《strakf》命题报告

佛山石中李子豪

2016 年 4 月 25 日

前言

前言

前言

本文主要写了两道字符串与数据结构结合的题目，以及KD-tree的一些调参。

试题大意

试题大意

试题大意

给了一个长度为N的仅由小写字母组成的初始字符串S.

试题大意

给了一个长度为N的仅由小写字母组成的初始字符串S.
之后有M次操作，操作分为三类：

试题大意

给了一个长度为N的仅由小写字母组成的初始字符串S.

之后有M次操作，操作分为三类：

1　A:将原串S的所有等于A的子串权值+1；

试题大意

给了一个长度为N的仅由小写字母组成的初始字符串S.

之后有M次操作，操作分为三类：

1 A:将原串S的所有等于A的子串权值+1；

2 l r:将原串S的所有等于$S[l,r]$的子串权值+1；

试题大意

给了一个长度为N的仅由小写字母组成的初始字符串S.

之后有M次操作，操作分为三类：

1 A:将原串S的所有等于A的子串权值+1；

2 l r:将原串S的所有等于$S[l,r]$的子串权值+1；

3 a b:询问原串S所有子串$S[l,r]$满足$a \leq l \leq r \leq b$的权值和。

试题大意

给了一个长度为N的仅由小写字母组成的初始字符串S.

之后有M次操作，操作分为三类：

1 A:将原串S的所有等于A的子串权值+1；

2 l r:将原串S的所有等于$S[l,r]$的子串权值+1；

3 a b:询问原串S所有子串$S[l,r]$满足$a \le l \le r \le b$的权值和。

数据规模

数据规模

数据规模

数据点编号	N, M	操作类型	数据特点
1	5	2,3	无
2	10	2,3	无
3	100	2,3	无
4	$2*10^5$	1,2	无
5	$2*10^4$	1,3	所有的3操作a=1
6	$2*10^4$	1,3	所有的3操作b=N
7-8	$2*10^4$	1,3	所有的1操作的字符串长度为不下降序列
9-10	$6*10^3$	2,3	无
11-12	$2*10^4$	2,3	所有的3操作b=N
13-15	$2*10^4$	2,3	所有的2操作的字符串长度为不下降序列
16-20	$2*10^4$	2,3	无

数据规模

数据点编号	N, M	操作类型	数据特点
1	5	2,3	无
2	10	2,3	无
3	100	2,3	无
4	$2 * 10^5$	1,2	无
5	$2 * 10^4$	1,3	所有的3操作a=1
6	$2 * 10^4$	1,3	所有的3操作b=N
7-8	$2 * 10^4$	1,3	所有的1操作的字符串长度为不下降序列
9-10	$6 * 10^3$	2,3	无
11-12	$2 * 10^4$	2,3	所有的3操作b=N
13-15	$2 * 10^4$	2,3	所有的2操作的字符串长度为不下降序列
16-20	$2 * 10^4$	2,3	无

输入文件大小不超过0.3M。

出题想法来源

出题想法来源

一道题目：

出题想法来源

一道题目：

给了一个长度为N的仅有小写字母组成的字符串S.
定义$S[l][r]$为S从l到r的子串。

出题想法来源

一道题目：

给了一个长度为N的仅有小写字母组成的字符串S.

定义$S[l][r]$为S从l到r的子串。

有q次询问，每次对一个区间$[l_i, r_i]$进行询问，询问最大的x满足$S[l][l+x-1] = S[r-x+1][r]$.

出题想法来源

一道题目：

给了一个长度为N的仅有小写字母组成的字符串S.

定义$S[l][r]$为S从l到r的子串。

有q次询问，每次对一个区间$[l_i, r_i]$进行询问，询问最大的x满足$S[l][l+x-1] = S[r-x+1][r]$.

$N, q \leq 2 * 10^5$

题解

题解

这道题，假设对于一个询问区间$[l, r]$,那么我们实际上就是求一个最小的x满足以l开始的后缀以及与x开始的后缀的最长公共前缀$> r - x$.

因此，我们可以从小到大逐一枚举x,然后离线按顺序处理所有的询问。

题解

这道题，假设对于一个询问区间$[l, r]$,那么我们实际上就是求一个最小的x满足以l开始的后缀以及与x开始的后缀的最长公共前缀$> r - x$.

因此，我们可以从小到大逐一枚举x,然后离线按顺序处理所有的询问。

我们对询问以左端点进行排序，然后若当前枚举到i,那么则把左端点$< i$的询问插入，然后就删去所有满足答案可以为i的询问。那么，就可以解决了。

因此，我们现在的问题就是要去维护一个可以求出满足答案可以为i的询问。

题解

题解

而根据上面所说，我们有这么一条式子：$i + len - 1 \geq r$, len是最长公共前缀，即$i \geq r - len + 1$.

