字符串相关

wh_ZH

？？？

- 内容主要分为2部分，概念部分和题目部分

？？？

- 内容主要分为2部分，概念部分和题目部分
- 讲题人水平有限，如果出锅了请大家多包涵QWQ

Part1:知识点总结

- 因为本节课要讲的知识点，之前已经有几位学长给大家讲过了，所以不再重新讲一遍了

Part1:知识点总结

- 因为本节课要讲的知识点，之前已经有几位学长给大家讲过了，所以不再重新讲一遍了
- ~~但是直接讲题又不太好，~~所以我还是总结了一些应用方面的内容。

Part1:知识点总结

- 因为本节课要讲的知识点，之前已经有几位学长给大家讲过了，所以不再重新讲一遍了
- ~~但是直接讲题又不太好，~~所以我还是总结了一些应用方面的内容。
- 这部分不会有什么难度，但是这些经典应用还是很有总结一下的必要的。

哈希

- 一种把长字符串转换成一个/一些大数字的方法

哈希

- 一种把长字符串转换成一个/一些大数字的方法
- 有错误概率，但是一般小到可以忽略不计

哈希

- 一种把长字符串转换成一个/一些大数字的方法
- 有错误概率，但是一般小到可以忽略不计
- 用途非常地广泛

哈希

- 一种把长字符串转换成一个/一些大数字的方法
- 有错误概率，但是一般小到可以忽略不计
- 用途非常地广泛
- zxp学长：二分+哈希秒天秒地

哈希

- 一种把长字符串转换成一个/一些大数字的方法
- 有错误概率，但是一般小到可以忽略不计
- 用途非常地广泛
- zxp学长：二分+哈希秒天秒地
- 但是本节课中，希望大家尽量避免使用哈希，多想想确定性算法

哈希

- 一种把长字符串转换成一个/一些大数字的方法
- 有错误概率，但是一般小到可以忽略不计
- 用途非常地广泛
- zxp学长：二分+哈希秒天秒地
- 但是本节课中，希望大家尽量避免使用哈希，多想想确定性算法
- 应用太多了，没法总结，也举不出有代表性的例子

KMP算法

- 单串匹配利器，可以$O(n+m)$对两个长度分别为n和m的串进行匹配

KMP算法

- 单串匹配利器，可以$O(n+m)$对两个长度分别为n和m的串进行匹配
- 在算法竞赛中，更多考察的是$next$数组的性质和灵活运用

KMP算法 应用举例

- 1.求一个串的最短循环节

KMP算法 应用举例

- 1.求一个串的最短循环节
- 如果$n\%(n-next[n])=0$，那么答案是$n-next[n]$，否则是n。考虑$next$数组的性质，以$(n-next[n])$为单位向前推即可证明。

KMP算法 应用举例

- 1.求一个串的最短循环节
- 如果$n\%(n-next[n])=0$，那么答案是$n-next[n]$，否则是n。考虑$next$数组的性质，以$(n-next[n])$为单位向前推即可证明。
- 2.对于一个串每个前缀，求它的不互相重叠的公共前后缀个数

KMP算法 应用举例

- 1.求一个串的最短循环节
- 如果$n\%(n-next[n])=0$，那么答案是$n-next[n]$，否则是n。考虑$next$数组的性质，以$(n-next[n])$为单位向前推即可证明。
- 2.对于一个串每个前缀，求它的不互相重叠的公共前后缀个数
- 求出$next$，考虑不互相重叠这一性质，可以记录$to[i]$表示i一直跳$next$到的第一个长度小于等于二分之一的位置，然后to每次至多增长1，这样就可以暴力做了。

KMP算法 应用举例

- 1.求一个串的最短循环节
- 如果$n\%(n-next[n])=0$，那么答案是$n-next[n]$，否则是n。考虑$next$数组的性质，以$(n-next[n])$为单位向前推即可证明。
- 2.对于一个串每个前缀，求它的不互相重叠的公共前后缀个数
- 求出$next$，考虑不互相重叠这一性质，可以记录$to[i]$表示i一直跳$next$到的第一个长度小于等于二分之一的位置，然后to每次至多增长1，这样就可以暴力做了。
- 3.给一个长度20的串，问长度为$1e9$的串中有多少个不包含它

KMP算法 应用举例

- 1.求一个串的最短循环节
- 如果$n\%(n-next[n])=0$，那么答案是$n-next[n]$，否则是n。考虑$next$数组的性质，以$(n-next[n])$为单位向前推即可证明。
- 2.对于一个串每个前缀，求它的不互相重叠的公共前后缀个数
- 求出$next$，考虑不互相重叠这一性质，可以记录$to[i]$表示i一直跳$next$到的第一个长度小于等于二分之一的位置，然后to每次至多增长1，这样就可以暴力做了。
- 3.给一个长度20的串，问长度为$1e9$的串中有多少个不包含它
- $F[i][j]$表示长串到i，短串到j的方案数，$G[i][j]$表示有多少种加一个字符的方法使匹配长度由i变成j，用$next$数组求，转移显然，上矩阵优化就完了。

AC自动机（Trie图）

- 一种可以进行字符串多串匹配的东西

AC自动机（Trie图）

- 一种可以进行字符串多串匹配的东西
- 我们可以通过一次bfs完成AC自动机的构建。

AC自动机（Trie图）

- 一种可以进行字符串多串匹配的东西
- 我们可以通过一次bfs完成AC自动机的构建。
- AC自动机可以支持多串匹配单串，求一个字符串在另一个串中的出现次数等等，大致流程如下：

AC自动机（Trie图）

- 一种可以进行字符串多串匹配的东西
- 我们可以通过一次bfs完成AC自动机的构建。
- AC自动机可以支持多串匹配单串，求一个字符串在另一个串中的出现次数等等，大致流程如下：
- 主串从左到右，就顺着每一个字符向下跳，每一次沿着Fail链向上。这样会可重而不漏的得到模式串中的所有子串，视题目要求进行操作即可。

AC自动机（Trie图）应用举例

- 1.求有多少个模式串在文本串里出现过

AC自动机（Trie图）应用举例

- 1.求有多少个模式串在文本串里出现过
- 按模式串建AC自动机，然后把文本串扔上去跑就行了

AC自动机（Trie图）应用举例

- 1.求有多少个模式串在文本串里出现过
- 按模式串建AC自动机，然后把文本串扔上去跑就行了
- 2.给一堆单词，问每个单词分别在所有单词中共出现了多少次。

AC自动机（Trie图）应用举例

- 1.求有多少个模式串在文本串里出现过
- 按模式串建AC自动机，然后把文本串扔上去跑就行了
- 2.给一堆单词，问每个单词分别在所有单词中共出现了多少次。
- 建AC自动机，每个节点保存它属于多少字符串。然后一个点表示的字符串在整个字典中出现次数等于其在Fail树中的子树和。

AC自动机（Trie图）应用举例

- 1.求有多少个模式串在文本串里出现过
- 按模式串建AC自动机，然后把文本串扔上去跑就行了
- 2.给一堆单词，问每个单词分别在所有单词中共出现了多少次。
- 建AC自动机，每个节点保存它属于多少字符串。然后一个点表示的字符串在整个字典中出现次数等于其在Fail树中的子树和。
- 3.给出一棵Trie树，多次询问树上一个串在另一个串的出现次数。

AC自动机（Trie图）应用举例

- 1.求有多少个模式串在文本串里出现过
- 按模式串建AC自动机，然后把文本串扔上去跑就行了
- 2.给一堆单词，问每个单词分别在所有单词中共出现了多少次。
- 建AC自动机，每个节点保存它属于多少字符串。然后一个点表示的字符串在整个字典中出现次数等于其在Fail树中的子树和。
- 3.给出一棵Trie树，多次询问树上一个串在另一个串的出现次数。
- 对着这棵Trie建AC自动机，x在y中的出现次数即为Fail树中x的子树与Trie树中y到根节点的路径的公共节点数。如果强制在线，可以用主席树，否则离线+树状数组即可.

