「KDOI-11」& MX-S6 讲解

Daniel_lele

KDOI 出题组

2024/11/17

比赛情况

（以下数据均为梦熊数据）

共有 1 人 AK，（**提交的所有人中**）前 10% 线为 262，前 20% 线为 208，前 30% 线为 175，前 50% 线为 105。

四题通过率分别为 44.3%，36%，3.1%，0.6%，预期难度分别为 $4, 5, 8, 9$。

感谢大家参加比赛！预祝大家在 NOIp 中获得好成绩

题意简述

有 n 个文件和 m 个打印机，第 i 个会在 t_i 时刻下发并需要占用 s_i 打印时间。
每个文件会选择等待时间最短，等待时间相同情况下编号最小的打印机打印。
问每个打印机会打印哪些文件。
$n, m \le 2 \times 10^5$，$s_i, t_i \le 10^9$，t_i 互不相同。
Author: Inaba_Meguru

算法 1

首先，我们会发现后下发的文件不会影响先下发的文件的选择。于是，我们可以将文件按 t_i 排序，记录对应文件编号，并从小到大依次选择。

维护每个打印机上一次完成打印的时间（初始为 0）。那么，我们将所有打印机遍历一遍，求出等待时间的最小值与等待时间相同情况下编号最小的打印机，并选择这个打印机打印即可，注意更新完成打印的时间。

总复杂度 $O(nm)$，期望得分 45 分。

算法 2

考虑如何维护哪个打印机等待时间最小。
维护两个**小根堆** p, q，p 中存储所有**已经完成打印**的打印机编号，q 中存储所有**未完成打印**的打印机编号与对应完成时间。
于是，我们在 p 中有打印机的情况下一定会选择 p 中的打印机，由于我们要取编号最小的，可以直接取出 p 的堆顶。否则，我们会选择 q 中完成时间最小的，完成时间相同情况下编号最小的，也可以直接取出 q 的堆顶。
注意，当 q 堆顶完成时间比目前考虑的 t_i 小，那么要从 q 中弹出并塞入 p。
总复杂度 $O((n+m)\log m)$，期望得分 100 分。

题意简述

在无限上的跑道上有 n 个加油站，在 p_i 位置可以花 t_i 单位时间加 x_i 编号的油。
q 次询问，每次询问假设有一艘飞船在 0 位置，初始速度为 1 距离每单位时间。加 x 编号的油可以将速度乘 x，求飞到 y 位置的最小时间。
$n,q \le 10^5$，$x_i \le 4$，$t_i,y_i,p_i \le 10^9$，p_i 递增，输入均为正整数。
Author: Inaba_Meguru

算法 1

考虑对于每个询问，暴力枚举在哪些加油站加油，那么可以计算每一段路程对应的速度，将所有时间取最小值。总复杂度 $O(q2^n n)$，期望得分 15。

算法 2

以下设 V 为 p_i, y_i 的值域。

考虑动态规划，设 $dp_{i,j}$ 表示飞船飞到 i 位置，速度为 j 的，到 i 的最小时间。

对于转移，如果一个位置有加油站 k，那么可以选择这个加油站，并 $dp_{i,j\times x_k} \leftarrow dp_{i,j} + t_k$。由于只能加油一次，故需使用合适的转移顺序。

总复杂度 $O(qV4^n)$。

算法 2

考虑优化，注意到加油一次的时间 $t_i \geq 1$。
如果目前速度为 v_0，到终点的距离为 s_0，那么若加油，还需耗时为 $\frac{s_0}{v_0 \times x_i} + t_i$；若不加油，还需耗时为 $\frac{s_0}{v_0}$。
当 $v_0 > s_0$ 时，容易得到 $\frac{s_0}{v_0 \times x_i} + t_i > \frac{s_0}{v_0}$。
也就是说，飞船速度不会超过 V，即第二维不超过 V。
总复杂度 $O(qV^2)$。预处理所有 $dp_{i,j}$ 的值即可做到 $O(V^2 + q)$。