因此，我们只需要有办法维护$r - len + 1$的值，并从小到大取出即可。

题解

而根据上面所说，我们有这么一条式子：$i + len - 1 \geq r$, len是最长公共前缀，即$i \geq r - len + 1$.

因此，我们只需要有办法维护$r - len + 1$的值，并从小到大取出即可。

对于这个问题，我们可以先进行后缀数组并求出$height$数组。

题解

题解

然后，根据*height*，我们可以构造出一棵树。大概就是这样：

题解

然后，根据*height*，我们可以构造出一棵树。大概就是这样：

假设字符串为*ababcba*,那么*height*数组为：

题解

然后，根据*height*，我们可以构造出一棵树。大概就是这样：

假设字符串为*ababcba*,那么*height*数组为：

0 a

1 ababcba

2 abcba

0 ba

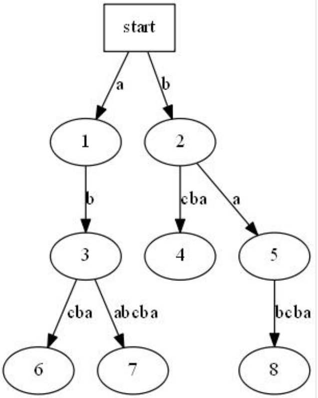
2 babcba

1 bcba

题解

题解

然后构造的树为：



感受一下大概就懂了。

题解

题解

之后，我们对于一个询问$[l, r]$的插入，则先找出l所对应的节点，然后从这个点不断往根，对于路径上每个节点插入一个pair为$(r - len, num)$,len为这个节点能表示的最长串。

题解

之后，我们对于一个询问$[l, r]$的插入，则先找出l所对应的节点，然后从这个点不断往根，对于路径上每个节点插入一个pair为$(r - len, num)$,len为这个节点能表示的最长串。

然后，对于处理x为i的询问，则从x对应的节点，不断往根走，对于路径上每个节点询问比i小的所有pair，然后更新对应答案，删除对应询问即可。

题解

之后，我们对于一个询问$[l, r]$的插入，则先找出l所对应的节点，然后从这个点不断往根，对于路径上每个节点插入一个pair为$(r - len, num)$,len为这个节点能表示的最长串。

然后，对于处理x为i的询问，则从x对应的节点，不断往根走，对于路径上每个节点询问比i小的所有pair，然后更新对应答案，删除对应询问即可。

然后，这个问题，我们可以用树链剖分和dfs序来解决。

题解

之后，我们对于一个询问$[l, r]$的插入，则先找出l所对应的节点，然后从这个点不断往根，对于路径上每个节点插入一个pair为$(r - len, num)$,len为这个节点能表示的最长串。

然后，对于处理x为i的询问，则从x对应的节点，不断往根走，对于路径上每个节点询问比i小的所有pair，然后更新对应答案，删除对应询问即可。

然后，这个问题，我们可以用树链剖分和dfs序来解决。

我们大概可以分为三类询问：

题解

之后，我们对于一个询问$[l, r]$的插入，则先找出l所对应的节点，然后从这个点不断往根，对于路径上每个节点插入一个pair为$(r - len, num)$,len为这个节点能表示的最长串。

然后，对于处理x为i的询问，则从x对应的节点，不断往根走，对于路径上每个节点询问比i小的所有pair，然后更新对应答案，删除对应询问即可。

然后，这个问题，我们可以用树链剖分和dfs序来解决。

我们大概可以分为三类询问：

1.询问子树的最小值；

题解

之后，我们对于一个询问$[l, r]$的插入，则先找出l所对应的节点，然后从这个点不断往根，对于路径上每个节点插入一个pair为$(r - len, num)$,len为这个节点能表示的最长串。

然后，对于处理x为i的询问，则从x对应的节点，不断往根走，对于路径上每个节点询问比i小的所有pair，然后更新对应答案，删除对应询问即可。

然后，这个问题，我们可以用树链剖分和dfs序来解决。

我们大概可以分为三类询问：

1.询问子树的最小值；

2.询问重链上节点的最小值；

题解

之后，我们对于一个询问$[l, r]$的插入，则先找出l所对应的节点，然后从这个点不断往根，对于路径上每个节点插入一个pair为$(r - len, num)$,len为这个节点能表示的最长串。