后缀数组

- 可以通过倍增($O(nlogn)$)或DC3(O(n))求得，当然你喜欢拿SAM求也行。

后缀数组

- 可以通过倍增($O(nlogn)$)或DC3(O(n))求得，当然你喜欢拿SAM求也行。
- 我们求得的数组有sa和$rank$，有很多时候还需要用到$height$数组。

后缀数组

- 可以通过倍增($O(nlogn)$)或DC3(O(n))求得，当然你喜欢拿SAM求也行。
- 我们求得的数组有sa和$rank$，有很多时候还需要用到$height$数组。
- 其中$sa[i]$表示排名第i的后缀是从哪里开始的，$rank[i]$表示第i个后缀排名第几，$height[i]$的意义是$rank[i]$与$rank[i-1]$的字符串所对应的LCP（最长公共前缀）。

后缀数组

- 可以通过倍增($O(nlogn)$)或DC3(O(n))求得，当然你喜欢拿SAM求也行。
- 我们求得的数组有sa和$rank$，有很多时候还需要用到$height$数组。
- 其中$sa[i]$表示排名第i的后缀是从哪里开始的，$rank[i]$表示第i个后缀排名第几，$height[i]$的意义是$rank[i]$与$rank[i-1]$的字符串所对应的LCP（最长公共前缀）。
- 任意两个后缀的LCP，是这个后缀所对应的$rank$之间的$height$的最小值。

后缀数组 应用举例

- 1.求至少出现过两次的最长的子串（可重叠）

后缀数组 应用举例

- 1.求至少出现过两次的最长的子串（可重叠）
- 答案即为$\max(height[i])$，容易看出这就是LCP。根据定义，LCP最大的只会出现在$rank$相邻的之间。

后缀数组 应用举例

- 1.求至少出现过两次的最长的子串（可重叠）
- 答案即为$\max(height[i])$，容易看出这就是LCP。根据定义，LCP最大的只会出现在$rank$相邻的之间。
- 2.求至少出现过两次的最长的子串（不可重叠）

后缀数组 应用举例

- 1.求至少出现过两次的最长的子串（可重叠）
- 答案即为$\max(height[i])$，容易看出这就是LCP。根据定义，LCP最大的只会出现在$rank$相邻的之间。
- 2.求至少出现过两次的最长的子串（不可重叠）
- 二分答案，对$height$分组(小于二分的答案就分组)，判断存在的条件是某一组中sa的最大最小值相减大于二分长度。

后缀数组 应用举例

- 1.求至少出现过两次的最长的子串（可重叠）
- 答案即为$\max(height[i])$，容易看出这就是LCP。根据定义，LCP最大的只会出现在$rank$相邻的之间。
- 2.求至少出现过两次的最长的子串（不可重叠）
- 二分答案，对$height$分组(小于二分的答案就分组)，判断存在的条件是某一组中sa的最大最小值相减大于二分长度。
- 3.求至少出现过k次的最长的子串（可重叠）

后缀数组 应用举例

- 1.求至少出现过两次的最长的子串（可重叠）
- 答案即为$\max(height[i])$，容易看出这就是LCP。根据定义，LCP最大的只会出现在$rank$相邻的之间。
- 2.求至少出现过两次的最长的子串（不可重叠）
- 二分答案，对$height$分组(小于二分的答案就分组)，判断存在的条件是某一组中sa的最大最小值相减大于二分长度。
- 3.求至少出现过k次的最长的子串（可重叠）
- 枚举i，维护$height[i->i+k-1]$的$\min$，用单调队列即可O(N)解决

后缀数组 应用举例

- 4.求不同子串个数

后缀数组 应用举例

- 4.求不同子串个数
- 因为相同的前缀个数，就是所有$height$之和。所以答案可以表示为$n*(n-1)/2-\sum height[i]$

后缀数组 应用举例

- 4.求不同子串个数
- 因为相同的前缀个数，就是所有$height$之和。所以答案可以表示为$n*(n-1)/2-\sum height[i]$
- 5.求一个字符串中有多少个出现多次的子串

后缀数组 应用举例

- 4.求不同子串个数
- 因为相同的前缀个数，就是所有$height$之和。所以答案可以表示为$n*(n-1)/2-\sum height[i]$
- 5.求一个字符串中有多少个出现多次的子串
- 设每个后缀$rank$为i，那么它最多贡献$height[i]-height[i-1]$个不同重复子串，所以答案就是$\sum \max(height[i]-height[i-1],0)$

后缀数组 应用举例

- 4.求不同子串个数
- 因为相同的前缀个数，就是所有$height$之和。所以答案可以表示为$n*(n-1)/2-\sum height[i]$
- 5.求一个字符串中有多少个出现多次的子串
- 设每个后缀$rank$为i，那么它最多贡献$height[i]-height[i-1]$个不同重复子串，所以答案就是$\sum \max(height[i]-height[i-1],0)$
- 6.求重复次数最多的连续重复子串

后缀数组 应用举例

- 4.求不同子串个数
- 因为相同的前缀个数，就是所有$height$之和。所以答案可以表示为$n * (n-1)/2 - \sum height[i]$
- 5.求一个字符串中有多少个出现多次的子串
- 设每个后缀$rank$为i，那么它最多贡献$height[i] - height[i-1]$个不同重复子串，所以答案就是$\sum \max(height[i] - height[i-1], 0)$
- 6.求重复次数最多的连续重复子串
- 枚举长度，用LCP判断能否拓展。复杂度是调和级数，近似于$O(nlogn)$

后缀自动机

- 后缀自动机是一个DAG。

后缀自动机

- 后缀自动机是一个DAG。
- 我们可以实现O(n)的在线构造，并且可以证明其点数和边数都是O(n)级别的

后缀自动机

- 后缀自动机是一个DAG。
- 我们可以实现O(n)的在线构造，并且可以证明其点数和边数都是O(n)级别的
- 它可以识别一个字符串的所有子串（最强的性质

后缀自动机

- 后缀自动机是一个DAG。
- 我们可以实现O(n)的在线构造，并且可以证明其点数和边数都是O(n)级别的
- 它可以识别一个字符串的所有子串（最强的性质
- 因此它有非常多的用途，做法也比后缀数组自然

后缀自动机 应用举例

- 1.判断某些串是否为某一个串的子串

后缀自动机 应用举例

- 1.判断某些串是否为某一个串的子串
- 把长串建SAM，短串放在上面挨个跑就行了

后缀自动机 应用举例

- 1.判断某些串是否为某一个串的子串
- 把长串建SAM，短串放在上面挨个跑就行了
- 2.不同子串个数

后缀自动机 应用举例

- 1.判断某些串是否为某一个串的子串
- 把长串建SAM，短串放在上面挨个跑就行了
- 2.不同子串个数
- 设f[i]表示从i出发的子串个数，显然，$f[i]=\sum_{(i,j)}(f[j]+1)$

后缀自动机 应用举例

- 1.判断某些串是否为某一个串的子串
- 把长串建SAM，短串放在上面挨个跑就行了
- 2.不同子串个数
- 设f[i]表示从i出发的子串个数，显然，$f[i]=\sum_{(i,j)}(f[j]+1)$
- DAG上DP即可

