# NOI 模拟赛试题讨论

wzj52501

Peking University

2018 年 11 月 26 日

# Results

# password

## 玄机

求出至少包含 $m$ 个模板串各一次的长度为 $n$ 的匹配串有多少个，答案对 $998244353$ 取模。

$m \leq 4, \sum |str| \leq 50, n \leq 10^9$

# 20pts

- 对于这一部分数据，$n \leq 7$。

# 20pts

- 对于这一部分数据，$n \leq 7$。
- 直接枚举长度为 $n$ 的所有密码串，分别检验即可。

# 20pts

- 对于这一部分数据，$n \leq 7$。
- 直接枚举长度为 $n$ 的所有密码串，分别检验即可。
- 时间复杂度为 $O(n^2 \times 10^n)$。

# 进一步的分析

- 面对这种多串匹配的问题，我们自然联想到了 AC 自动机。

# 进一步的分析

- 面对这种多串匹配的问题，我们自然联想到了 AC 自动机。
- 首先对 $m$ 个串建立 AC 自动机模型，这样我们就可以把问题转化成下面的形式：
  - 在一个 DAG 上行走 $n$ 步，要求所经过的结点包含了 $m$ 个给定结点。

# 40pts

- 对于这一部分数据，$n \leq 2501$。

# 40pts

- 对于这一部分数据，$n \leq 2501$。
- 设 $f(i, j, S)$ 表示已经走了 $i$ 步，当前在结点 $j$，已经包含的特殊结点集合为 $S$ 的方案数。

# 40pts

- 对于这一部分数据，$n \leq 2501$。
- 设 $f(i, j, S)$ 表示已经走了 $i$ 步，当前在结点 $j$，已经包含的特殊结点集合为 $S$ 的方案数。
- 转移时枚举第 $i+1$ 步走了哪一条边（相当于枚举密码串的第 $i+1$ 位填写了哪个数字），如果是数字 $c$ 对应的那条边，则 $f(i, j, S) \to f(i+1, trans(j, c), S_2)$。

# 40pts

- 对于这一部分数据，$n \leq 2501$。
- 设 $f(i, j, S)$ 表示已经走了 $i$ 步，当前在结点 $j$，已经包含的特殊结点集合为 $S$ 的方案数。
- 转移时枚举第 $i+1$ 步走了哪一条边（相当于枚举密码串的第 $i+1$ 位填写了哪个数字），如果是数字 $c$ 对应的那条边，则 $f(i, j, S) \to f(i+1, trans(j, c), S_2)$。
- 初始将 $f(0, 0, 0)$ 设为 $1$，答案即为 $\sum_{j=0}^{\sum |str|} f(n, j, all)$。

# 40pts

- 对于这一部分数据，$n \leq 2501$。
- 设 $f(i, j, S)$ 表示已经走了 $i$ 步，当前在结点 $j$，已经包含的特殊结点集合为 $S$ 的方案数。
- 转移时枚举第 $i+1$ 步走了哪一条边（相当于枚举密码串的第 $i+1$ 位填写了哪个数字），如果是数字 $c$ 对应的那条边，则 $f(i, j, S) \to f(i+1, trans(j, c), S_2)$。
- 初始将 $f(0, 0, 0)$ 设为 $1$，答案即为 $\sum_{j=0}^{\sum |str|} f(n, j, all)$。
- 时间复杂度为 $O(n \times \sum |str| \times 2^m)$。

# 80 - 90pts

- 对于这一部分数据，$m \leq 2$。

# 80 - 90pts

- 对于这一部分数据，$m \leq 2$。
- 以上的递推算法显然可以通过矩阵乘法加速，如果常数较好可以通过 $m = 3$ 的数据。

# 80 - 90pts

- 对于这一部分数据，$m \leq 2$。
- 以上的递推算法显然可以通过矩阵乘法加速，如果常数较好可以通过 $m = 3$ 的数据。
- 时间复杂度为 $O(k^3 \log n)$，其中 $k = \sum |str| \times 2^m$。

# 100pts

- 我们发现以上的算法瓶颈在于矩阵的阶太高，考虑用容斥原理代替状态压缩。

# 100pts

- 我们发现以上的算法瓶颈在于矩阵的阶太高，考虑用容斥原理代替状态压缩。
- 设 $g(S)$ 表示肯定不会包含 $S$ 中的模板串的长度为 $n$ 的密码串数，则 $ans = \sum (-1)^{|S|} \times g(S)$。