然后，对于处理x为i的询问，则从x对应的节点，不断往根走，对于路径上每个节点询问比i小的所有pair，然后更新对应答案，删除对应询问即可。

然后，这个问题，我们可以用树链剖分和dfs序来解决。

我们大概可以分为三类询问：

1.询问子树的最小值；

2.询问重链上节点的最小值；

3.询问重链上节点的非重儿子的所有儿子的最小值。

题解

题解

然后，对于这三类问题，则可以分别通过整棵树的dfs序、轻重链、轻儿子dfs序来解决。

题解

然后，对于这三类问题，则可以分别通过整棵树的dfs序、轻重链、轻儿子dfs序来解决。

因此，就可以通过$O((N+M)log_2^2 N)$解决了。

小结

小结

这一题比较好的结合了字符串和数据结构的问题。比较考验选手代码能力。

回到原题

算法介绍

第1-3个数据点

第1-3个数据点

采取*bruteforce*的方法。

对于每一个修改操作，则把所有符合的子串的权值+1.

对于每一个询问操作，则直接询问范围内的所有子串的权值和。

第1-3个数据点

采取*bruteforce*的方法。

对于每一个修改操作，则把所有符合的子串的权值+1.

对于每一个询问操作，则直接询问范围内的所有子串的权值和。

时间复杂度为$O(n^3)$,预计得分15分。

第4个数据点

第4个数据点

注意到这个数据点并没有操作3，因此直接结束即可。

第4个数据点

注意到这个数据点并没有操作3，因此直接结束即可。

算上前面的15分，预计得分20分。

第5-8个数据点

第5-8个数据点

这一类数据点的特点在于只有操作1的修改，而没有操作2的修改。

第5-8个数据点

这一类数据点的特点在于只有操作1的修改，而没有操作2的修改。
分析数据特点:

第5-8个数据点

这一类数据点的特点在于只有操作1的修改，而没有操作2的修改。

分析数据特点:

由于输入保证不超过0.3M,可以得到插入总长度不超过$3 * 10^5$。

因此，我们可以得到一个结论：插入字符串的长度最多只有$O(\sqrt{N})$种。

第5-8个数据点

这一类数据点的特点在于只有操作1的修改，而没有操作2的修改。

分析数据特点:

由于输入保证不超过0.3M,可以得到插入总长度不超过$3 * 10^5$。

因此，我们可以得到一个结论：插入字符串的长度最多只有$O(\sqrt{N})$种。

证明：

第5-8个数据点

这一类数据点的特点在于只有操作1的修改，而没有操作2的修改。

分析数据特点:

由于输入保证不超过0.3M,可以得到插入总长度不超过$3 * 10^5$。

因此，我们可以得到一个结论：插入字符串的长度最多只有$O(\sqrt{N})$种。

证明：

假设长度恰好为1到$\sqrt{N}$各一个，那么总长度已达到$O(N)$级别。

证毕。

第5-8个数据点

第5-8个数据点

因此，我们对于询问操作，可以对于各个不同长度的询问分开处理。

第5-8个数据点

因此，我们对于询问操作，可以对于各个不同长度的询问分开处理。

那么，问题变成要维护各个不同长度的左端点在某一区间范围的方案数。

第5-8个数据点

因此，我们对于询问操作，可以对于各个不同长度的询问分开处理。

那么，问题变成要维护各个不同长度的左端点在某一区间范围的方案数。

而对于插入操作，我们可以得出：插入操作所能影响的左端点的后缀排名处于某一个区间范围内。

第5-8个数据点

因此，我们对于询问操作，可以对于各个不同长度的询问分开处理。

那么，问题变成要维护各个不同长度的左端点在某一区间范围的方案数。

而对于插入操作，我们可以得出：插入操作所能影响的左端点的后缀排名处于某一个区间范围内。

因此，我们可以对于每一个左端点记录一个二元组(a,b)表示左端点位置在a，对应后缀排名为b。

第5-8个数据点

因此，我们对于询问操作，可以对于各个不同长度的询问分开处理。

那么，问题变成要维护各个不同长度的左端点在某一区间范围的方案数。

而对于插入操作，我们可以得出：插入操作所能影响的左端点的后缀排名处于某一个区间范围内。

因此，我们可以对于每一个左端点记录一个二元组(a,b)表示左端点位置在a，对应后缀排名为b。

然后插入则是对于所有$b \in [l, r]$的二元组进行权值+1.询问则是对于所有$a \in [l, r]$的二元组询问权值和。

我们可以采取分块来解决这个问题。

第5-8个数据点

第5-8个数据点

假设我们按照位置序列来分块，块长为L.

第5-8个数据点

假设我们按照位置序列来分块，块长为L.

那么我们可以对于每一块记录块内权值和以及按照b排序的块内端点排名从而记录对于每一个给定的r得到的最大的k满足$b_k \leq r$.