后缀自动机 应用举例

- 1.判断某些串是否为某一个串的子串
- 把长串建SAM，短串放在上面挨个跑就行了
- 2.不同子串个数
- 设f[i]表示从i出发的子串个数，显然，$f[i]=\sum_{(i,j)}(f[j]+1)$
- DAG上DP即可
- 3.求第k大子串（相同的串算一个）

后缀自动机 应用举例

- 1.判断某些串是否为某一个串的子串
- 把长串建SAM，短串放在上面挨个跑就行了
- 2.不同子串个数
- 设f[i]表示从i出发的子串个数，显然，$f[i] = \sum_{(i,j)}(f[j] + 1)$
- DAG上DP即可
- 3.求第k大子串（相同的串算一个）
- 像上一题一样预处理从i出发的子串个数，然后贪心地在SAM上走就行了

后缀自动机 应用举例

- 4.求第k大子串（相同的串不算一个）

后缀自动机 应用举例

- 4.求第k大子串（相同的串不算一个）
- 先在DAG上通过DP处理right集合大小，再用3的方法递推

后缀自动机 应用举例

- 4.求第k大子串（相同的串不算一个）
- 先在DAG上通过DP处理right集合大小，再用3的方法递推
- 5.整个串排列成环，求从哪一个断点得到的字符串字典序最小

后缀自动机 应用举例

- 4.求第k大子串（相同的串不算一个）
- 先在DAG上通过DP处理right集合大小，再用3的方法递推
- 5.整个串排列成环，求从哪一个断点得到的字符串字典序最小
- 把整个串插入两遍，然后在SAM上贪心地走len步即可

后缀自动机 应用举例

- 4.求第k大子串（相同的串不算一个）
- 先在DAG上通过DP处理right集合大小，再用3的方法递推
- 5.整个串排列成环，求从哪一个断点得到的字符串字典序最小
- 把整个串插入两遍，然后在SAM上贪心地走len步即可
- 6.求两个串的最长公共子串

后缀自动机 应用举例

- 4.求第k大子串（相同的串不算一个）
- 先在DAG上通过DP处理right集合大小，再用3的方法递推
- 5.整个串排列成环，求从哪一个断点得到的字符串字典序最小
- 把整个串插入两遍，然后在SAM上贪心地走len步即可
- 6.求两个串的最长公共子串
- 做法特别多，一种方法是对第一个串建SAM，然后把第二个串放在SAM上跑，对经过的节点len取max

后缀自动机 应用举例

- 7.求k个串的最长公共子串

后缀自动机 应用举例

- 7.求k个串的最长公共子串
- 做法也是特别多，这里也是只说一种。我们还是对于第一个串建立SAM，对于每一个节点，记cnt[i]为第i个节点表示的字符串在多少个串里出现过，容易发现，cnt[i]只要在一个串跑完之后，自下向上累加即可维护，时间复杂度$O(\sum len_i)$。

后缀自动机 应用举例

- 7.求k个串的最长公共子串
- 做法也是特别多，这里也是只说一种。我们还是对于第一个串建立SAM，对于每一个节点，记cnt[i]为第i个节点表示的字符串在多少个串里出现过，容易发现，cnt[i]只要在一个串跑完之后，自下向上累加即可维护，时间复杂度$O(\sum len_i)$。
- （其实广义SAM直接插就做完了QWQ）

序列自动机

- 有同学说没听说过这种科技？那是因为这东西没啥用。

序列自动机

- 有同学说没听说过这种科技？那是因为这东西没啥用。
- 大概讲一下这是个啥（其实很简单）

序列自动机

- 有同学说没听说过这种科技？那是因为这东西没啥用。
- 大概讲一下这是个啥（其实很简单）
- $next[i][j]$表示在原串第i位之后第一个字符j的位置

序列自动机

- 有同学说没听说过这种科技？那是因为这东西没啥用。
- 大概讲一下这是个啥（其实很简单）
- $next[i][j]$表示在原串第i位之后第一个字符j的位置
- 构造很显然

序列自动机

- 有同学说没听说过这种科技？那是因为这东西没啥用。
- 大概讲一下这是个啥（其实很简单）
- $next[i][j]$表示在原串第i位之后第一个字符j的位置
- 构造很显然
- 能识别所有子序列

序列自动机

- 有同学说没听说过这种科技？那是因为这东西没啥用。
- 大概讲一下这是个啥（其实很简单）
- $next[i][j]$表示在原串第i位之后第一个字符j的位置
- 构造很显然
- 能识别所有子序列
- 可以用于求不同子序列个数等,DP方式同SAM

回文自动机

- 为什么不讲manacher?

回文自动机

- 为什么不讲manacher?
- 因为PAM基本能做所有manacher能做的吧。

回文自动机

- 为什么不讲manacher?
- 因为PAM基本能做所有manacher能做的吧。
- 回文自动机中的每个节点都对应了一个回文串。

回文自动机

- 为什么不讲manacher?
- 因为PAM基本能做所有manacher能做的吧。
- 回文自动机中的每个节点都对应了一个回文串。
- 可以用于求回文串个数之类的东西

Part2:例题

- 下面是一些例题，因为大家切题太勤快了，比较经典的题大家都做过，所以我选择的题大多数是近几年比赛的字符串题目。

Part2:例题

- 下面是一些例题，因为大家切题太勤快了，比较经典的题大家都做过，所以我选择的题大多数是近几年比赛的字符串题目。
- 希望大家踊跃发表自己对于题目的想法。

Part2:例题

- 下面是一些例题，因为大家切题太勤快了，比较经典的题大家都做过，所以我选择的题大多数是近几年比赛的字符串题目。
- 希望大家踊跃发表自己对于题目的想法。
- 如果有同学做过/一眼秒了，请给其他同学留几分钟思考时间再上台。

Part2:例题

- 下面是一些例题，因为大家切题太勤快了，比较经典的题大家都做过，所以我选择的题大多数是近几年比赛的字符串题目。
- 希望大家踊跃发表自己对于题目的想法。
- 如果有同学做过/一眼秒了，请给其他同学留几分钟思考时间再上台。
- （如果没有同学愿意主动上台，我可能会钦定一位同学

[HAOI2016]找相同字符

- 给定两个字符串，求出在两个字符串中各取出一个子串使得这两个子串相同的方案数。

[HAOI2016]找相同字符

- 给定两个字符串，求出在两个字符串中各取出一个子串使得这两个子串相同的方案数。
- $1 \leq len1, len2 \leq 200000$

Solution

- 后缀数组做法:

Solution

- 后缀数组做法：
- 首先，考虑一个不那么优秀的做法，我们可以直接暴力枚举任意两个后缀，一个属于A串，另一个属于B串，他们的LCP就是会重复的子串个数。这样做显然是O(n^2)的。

Solution

- 后缀数组做法：
- 首先，考虑一个不那么优秀的做法，我们可以直接暴力枚举任意两个后缀，一个属于A串，另一个属于B串，他们的LCP就是会重复的子串个数。这样做显然是O(n^2)的。
- 那么，我们对于每一个height通过单调栈求出来以它为最小值的最靠左的点l[i] 和最靠右的点r[i] 。然后对于一个height它的贡献就是左边的A串后缀数乘上右边的B串后缀数+左边的B串后缀数乘上右边的A串后缀数（公式效果太差，所以放个截图）