# 100pts

- 我们发现以上的算法瓶颈在于矩阵的阶太高，考虑用容斥原理代替状态压缩。
- 设 $g(S)$ 表示肯定不会包含 $S$ 中的模板串的长度为 $n$ 的密码串数，则 $ans = \sum (-1)^{|S|} \times g(S)$。
- $g(S)$ 可以很容易用矩阵乘法加速动态规划的算法求出，时间复杂度为 $O(2^m \times k^3 \log n)$，其中 $k = \sum |str|$。

## 画心

渠是一名画师。渠有一支神奇的画笔，可以画尽因果。
渠要画一幅画，这幅画由 $n$ 个线段组成，线段从 $1$ 开始编号，第 $i$ 条线段有一个特殊的因果值 $A_i$。
渠打算将这 $n$ 条线段分成若干组来画，每一组的长度要求在 $[l, r]$ 之间，且必须是编号连续的一段。
对于因果值之和为 $x$ 的一组线段，渠画完后种下的因果为 $ax^2 + bx + c$。渠想知道渠将这 $n$ 条线段画完，最多收获多少因果呢？
$1 \leq l \leq r \leq n \leq 152501, |A_i| \leq 10^9$

# 20pts

- 对于这一部分数据，$n \leq 10$。

# 20pts

- 对于这一部分数据，$n \leq 10$。
- 直接枚举分段情况，简单计算即可。

# 20pts

- 对于这一部分数据，$n \leq 10$。
- 直接枚举分段情况，简单计算即可。
- 时间复杂度为 $O(n \times 2^n)$。

# 40pts

- 对于这一部分数据，$n \leq 501$。

# 40pts

- 对于这一部分数据，$n \leq 501$。
- 设 $f_i$ 表示前 $i$ 条线段全画完最多收获的因果数，设 $S_i$ 表示 $A_i$ 的前缀和。

# 40pts

- 对于这一部分数据，$n \leq 501$。
- 设 $f_i$ 表示前 $i$ 条线段全画完最多收获的因果数，设 $S_i$ 表示 $A_i$ 的前缀和。
- 易得 $f_i = max\left\{f_j + a \times (S_i - S_j)^2 + b(S_i - S_j) + c\right\}, j \in [i - r, i - l]$。

# 40pts

- 对于这一部分数据，$n \leq 501$。
- 设 $f_i$ 表示前 $i$ 条线段全画完最多收获的因果数，设 $S_i$ 表示 $A_i$ 的前缀和。
- 易得 $f_i = max\left\{f_j + a \times (S_i - S_j)^2 + b(S_i - S_j) + c\right\}, j \in [i - r, i - l]$。
- 时间复杂度为 $O(n^2)$。

# extra 20pts

- 对于这一部分数据，$l = 1, r = n, 0 \leq A_i$。

# extra 20pts

- 对于这一部分数据，$l = 1, r = n, 0 \leq A_i$。
- 这是一个经典的斜率优化问题，来源是 APIO 特别行动队。

# extra 20pts

- 对于这一部分数据，$l = 1, r = n, 0 \leq A_i$。
- 这是一个经典的斜率优化问题，来源是 APIO 特别行动队。
- 我们发现 $b$ 在这里面没什么用，我们可以将它忽视，最后答案再加上 $S_n \times b$。

# extra 20pts

- 对于这一部分数据，$l = 1, r = n, 0 \leq A_i$。
- 这是一个经典的斜率优化问题，来源是 APIO 特别行动队。
- 我们发现 $b$ 在这里面没什么用，我们可以将它忽视，最后答案再加上 $S_n \times b$。
- 然后把式子变形，$f_i = max\left\{f_j + a \times S_j^2 - 2aS_j \times S_i\right\} + a \times S_i^2 + c$。设 $y_j = f_j + a \times S_j^2$，$x_j = 2aS_j$，$k = S_i$，则 $j$ 向 $i$ 转移的贡献可以认为是一条过 $(x_j, y_j)$ 且斜率为 $k$ 的直线在 y 轴上的截距。