第5-8个数据点

假设我们按照位置序列来分块，块长为L.

那么我们可以对于每一块记录块内权值和以及按照b排序的块内端点排名从而记录对于每一个给定的r得到的最大的k满足$b_k \leq r$.

然后，再对前i块记录权值和sum_i.

第5-8个数据点

假设我们按照位置序列来分块，块长为L.

那么我们可以对于每一块记录块内权值和以及按照b排序的块内端点排名从而记录对于每一个给定的r得到的最大的k满足$b_k \leq r$.

然后，再对前i块记录权值和sum_i.

对于修改操作，我们直接求出对于每一块的贡献以及对于前i块的贡献值，修改对应的块内权值和以及sum_i.并且对于每一块记录一个修改标记。

第5-8个数据点

假设我们按照位置序列来分块，块长为L.

那么我们可以对于每一块记录块内权值和以及按照b排序的块内端点排名从而记录对于每一个给定的r得到的最大的k满足$b_k \leq r$.

然后，再对前i块记录权值和sum_i.

对于修改操作，我们直接求出对于每一块的贡献以及对于前i块的贡献值，修改对应的块内权值和以及sum_i.并且对于每一块记录一个修改标记。

对于询问操作，则先利用求好的sum_i求出整块的权值和，然后对于非整块部分，则直接打算标记，暴力询问即可。

第5-8个数据点

假设我们按照位置序列来分块，块长为L.

那么我们可以对于每一块记录块内权值和以及按照b排序的块内端点排名从而记录对于每一个给定的r得到的最大的k满足$b_k \le r$.

然后，再对前i块记录权值和sum_i.

对于修改操作，我们直接求出对于每一块的贡献以及对于前i块的贡献值，修改对应的块内权值和以及sum_i.并且对于每一块记录一个修改标记。

对于询问操作，则先利用求好的sum_i求出整块的权值和，然后对于非整块部分，则直接打算标记，暴力询问即可。

分析复杂度，修改操作复杂度为$O(\frac{N}{L})$,询问操作复杂度为$O(L\sqrt{N})$.当$L = N^{0.25}$时，有最小复杂度为$O(N^{0.75})$.