Solution

- 后缀数组做法：
- 首先，考虑一个不那么优秀的做法，我们可以直接暴力枚举任意两个后缀，一个属于A串，另一个属于B串，他们的LCP就是会重复的子串个数。这样做显然是O(n^2)的。
- 那么，我们对于每一个height通过单调栈求出来以它为最小值的最靠左的点l[i] 和最靠右的点r[i] 。然后对于一个height它的贡献就是左边的A串后缀数乘上右边的B串后缀数+左边的B串后缀数乘上右边的A串后缀数（公式效果太差，所以放个截图）

$$height[i] * (geta(i-1, l[i]-1) * getb(r[i], i) + getb(i-1, l[i]-1) * geta(r[i], i))$$

Solution

- 后缀数组做法:
- 首先，考虑一个不那么优秀的做法，我们可以直接暴力枚举任意两个后缀，一个属于A串，另一个属于B串，他们的LCP就是会重复的子串个数。这样做显然是O(n^2)的。
- 那么，我们对于每一个height通过单调栈求出来以它为最小值的最靠左的点l[i] 和最靠右的点r[i] 。然后对于一个height它的贡献就是左边的A串后缀数乘上右边的B串后缀数+左边的B串后缀数乘上右边的A串后缀数（公式效果太差，所以放个截图）

$$height[i] * (geta(i-1, l[i]-1) * getb(r[i], i) + getb(i-1, l[i]-1) * geta(r[i], i))$$

- 瓶颈在于求后缀数组，如果你开心写个DC3，那就是O(n)了

Solution

- 后缀自动机做法:

Solution

- 后缀自动机做法：
- 显然，这个SA做法目测100行之内写不完，我们有没有稍微好写一点的做法？

Solution

- 后缀自动机做法：
- 显然，这个SA做法目测100行之内写不完，我们有没有稍微好写一点的做法？
- 我们可以将两个串建立广义SAM，然后对于SAM上的每一个结点记录$size[i][0/1]$，表示在第一/二个串中，i结点表示的串出现了多少次，这个直接在parent树上dp即可

Solution

- 后缀自动机做法：
- 显然，这个SA做法目测100行之内写不完，我们有没有稍微好写一点的做法？
- 我们可以将两个串建立广义SAM，然后对于SAM上的每一个结点记录$size[i][0/1]$，表示在第一/二个串中，i结点表示的串出现了多少次，这个直接在parent树上dp即可
- 我们知道，$len[i] - len[fa[i]]$相当于到这个节点的路径数目

Solution

- 后缀自动机做法:
- 显然，这个SA做法目测100行之内写不完，我们有没有稍微好写一点的做法?
- 我们可以将两个串建立广义SAM，然后对于SAM上的每一个结点记录$size[i][0/1]$，表示在第一/二个串中，i结点表示的串出现了多少次，这个直接在parent树上dp即可
- 我们知道，$len[i]-len[fa[i]]$相当于到这个节点的路径数目
- 那么，每一个结点对答案的贡献就是:

Solution

- 后缀自动机做法:
- 显然，这个SA做法目测100行之内写不完，我们有没有稍微好写一点的做法?
- 我们可以将两个串建立广义SAM，然后对于SAM上的每一个结点记录$size[i][0/1]$，表示在第一/二个串中，i结点表示的串出现了多少次，这个直接在parent树上dp即可
- 我们知道，$len[i]-len[fa[i]]$相当于到这个节点的路径数目
- 那么，每一个结点对答案的贡献就是:
- $size[i][0]*size[i][1]*(len[i]-len[fa[i]])$

Solution

- 后缀自动机做法：
- 显然，这个SA做法目测100行之内写不完，我们有没有稍微好写一点的做法？
- 我们可以将两个串建立广义SAM，然后对于SAM上的每一个结点记录$size[i][0/1]$，表示在第一/二个串中，i结点表示的串出现了多少次，这个直接在parent树上dp即可
- 我们知道，$len[i]-len[fa[i]]$相当于到这个节点的路径数目
- 那么，每一个结点对答案的贡献就是：
- $size[i][0]*size[i][1]*(len[i]-len[fa[i]])$
- 好写很多

[HEOI2015]最短不公共子串

- 下面，给两个小写字母串A，B，请你计算：
- (1) A的一个最短的子串，它不是B的子串
- (2) A的一个最短的子串，它不是B的子序列
- (3) A的一个最短的子序列，它不是B的子串
- (4) A的一个最短的子序列，它不是B的子序列
- 对于100%的数据，A和B的长度都不超过2000

[HEOI2015]最短不公共子串

- 下面，给两个小写字母串A，B，请你计算：
- (1) A的一个最短的子串，它不是B的子串
- (2) A的一个最短的子串，它不是B的子序列
- (3) A的一个最短的子序列，它不是B的子串
- (4) A的一个最短的子序列，它不是B的子序列
- 对于100%的数据，A和B的长度都不超过2000

- 因为这个题强行四合一，所以如果能解决任何一个都可以上来讲

Solution

- Subtask1：

Solution

- Subtask1：
- 最短的不存在的->最长的存在的+1

Solution

- Subtask1：
- 最短的不存在的->最长的存在的+1
- 所以我们可以对于A串的每一个后缀求一下和B串每一个后缀的LCP，并取max，这就是能匹配的最长长度

Solution

- Subtask1：
- 最短的不存在的->最长的存在的+1
- 所以我们可以对于A串的每一个后缀求一下和B串每一个后缀的LCP，并取max，这就是能匹配的最长长度
- 具体做法，可以考虑dp

Solution

- Subtask1：
- 最短的不存在的->最长的存在的+1
- 所以我们可以对于A串的每一个后缀求一下和B串每一个后缀的LCP，并取max，这就是能匹配的最长长度
- 具体做法，可以考虑dp
- $f[i][j]$表示A的第i个后缀和B的第j个后缀的LCP

Solution

- Subtask1：
- 最短的不存在的->最长的存在的+1
- 所以我们可以对于A串的每一个后缀求一下和B串每一个后缀的LCP，并取max，这就是能匹配的最长长度
- 具体做法，可以考虑dp
- $f[i][j]$表示A的第i个后缀和B的第j个后缀的LCP
- 转移显然

Solution

- Subtask1：
- 最短的不存在的->最长的存在的+1
- 所以我们可以对于A串的每一个后缀求一下和B串每一个后缀的LCP，并取max，这就是能匹配的最长长度
- 具体做法，可以考虑dp
- $f[i][j]$表示A的第i个后缀和B的第j个后缀的LCP
- 转移显然
- 时间复杂度$O(nm)$

Solution

- Subtask2：

Solution

- Subtask2：
- 对B串建立序列自动机，在上面贪心

Solution

- Subtask2：
- 对B串建立序列自动机，在上面贪心
- $f[i]$表示A的后缀i在序列自动机上能走几步

Solution

- Subtask2：
- 对B串建立序列自动机，在上面贪心
- $f[i]$表示A的后缀i在序列自动机上能走几步
- 失配就表示不能再走下去，更新答案

Solution

- Subtask2：
- 对B串建立序列自动机，在上面贪心
- $f[i]$表示A的后缀i在序列自动机上能走几步
- 失配就表示不能再走下去，更新答案
- ans就是$\min(f[i])$

Solution

- Subtask2：
- 对B串建立序列自动机，在上面贪心
- $f[i]$表示A的后缀i在序列自动机上能走几步
- 失配就表示不能再走下去，更新答案
- ans就是$\min(f[i])$
- 感觉比Subtask1还要简单些？

Solution

- Subtask2：
- 对B串建立序列自动机，在上面贪心
- $f[i]$表示A的后缀i在序列自动机上能走几步
- 失配就表示不能再走下去，更新答案
- ans就是$\min(f[i])$
- 感觉比Subtask1还要简单些?
- 时间复杂度$O(nm)$