算法 2

注意到速度提升只会在加油站处发生，记录所有 i 的 $dp_{i,j}$ 是没有意义的。
改变状态为 $dp_{i,j}$ 表示第 i 个加油站，速度为 j，到这里的最小耗时。
对于查询，我们只需要找出查询位置前的第一个加油站，并用那里所有的 $dp_{i,j}$ 计算答案即可。
总复杂度 $O(nV + q\log n)$，期望得分 15。

算法 3

考察 $x_i \in 1,2$ 的情况。
首先 $x_i = 1$ 不会对飞船速度有任何提升，故没有意义。
由于只会加 $x_i = 2$ 的油，速度一定形如 2^p。不妨改设 dp 状态为 $dp_{i,p}$，表示第 i 个加油站，速度为 2^p，到这里的最小耗时。
总复杂度 $O(n \log V + q \log n)$，期望得分 15。

算法 3

将上述解法推广到 $x_i \in 1, 2, 4$ 的情况。$4 = 2^2$，也就是在转移 4 的时候只需要将 $dp_{i,p}$ 转到 $dp_{i,p+2}$ 即可。
总复杂度 $O(n \log V + q \log n)$，期望得分 35。
将上述解法推广到 $x_i \in 1, 2, 3, 4$ 的情况。考虑增设一维，$dp_{i,p,q}$ 表示第 i 个加油站，速度为 $2^p 3^q$ 的最小耗时。
总复杂度 $O(n \log^2 V + q \log n)$，期望得分 100。

题意简述

给定 n 个字符串 S_i 和一个字符串 T。
定义一个三元组 (l,r,k)（$k \le K$）是好的当且仅当 $T_{[l,r]}$ 可以被分割成 k 个 S_i 的可以为空的前缀。
求好的三元组的数量，并对于每个 i 求出 $l \le i \le r$ 的好的三元组数量。
$n \le 10$，$|S_i| \le 5\times 10^4$，$|T|, K \le 5\times 10^5$，字符集为小写字母。
Author: Daniel_lele

算法 1

设 $f_{l,r,k}$ 为 (l,r,k) 是否符合要求。考虑如何计算。
首先，将 T 从每个位置开始的后缀匹配上所有 S_i，这样可以得出所有 $f_{l,r,1}$ 的值。
对于 $f_{l,r,k}$（$k \geq 2$），考虑从 $f_{l,p,k-1}$ 转移。即，$f_{l,r,k}=1$ 当且仅当存在 p 使得 $f_{l,p,k-1}=1$ 且 $f_{p+1,r,1}=1$。
总复杂度 $O(k|T|^3+n|T||S_i|)$，期望得分 $15\sim 30$。

算法 2

容易发现 $f_{l,r,k}$ 是单调的。即，$f_{l,r,k} \geq f_{l,r+1,k}$，$f_{l,r,k} \leq f_{l,r,k+1}$。后者是容易证明的，前者考虑反证，不妨假设 $f_{l,r,k} = 0$ 且 $f_{l,r+1,k} = 1$，设 $R_1 + R_2 + \cdots + R_k = T_{l,r+1}$，不妨设 R_k 不为空。删去 R_k 的最后一个位置即可得到 $T_{l,r}$，与 $f_{l,r,k} = 0$ 矛盾。

设 $g_{l,k}$ 表示对于 (l,k)，使得 $f_{l,r,k} = 1$ 的最大 r。

$g_{l,1}$ 是容易求出的。$g_{l,k} = \max_{i=l}^{g_{l,k-1}+1} g_{i,1}$。这里可以使用倍增求 RMQ，复杂度为 $O(|T| \log |T|)$（对于每个 k），当然也有 $O(|T|)$ 的四毛子解法，由于超纲，在此不详细阐述。

于是，我们可以 $O(n|T||S_i| + K|T| \log |T|)$ 求出第一问的答案，期望得分 30。

算法 2

考虑第二问，统计出 $h_{l,r}$ 表示 k 至少为多少时 $f_{l,r,k}=1$。同样可以 $O(|T|^2)$ 求出。
然后，做一个二维前缀和即可得到每个 i 对应的好的三元组包含。
总复杂度 $O(n|T||S_i|+K|T|\log|T|+|T|^2)$，期望得分 50。

