贪心

hydd

October 19, 2022

Notes

题目翻译大部分来自洛谷，可能和原题面有较大差别。

难度不高。

Well played!

小 Y 有 n 只精灵。每只精灵都有对应的生命值 hp_i 和攻击值 d_i。在比赛过程中，小 Y 可以借助巴拉拉小魔仙之力，说出这两种咒语：

1、“乌卡拉！血量！加倍！“意即将当前精灵的生命值加倍。

2、“乌卡拉！生命之力！”意即将当前精灵的生命值赋给当前精灵的攻击值（使得 $d_i = hp_i$）。

小 Y 当然不能无限使用这两种咒语。在一局游戏中，他可以使用第一种咒语 a 次，第二次咒语 b 次。由于小 Y 购买了超级 Nono，所以这两种咒语都可以被多次用在同一精灵身上，且咒语的使用顺序没有限制，可以不用完所有的咒语。

小 Y 非常希望通过使用这些咒语使得自己的精灵战斗群的攻击值达到最大。现在，小 Y 想知道这个最大值。

$1 \leq n \leq 2 \cdot 10^5, 0 \leq a \leq 20, 0 \leq b \leq 2 \cdot 10^5, 1 \leq hp_i, dmg_i <= 10^9$。[1]

Well played!

$\times 2$ 只给同一个人最优。

剩下的一定是选替换掉新增的贡献最大的若干个。

排序求一下即可。

[1] CodeForces 976E

Koa and the Beach

Koa 要从海岸游到岛上。海岸的位置是 0，岛的位置是 $n+1$。

除了海岸和岛上，在 $[1,n]$ 之间的位置 i 都有一个深度 d_i。海里有潮汐，具体来讲，给定值 k，则有下标从 0 开始的数组 $p=[0,1,2,...,k,k-1,...,1]$。在时间 t，海里的潮汐高度为 $p_{t \mod 2k}$。在任意时间，如果 Koa 在位置 x，可以选择下一个时间停留位置 x，或游到位置 $x+1$。

Koa 能在时间 t 到达位置 i 当且仅当时间 t 的潮汐高度与位置 i 的深度之和不超过一个给定的值 l，即：$p_{t \mod 2k}+d_i \le l$，否则她会溺水。海岸和岛上是安全的，不会受潮汐影响，所以在任何时间停留在那里都不会溺水。Koa 在游泳的时候不会受潮汐的影响（不会溺水）。

求出 Koa 是否能安全地到达岛上。

多组数据，数据组数 $t \le 10^4$，$1 \le n \le 3 \cdot 10^5$，$1 \le k,l \le 10^9$，$0 \le d_i \le 10^9$。[2]

Koa and the Beach

称无论什么时候在当前位置都是安全的点为安全点，否则为不安全点。显然不安全点能走的时间是一些区间。

若当前点为安全点且下一个点也是安全点，直接走过去即可；

若当前点为安全点且下一个点不是安全点，则在最优的时间走过去（递减且正好能走过去）。

若当前点不为安全点且下一个点是安全点，直接走过去即可；

若当前点不为安全点且下一个点不是安全点，则只要能走过去且不死就走过去，一直不走必定会死，能跑路就赶紧跑路。

[2] CodeForces 1384B2

Buy Low Sell High

已知接下来 N 天的股票价格，每天你可以买进一股股票，卖出一股股票，或者什么也不做。N 天之后你拥有的股票应为 0，当然，希望这 N 天内能够赚足够多的钱。

输出最大收益。

$2 \leq N \leq 300000$，$1 \leq p_i \leq 10^6$。[3]

Buy Low Sell High

将每天的价格当作一种选项，压入小根堆中。

从前往后扫，对于现在的价格 p_i，如果堆顶元素 p_j 满足 $p_j < p_i$，那么取出堆顶，在第 j 天买入股票，在第 i 天卖出股票，此时，我们就可以获得 $p_i - p_j$ 的收益。

然而，如果之后有 p_k 满足 $p_k > p_i$，我们当前作出的决策可能并不是最优的，如何反悔呢？

当我们进行上述操作时，我们将 p_i 也压入堆中，增加一个 p_i 的选项，弹出时，我们相当于将 p_j 按照 p_i 的价格又买了回来。

[3] CodeForces 865D

Fishes

有一个长为 n ，宽为 m 的鱼缸，还有一个边长为 r 的正方形渔网。

你可以往鱼缸里放 k 条鱼，问用渔网随机在浴缸里捞鱼的最大期望是多少。

$1 \leq n, m \leq 10^5$ ，$1 \leq r \leq \min(n, m)$ ，$1 \leq k \leq \min(n \cdot m, 10^5)$。[4]