Solution

- Subtask3:

Solution

- Subtask3:
- 对B串建立后缀自动机，在后缀自动机上dp，找出最长的可匹配部分并使它尽量短

Solution

- Subtask3:
- 对B串建立后缀自动机，在后缀自动机上dp，找出最长的可匹配部分并使它尽量短
- 设f[i]表示在SAM上跑到位置i的最短长度是多少

Solution

- Subtask3:
- 对B串建立后缀自动机，在后缀自动机上dp，找出最长的可匹配部分并使它尽量短
- 设f[i]表示在SAM上跑到位置i的最短长度是多少
- 每次枚举A的一个字符，更新后缀自动机上的所有结点答案

Solution

- Subtask3:
- 对B串建立后缀自动机，在后缀自动机上dp，找出最长的可匹配部分并使它尽量短
- 设f[i]表示在SAM上跑到位置i的最短长度是多少
- 每次枚举A的一个字符，更新后缀自动机上的所有结点答案
- 如果一个结点的答案之前被更新过，但是现在失配了，那么可以更新答案，原因显然

Solution

- Subtask3:
- 对B串建立后缀自动机，在后缀自动机上dp，找出最长的可匹配部分并使它尽量短
- 设f[i]表示在SAM上跑到位置i的最短长度是多少
- 每次枚举A的一个字符，更新后缀自动机上的所有结点答案
- 如果一个结点的答案之前被更新过，但是现在失配了，那么可以更新答案，原因显然
- 时间复杂度$O(nm)$

Solution

- Subtask4:

Solution

- Subtask4:
- 把后缀自动机改成序列自动机，做法和Subtask3就一模一样了

Solution

- Subtask4:
- 把后缀自动机改成序列自动机，做法和Subtask3就一模一样了
- 时间复杂度$O(nm)$

Solution

- Subtask4:
- 把后缀自动机改成序列自动机，做法和Subtask3就一模一样了
- 时间复杂度$O(nm)$

- 但是这样做的话，相当于做了四道题，有没有好一些的方法呢？

Solution

- 其实可以每个串建一个后缀自动机，以及一个序列自动机，分别用于识别子串和子序列

Solution

- 其实可以每个串建一个后缀自动机，以及一个序列自动机，分别用于识别子串和子序列
- 每一问就相当于在A、B串相应的自动机上跑广搜，遇到第一个A串匹配、B串失配的字符就输出当前深度

Solution

- 其实可以每个串建一个后缀自动机，以及一个序列自动机，分别用于识别子串和子序列
- 每一问就相当于在A、B串相应的自动机上跑广搜，遇到第一个A串匹配、B串失配的字符就输出当前深度
- 这样我们就可以写一个广搜，然后复制3遍，改几个字母就做好了。

Solution

- 其实可以每个串建一个后缀自动机，以及一个序列自动机，分别用于识别子串和子序列
- 每一问就相当于在A、B串相应的自动机上跑广搜，遇到第一个A串匹配、B串失配的字符就输出当前深度
- 这样我们就可以写一个广搜，然后复制3遍，改几个字母就做好了。
- 是不是好写很多?

[AHOI2013]差异

- 求一个字符串的所有后缀中，任选2个，计算 $\sum_{1 \le i < j \le n} len(Ti) + len(Tj) - 2 \times lcp(Ti, Tj)$
- $N \le 1e5$

[AHOI2013]差异

- 求一个字符串的所有后缀中，任选2个，计算 $\sum_{1\le i<j\le n} len(Ti) + len(Tj) - 2 \times lcp(Ti, Tj)$
- $N \le 1e5$
- 一眼秒了？看个加强版

[AHOI2013]差异

- 求一个字符串的所有后缀中，任选2个，计算 $\sum_{1\le i<j\le n} len(Ti) + len(Tj) - 2 \times lcp(Ti, Tj)$
- $N \le 1e5$
- 一眼秒了？看个加强版
- 求一个字符串的所有子串中，任选2个，计算$\sum[|A| + |B| - 2 \times LCP(A, B) \le L]$。
- $N \le 1e5$

Solution

- 对于原题，很多人选择了把式子拆开然后分别计算贡献

Solution

- 对于原题，很多人选择了把式子拆开然后分别计算贡献
- 但是为什么要拆掉这个美观的式子呢？我们考虑把它凑到后缀树上，就变成了两个节点的距离（LCP对应的点就是它们的LCA）！

Solution

- 对于原题，很多人选择了把式子拆开然后分别计算贡献
- 但是为什么要拆掉这个美观的式子呢？我们考虑把它凑到后缀树上，就变成了两个节点的距离（LCP对应的点就是它们的LCA）！
- 这样，我们只要进行一次树形dp就可以求出总路径条数

Solution

- 对于原题，很多人选择了把式子拆开然后分别计算贡献
- 但是为什么要拆掉这个美观的式子呢？我们考虑把它凑到后缀树上，就变成了两个节点的距离（LCP对应的点就是它们的LCA）！
- 这样，我们只要进行一次树形dp就可以求出总路径条数
- 时间复杂度O(n)

Solution

- 对于加强版， 类似进行分析

Solution

- 对于加强版，类似进行分析
- 得出结论大概就是，要求的式子等于树上路径<=L的条数

Solution

- 对于加强版，类似进行分析
- 得出结论大概就是，要求的式子等于树上路径<=L的条数
- 这显然是个点分治模板题。

Solution

- 对于加强版，类似进行分析
- 得出结论大概就是，要求的式子等于树上路径<=L的条数
- 这显然是个点分治模板题。
- 那么直接在后缀树上点分治就好了。

原样输出

- 给n个字符串，从每一个字符串中取出一个可以为空的子串，问有多少种不同取法，答案膜 1e9+7
- $n \leq 1e6, \sum len \leq 2e6$，字符集大小=4

原样输出

- 给n个字符串，从每一个字符串中取出一个可以为空的子串，问有多少种不同取法，答案膜 1e9+7
- $n \le 1e6, \sum len \le 2e6$，字符集大小=4

- 这个题的讲评大家应该都听了，所以大家都会，我就简单讲一下

Solution

- 考虑$n = 1$，直接建个SAM跑就行了

Solution

- 考虑$n = 1$，直接建个SAM跑就行了
- 现在n很大，我们建n个SAM

Solution

- 考虑$n = 1$，直接建个SAM跑就行了
- 现在n很大，我们建n个SAM
- 对于某个结点，如果它识别不了某个字符了，那就应该跳到后面的SAM中。

Solution

- 考虑$n = 1$，直接建个SAM跑就行了
- 现在n很大，我们建n个SAM
- 对于某个结点，如果它识别不了某个字符了，那就应该跳到后面的SAM中。
- 对于这个过程，我们可以预处理能识别某个字符的下一个SAM的编号，直接跳过去即可

Solution

- 考虑$n = 1$，直接建个SAM跑就行了
- 现在n很大，我们建n个SAM
- 对于某个结点，如果它识别不了某个字符了，那就应该跳到后面的SAM中。
- 对于这个过程，我们可以预处理能识别某个字符的下一个SAM的编号，直接跳过去即可
- 计算个数只要dp一下就好了

Solution

- 考虑$n = 1$，直接建个SAM跑就行了
- 现在n很大，我们建n个SAM
- 对于某个结点，如果它识别不了某个字符了，那就应该跳到后面的SAM中。
- 对于这个过程，我们可以预处理能识别某个字符的下一个SAM的编号，直接跳过去即可
- 计算个数只要dp一下就好了
- 时间复杂度$O(n)$

