《分组》解题报告

杭州学军中学 金策

1 试题来源

PA 2014 Round 3 Drużyny

提交地址：http://main.edu.pl/pl/archive/pa/2014/dru

BZOJ：http://www.lydsy.com/JudgeOnline/problem.php?id=3711

2 试题大意

有n个人排成一列，按顺序编号为$1, 2, \cdots, n$。

将其分为若干连续段，使得第i个人所在段的人数不超过d_i，不少于c_i。

判断是否有解，若有解则求出段的数量的最大值，以及达到此最大值的方案数量。

数据规模：$1 \le n \le 10^6, 1 \le c_i \le d_i \le n$。

3 算法介绍

我们首先不考虑方案计数，只关心段数最大值。

3.1 朴素DP

令$f[i]$表示仅将$1 \sim i$分段，所得到的最大段数（若无解则视作$-\infty$)，则

$f[0] = 0$

$f[i] = 1+\max\left\{ f[j-1] \middle| 1 \le j \le i, \max\{c_j, \cdots, c_i\} \le i-j+1 \le \min\{d_j, \cdots, d_i\} \right\}$

直接根据此式，按照i升序，j降序进行DP，可以做到$O(n^2)$。

3.2 分治优化(不考虑d_i的情况)

现在假设所有$d_i = n$，即取消上界限制。

采用分治的方法优化这个朴素DP。在调用$\textsc{Solve}(l, r)$时，我们假设区间$[l, r]$内的i已被所有$j \in [0, l-1]$更新过，此次需要进行所有$l \le j \le i \le r$的更新。具体流程为：

(1)取一个$k \in [l, r]$，使得$c_k = \max\{c_l, \cdots, c_r\}$；

(2)调用$\textsc{Solve}(l, k-1)$；

(3)进行所有$l \le j \le k \le i \le r$的更新；

(4)调用$\textsc{Solve}(k+1, r)$。

3.2.1 第(1)步的实现

用线段树查询k即可。

3.2.2 第(3)步的实现

这一步的实现是本题关键。

由于区间$[j, i]$包含k，所以只需要满足$i - j + 1 \ge \max\{c_j, \cdots, c_i\} = c_k$即可进行更新，即$j \in [l, \min\{i - c_k + 1, k\}]$，其中$i \ge i_{\min} = \max\{k, c_k + l - 1\}$以保证此区间非空。

我们从$i = i_{\min}$开始左往右枚举i，同时记录可用来更新i的最大$f[j-1]$。初始时j的取值范围为$[l, i_{\min} - c_k + 1]$。i每往右移动一位，j的取值范围也会往右增加一位。如果$c_k + k - 1 > r$，则循环到$i = r$后终止；否则当$i - c_k + 1 \ge k$时，j的取值范围为$[l, k]$且不再改变，我们用此时最大的$f[j-1]$去更新所有$i \in [c_k + k - 1, r]$。

在这一步中，开始进行一次区间查询，最后进行一次区间更新，中间i的循环过程执行次数$t = \min\{r, c_k + k - 1\} - \max\{k, c_k + l - 1\}$，有$t \le r - k$，$t \le (c_k + k - 1) - (c_k + l - 1) = k - l$。即执行次数不超过$\max\{k - l, r - k\}$。

3.2.3 利用数据结构维护

用线段树维护$f[]$，则区间查询和区间更新是$O(\log n)$的。

第(3)步i的循环过程中，每次需要单点查询和单点更新，这都可以用数组$O(1)$记录。

等到第(3)步结束，$f[k]$已经被所有$1 \le j \le k$更新完毕后，再将数组中记录的$f[k]$值用$O(\log n)$时间插入线段树。

3.2.4 时间复杂度分析

可以看出$\textsc{Solve}(l,r)$的时间复杂度为$T(l,r) = T(l,k-1) + T(k+1,r) + O(\log n) + \min\{k-l, r-k\}$。

每次调用$\textsc{Solve}$时会选择一个k，而每一个k恰被选择一次，所以$\textsc{Solve}$被调用了n次。上式第三项的总贡献为$O(n \log n)$。

第四项的复杂度与启发式合并类似（启发式分割），总复杂度为$O(n \log n)$。

从而$T(1,n) = O(n \log n)$。

3.3 考虑d_i的情况

如果存在$d_i \neq n$的情况，则需对第(3)步中i的循环和最后区间更新的部分做一些修改。

在i右移的过程中，区间长度$i-j+1$的上界可能缩小，此时j的取值区间的左端点也会右移。假设某次左端点右移后变为$i-d_i+1$，则需要有$l \le i-d_i+1 \le k$。而在分治过程中，用来更新某个i的所有区间$[l,k]$不会有交集，所以对于每个i只发生不超过一次右移。

于是，每次右移时用$O(\log n)$的时间查询新区间内$f[j-1]$的最大值即可。总复杂度仍然为$O(n \log n)$。

3.4 方案计数

容易看出我们的更新过程是不重不漏的，所以在记录最大值的同时记录取到最大值的方案数即可。

4 总结

利用分治配合数据结构优化DP是一个经典的思路。但本题的亮点在于分治时并不取中点断开，而是取一个c_k最大的k，从而跨越k点的转移都自然地满足

了下界条件；同时每层分治的开销只与两段中较短者的长度有关，保证了复杂度。