app

WJMZBMR

Universal OJ用户群

May 13, 2015

intro.

大家好，我是萌萌哒陈老师。
考场送温暖！

题目简述

有一些任务，每个任务i有两个属性截止日期t_i和完成收益p_i。
我们一共有T天，从1到T标号。
每天只能完成一个任务。截止日期为t_i的任务必须在前t_i天完成。
要求能够完成的任务的最大收益和。
同时支持两个操作，加入一个新任务和删除一个任务。

数据范围

测试点编号	T	Q	特殊说明
1	≤ 1000	≤ 1000	无
2	≤ 10	≤ 100000	无
3~6	≤ 100000	≤ 100000	所有操作均为 ADD 操作
7~8			所有 ADD 操作都在所有 DEL 操作之前
9~10	≤ 50000	≤ 50000	无
11~14	≤ 100000	≤ 100000	
15~20	≤ 300000	≤ 300000	

5%的数据

对于$T \leq 10$的数据，由于T很小，让我们不妨考虑较为暴力的做法。
首先很容易看到，对于所有t_i值恰好为t的任务，我们只需要保留p_i值最大的t个，多了也没用。
那么现在最多只有$T^2 = 100$个任务。
我们注意到，如果我们确定了哪些任务要完成，那么显然应该按照t_i值升序依次完成。
那我们考虑这样一个算法，令$dp_{i,j}$表示当前考虑到了t值为i的那些任务，你在第j天，你能获得的最大收益。转移就是枚举t值为i的任务你要完成几个。
复杂度为$O(T^3 \cdot Q)$。
这样你就能获得5分了！

10%的数据

如果没有操作的话，对这个问题容易想到使用贪心算法。
对于一个任务的集合P，如果存在一个方案完成P中每一个任务，则称P为合法的。
考虑如下简单贪心：

- 令当前任务集合P为空。
- 将所有任务按照p_i从大到小排序，然后依次考虑，如果当前任务加入P之后P任然合法，则将该任务加入P。

10%的数据

为什么如此贪心是对的？
假设在任意时候我们在当前任务加入P合法的情况下我们没有选择当前任务i，那么在完成以后我们强行加入该任务。
这时我们可以发现，我们能够删除一个i之后的任务使得方案重新合法，这意味着能得到更优的解。

10%的数据

如何判断方案合法？
等价于对于任意$1 \leq t \leq T$，P中$t_i \leq t$的任务数量不超过t。
为什么？如果存在t违反这个条件，那么你显然没法完成P中所有任务。
否则的话，将所有任务按t从小到大依次来，就能完成。

10%的数据

那么，容易看到，对于n个任务，我们可以使用一个$O(n^2)$的算法来暴力实现贪心。
那么总计就能得到一个时间复杂度为$O(n^2 \cdot Q)$的算法，由于常数不大，实现的好能够通过数据。

10%的数据

考虑判断条件，容易看出，我们可以给第i天维护一个b_i值，表示i减去$t_j \leq i$的任务数量。
那么任务集合合法等价于b_i的最小值非负。
考虑加入一个任务，可以看出就等价于给一段b_i减一。
那么这就是一个经典的线段树问题。
那么我们就能实现一个复杂度为$O(n \log n \cdot Q)$的算法，就能通过$T, Q \leq 1000$的数据。
对于$T \leq 10$的情况，我们和之前一样只保留有意义的T^2个数据即可。
这样你就能获得10分了！

另20%的数据

方便起见，假定所有任务p值都不同，那么最优任务集合任何时刻都唯一。
现在我们来考虑只有加入新任务的情况。
考虑当前最优任务集合为P，假如我们加入新任务a。
有如下几种情况：

- 最优任务集合不变。
- 最优任务集合变为P加上a。
- 其它情况。

我们可以证明如果不是前两种情况，那么新的最优任务集合必然是P中删去一个并上a。

另20%的数据

如何证明？
将P中所有任务按照p值从大到小排序。令其为$u_1, u_2, \ldots, u_n$。
假设$p_{u_i} < p_a < p_{u_{i+1}}$。
那么考虑贪心算法，显然$u_1, u_2, \ldots, u_i$都会被选。
当贪心算法考虑到a时，如果a无法加入，那么此时显然就是情况1。
如果a可以加入，假设存在一个v，不在原最优任务集合P当中，但是却被贪心算法选中了。 那么在v的前面，必然存在一个被贪心算法拒绝的u_j(不然的话，v肯定无法被选中)。
同时必然只有一个u_j被拒绝，因为在v之前没有新的任务被选择，a一个任务的加入最多只能使得一个u_j被拒绝。
由于a的加入使得u_j被拒绝，那么我们必然有$t_a \leq t_{u_j}$。否则a无法影响u_j，然而此时可以发现，由于$t_a \leq t_{u_j}$，在u_j存在的情况下v无法被选，a存在的情况下就更加不可能了。

另20%的数据

以上我们证明了新的最优任务集合P'必然由P中的任务和a组成。
我们还可以证明P中的元素最多被删除一个。
这样的话，我们就能得到一个较为简单的算法。
首先维护一个关于b_i值的线段树，对于新增任务$a = (t, p)$，先看一下$b_t, b_{t+1}, \cdots, b_T$的最小值是否为0，不是的话我们就直接加入$a$。
否则，不妨令第一个0的位置在x。那么我们注意到，加入a以后，我们只需要删除一个t值在$[1, x]$的任务，就能是的任务集合依然合法。
我们当然是删除p值最小的那个。比较一下这个的p值和a的p值，来决定是否删除。
这样我们就有了一个时间复杂度$O(n \log n)$的算法，虽然只支持加入。
结合之前所说的，你就能轻松获得30分了！

另10%的数据

如果删除都在插入之后，由于我们不要求在线，你完全可以倒过来看成加入。
那么问题就变成了两个只有加入的问题。
结合之前所说的，你就能轻松获得30分了！

100%的数据

接下来我们要处理删除的情况。
不妨设当前最优任务集合为P。我们要删除任务a。
如果任务a不在P中，那么很好，我们什么也不用做。
否则的话，令删除a之后的最优任务集合为Q。
那么加入任务a之后，Q就变成了P。注意由我们之前的假设，任意时刻最优任务集合都是唯一的。 那么此时只可能是两种情况，Q是P删去a和Q是P删去a后再新插入一个元素x。
我们考虑删除a。此时我们观察所有位置的b值，不妨令最后一个0的位置为y。那么显然，新插入的元素x的t值一定要大于y。我们选择一个p值最大的加入即可。
这样的话，我们还得用一个线段树维护一下所有不在P中的任务。
时间复杂度$O(n \log n)$，你就能直接获得100分了！

感谢

richard peng和顾昱洲对于出题想法的帮助。
罗雨屏对验题的帮助。