Fishes

显然期望可以拆开，变成期望最大的 k 个位置的和。

对于 (x, y)，能覆盖它的渔网数量
$= (\min(x + r - 1, m) - \max(x, r) + 1) \times (\min(y + r - 1, m) - \max(y, r) + 1)$。

那么两维相对独立，求出 $(\min(x + r - 1, m) - \max(x, r) + 1)$ 的所有 x 的取值从大到小排，对于 y 同理。

然后可以使用堆贪心，因为 $u[1] * v[1] > u[2] * v[1], u[1] * v[1] > u[1] * v[2]$，取出前 k 个即可。

[4] CodeForces 912D

Monster Invaders

这是一个 RPG 枪战游戏。有 n 个关卡，每一个关卡都有 a_i 个生命值为 1 的小怪，1 个生命值为 2 的 boss。有三种武器：

- 手枪，可以对一个怪物造成 1 点伤害，每次使用前需要 r_1 秒装弹。
- 激光枪，可以对目前关卡所有怪物造成 1 点伤害，每次使用前需要 r_2 秒装弹。
- AWP，可以直接杀死任意怪物，每次使用前需要 r_3 秒装弹。

由于游戏 feature，用手枪或 AWP 攻击 boss 前必须先杀死 boss 所在关卡的所有小怪。如果攻击 boss 但此次攻击并没有杀死 boss，必须移动到该关卡的相邻关卡。除此之外，可以在任意时间移动到所在关卡的相邻关卡，每一次移动需要 d 秒，此时什么都不能做。

从第一关开始游戏，游戏目标是击杀所有 boss，求完成游戏的最短时间。

$2 \le n \le 10^6, 1 \le r1 \le r2 \le r3 \le 10^9, 1 \le d \le 10^9, 1 \le a_i \le 10^6$。[5]

Monster Invaders

由于 $r_1 \leq r_2 \leq r_3$，而手枪和 AWP 都只能打一只怪物，所以 AWP 只可能用来打 BOSS。

如果用手枪一只一只打小怪，最后一定是一起把 BOSS 一起解决了（不解决放着之后需要的时间不会更少）。

如果用激光枪，那么打完一定要走，如果之前 $i-1$ BOSS 也没死，就去打 $i-1$，否则打 $i+1$（你之后还得打 $i-1$ 的 BOSS，不如现在先去了）。

不过即使你用手枪打，如果 $i-1$ 的 BOSS 没死，你打完后一定也是先去 $i-1$，之后再折返一定不优。

Monster Invaders

所以说，假设 i 之前的怪全都死了，一种是你用手枪 +AWP 解决，到 $i+1$；另一种是用激光枪/手枪给每个打一枪，然后到 $i+1$，再用任意方法（都打一枪或直接解决），再回到 i，手枪一枪打死 BOSS，又到 $i+1$，如果 BOSS 没死也一枪打死，最后到 $i+2$。

所以一种是新增 $a_i \times r_1 + r_3 + d$ 的代价走到 $i+1$，一种是新增 $\min((a_i+1)\times r_1, r_3)+r_1+\min(\min((a_{i+1}+1)\times r_1, r_3)+r_1, a_{i+1}\times r_1+r_3)+4d$ 走到 $i+2$。

dp 即可。

[5] CodeForces 1396C

Voting

有 n 个选民，你可以付出 p_i 的代价让第 i 个选民为你投票，或者，在为你投票的人数达到 m_i 时，他会主动为你投票而不用你付出任何代价。

问得到所有选民投票的最小代价。t 组数据。

$1 \leq t \leq 2 \cdot 10^5$, $1 \leq n \leq 2 \cdot 10^5$, $1 \leq p_i \leq 10^9, 0 \leq m_i < n$。 [6]

Voting

按照 m_i 从小到大排序，有个朴素的想法是设 $dp[i][j]$ 表示考虑完前 i 个，$i+1..n$ 还有 j 个被贿赂的最小代价。

转移第一种是 $i+1$ 被贿赂，就转移到 $dp[i+1][j-1]+q_{i+1}$；第二种是没被贿赂，那么要求 $i+j \geq m_{i+1}$，转移到 $dp[i+1][j]$。这足以通过 E1。