第5-8个数据点

第5-8个数据点

因此，整个算法复杂度为$O(N^{1.75})$.预计得分20分。

第5-8个数据点

因此，整个算法复杂度为$O(N^{1.75})$.预计得分20分。

算上上面其余部分的分数，预计得分40分。

第9-10个数据点

第9-10个数据点

对于这一类数据，同样的，我们可以先构造一个后缀数组。

然后，我们对于每一个位置维护一个结构，维护以该点为左端点的所有修改长度。

第9-10个数据点

对于这一类数据，同样的，我们可以先构造一个后缀数组。

然后，我们对于每一个位置维护一个结构，维护以该点为左端点的所有修改长度。

对于修改操作，我们可以通过二分求出包含这一子串的后缀排名区间。我们可以对于排名区间里面对应的每个位置，在其对应的结构里面插入当前的修改长度。

第9-10个数据点

对于这一类数据，同样的，我们可以先构造一个后缀数组。

然后，我们对于每一个位置维护一个结构，维护以该点为左端点的所有修改长度。

对于修改操作，我们可以通过二分求出包含这一子串的后缀排名区间。我们可以对于排名区间里面对应的每个位置，在其对应的结构里面插入当前的修改长度。

然后，对于询问操作，则对操作区间内的所有点分别询问其结构里面长度不超过某一个值的修改个数。

第9-10个数据点

对于这一类数据，同样的，我们可以先构造一个后缀数组。

然后，我们对于每一个位置维护一个结构，维护以该点为左端点的所有修改长度。

对于修改操作，我们可以通过二分求出包含这一子串的后缀排名区间。我们可以对于排名区间里面对应的每个位置，在其对应的结构里面插入当前的修改长度。

然后，对于询问操作，则对操作区间内的所有点分别询问其结构里面长度不超过某一个值的修改个数。

因此，我们需要一个可以维护动态插入以及询问比某个值小的值的个数，可以使用线段树或者平衡树，也可以采用pb_ds库里面的黑科技。

第9-10个数据点

对于这一类数据，同样的，我们可以先构造一个后缀数组。

然后，我们对于每一个位置维护一个结构，维护以该点为左端点的所有修改长度。

对于修改操作，我们可以通过二分求出包含这一子串的后缀排名区间。我们可以对于排名区间里面对应的每个位置，在其对应的结构里面插入当前的修改长度。

然后，对于询问操作，则对操作区间内的所有点分别询问其结构里面长度不超过某一个值的修改个数。

因此，我们需要一个可以维护动态插入以及询问比某个值小的值的个数，可以使用线段树或者平衡树，也可以采用pb_ds库里面的黑科技。

时间复杂度为$O(NMlog_2N)$.预计得分10分。

第9-10个数据点

对于这一类数据，同样的，我们可以先构造一个后缀数组。

然后，我们对于每一个位置维护一个结构，维护以该点为左端点的所有修改长度。

对于修改操作，我们可以通过二分求出包含这一子串的后缀排名区间。我们可以对于排名区间里面对应的每个位置，在其对应的结构里面插入当前的修改长度。

然后，对于询问操作，则对操作区间内的所有点分别询问其结构里面长度不超过某一个值的修改个数。

因此，我们需要一个可以维护动态插入以及询问比某个值小的值的个数，可以使用线段树或者平衡树，也可以采用pb_ds库里面的黑科技。

时间复杂度为$O(NMlog_2N)$.预计得分10分。

算上上面其余部分的分数，预计得分50分。

第11-12个数据点

第11-12个数据点

这一类数据的特点在于右端点固定为N，即不存在右端点越界的情况。因此，我们不需要分开不同长度来处理。

第11-12个数据点

这一类数据的特点在于右端点固定为N，即不存在右端点越界的情况。因此，我们不需要分开不同长度来处理。

因此，我们可以采取类似第5-8个数据点的解决方法：

第11-12个数据点

这一类数据的特点在于右端点固定为N，即不存在右端点越界的情况。因此，我们不需要分开不同长度来处理。

因此，我们可以采取类似第5-8个数据点的解决方法：

对于每一个左端点记录一个二元组(a,b)表示左端点位置在a，对应后缀排名为b。

然后插入则是对于所有$b \in [l, r]$的二元组进行权值+1.询问则是对于所有$a \in [l, r]$的二元组询问权值和。

第11-12个数据点

这一类数据的特点在于右端点固定为N，即不存在右端点越界的情况。因此，我们不需要分开不同长度来处理。

因此，我们可以采取类似第5-8个数据点的解决方法：

对于每一个左端点记录一个二元组(a,b)表示左端点位置在a，对应后缀排名为b。

然后插入则是对于所有$b \in [l, r]$的二元组进行权值+1.询问则是对于所有$a \in [l, r]$的二元组询问权值和。

我们上面原本使用的是分块，但其实这里也是可以采用KD-tree来解决的。使用KD-tree可以减少一定的代码量以及运行速度上也会有一定的提升。

第11-12个数据点

这一类数据的特点在于右端点固定为N，即不存在右端点越界的情况。因此，我们不需要分开不同长度来处理。

因此，我们可以采取类似第5-8个数据点的解决方法：

对于每一个左端点记录一个二元组(a,b)表示左端点位置在a，对应后缀排名为b。

然后插入则是对于所有$b \in [l, r]$的二元组进行权值+1.询问则是对于所有$a \in [l, r]$的二元组询问权值和。

我们上面原本使用的是分块，但其实这里也是可以采用KD-tree来解决的。使用KD-tree可以减少一定的代码量以及运行速度上也会有一定的提升。

可以证明，对于坐标两两不相同的情况，KD-tree可以保证最坏$O(\sqrt{N})$的复杂度完成二维区间内的问题。

第11-12个数据点

这一类数据的特点在于右端点固定为N，即不存在右端点越界的情况。因此，我们不需要分开不同长度来处理。

因此，我们可以采取类似第5-8个数据点的解决方法：

对于每一个左端点记录一个二元组(a,b)表示左端点位置在a，对应后缀排名为b。

然后插入则是对于所有$b \in [l,r]$的二元组进行权值+1.询问则是对于所有$a \in [l,r]$的二元组询问权值和。

我们上面原本使用的是分块，但其实这里也是可以采用KD-tree来解决的。使用KD-tree可以减少一定的代码量以及运行速度上也会有一定的提升。

可以证明，对于坐标两两不相同的情况，KD-tree可以保证最坏$O(\sqrt{N})$的复杂度完成二维区间内的问题。

第11-12个数据点

第11-12个数据点

这一点，我们可以分两步来证明(开始凑页数了...)：

对于其中一维区间为全集的情况

对于其中一维区间为全集的情况

首先，比较容易可以知道另一维区间为任意区间[L,r]实际上可以等价于[1,r]区间的解决。

对于其中一维区间为全集的情况

首先，比较容易可以知道另一维区间为任意区间[L,r]实际上可以等价于[1,r]区间的解决。

这个可以简单证明得到:假设线段树中mid为使得L,r处于两个不同区间的节点，那么下一步问题变成了等价于[L,n]以及[1,r]的问题。

然后，对于[1,r]的情况，如果当前是以全集区间的那维分两边，那么显然两边节点都要遍历，而如果是以[1,r]区间的那维的话，那么必然有一边要么完全不覆盖，要么完全覆盖，都可以一步解决，因此，实际需要往下的只有一个节点。