算法 3

考虑优化 $O(n|T||S_i|)$ 部分。
一种办法是考虑字符串哈希。将 T, S_i 哈希。每次求 T 的某个后缀和 S_i 的 LCP 时二分它的长度，并使用哈希判断是否相等。
这部分复杂度 $O(n|T|\log|S_i|)$。
另一种办法是考虑 exKMP 算法（Z 函数）。该解法复杂度为 $O(n|T|)$。由于超纲，在此不详细阐述。
第一问的总复杂度变为 $O(n|T|\log|S_i| + K|T|\log|T|)$，期望得分 42。

算法 3

考虑优化第二问。
对于一个 $g_{l,k}$，他会使包含每一个 $l \le i \le g_{l,k}$ 的 i 的好的三元组数量增加 $g_{l,k} - i + 1$。
于是，我们可以在 $l+1$ 位置的二次前缀和加 -1，$g_{l,k}+2$ 位置的二次前缀和加 1，l 位置的一次前缀和加上 $g_{l,k} - l + 1$。然后跑两遍前缀和即可。
总复杂度 $O(n|T|\log|S_i| + k|T|\log|T|)$，期望得分 70。

算法 4

考虑 $g_{l,k}$ 的转移点，不妨设为 $tr_{l,k-1}$。即，$g_{l,k}=g_{tr_{l,k-1},1}$。另设 $tr_{i,0}=i$。
于是，我们可以得到 $g_{tr_{l,k-1},1}\ge \max_{i=l}^{tr_{l,k-1}} g_{i,1}$。
考虑 $g_{l,k+1}$ 的转移点，由上可以得到 $tr_{l,k-1}$ 优于 $l\sim tr_{l,k-1}-1$ 中的转移点。
也就是说，
$g_{l,k+1}=\max_{i=tr_{l,k-1}}^{g_{l,k}} g_{i,1}=\max_{i=tr_{l,k-1}}^{g_{tr_{l,k-1},1}} g_{i,1}=g_{tr_{l,k-1},1}$。
即，$tr_{l,k}=tr_{tr_{l,k-1},1}$。

算法 4

考虑将 i 向 $tr_{i,1}$ 连边，那么 i 的 k 级祖先即为 $tr_{i,k}$。
设 $fa_{i,k}$ 表示 i 的 k 级祖先。于是，第一问的答案即为 $\sum_{j=1}^{n}\sum_{i=0}^{K-1}(g_{fa_{j,i},1}-j+1)$。
考虑拆开：$\sum_{j=1}^{n}\sum_{i=0}^{K-1}g_{fa_{j,i},1}-\sum_{j=1}^{n}K(j-1)$。
设 cnt_i 为使得 $fa_{j,k}=i$ 的 $1\le j\le n$，$0\le k<K-1$ 的 (i,j) 数量。那么答案即为 $\sum_{j=1}^{n}cnt_j g_{j,1}-\sum_{j=1}^{n}K(j-1)$。
cnt_i 可以使用树上前缀和计算，注意跳到根之后有一个自环，特判一下即可。
这里前缀和要求树上 k 级祖先，可以倍增 $O(|T|\log|T|)$ 求。当然也有 $O(|T|)$ 长链剖分解法，由于超纲，在此不详细阐述。
第一问的总复杂度为 $O(n|T|\log|S_i|+|T|\log|T|)$，期望得分 60。

算法 4

考虑第二问，根据算法 3 的解法，对于每个 i，我们要在 $i+1$ 位置的二次前缀和加上 $-K$，在 $g_{i,1}+2$ 位置的二次前缀和加上 cnt_i，在 i 位置的一次前缀和加上 $\sum_{j=0}^{K-1} g_{i,j} - K(i-1)$，$\sum_{j=0}^{K-1} g_{i,j}$ 是 $0 \sim K-1$ 级祖先的某值之和，可以使用树上前缀和计算。