# extra 20pts

- 对于这一部分数据，$l = 1, r = n, 0 \leq A_i$。
- 这是一个经典的斜率优化问题，来源是 APIO 特别行动队。
- 我们发现 $b$ 在这里面没什么用，我们可以将它忽视，最后答案再加上 $S_n \times b$。
- 然后把式子变形，$f_i = max\left\{f_j + a \times S_j^2 - 2aS_j \times S_i\right\} + a \times S_i^2 + c$。设 $y_j = f_j + a \times S_j^2$，$x_j = 2aS_j$，$k = S_i$，则 $j$ 向 $i$ 转移的贡献可以认为是一条过 $(x_j, y_j)$ 且斜率为 $k$ 的直线在 y 轴上的截距。
- 由于数据有很好的单调性，可以很容易地维护答案的凸壳。

# extra 20pts

- 对于这一部分数据，$l = 1, r = n, 0 \leq A_i$。
- 这是一个经典的斜率优化问题，来源是 APIO 特别行动队。
- 我们发现 $b$ 在这里面没什么用，我们可以将它忽视，最后答案再加上 $S_n \times b$。
- 然后把式子变形，$f_i = max\left\{f_j + a \times S_j^2 - 2aS_j \times S_i\right\} + a \times S_i^2 + c$。设 $y_j = f_j + a \times S_j^2$，$x_j = 2aS_j$，$k = S_i$，则 $j$ 向 $i$ 转移的贡献可以认为是一条过 $(x_j, y_j)$ 且斜率为 $k$ 的直线在 y 轴上的截距。
- 由于数据有很好的单调性，可以很容易地维护答案的凸壳。
- 时间复杂度为 $O(n)$。

# extra 20pts

- 对于这一部分数据，$l = 1, r = n$。

# extra 20pts

- 对于这一部分数据，$l = 1, r = n$。
- 由于单调性的丧失，不能直接用单调队列维护凸壳，可以考虑使用 CDQ 分治来解决。

# extra 20pts

- 对于这一部分数据，$l = 1, r = n$。
- 由于单调性的丧失，不能直接用单调队列维护凸壳，可以考虑使用 CDQ 分治来解决。
- 介于此算法与最终解法类似而适用性较差，此处暂略。

# extra 20pts

- 对于这一部分数据，$l = 1, r = n$。
- 由于单调性的丧失，不能直接用单调队列维护凸壳，可以考虑使用 CDQ 分治来解决。
- 介于此算法与最终解法类似而适用性较差，此处暂略。
- 时间复杂度为 $O(n \log^2 n)$ 或 $O(n \log n)$。

# 100pts

- 我们在 DP 的时候，发现某个状态 $f_j$ 会更新一段连续区间中的 $f_i$，这与线段树的区间查询很相似。

# 100pts

- 我们在 DP 的时候，发现某个状态 $f_j$ 会更新一段连续区间中的 $f_i$，这与线段树的区间查询很相似。
- 我们对序列建立线段树结构，对于每个状态 $f_j$，将其影响的 $f_i$ 的区间在线段树上打上标记。最后中序遍历线段树，如果当前节点对应的区间是 $[l, r]$，则用完全覆盖在该区间标记上的 $f_j$ 构建出凸壳，并更新 $[l, r]$ 区间中的所有 $f_i$。

# 100pts

- 我们在 DP 的时候，发现某个状态 $f_j$ 会更新一段连续区间中的 $f_i$，这与线段树的区间查询很相似。
- 我们对序列建立线段树结构，对于每个状态 $f_j$，将其影响的 $f_i$ 的区间在线段树上打上标记。最后中序遍历线段树，如果当前节点对应的区间是 $[l, r]$，则用完全覆盖在该区间标记上的 $f_j$ 构建出凸壳，并更新 $[l, r]$ 区间中的所有 $f_i$。
- 这就是喜闻乐见的线段树分治算法，在许多离线问题上都有着重要的应用。

# 100pts

- 我们在 DP 的时候，发现某个状态 $f_j$ 会更新一段连续区间中的 $f_i$，这与线段树的区间查询很相似。
- 我们对序列建立线段树结构，对于每个状态 $f_j$，将其影响的 $f_i$ 的区间在线段树上打上标记。最后中序遍历线段树，如果当前节点对应的区间是 $[l, r]$，则用完全覆盖在该区间标记上的 $f_j$ 构建出凸壳，并更新 $[l, r]$ 区间中的所有 $f_i$。
- 这就是喜闻乐见的线段树分治算法，在许多离线问题上都有着重要的应用。
- 时间复杂度为 $O(n \log^2 n)$。