对于其中一维区间为全集的情况

首先，比较容易可以知道另一维区间为任意区间[L,r]实际上可以等价于[1,r]区间的解决。

这个可以简单证明得到:假设线段树中mid为使得L,r处于两个不同区间的节点，那么下一步问题变成了等价于[L,n]以及[1,r]的问题。

然后，对于[1,r]的情况，如果当前是以全集区间的那维分两边，那么显然两边节点都要遍历，而如果是以[1,r]区间的那维的话，那么必然有一边要么完全不覆盖，要么完全覆盖，都可以一步解决，因此，实际需要往下的只有一个节点。

然后，总层数为$log_2 N$层，然后只有$\frac{log_2 N}{2}$层需要遍历两边，因此复杂度为$O(2^{\frac{log_2 N}{2}}) = O(\sqrt{N})$.

一般情况

一般情况

首先，同样容易证明其中一维的任意区间[L,r]可以等价于[1,r]。

一般情况

首先，同样容易证明其中一维的任意区间[L,r]可以等价于[1,r]。

然后当当前是以这一维分两边的话，那么必然有一边这一维要么完全不覆盖，要么完全覆盖。最坏情况是完全覆盖那一种，那么这一部分我们已经证明是$O(\sqrt{N})$级别的了。而另一边，则是等价的问题。

一般情况

首先，同样容易证明其中一维的任意区间[L,r]可以等价于[1,r]。

然后当当前是以这一维分两边的话，那么必然有一边这一维要么完全不覆盖，要么完全覆盖。最坏情况是完全覆盖那一种，那么这一部分我们已经证明是$O(\sqrt{N})$级别的了。而另一边，则是等价的问题。

因此，我们有$F(N) = F(N/2) + O(\sqrt{N})$,根据等比数列我们可以得到$F(N) = O(\sqrt{N})$.

一般情况

首先，同样容易证明其中一维的任意区间[L,r]可以等价于[1,r]。

然后当当前是以这一维分两边的话，那么必然有一边这一维要么完全不覆盖，要么完全覆盖。最坏情况是完全覆盖那一种，那么这一部分我们已经证明是$O(\sqrt{N})$级别的了。而另一边，则是等价的问题。

因此，我们有$F(N) = F(N/2) + O(\sqrt{N})$,根据等比数列我们可以得到$F(N) = O(\sqrt{N})$.

证毕。

第11-12个数据点

第11-12个数据点

于是，我们可以通过$O(M\sqrt{N})$的复杂度解决。预计得分10分。
算上上面其余部分的分数，预计得分60分。

第13-15个数据点

第13-15个数据点

这一类数据点，保证了插入长度不下降。这个提示我们可以往长度方向想。

第13-15个数据点

这一类数据点，保证了插入长度不下降。这个提示我们可以往长度方向想。

考虑当前询问区间为[L,r]，我们设[1,L-1]为A区间，[L,r]为B区间，[r+1,N]为C区间。

第13-15个数据点

这一类数据点，保证了插入长度不下降。这个提示我们可以往长度方向想。

考虑当前询问区间为[L,r]，我们设[1,L-1]为A区间，[L,r]为B区间，[r+1,N]为C区间。然后设(x,y)意义为左端点在x区间，右端点在y区间的权值和。例如(A,B)表示左端点在A区间，右端点在B区间。我们求解的是(B,B)的值。

第13-15个数据点

这一类数据点，保证了插入长度不下降。这个提示我们可以往长度方向想。

考虑当前询问区间为[L,r]，我们设[1,L-1]为A区间，[L,r]为B区间，[r+1,N]为C区间。然后设(x,y)意义为左端点在x区间，右端点在y区间的权值和。例如(A,B)表示左端点在A区间，右端点在B区间。我们求解的是(B,B)的值。

那么，对于[1,r]区间的询问，我们得出了(A,A)+(A,B)+(B,B)。那么，如果我们有办法求解(A,A)+(A,B)就可以解决了。因为我们已经知道其中一个端点在边界的解法了。

第13-15个数据点

这一类数据点，保证了插入长度不下降。这个提示我们可以往长度方向想。

考虑当前询问区间为[L,r]，我们设[1,L-1]为A区间，[L,r]为B区间，[r+1,N]为C区间。然后设(x,y)意义为左端点在x区间，右端点在y区间的权值和。例如(A,B)表示左端点在A区间，右端点在B区间。我们求解的是(B,B)的值。

那么，对于[1,r]区间的询问，我们得出了(A,A)+(A,B)+(B,B)。那么，如果我们有办法求解(A,A)+(A,B)就可以解决了。因为我们已经知道其中一个端点在边界的解法了。

然后我们可以求[1,N]区间的询问，从而得出(A,A)+(A,B)+(A,C)+(B,B)+(B,C)+(C,C).

