前 k 贪心

这名字是我瞎取的, 不要在意

§ 模型

有时候会见到一类题：给出若干类东西, 每类有多个物品, 不同类物品任意组合形成一种带权方案, 求方案的前 k 大 / 小（下文直接讨论前 k 大, 前 k 小同理）

§ 2 类物品合并前 k 大

> 有两种物品 A, B, 其中 A 有 n 个物品, B 有 m 个物品, 每个物品带权, 用 A_i 或 B_i 表示
>
> 一种方案由一个 A 和一个 B 合成, 记 (x, y) 表示 A_x, B_y 合成了一个方案, 该方案的权为 A_x, B_y
>
> 现在求前 k 大的方案的权值, $n, m \leqslant 10^5, k \leqslant \min\{10^5, nm\}$

有一个显然的思路是 $O(nm)$ 暴力把所有方案算出来, 显然复杂度爆炸, 我们需要寻求一个更加优秀的解决方案

首先是把 A, B 从大到小排序, 因为这道题选取是任意的, 而有序比无序更具优良性质

我们考虑依次处理出答案, 也就是先找到最大的, 再找次大的, 次次大的, 依此类推

排序后重新标号, 注意到方案 $(1, 1)$ 一定是最大的, 次大的一定是 $(1, 2), (2, 1)$ 中的一个, 根据这点, 我们或许能够被启发, 得到一个有力的结论：$(x, y) \leqslant (x+1, y), (x, y) \leqslant (x, y+1)$

这种结论是解决这类问题的核心, 一切都是基于 **方案之间存在一定的大小关系**, 比如说如果方案 (x, y) 是前 k 中的一个, 那么意味着方案 $(x-1, y), (x, y-1)$ 一定也是前 k 个中的一个, 并且先于 (x, y) 出现

不难想到一个这样的处理办法：

1. 首先把 $(1, 1)$ 加入一个基于方案权值大小比较的大根堆中
2. 取出堆顶方案, 记为 (x, y), 如果 (x, y) 还没有出现过, 则执行 3, 否则反复操作 2, 直至堆为空或已取出前 k 个方案
3. 输出方案 (x, y), 把 $(x+1, y), (x, y+1)$ 加入堆中, 执行 2

虽然这是对的, 但很遗憾, 复杂度不对, 因为一个 (x,y) 可能被取出很多次, 然后复杂度爆炸

我们希望能构造一个算法:

1. 每种方案最多被取出一次
2. 比当前方案大的方案先于本方案出现

其实有一个很简单的解决方法, 一开始把 $(i,1)$ $(1\leqslant i\leqslant n)$ 放入堆中, 然后我们对于一个方案 (x,y) 只扩展 $(x,y+1)$, 显然满足我们上面的两个要求

但这种方法的可扩展性并不好, 因为接下来讲的多类物品合并它并不适用, 下面一起讲了

§ 多类物品合并前 k 大

> S 类物品记为 $A_1,A_2,...A_S$, A_i 类有 n_i 个物品, 每类物品各选一个作为方案, 方案权为方案选取的物品权和, 求前 k 大方案的权值 $\sum n_i\leqslant 10^5,k\leqslant 10^5$

一种方案我们记为 $(x_1,x_2,...x_S)$, 对于每类物品内部, 还是从大到小排好序, 但按照上面的方法的话, 显然扩展复杂度会爆炸, 我们需要寻求新路子

首先我们要解决存储问题, 显然不能用一个长为 S 的数组表示一个方案

观察到其实一个方案就像一个向量, 对于一个向量, 其实我们可以按照维度依次考虑问题, 这里直接给出解决方案, 然后再理解

先把只有一个物品的类去掉 (因为一个物品的类无论如何贡献是固定的), 所有的类 i 按照 $A_{i,1}-A_{i,2}$ 从大到小排好序 (至于为什么之后再解释)

我们用二元组 (x,y) 表示方案, 含义是当前考虑到第 x 维 (也就是第 x 类物品) 的第 y 个物品, 并且前 $x-1$ 维的状态已经完全确定 (意思就是以后扩展不会动这 $x-1$ 维了), 并且 x 后面的维还都是 1, 具体而言, 就是本方案形如 $(a_1,a_2,...a_{x-1},y,1,...1)$

然后扩展的时候有考虑三种转移:

- $(x,y)\to(x,y+1)$, 这个没什么好说的, 含义就是该维物品往后选
- $(x,y)\to(x+1,2)$, 就是所谓的前面维的状态完全确定, 维度往后移动 (注意, 这里第二项直接变为 2 是因为在此含义下 $(x+1,1)$ 其实就是 (x,y))
- 若 $y=2$, 则 $(x,y)\to(x+1,2)$, 同时要把第 x 维强制变为 1 并重新计算第 x 维的贡献

注意到一个状态 (x,y) 表示的不止一种方案, 但这是没有关系的

还是考虑是否满足我们上面的两个要求

首先一个方案只会唯一出现一次 (有一个小小的特殊情况, 等会会说), 因为这个转移就像一个分层图, 归纳一下, 假设前面 $x-1$ 维所有情况都被考虑到, 然后按照转移讨论讨论就好了

最优性也有保证, 前 2 种转移显然没问题, 对于第 3 种转移, 因为我们之前已经按照类排好序了, 根据排序的依据, 这也是没问题的

有点像反悔贪心

§ 扩展

上述这类思想也可以用于 k 短路之类的问题, 如果只有两维, 完全不需要带入第二种方法的 " 反悔 " 操作, 否则就需要用到更强的方法

§ 题目推荐

- UNR#4 追击圣诞老人

§ 参考

- 校内考试题