## 求索

给定一棵 $n$ 个结点的无根树，边上有权值 $w_i$。求一条路径满足路径上边权的最小值与路径边权之和的乘积尽量大，且这条路径上的边权的极差不能超过 $lim$，输出最大乘积。

$n \leq 152501, 0 \leq w_i, lim \leq 52501$

# 20pts

- 对于这一部分数据，$n \leq 2501$。

# 20pts

- 对于这一部分数据，$n \leq 2501$。
- 按题意枚举所有可行路径即可。

# 20pts

- 对于这一部分数据，$n \leq 2501$。
- 按题意枚举所有可行路径即可。
- 时间复杂度为 $O(n^2)$。

# extra 20pts

- 对于这一部分数据，保证树是一条链且无 *lim* 限制。

# extra 20pts

- 对于这一部分数据，保证树是一条链且无 *lim* 限制。
- 枚举路径上边权的最小值，利用单调队列可以计算出固定最小值的序列向左右延伸最多到达的位置，计算答案即可。

# extra 20pts

- 对于这一部分数据，保证树是一条链且无 $lim$ 限制。
- 枚举路径上边权的最小值，利用单调队列可以计算出固定最小值的序列向左右延伸最多到达的位置，计算答案即可。
- 时间复杂度为 $O(n)$。

# extra 20pts

- 对于这一部分数据，边权只有 $0$ 和 $1$ 两种情况且无 *lim* 限制。

# extra 20pts

- 对于这一部分数据，边权只有 $0$ 和 $1$ 两种情况且无 *lim* 限制。
- 边权为 $0$ 的边肯定不会出现在答案中，所以问题转化成在一棵森林中寻找直径，使用经典的两次 BFS 算法即可。

# extra 20pts

- 对于这一部分数据，边权只有 $0$ 和 $1$ 两种情况且无 *lim* 限制。
- 边权为 $0$ 的边肯定不会出现在答案中，所以问题转化成在一棵森林中寻找直径，使用经典的两次 BFS 算法即可。
- 时间复杂度为 $O(n)$。

# 80 pts

- 对于这一部分数据，无 *lim* 限制。

# 80 pts

- 对于这一部分数据，无 *lim* 限制。
- 考虑使用点分治算法，问题转化成寻找经过分治中心的路径的最大乘积，对于每一条从分治中心出发的路径，将 (路径边权的最小值，路径边权之和) 作为二元组加入合适的数据结构中，动态维护前缀最大值即可。

# 80 pts

- 对于这一部分数据，无 *lim* 限制。
- 考虑使用点分治算法，问题转化成寻找经过分治中心的路径的最大乘积，对于每一条从分治中心出发的路径，将 (路径边权的最小值，路径边权之和) 作为二元组加入合适的数据结构中，动态维护前缀最大值即可。
- 时间复杂度为 $O(n \log^2 n)$。

# 进一步的思考

我们先给出一个结论：

- 对于树上两个不相交的点集 $s$、$t$，设集合 $s$ 内任意两点间路径长度最长的路径之一为 $(x_1 \to y_1)$，集合 $t$ 内任意两点间路径长度最长的路径之一为 $(x_2 \to y_2)$。则 $s \cup t$ 集合内任意两点间路径长度最长的路径之一必定在集合 $\{(x_1 \to y_1), (x_2 \to y_2), (x_1 \to x_2)\} \cup \{(x_1 \to y_2), (x_2 \to x_1), (x_2 \to y_1)\}$ 中。

# 证明

反证法，设 $s \cup t$ 内有一条边权最大的路径 $(u \to v)$ 不属于答案集合且满足其边权大于集合内任意一条路径。

# 证明

反证法，设 $s \cup t$ 内有一条边权最大的路径 $(u \to v)$ 不属于答案集合且满足其边权大于集合内任意一条路径。

1. 若 $u, v$ 同属于 $s$ 或 $t$ 集合，设它们同属于集合 $s$，则 $(u \to v)$ 路径会比 $(x_1 \to y_1)$ 优，这显然与假设相悖。