[COCI2015] Divljak

- 有n个字符串$S[1 \dots n]$和一个字符串集合T，一开始集合是空的。
- 接下来会发生q个操作，操作有两种形式：
- 往集合里添加一个字符串P。
- 询问集合T中有多少个字符串包含串Sx。
- $1 \le n, q \le 1e5$,长度总和 $\le 4e6$

Solution

- 对S中的n个串构建AC自动机，并顺手把fail树搞出来。

Solution

- 对S中的n个串构建AC自动机，并顺手把fail树搞出来。
- 然后对于新加入T中的串，就把这个串扔进AC自动机里走一遍,会经过一些节点，每个节点在fail树上到根的路径上的节点对应的串都在这个串里出现。

Solution

- 对S中的n个串构建AC自动机，并顺手把fail树搞出来。
- 然后对于新加入T中的串，就把这个串扔进AC自动机里走一遍,会经过一些节点，每个节点在fail树上到根的路径上的节点对应的串都在这个串里出现。
- 那么我们把这些节点到根节点的路径的并上的每个节点都+1，那么按节点的dfs序排序，每个节点处+1，相邻节点（**不包括第一个和最后一个**）处-1即可。

Solution

- 对S中的n个串构建AC自动机，并顺手把fail树搞出来。
- 然后对于新加入T中的串，就把这个串扔进AC自动机里走一遍,会经过一些节点，每个节点在fail树上到根的路径上的节点对应的串都在这个串里出现。
- 那么我们把这些节点到根节点的路径的并上的每个节点都+1，那么按节点的dfs序排序，每个节点处+1，相邻节点（**不包括第一个和最后一个**）处-1即可。
- 子树和可以使用简单的树状数组维护。

[SDOI2014]数数

- 我们称一个正整数N是幸运数，当且仅当它的十进制表示中不包含数字串集合S中任意一个元素作为其子串。例如当$S =$ (22，333，0233)时，233是幸运数，2333、20233、3223不是幸运数。给定N和S，计算不大于N的幸运数个数。
- $N \leq 10^{1200}, \sum|S| \leq 1500\ M \leq 100$

Solution

- 对给定的若干个串建立AC自动机。对自动机上的每个节点，维护一个标记，表示在fail树上，它到根的路径中是否存在词尾节点。（词尾节点即每次插入一个串时最后到达的那个节点）

Solution

- 对给定的若干个串建立AC自动机。对自动机上的每个节点，维护一个标记，表示在fail树上，它到根的路径中是否存在词尾节点。（词尾节点即每次插入一个串时最后到达的那个节点）
- ~~因为打公式太麻烦了，所以一部分就以图片形式展示了~~

Solution

- 对给定的若干个串建立AC自动机。对自动机上的每个节点，维护一个标记，表示在fail树上，它到根的路径中是否存在词尾节点。（词尾节点即每次插入一个串时最后到达的那个节点）
- ~~因为打公式太麻烦了，所以一部分就以图片形式展示了~~

设$f_{i,j,0/1}$表示考虑到n的第i位（从高到低），当前在自动机的j号节点，目前为止 没有挨着 / 正在挨着 上界时，答案是多少。设ch(x,c)表示x号节点的c号子节点，于是可以写出转移：

1. 若当前没有挨着上界，前一位不挨着上界，则：$f_{i-1,p,0} \longrightarrow f_{i,ch(p,1\sim 9),0}$
2. 若当前没有挨着上界，前一位挨着上界，则：$f_{i-1,p,1} \longrightarrow f_{i,ch(p,1\sim upper_i-1),0}$
3. 若当前要挨着上界，那前一位必然要挨着上界，则：$f_{i-1,p,1} \longrightarrow f_{i,ch(p,upper_i),1}$

最后的答案显然就是：$\sum_{i=0}^{tot} f_{n,i,0} + f_{n,i,1}$

喵星球上的点名（加强版）

- 有n只喵，每只喵有两个字符串Ai,Bi，q次询问一个字符串是多少喵的Ai或Bi的子串，最后输出每只喵被点了多少次
- **$\boldsymbol{n = 1e5, m = 5e5}$,字符集大小$\boldsymbol{1e5}$,字符串总长$\boldsymbol{1e6}$，关于第一问强制在线**

喵星球上的点名（加强版）

- 有n只喵，每只喵有两个字符串Ai,Bi，q次询问一个字符串是多少喵的Ai或Bi的子串，最后输出每只喵被点了多少次
- **$\boldsymbol{n = 1e5, m = 5e5}$,字符集大小$\boldsymbol{1e5}$,字符串总长$\boldsymbol{1e6}$，关于第一问强制在线**
- （原题：$N = 2e4$，$M = 5e4$，字符集大小$1e4$,字符串总长$1e5$,可离线

喵星球上的点名（加强版）

- 有n只喵，每只喵有两个字符串Ai,Bi，q次询问一个字符串是多少喵的Ai或Bi的子串，最后输出每只喵被点了多少次
- **$n = 1e5, m = 5e5$,字符集大小$1e5$,字符串总长$1e6$，关于第一问强制在线**
- （原题：$N = 2e4$，$M = 5e4$，字符集大小$1e4$,字符串总长$1e5$,可离线
- 这个题好多乱搞&复杂度根本不对的做法

喵星球上的点名（加强版）

- 有n只喵，每只喵有两个字符串Ai,Bi，q次询问一个字符串是多少喵的Ai或Bi的子串，最后输出每只喵被点了多少次
- **$\boldsymbol{n = 1e5, m = 5e5}$,字符集大小$\boldsymbol{1e5}$,字符串总长$\boldsymbol{1e6}$，关于第一问强制在线**
- （原题：$N = 2e4$，$M = 5e4$，字符集大小$1e4$,字符串总长$1e5$,可离线
- 这个题好多乱搞&复杂度根本不对的做法
- 不知道为什么我卡了校内OJ的rank1，可能大家都没写正解啊
- （今天课讲完我可能就不是rank1了

Solution

- 先考虑一个离线的$O(msqrtn + nlogn)$的做法

Solution

- 先考虑一个离线的$O(msqrtn + nlogn)$的做法
- 后缀数组+莫队

Solution

- 先考虑一个离线的$O(msqrtn + nlogn)$的做法
- 后缀数组+莫队
- 首先我们考虑将所有喵的姓和名拼接在一起（注意中间还是要用特殊字符连接一下），并记录每一个位置上的字符是哪只喵的

Solution

- 先考虑一个离线的$O(msqrtn + nlogn)$的做法
- 后缀数组+莫队
- 首先我们考虑将所有喵的姓和名拼接在一起（注意中间还是要用特殊字符连接一下），并记录每一个位置上的字符是哪只喵的
- 然后考虑对于询问，由于后缀数组的sa数组表示的是排名为i的后缀的位置，因此我们可以根据排名二分出每个询问在sa数组上对应的区间。

Solution

- 先考虑一个离线的$O(msqrtn + nlogn)$的做法
- 后缀数组+莫队
- 首先我们考虑将所有喵的姓和名拼接在一起（注意中间还是要用特殊字符连接一下），并记录每一个位置上的字符是哪只喵的
- 然后考虑对于询问，由于后缀数组的sa数组表示的是排名为i的后缀的位置，因此我们可以根据排名二分出每个询问在sa数组上对应的区间。
- 剩下的部分：对于第一问就是区间询问不同数字的个数