总复杂度 $O(n|T|\log|S_i| + |T|\log|T|)$，期望得分 100。

使用四毛子，exKMP，长链剖分即可做到 $O(n|T|)$。

题意简述

给出 n 个点 m 条边的有向无环图，求对于 K^n 种不同的给每个点涂色的方案，每种颜色长度为 l 的链数量的平方和之和，对 $P=998244353$ 取模。

$n\le 300$，$k<P$，$l\le 20$。

Author: Daniel_lele

算法 1

考虑暴力枚举 k^n 种状态的每一种，并暴力枚举每一条长度为 l 的链，求出 cnt_i，并计算答案。
总复杂度 $O(nk^n2^m)$，期望得分 4。

分析

我们需要求 $\sum_{i=1}^{k} cnt_i^2$。将 cnt_i^2 视作从所有颜色为 i 的链中**有序**选出**可以相同**两条的方案数。
考虑交换求和顺序，将问题转化为：求对于每一对长度为 l 的链，他们颜色相同的方案数之和。

算法 2

考虑找出所有长度为 l 的链，暴力枚举两条，并算出他们颜色相同的方案数。
此时，这两条链上的点颜色必须相同，其他点没有限制。
设这两条链的并中的点有 t 个。那么他们颜色相同的方案数即为 k^{n-t+1}。
总复杂度 $O(C^2)$，其中 C 为链的数量，可以发现 $1 \sim 3$，$10 \sim 12$ 测试点中 C 均不大，此做法可以通过，期望得分 24。

算法 3

两条链并的颜色个数是 $2l$ 减去两条链交的颜色个数。
于是，我们要关注两条链的交。
对于 $l=1$，$l=2$ 和 $l=3$ 的情况，可以暴力枚举两条链交的大致情况。由于细节繁多，在此不一一列举。
以上做法期望得分 $8\sim 32$。

算法 4

考虑动态规划。
设计 $dp_{i_1,i_2,j_1,j_2,k}$ 表示两条链目前链头是 i_1,i_2，长度分别为 j_1,j_2，重叠了 k 次的方案数。
我们希望能够准确计算所有重叠的时刻，也就是两条链的交。
取出原图的拓扑序，在拓扑序上 dp。每次将 i_1,i_2 中拓扑序靠前的一个转移，并枚举这条链的下一个点。
对于转移前 $i_1=i_2$ 的情况，钦定先转移 i_1。如果 $j_1=l$ 或 $j_2=l$，那么不论谁的拓扑序靠前都要转移那个还没满的链。
对于转移后 $i_1=i_2$ 的情况，将 k 转移的时候同样加一。
这样，我们就可以不重不漏的计算出所有重叠 k 次的方案。总复杂度 $O(n^3l^3)$，期望得分 $48\sim 56$。

算法 5

考虑优化算法 4。发现 $dp_{i_1,i_2,l,l,k}$ 给答案的贡献为 $K^{n-2l+k+1}$，也就是 $K^{n-2l+1}\times K^k$。
换句话说，每重叠一次，方案数都会乘 K。
于是，我们可以省略上述 dp 的最后一维，对于转移后 $i_1=i_2$ 的情况转移系数乘 K 即可。
总复杂度 $O(n^3l^2)$，期望得分 $56\sim 64$。

算法 6

上述 dp 的主要问题是需要记录两条链的端点，然后转移还需要 $O(n)$。

注意到我们仅需考虑两条链的交，其他链上的点对我们不重要。

设计 dp_{i,j_1,j_2,j_3} 表示两条链上一次相交是在 i 点，到 i 点为止两条链长度分别为 j_1,j_2，重叠了 j_3 个点。

考虑转移，同样按拓扑序做，我们枚举两条链下一次相交的点 i'，求出 $i \to i'$ 的长度为 k_1,k_2 的链的数量，并转移到 $dp_{i',j_1+k_1,j_2+k_2,j_3+1}$。