# 证明

反证法，设 $s \cup t$ 内有一条边权最大的路径 $(u \to v)$ 不属于答案集合且满足其边权大于集合内任意一条路径。

1. 若 $u, v$ 同属于 $s$ 或 $t$ 集合，设它们同属于集合 $s$，则 $(u \to v)$ 路径会比 $(x_1 \to y_1)$ 优，这显然与假设相悖。
2. 若 $u, v$ 不同属于 $s$ 或 $t$ 集合，设 $u$ 属于集合 $s$。

# 证明

反证法，设 $s \cup t$ 内有一条边权最大的路径 $(u \to v)$ 不属于答案集合且满足其边权大于集合内任意一条路径。

1. 若 $u, v$ 同属于 $s$ 或 $t$ 集合，设它们同属于集合 $s$，则 $(u \to v)$ 路径会比 $(x_1 \to y_1)$ 优，这显然与假设相悖。
2. 若 $u, v$ 不同属于 $s$ 或 $t$ 集合，设 $u$ 属于集合 $s$。
    1. 若 $x_1, y_1$ 任意一个在路径 $(u \to v)$ 上，设 $x_1$ 在路径上，我们取 $(y_1 \to v)$ 会优于路径 $(u \to v)$。

# 证明

反证法，设 $s \cup t$ 内有一条边权最大的路径 $(u \to v)$ 不属于答案集合且满足其边权大于集合内任意一条路径。

1. 若 $u, v$ 同属于 $s$ 或 $t$ 集合，设它们同属于集合 $s$，则 $(u \to v)$ 路径会比 $(x_1 \to y_1)$ 优，这显然与假设相悖。
2. 若 $u, v$ 不同属于 $s$ 或 $t$ 集合，设 $u$ 属于集合 $s$。
   1. 若 $x_1, y_1$ 任意一个在路径 $(u \to v)$ 上，设 $x_1$ 在路径上，我们取 $(y_1 \to v)$ 会优于路径 $(u \to v)$。
   2. 若 $x_1, y_1$ 均不在路径 $(u \to v)$ 上，则路径 $(y_1 \to v)$ 和 $(x_1 \to v)$ 中肯定存在一个比路径 $(u \to v)$ 更优。

# 证明

反证法，设 $s \cup t$ 内有一条边权最大的路径 $(u \to v)$ 不属于答案集合且满足其边权大于集合内任意一条路径。

1. 若 $u, v$ 同属于 $s$ 或 $t$ 集合，设它们同属于集合 $s$，则 $(u \to v)$ 路径会比 $(x_1 \to y_1)$ 优，这显然与假设相悖。
2. 若 $u, v$ 不同属于 $s$ 或 $t$ 集合，设 $u$ 属于集合 $s$。
   1. 若 $x_1, y_1$ 任意一个在路径 $(u \to v)$ 上，设 $x_1$ 在路径上，我们取 $(y_1 \to v)$ 会优于路径 $(u \to v)$。
   2. 若 $x_1, y_1$ 均不在路径 $(u \to v)$ 上，则路径 $(y_1 \to v)$ 和 $(x_1 \to v)$ 中肯定存在一个比路径 $(u \to v)$ 更优。

# Q.E.D

# 100 pts

- 有了以上的结论，可以很容易地在加入边的同时维护直径，从而顺利通过 $80pts$ 的任务。

# 100 pts

- 有了以上的结论，可以很容易地在加入边的同时维护直径，从而顺利通过 $80pts$ 的任务。
- 加入 *lim* 限制后，我们顺序枚举最小边权时需要删除已经不合条件的边。可以使用和上题类似的办法进行线段树分治，同时用按秩合并的并查集（为了撤销操作）维护答案。

# 100 pts

- 有了以上的结论，可以很容易地在加入边的同时维护直径，从而顺利通过 $80pts$ 的任务。
- 加入 $lim$ 限制后，我们顺序枚举最小边权时需要删除已经不合条件的边。可以使用和上题类似的办法进行线段树分治，同时用按秩合并的并查集（为了撤销操作）维护答案。
- 时间复杂度为 $O(n\log^2 n)$。

# Q & A