Voting

发现 $i+j \geq m_{i+1}$ 可以移项成为 $j \geq m_{i+1}-i$，也就是一个人跟风为你投票当且仅当后面被贿赂的人数 $\geq m_i - i + 1$。

把 m_i 相同的放在一起，倒过来考虑，设当前 $\geq m_i$ 被贿赂的人数为 $cnt1$，$< m_i$ 的人数为 $cnt2$，相当于现在一定有 $cnt1 + cnt2$ 个人投票，如果够了，那么能全部都选；否则不够，就需要在 $\geq m_i$ 里的再贿赂一些，显然贿赂的越少越好（如果不够之后可以再贿赂），就贿赂直到 $cnt1 + cnt2$ 足够，拿个小根堆存一下。

[6] CodeForces 1251E2

The Next Good String

如果不包含长度 $\geq d$ 的回文子串，则该字符串为“好”。

给你一个仅有小写字母组成的字符串 s，要你找到一个“好”的串 t，字典序比 s 大。在这所有满足条件的字符串中，t 得是字典序最小的。

可能无解。

$1 \leq d \leq |s| \leq 4 \times 10^5$。[7]

The Next Good String

回文串不存在长度 $\geq d$ 的回文子串，等价于不存在长度为 d 或 $d+1$ 的回文子串。

由于字典序要 $> s$ 且最小，所以让 s 和答案串的 lcp 尽量长。

如果发现某个位置和前若干个形成了长度为 d 或 $d+1$ 的回文子串，那么最优的策略是把它替换了，且一定是替换成比原先大的且与之前不会形成长度为 d 或 $d+1$ 的回文子串的。如果这个位置没有可选的替换方案就考虑前一个位置以此类推。

The Next Good String

之后的问题就没有字典序大小的限制了。还是一样的用贪心，每个位置尽量填最小的，且不用考虑没有能填的问题（26 个字符，最多 ban 两个）。

是否是回文怎么判断？直接哈希，每次是在最后插入字符，容易维护前缀哈希值。

[7] Codeforces 196D

Difficult Mountain

n 个人相约去爬山。山的初始攀登难度为 d。每位登山者有两个属性：技巧 s 和整洁度 a。

技巧为 s 的登山者能登上攀登难度为 p 的山当且仅当 $p \leq s$。在一位整洁度为 a 的登山者登上攀登难度为 p 的山后，山的攀登难度会变为 $\max(p, a)$。

请给这些登山者指定一个爬山的先后顺序，最大化登上山的人数。如果轮到一位登山者时他能登上山，则他一定会选择登山。

$1 \leq n \leq 5 \times 10^5$，$0 \leq d \leq 10^9$。[8]

Difficult Mountain

分情况讨论这个最大值，这种一般就考虑调整：

- 最大值都是 a，不妨假设 $a_i < a_j$。j 先上 i 一定上不去，不如先让 i 上且 j 有可能上去且最后高度不会更高（结论是 i 先上）；
- 如果一个最大值是 a 一个最大值是 s，不妨设 i 的最大值是 a，j 的最大值是 s。如果 $a_i > s_j$ 则还是 i 先上 j 一定上不去，不如先让 j 上可能上去且最后高度不会更高（结论是 j 先上）；不如 $a_i \leq s_j$ 则 i 先上不会影响 j，j 先上则不一定（结论是 i 先上）。
- 最大值都是 s，不妨假设 $s_i < s_j$。i 先上，一定不会影响 j；j 先上，可能会影响 i（结论是 i 先上）。

所以发现最大值小的一定优先，最大值相同的可以直接贪心思想，一定是 s 小的先上。

[8] Codeforces 1601D

Tournament Construction

一个竞赛图的度数集合是由该竞赛图中每个点的出度所构成的集合。

现给定一个 m 个元素的集合，第 i 个元素是 ai。判断其是否是一个竞赛图的度数集合，如果是，找到点数最小的满足条件的竞赛图。

$1 \leq m \leq 31$，$0 \leq a_i \leq 30$，a_i 互不相同。[9]