考虑使用另一个 $f_{i,j,k}$ 预处理 $i \to j$ 的长度为 k 的链的数量。具体地，$f_{i,j,k}$ 可以转移到 $f_{i,p,k+1}$，系数为 $j \to p$ 的边的数量。这么做是 $O(n^3l)$ 的。

算法 6

然而，上述做法是错误的。
考虑什么情况会出错。由于我们转移的时候分开考虑了 $i \to i'$ 的长度为 k_1, k_2 的链的数量，如果在其中还有交那么就会将 j_3 算小。同时，这样也会导致重复计算。

算法 6

考虑容斥。
改变状态设计，设 dp_{i,j_1,j_2,j_3} 表示两条链上一次相交是在 i 点，到 i 点为止两条链长度分别为 j_1,j_2，并钦定了重叠的 j_3 个点。
转移没有变化。
设 g'_j 表示钦定重叠了 j 个点的情况，$'_j$ 表示恰好重叠了 j 个点的情况。通过二项式反演计算出 $'_j$ 并求出答案。
注意，我们还需要统计以 i 为起点/终点的长度为 k 的链的数量，以计算 $dp_{i,j_1,j_2,1}$ 的初值以及将 dp_{i,j_1,j_2,j_3} 统计入 g'_{j_3}。
g'_0 可以使用预处理的 $f_{i,j,k}$ 求出。
总复杂度 $O(n^3l+n^2l^5)$，期望得分 $64\sim 80$。

算法 7

考虑优化算法 6。
我们发现枚举 k_1, k_2 是浪费的。考虑分步转移，额外设一个 tmp_{j_1,j_2,j_3} 数组。
先将 dp_{i,j_1,j_2,j_3} 转移到 tmp_{j_1+k,j_2,j_3}，再将 tmp_{j_1,j_2,j_3} 转移到 dp_{i',j_1,j_2+k,j_3+1}。
两次转移的系数均为 $f_{i,i',k}$。
总复杂度 $O(n^3l + n^2l^4)$，期望得分 $76 \sim 92$。

算法 8

考虑类比算法 5 优化算法 6。
写出统计答案的式子：$g_i=\sum_{j=i}^{l}(-1)^{j-i}\binom{j}{i}g'_j$，
$ans=\sum_{i=0}^{l}g_iK^{n-2l+1+i}$。

$$\begin{aligned}ans&=\sum_{i=0}^{l}\sum_{j=i}^{l}(-1)^{j-i}\binom{j}{i}g'_jK^{n-2l+1+i}\\&=K^{n-2l+1}\sum_{i=0}^{n}\sum_{j=i}^{l}(-1)^{j-i}\binom{j}{i}g'_jK^{i}\end{aligned}$$

算法 8

$$
\begin{aligned}
&= K^{n-2l+1}\sum_{i=0}^{n}\sum_{j=i}^{n}K^j g'_j(-1)^{j-i}\binom{j}{i}(\frac{1}{K})^{j-i} \\
&= K^{n-2l+1}\sum_{j=0}^{n}K^j g'_j\sum_{i=0}^{j}\binom{j}{i}(-\frac{1}{K})^{j-i} \\
&= K^{n-2l+1}\sum_{j=0}^{n}K^j g'_j(1-\frac{1}{K})^j \\
&= K^{n-2l+1}\sum_{j=0}^{n}g'_j(K-1)^j
\end{aligned}
$$

算法 8

也就是说，我们发现，每重叠一次，都会给答案 $K-1$ 的贡献。
于是，我们可以省略最后一维，并将转移系数乘 $K-1$。
总复杂度 $O(n^3l+n^2l^4)$，期望得分 $76\sim92$。
当然，上述结论也可以通过一些组合意义理解。

算法 9

结合算法 7 和算法 8 的优化，即可做到 $O(n^3l + n^2l^3)$，期望得分 100。

也可以使用 NTT 将理论复杂度进一步优化到 $O(n^3l + n^2l^2 \log n)$。

吐槽&答疑

如果有任何问题/想吐槽的欢迎发在聊天区!