第13-15个数据点

这一类数据点，保证了插入长度不下降。这个提示我们可以往长度方向想。

考虑当前询问区间为[L,r]，我们设[1,L-1]为A区间，[L,r]为B区间，[r+1,N]为C区间。然后设(x,y)意义为左端点在x区间，右端点在y区间的权值和。例如(A,B)表示左端点在A区间，右端点在B区间。我们求解的是(B,B)的值。

那么，对于[1,r]区间的询问，我们得出了(A,A)+(A,B)+(B,B)。那么，如果我们有办法求解(A,A)+(A,B)就可以解决了。因为我们已经知道其中一个端点在边界的解法了。

然后我们可以求[1,N]区间的询问，从而得出(A,A)+(A,B)+(A,C)+(B,B)+(B,C)+(C,C).

然后又由(B,B)+(B,C)+(C,C)为[L,N]区间的询问，可以得到。

第13-15个数据点

第13-15个数据点

因此，现在唯一的问题就是(A,C)问题的求解了。但这一类问题似乎没有什么好的解决方法。

第13-15个数据点

因此，现在唯一的问题就是(A,C)问题的求解了。但这一类问题似乎没有什么好的解决方法。

这时，我们回过来看长度的提示。然后我们可以得到一个突破点：如果保证长度不超过$r-l+1$的话，那么不存在(A,C)这个问题。

第13-15个数据点

因此，现在唯一的问题就是(A,C)问题的求解了。但这一类问题似乎没有什么好的解决方法。

这时，我们回过来看长度的提示。然后我们可以得到一个突破点：如果保证长度不超过$r-l+1$的话，那么不存在(A,C)这个问题。

因此，我们可以根据长度来维护一个可持久化KD-tree，维护只包含长度不超过某个值的答案。

第13-15个数据点

因此，现在唯一的问题就是(A,C)问题的求解了。但这一类问题似乎没有什么好的解决方法。

这时，我们回过来看长度的提示。然后我们可以得到一个突破点：如果保证长度不超过$r-l+1$的话，那么不存在(A,C)这个问题。

因此，我们可以根据长度来维护一个可持久化KD-tree，维护只包含长度不超过某个值的答案。

那么，对于询问操作，我们只需要先二分出对应最大长度对应的KD-tree，然后在这棵KD-tree进行询问即可。

第13-15个数据点

因此，现在唯一的问题就是(A,C)问题的求解了。但这一类问题似乎没有什么好的解决方法。

这时，我们回过来看长度的提示。然后我们可以得到一个突破点：如果保证长度不超过$r-l+1$的话，那么不存在(A,C)这个问题。

因此，我们可以根据长度来维护一个可持久化KD-tree，维护只包含长度不超过某个值的答案。

那么，对于询问操作，我们只需要先二分出对应最大长度对应的KD-tree，然后在这棵KD-tree进行询问即可。

时间复杂度为$O(M\sqrt{N})$,预计得分15分。

第13-15个数据点

因此，现在唯一的问题就是(A,C)问题的求解了。但这一类问题似乎没有什么好的解决方法。

这时，我们回过来看长度的提示。然后我们可以得到一个突破点：如果保证长度不超过$r-l+1$的话，那么不存在(A,C)这个问题。

因此，我们可以根据长度来维护一个可持久化KD-tree，维护只包含长度不超过某个值的答案。

那么，对于询问操作，我们只需要先二分出对应最大长度对应的KD-tree，然后在这棵KD-tree进行询问即可。

时间复杂度为$O(M\sqrt{N})$,预计得分15分。

算上上面其余部分的分数，预计得分75分。

第16-20个数据点

第16-20个数据点

最后这一类数据点，实际上只需要对第13-15个数据点的解决方法进行少量修改即可。

第16-20个数据点

最后这一类数据点，实际上只需要对第13-15个数据点的解决方法进行少量修改即可。

我们观察到这一题允许离线，并且操作独立，因此，我们只需要在外面套上一个cdq分治，然后就能保证插入长度单调，从而就能直接利用第13-15个数据点的解决方法解决了。

第16-20个数据点

最后这一类数据点，实际上只需要对第13-15个数据点的解决方法进行少量修改即可。

我们观察到这一题允许离线，并且操作独立，因此，我们只需要在外面套上一个cdq分治，然后就能保证插入长度单调，从而就能直接利用第13-15个数据点的解决方法解决了。

由于多套了个cdq分治，时间复杂度为$O(M\sqrt{N}log_2M)$.