Tournament Construction

根据兰道定理，把比分序列从小到大排序之后要求前 k 个之和 $\geq \frac{k(k-1)}{2}$。

用这个定理，先把 a 排序，dp 一下记录集合大小，构造出的点数，总共边数，dp 一下求出原度数。

构造就比较容易，由于竞赛图删去一个点和与它相邻的所有边还是一个竞赛图，按照兰道定理贪心的删去一个出度最小的即可。

[9] CodeForces 850D

Foolprüf Security

给你 $n+m$ 个点，组成一棵树，并且左半边 n 个点，右半边 m 个点组成一张二分图。

告诉你两个子序列 a, b。他们都是这个 Prufer code 的一个子序列，并且 a 中元素在 $1, 2, \cdots, n$ 内，b 中元素在 $n+1, n+2, \cdots, n+m$ 内。

问你是否能构造这样一棵树，并且将这棵树输出。

$2 \leq n, m \leq 10^5$，$1 \leq |a|, |b|$，$|a| + |b| \leq n + m - 2$。[10]

Foolprüf Security

各自的个数得满足限制，剩下的一定是一个左部点一个右部点，所以要求 $|a| \leq m-1, |b| \leq n-1$。

然后就直接贪心，度数为 0 的就是现在的叶子节点，拿出来，和另一部的开头连边，那个点度数 -1。

度数为 0 就可以加入到叶子节点，如果另一部的序列为空了就随便连。

[10] CodeForces 1267F

April Fools' Problem

n 道题，第 i 天可以花费 a_i 准备一道题，花费 b_i 打印一道题，每天最多准备一道，最多打印一道，准备的题可以留到以后打印，求最少花费使得准备并打印 k 道题。

$1 \leq k \leq n \leq 500000$。[11]

April Fools' Problem

首先有个裸的最小费用最大流建图，不过显然过不去。

毕竟都可以网络流了，所以一定是个凸函数。

考虑 wqs 二分，二分完之后 a, b 可能变成负的。

考虑负的怎么做，可以用带悔贪心，从后往前，要么选择后面 b 最小的配对，要么替换掉 a 最大的匹配中的 a，要么不做操作。如果现在 a 选了把 a 加入优先队列，b 没配对把 b 加入优先队列。每次选最优决策即可。

[11] CodeForces 802O

Puzzle

给定两个 $2 \times n$ 的 01 矩阵 A 和 B，定义一次操作为交换 A 中任意两个相邻的位置中的值，输出使得 $A = B$ 的最小操作次数，如果无法使 $A = B$ 则输出 -1。

$1 \leq n \leq 200000$。[12]

Puzzle

能不能把所有上下交换的移到最前面，之后只考虑左右交换？

但是其实直接这样是不行的，比如这种情况：

0 1

0 1

变成

1 1

0 0

不存在先上下后左右的方案。

Puzzle

需要注意到，在任何时刻都不会交换两个相同的位置，且 0 和 1 是对称的，只需要考虑一种。

那么，假设知道了 A 中每个 (x_1, y_1) 的 1 移到了 B 中的 (x_2, y_2)，那么答案一定不超过 $\sum(|x_2 - x_1| + |y_2 - y_1|)$。

原因是一次交换最多使 A 的有且仅有一个 1 离目标更近一步。

Puzzle

可以把操作改为交换任意两个不同的位置，代价是它们的曼哈顿距离，具体方法就是找一条曼哈顿距离最小的路径，从后往前依次移动每个 1。

这样就可以把问题看作把原来每个 1 和目标状态的每个 1 进行匹配，求它们曼哈顿距离和的最小值。答案恰好为 $\sum(|x_2 - x_1| + |y_2 - y_1|)$。

还有点小问题，你不能将某个 1 和原先是 1 目标也是 1 的位置匹配，但是这样一定不优，可以强制所有原先是 1 目标也是 1 的位置不动。

Puzzle

设 $C = A - B$，现在要把 1 和 -1 两两匹配。只有一行的时候，只需要考虑对于相邻两个位置中间的分界线被跨过多少次。

考虑两行的情况，从前往后一列一列考虑，贪心把里面的 1 和 -1 优先匹配完。因为如果留到下一列和之后的匹配，它们都需要增加 1 的代价，而更换匹配方式会减少这 2 的代价，而最多增加 2 的跨行代价。同行的优先匹配，原因同理。

[12] CodeForces 1700F