Solution

- 先考虑一个离线的$O(msqrtn + nlogn)$的做法
- 后缀数组+莫队
- 首先我们考虑将所有喵的姓和名拼接在一起（注意中间还是要用特殊字符连接一下），并记录每一个位置上的字符是哪只喵的
- 然后考虑对于询问，由于后缀数组的sa数组表示的是排名为i的后缀的位置，因此我们可以根据排名二分出每个询问在sa数组上对应的区间。
- 剩下的部分：对于第一问就是区间询问不同数字的个数
- 对于第二问，计算进入和退出该区间的时间即可

Solution

- 先考虑一个离线的$O(msqrtn + nlogn)$的做法
- 后缀数组+莫队
- 首先我们考虑将所有喵的姓和名拼接在一起（注意中间还是要用特殊字符连接一下），并记录每一个位置上的字符是哪只喵的
- 然后考虑对于询问，由于后缀数组的sa数组表示的是排名为i的后缀的位置，因此我们可以根据排名二分出每个询问在sa数组上对应的区间。
- 剩下的部分：对于第一问就是区间询问不同数字的个数
- 对于第二问，计算进入和退出该区间的时间即可
- 莫队即可

Solution

- 现在我们来优化这个做法

Solution

- 现在我们来优化这个做法
- 已知上述做法的瓶颈在于莫队是根号级别的

Solution

- 现在我们来优化这个做法
- 已知上述做法的瓶颈在于莫队是根号级别的
- 我们用其他数据结构代替掉莫队，比如树状数组

Solution

- 现在我们来优化这个做法
- 已知上述做法的瓶颈在于莫队是根号级别的
- 我们用其他数据结构代替掉莫队，比如树状数组
- 如果你做过[SDOI2009]HH的项链，那么这就不是难事了

Solution

- 现在我们来优化这个做法
- 已知上述做法的瓶颈在于莫队是根号级别的
- 我们用其他数据结构代替掉莫队，比如树状数组
- 如果你做过[SDOI2009]HH的项链，那么这就不是难事了
- 用这种做法，第二问大同小异

Solution

- 现在我们来优化这个做法
- 已知上述做法的瓶颈在于莫队是根号级别的
- 我们用其他数据结构代替掉莫队，比如树状数组
- 如果你做过[SDOI2009]HH的项链，那么这就不是难事了
- 用这种做法，第二问大同小异
- 现在时间复杂度降到了$O((n+m)logL)$

Solution

- 现在我们来优化这个做法
- 已知上述做法的瓶颈在于莫队是根号级别的
- 我们用其他数据结构代替掉莫队，比如树状数组
- 如果你做过[SDOI2009]HH的项链，那么这就不是难事了
- 用这种做法，第二问大同小异
- 现在时间复杂度降到了$O((n+m)logL)$
- 考虑把树状数组改成主席树，就可以在线回答询问了

Solution

- 现在我们来优化这个做法
- 已知上述做法的瓶颈在于莫队是根号级别的
- 我们用其他数据结构代替掉莫队，比如树状数组
- 如果你做过[SDOI2009]HH的项链，那么这就不是难事了
- 用这种做法，第二问大同小异
- 现在时间复杂度降到了$O((n+m)logL)$
- 考虑把树状数组改成主席树，就可以在线回答询问了
- 时间复杂度$O((n+m)\ logL)$

Solution

- 现在我们来优化这个做法
- 已知上述做法的瓶颈在于莫队是根号级别的
- 我们用其他数据结构代替掉莫队，比如树状数组
- 如果你做过[SDOI2009]HH的项链，那么这就不是难事了
- 用这种做法，第二问大同小异
- 现在时间复杂度降到了$O((n+m)logL)$
- 考虑把树状数组改成主席树，就可以在线回答询问了
- 时间复杂度$O((n+m)\ logL)$
- 原题数据跑不过莫队，但是数据大了优势就会很明显

其他神仙做法

M 个模式串建 AC 自动机，bit 维护 fail 树的 dfs 序。设每个姓和名为一组，每组两个串做一次树链求并，最终的子树和就是第一问。若是把每组两个串往下跑，就能得到出现的所有模式串的个数，但是显然有重复，所以将匹配过程中到达的所有结点取出，按 dfs 序排序，加上每个点的 f ，减去相邻两点 LCA的 f(f是fail链上的end节点数)。O(nlogn)，常数略大跑肯定不过暴力

13:03:49

orz

[NOI2015]品酒大会

- 求有多少对后缀满足$LCP(i,j) \geq len$，每个后缀有一个权值ai，并同时求出满足条件的后缀的权值乘积的最大值。
- $N \leq 3e5$，$|ai| \leq 1e9$

Solution

- 首先，大于等于可以转化成等于的前缀和与前缀max

Solution

- 首先，大于等于可以转化成等于的前缀和与前缀max
- 根据SAM的性质，两后缀的LCP长度等于parent树上它们LCA的深度

Solution

- 首先，大于等于可以转化成等于的前缀和与前缀max
- 根据SAM的性质，两后缀的LCP长度等于parent树上它们LCA的深度
- 对于第一问，就变成了一个简单的树形dp，求出每个节点是多少对节点的LCA即可

Solution

- 首先，大于等于可以转化成等于的前缀和与前缀max
- 根据SAM的性质，两后缀的LCP长度等于parent树上它们LCA的深度
- 对于第一问，就变成了一个简单的树形dp，求出每个节点是多少对节点的LCA即可
- 对于第二问，因为权值可能为负，所以对每个节点记录最大值，次大值，最小值，次小值，也是一个很简单的树形dp

[NOI2016]优秀的拆分

- 给出一个长度为n的串，求它的每一个子串拆成$AABB$形式的方案数之和。

[NOI2016]优秀的拆分

- 给出一个长度为n的串，求它的每一个子串拆成$AABB$形式的方案数之和。
- 对于95%的数据,$n <= 2000$

[NOI2016]优秀的拆分

- 给出一个长度为n的串，求它的每一个子串拆成$AABB$形式的方案数之和。
- 对于95%的数据,$n <= 2000$
- 对于100%的数据,$n <= 1e5$

Solution

- ~~送95分好良心啊~~

Solution

- ~~送95分好良心啊~~
- 对于前95分， 我们设$f[i]$表示以i结尾的可以写成AA形式的方案数

Solution

- ~~送95分好良心啊~~
- 对于前95分，我们设$f[i]$表示以i结尾的可以写成AA形式的方案数
- 同样的，设$g[i]$表示以j开头的可以写成AA形式的方案数

Solution

- ~~送95分好良心啊~~
- 对于前95分，我们设$f[i]$表示以i结尾的可以写成AA形式的方案数
- 同样的，设$g[i]$表示以j开头的可以写成AA形式的方案数
- 显然，最后答案是$\Sigma f[i] * g[i+1]$，f和g的计算互相独立

Solution

- ~~送95分好良心啊~~
- 对于前95分，我们设$f[i]$表示以i结尾的可以写成AA形式的方案数
- 同样的，设$g[i]$表示以j开头的可以写成AA形式的方案数
- 显然，最后答案是$\Sigma f[i] * g[i+1]$，f和g的计算互相独立
- 暴力计算f和g是$O(n^2)$的，这样我们就获得了95分的好成绩

Solution

- 想做100分可能要难一些了

Solution

- 想做100分可能要难一些了
- 我们枚举A的长度len，把所有len的倍数的点设为关键点

Solution

- 想做100分可能要难一些了
- 我们枚举A的长度len，把所有len的倍数的点设为关键点
- 如果一个AA满足要求，显然它经过至少两个关键点

Solution

- 想做100分可能要难一些了
- 我们枚举A的长度len，把所有len的倍数的点设为关键点
- 如果一个AA满足要求，显然它经过至少两个关键点
- 那么我们要算的就是相邻两个关键点对答案的贡献：

