_i \le n$

必要性是显然的。由于使用一张符卡会使场上的符卡数减少 $level$，因此使用的符卡的 $level$ 之和不可能大于 n，否则将存在一个时刻使场上的符卡数为负。

充分性是比较显然的。我们使用归纳法证明它：

1. 首先，若 S 为空，则结论是显然的。

2. 若 S 不为空，我们将 S 中的符卡在环上标记出来，按顺时针方向的顺序编号，把他们到下一张 S 中的符卡的距离记为 S_i。注意到有 $\sum S_i = n$ 和 $\sum L_i \le n$ 成立，根据鸽巢原理存

在一个 i 使得 $L_i \le S_i$，那么我们使用这张符卡不会丢弃任何 S 内的其它符卡，从而将 S 的规模缩小了 1。

证明了这个结论，原题就变成了

求一个符卡的集合，满足 $\sum L_i \le n$ 且 $\sum D_i$ 最大。

这是一个典型的 01 背包问题，可以用朴素的动态规划解决。

时间复杂度 $O(n^2)$。

4 魔法师

权值线段树 + 动态开节点。

我们的问题是：我们有若干个可重集合，然后我们从第 i 个可重集合中拿前 i 大组成一个新的可重集合 S。我们的目的是动态维护 S 的前 n 大的和。

显然，我们需要维护所有小集合（小集合就是类型相同的书本）以及 S。

维护 $\max t$ 棵权值线段树，第 i 棵线段树对应第 i 个小集合。我们称这些线段树为小树。

另外维护一棵权值线段树，用来维护 S。我们称这棵线段树为大树。

权值线段树上只需维护一个值，就是在 $[l, r]$ 这个区间内有多少个元素，以及元素和。

需要支持的操作为

- 加入或减少一个元素。
- 查询一个元素的排名。
- 查询第 k 大。
- 查询前 k 大元素和。

后三者只需要在线段树上面二分即可。

- 对于 BORROW 操作，我们就是在第 t 棵小树中加入一个元素 p。这个可以通过线段树单点修改完成。

  之后，我们要确定要不要将 p 加入 S 中。这个只要知道 p 在第 t 个集合中的排名是否 $\le t$ 即可。

  加入了 S 以后，我们还要将原来的第 t 大，即现在的第 $t+1$ 大从 S 中删去。
- 对于 RETURN 操作，我们可以类似地，将 BORROW 操作取反即可。

  我们就是在第 t 棵小树中减少一个元素 p。线段树单点修改。

  之后，我们要确定要不要将 p 从 S 中删除。这个只要知道 p 在第 t 个集合中的排名是否 $\le t$ 即可。

  删除以后，我们还要将原来的第 $t+1$ 大，即现在的第 t 大从加入 S。

对于整棵大树，只需要支持寻找前 n 大的操作即可。

以及大树和小树不用分开写，放在一起写线段树模板即可。

时间复杂度 $O(m \log maxt)$，空间复杂度 $O(maxt \log maxt)$