字符串算法选讲-解题报告

徐沐杰

南京大学

2023 年 7 月 15 日

成裘

- 70pts 拼凑出 $|S_n|$ 后跑 KMP 或字符串哈希均可。

成裘

- 70pts 拼凑出 $|S_n|$ 后跑 KMP 或字符串哈希均可。
- 100pts 发现目标串太大，没办法建出来，我们抽象地考虑目标串的匹配过程。

成裘

- 70pts 拼凑出 $|S_n|$ 后跑 KMP 或字符串哈希均可。
- 100pts 发现目标串太大，没办法建出来，我们抽象地考虑目标串的匹配过程。
- 目标串的匹配过程即是在前缀自动机上的匹配过程，匹配的次数就是经过前缀自动机上某一状态的次数。

成裘

- 70pts 拼凑出 $|S_n|$ 后跑 KMP 或字符串哈希均可。
- 100pts 发现目标串太大，没办法建出来，我们抽象地考虑目标串的匹配过程。
- 目标串的匹配过程即是在前缀自动机上的匹配过程，匹配的次数就是经过前缀自动机上某一状态的次数。
- 发现可以容易递推，令 $go_{j,i}$ 表示 j 点接受 S_i 序列后所在的位置，$K_{j,i}$ 表示 j 点接受 S_i 序列后经过目标状态的次数。

成裘

- 70pts 拼凑出 $|S_n|$ 后跑 KMP 或字符串哈希均可。
- 100pts 发现目标串太大，没办法建出来，我们抽象地考虑目标串的匹配过程。
- 目标串的匹配过程即是在前缀自动机上的匹配过程，匹配的次数就是经过前缀自动机上某一状态的次数。
- 发现可以容易递推，令 $go_{j,i}$ 表示 j 点接受 S_i 序列后所在的位置，$K_{j,i}$ 表示 j 点接受 S_i 序列后经过目标状态的次数。
- 于是 $go_{j,i} = go_{go_{j,i-2},i-1}$, $K_{j,i} = K_{j,i-2} + K_{K_{j,i-2},i-1}$，递推即可。

成裘

- 70pts 拼凑出 $|S_n|$ 后跑 KMP 或字符串哈希均可。
- 100pts 发现目标串太大，没办法建出来，我们抽象地考虑目标串的匹配过程。
- 目标串的匹配过程即是在前缀自动机上的匹配过程，匹配的次数就是经过前缀自动机上某一状态的次数。
- 发现可以容易递推，令 $go_{j,i}$ 表示 j 点接受 S_i 序列后所在的位置，$K_{j,i}$ 表示 j 点接受 S_i 序列后经过目标状态的次数。
- 于是 $go_{j,i} = go_{go_{j,i-2},i-1}$, $K_{j,i} = K_{j,i-2} + K_{K_{j,i-2},i-1}$，递推即可。
- 复杂度 $O((n + \max\{|S_1|, |S_2|\})|M|)$。

水文

- 60pts 把后缀反着插到 Trie 里面，统计叶子节点的个数。

水文

- 60pts 把后缀反着插到 Trie 里面，统计叶子节点的个数。
- 100pts 既然这些字符串都是某一字符串的前缀，那么如果某一字符串是某一字符串的后缀，说明它就是该字符串的某一公共前后缀 (即 CPS，或谓 border)。

水文

- 60pts 把后缀反着插到 Trie 里面，统计叶子节点的个数。
- 100pts 既然这些字符串都是某一字符串的前缀，那么如果某一字符串是某一字符串的后缀，说明它就是该字符串的某一公共前后缀 (即 CPS，或谓 border)。
- 那么我们根据失配树的性质，可以发现只需统计所有失配树的叶子节点即可。

水文

- 60pts 把后缀反着插到 Trie 里面，统计叶子节点的个数。
- 100pts 既然这些字符串都是某一字符串的前缀，那么如果某一字符串是某一字符串的后缀，说明它就是该字符串的某一公共前后缀 (即 CPS，或谓 border)。
- 那么我们根据失配树的性质，可以发现只需统计所有失配树的叶子节点即可。
- 失配树裸题，秒了。

寻迹

- 90pts 直接对 S 的正串和反串同时建出两个 Trie 树，如果通配符出现在尾部，就从正串建出的 Trie 树走，统计子树和即可。如果通配符出现在首部就在反串的 Trie 树上统计子树和。