第16-20个数据点

最后这一类数据点，实际上只需要对第13-15个数据点的解决方法进行少量修改即可。

我们观察到这一题允许离线，并且操作独立，因此，我们只需要在外面套上一个cdq分治，然后就能保证插入长度单调，从而就能直接利用第13-15个数据点的解决方法解决了。

由于多套了个cdq分治，时间复杂度为$O(M\sqrt{N}log_2 M)$.

进行少量修改，可以通过全部数据。预计得分100分。

对于KD-tree的调参(继续凑...)

对于KD-tree的调参

对于KD-tree的调参

对于前面的第5-8个数据点的分块方法，实际上也可以用KD-tree来解决。

对于KD-tree的调参

对于前面的第5-8个数据点的分块方法，实际上也可以用KD-tree来解决。

现在问题就是对于某一维询问区间、另一维询问全集的情况，可以根据不同的情况得到不同的复杂度。

对于KD-tree的调参

对于前面的第5-8个数据点的分块方法，实际上也可以用KD-tree来解决。

现在问题就是对于某一维询问区间、另一维询问全集的情况，可以根据不同的情况得到不同的复杂度。

其实，这个似乎很好搞，假设我们要求第一维为区间的复杂度为N^a,第二维为区间的复杂度为N^{1-a}的话。

对于KD-tree的调参

对于前面的第5-8个数据点的分块方法，实际上也可以用KD-tree来解决。

现在问题就是对于某一维询问区间、另一维询问全集的情况，可以根据不同的情况得到不同的复杂度。

其实，这个似乎很好搞，假设我们要求第一维为区间的复杂度为N^a,第二维为区间的复杂度为N^{1-a}的话。

那么，我们只需要选择以第二维为分离标准的层数设为$a\log_2 N$即可。

对于KD-tree的调参

对于前面的第5-8个数据点的分块方法，实际上也可以用KD-tree来解决。

现在问题就是对于某一维询问区间、另一维询问全集的情况，可以根据不同的情况得到不同的复杂度。

其实，这个似乎很好搞，假设我们要求第一维为区间的复杂度为N^a,第二维为区间的复杂度为N^{1-a}的话。

那么，我们只需要选择以第二维为分离标准的层数设为$alog_2N$即可。

但是，要注意安排层的顺序，不然有可能复杂度会多乘了一个$O(log_2N)$。

对于KD-tree的调参

对于KD-tree的调参

而一种比较好的安排方法，就是假设$a < 1 - a$,那么先把多出来的$(1-2a)log_2 N$层先进行分离，之后剩余层数正常的轮流分即可。

对于KD-tree的调参

而一种比较好的安排方法，就是假设$a < 1 - a$,那么先把多出来的$(1-2a)log_2 N$层先进行分离，之后剩余层数正常的轮流分即可。

用KD-tree来调参的话，相较于分块，应该可以减少一定的空间和常数。

总结

总结

总结

总的来说，这道题难度与NOI第二题接近。

主要考察对题目性质的挖掘以及转化为简单问题的解决方法。旨在将字符串类型题目和数据结构类型的题目联系起来。

感谢

感谢

感谢

感谢中国计算机学会提供学习和交流的机会。
感谢佛山石中的江涛老师、梁冠健老师多年来给予的关怀与指导。
感谢国家集训队教练余林韵和陈许旻的指导。
感谢佛山石中的龙耀为、麦景、杨嘉宏等同学对我的帮助和启发。
感谢绍兴一中王鉴浩学长以及雅礼中学匡正非学长提供的模板。
感谢其他对我有过帮助和启发的老师和同学。
感谢父母对我的关心和照顾。

谢谢！

参考文献

参考文献

[1] 于纪平：《C++的的pb_ds库在OI中的应用》

[2] 任之洲：《k-d tree在传统OI数据结构题中的应用》

[3] 罗穗骞《后缀数组——处理字符串的有力工具》

[4] 陈立杰《后缀自动机》

[5] 罗剑桥《浅谈分块思想在一类数据处理问题中的应用》

[6] 陈丹琦《从《Cash》谈一类分治算法的应用》

[7] 刘汝佳, 黄亮, 《算法艺术与信息学竞赛》, 清华大学出版社。

[8] 刘汝佳, 《算法竞赛入门经典》, 清华大学出版社。