寻迹

- 90pts 直接对 S 的正串和反串同时建出两个 Trie 树，如果通配符出现在尾部，就从正串建出的 Trie 树走，统计子树和即可。如果通配符出现在首部就在反串的 Trie 树上统计子树和。
- 100pts 仍然建出正反两棵 Trie 树，考虑任何一 T，我们发现它能匹配到的串就是其前缀在正串 Trie 的子树中涉及到的串和反串 Trie 的子树中涉及到的串的交集

寻迹

- 90pts 直接对 S 的正串和反串同时建出两个 Trie 树，如果通配符出现在尾部，就从正串建出的 Trie 树走，统计子树和即可。如果通配符出现在首部就在反串的 Trie 树上统计子树和。
- 100pts 仍然建出正反两棵 Trie 树，考虑任何一 T，我们发现它能匹配到的串就是其前缀在正串 Trie 的子树中涉及到的串和反串 Trie 的子树中涉及到的串的交集
- 这个就是两树 DFS 序上的二维前缀和。假设一个 T 在正 Trie 上走到 pos，在反 Trie 上走到 pos'，即是针对 (pos, pos') 矩形的求和。

寻迹

- 90pts 直接对 S 的正串和反串同时建出两个 Trie 树，如果通配符出现在尾部，就从正串建出的 Trie 树走，统计子树和即可。如果通配符出现在首部就在反串的 Trie 树上统计子树和。
- 100pts 仍然建出正反两棵 Trie 树，考虑任何一 T，我们发现它能匹配到的串就是其前缀在正串 Trie 的子树中涉及到的串和反串 Trie 的子树中涉及到的串的交集
- 这个就是两树 DFS 序上的二维前缀和。假设一个 T 在正 Trie 上走到 pos，在反 Trie 上走到 pos'，即是针对 (pos, pos') 矩形的求和。
- 但是两维太大。考虑离线询问，以正 Trie DFS 序做第一维前缀和 (但是不保存)，树状数组维护反 Trie DFS 序上的第二维前缀和。

寻迹

- 90pts 直接对 S 的正串和反串同时建出两个 Trie 树，如果通配符出现在尾部，就从正串建出的 Trie 树走，统计子树和即可。如果通配符出现在首部就在反串的 Trie 树上统计子树和。
- 100pts 仍然建出正反两棵 Trie 树，考虑任何一 T，我们发现它能匹配到的串就是其前缀在正串 Trie 的子树中涉及到的串和反串 Trie 的子树中涉及到的串的交集
- 这个就是两树 DFS 序上的二维前缀和。假设一个 T 在正 Trie 上走到 pos，在反 Trie 上走到 pos'，即是针对 (pos, pos') 矩形的求和。
- 但是两维太大。考虑离线询问，以正 Trie DFS 序做第一维前缀和(但是不保存)，树状数组维护反 Trie DFS 序上的第二维前缀和。
- 处理到某询问正 Trie DFS 区间左端点 -1、右端点时各处理一次答案，最后把每个询问的两次答案相减就可以计算出此二维前缀和。

寻迹

- 90pts 直接对 S 的正串和反串同时建出两个 Trie 树，如果通配符出现在尾部，就从正串建出的 Trie 树走，统计子树和即可。如果通配符出现在首部就在反串的 Trie 树上统计子树和。
- 100pts 仍然建出正反两棵 Trie 树，考虑任何一 T，我们发现它能匹配到的串就是其前缀在正串 Trie 的子树中涉及到的串和反串 Trie 的子树中涉及到的串的交集
- 这个就是两树 DFS 序上的二维前缀和。假设一个 T 在正 Trie 上走到 pos，在反 Trie 上走到 pos'，即是针对 (pos, pos') 矩形的求和。
- 但是两维太大。考虑离线询问，以正 Trie DFS 序做第一维前缀和(但是不保存)，树状数组维护反 Trie DFS 序上的第二维前缀和。
- 处理到某询问正 Trie DFS 区间左端点 -1、右端点时各处理一次答案，最后把每个询问的两次答案相减就可以计算出此二维前缀和。
- 复杂度 $O(\sum_{i\in[n]}|T_i| + \sum_{i\in[n]}|S_i| + m\log\sum_{i\in[n]}|S_i|)$

提问 || 完结

还有问题吗?

提问 || 完结

还有问题吗？

谢谢大家陪伴！后会有期。

~~（此处脑补一曲《送别》）~~