Solution

- 想做100分可能要难一些了
- 我们枚举A的长度len，把所有len的倍数的点设为关键点
- 如果一个AA满足要求，显然它经过至少两个关键点
- 那么我们要算的就是相邻两个关键点对答案的贡献：
- 记相邻两个关键点为i,j，那么$j = i + len$

Solution

- 想做100分可能要难一些了
- 我们枚举A的长度len，把所有len的倍数的点设为关键点
- 如果一个AA满足要求，显然它经过至少两个关键点
- 那么我们要算的就是相邻两个关键点对答案的贡献：
- 记相邻两个关键点为i,j，那么$j = i + len$
- 先放一下结论
- 求出$L1 = LCP(suf(i), suf(j)), L2 = LCS(pre(i-1), pre(j-1))$
- 如果$L1 + L2 < len$，那么就不能构成AA

Solution

- 想做100分可能要难一些了
- 我们枚举A的长度len，把所有len的倍数的点设为关键点
- 如果一个AA满足要求，显然它经过至少两个关键点
- 那么我们要算的就是相邻两个关键点对答案的贡献：
- 记相邻两个关键点为i,j，那么$j = i + len$
- 先放一下结论
- 求出$L1 = LCP(suf(i), suf(j)), L2 = LCS(pre(i-1), pre(j-1))$
- 如果$L1 + L2 < len$，那么就不能构成AA
- 为什么呢？

Solution

$$\underbrace{\dots\dots i-1}_{Lcs}\ \overbrace{\underbrace{i\dots\dots}_{Lcp}\ \dots\underbrace{\dots\dots j-1}_{Lcs}}^{len}\ \underbrace{j\dots\dots}_{Lcp}$$

Solution

$$\underbrace{......i-1}_{Lcs}\ \overbrace{\underbrace{i.......}_{Lcp}\ ....\underbrace{.......j-1}_{Lcs}}^{len}\ \underbrace{j.......}_{Lcp}$$

- 反之，如果中间两段LCS,LCP有交，那么A串的端点落在交的区间的任意一个点上都是可以的，因为我们可以保证向后平移之后会出现一个完全一样的A串。

Solution

$$\underbrace{\ldots\ldots i-1}_{Lcs}\ \overbrace{\underbrace{i\ldots\ldots}_{Lcp}\ \ldots\underbrace{\ldots\ldots j-1}_{Lcs}}^{len}\ \underbrace{j\ldots\ldots}_{Lcp}$$

- 反之，如果中间两段LCS,LCP有交，那么A串的端点落在交的区间的任意一个点上都是可以的，因为我们可以保证向后平移之后会出现一个完全一样的A串。
- LCS,LCP可以使用后缀数组+ST表快速计算

Solution

$$\underbrace{\ldots\ldots i-1}_{Lcs}\ \overbrace{\underbrace{i\ldots\ldots}_{Lcp}\ \ldots\underbrace{\ldots\ldots j-1}_{Lcs}}^{len}\ \underbrace{j\ldots\ldots}_{Lcp}$$

- 反之，如果中间两段LCS,LCP有交，那么A串的端点落在交的区间的任意一个点上都是可以的，因为我们可以保证向后平移之后会出现一个完全一样的A串。
- LCS,LCP可以使用后缀数组+ST表快速计算
- 时间复杂度$O(nlogn)$

[NOI2018]你的名字

- 给定一个模板串S,多组询问,每次给出一个询问字符串T和一个区间(l,r)，要求你输出T有多少个**本质不同**的子串,满足这个子串没有在S的(l,r)这段区间当中出现

[NOI2018]你的名字

- 给定一个模板串S,多组询问,每次给出一个询问字符串T和一个区间(l,r)，要求你输出T有多少个**本质不同**的子串,满足这个子串没有在S的(l,r)这段区间当中出现
- 对于68%的测试数据，$l = 1, r = |S|$

[NOI2018]你的名字

- 给定一个模板串S,多组询问,每次给出一个询问字符串T和一个区间(l,r)，要求你输出T有多少个**本质不同**的子串,满足这个子串没有在S的(l,r)这段区间当中出现
- 对于68%的测试数据，$l=1,r=|S|$
- 对于100%的测试数据，$|S| \le 5e5, q \le 1e5, \Sigma|T| \le 1e6$

Solution

- 显然，我们要对S和每一个T建立SAM

Solution

- 显然，我们要对S和每一个T建立SAM
- 我们考虑计算一个$\lim$ 数组，$\lim[i]$表示当前询问的T的第i个前缀的最长在S中出现的后缀，显然答案就是 $\sum_{i=2}^{cnt}\max(0, len_i - \max(len_{fa}, lim_{pos}))$

Solution

- 显然，我们要对S和每一个T建立SAM
- 我们考虑计算一个$\lim$ 数组，$\lim[i]$表示当前询问的T的第i个前缀的最长在S中出现的后缀，显然答案就是$\sum_{i=2}^{cnt}\max(0, len_i - \max(len_{fa}, lim_{pos}))$
- len表示意义和SAM中相同，pos表示当前节点表示的第一个字符串位置，现在的问题就是怎么求出$\lim$

Solution

- 对于$l = 1, r = |S|$

Solution

- 对于$l = 1, r = |S|$
- 可以直接在S的SAM上跳求lim

Solution

- 对于$l = 1, r = |S|$
- 可以直接在S的SAM上跳求lim
- 对于l, r任意

Solution

- 对于$l = 1, r = |S|$
- 可以直接在S的SAM上跳求$\lim$
- 对于l, r任意
- 考虑维护每个节点的$right$集合，可以使用线段树合并实现

Solution

- 对于$l = 1, r = |S|$
- 可以直接在S的SAM上跳求lim
- 对于l, r任意
- 考虑维护每个节点的$right$集合，可以使用线段树合并实现
- 这样，区间$[l, r]$的SAM上是否有这个节点可以转化成询问$right$集合中有没有在$[l + len, r]$ (len为当前匹配长度)中的元素。然后就可以做了。

Solution

- 对于$l = 1, r = |S|$
- 可以直接在S的SAM上跳求lim
- 对于l, r任意
- 考虑维护每个节点的$right$集合，可以使用线段树合并实现
- 这样，区间$[l, r]$的SAM上是否有这个节点可以转化成询问$right$集合中有没有在$[l + len, r]$ (len为当前匹配长度)中的元素。然后就可以做了。
- 时间复杂度$O(Llogn)$

CF666E Forensic Examination

- 给定一个串S和一个字符串数组$T[1 \dots m]$，有q次询问，每次给定$l1, r1, l2, r2$，请求出S的$l1 \dots r1$的子串在T的$l2 \dots r2$中哪个串出现次数最多，并输出出现次数。
- $|S| \le 5e5, m \le 5e4, q \le 5e5, \sum|T| \le 5e4$ 的长度总和
- $6000ms\ /\ 768MB$

Solution

- 先对所有的模板串建一个广义SAM

Solution

- 先对所有的模板串建一个广义SAM
- 因为查询的是模板串的区间，我们使用权值线段树，下标是串的id

Solution

- 先对所有的模板串建一个广义SAM
- 因为查询的是模板串的区间，我们使用权值线段树，下标是串的id
- 我们为SAM上每个节点建立一个动态开点线段树，记录区间的最大值，以及最大值所在的下标

Solution

- 先对所有的模板串建一个广义SAM
- 因为查询的是模板串的区间，我们使用权值线段树，下标是串的 id
- 我们为SAM上每个节点建立一个动态开点线段树，记录区间的最大值，以及最大值所在的下标
- 建完广义SAM后跑一个线段树合并

Solution

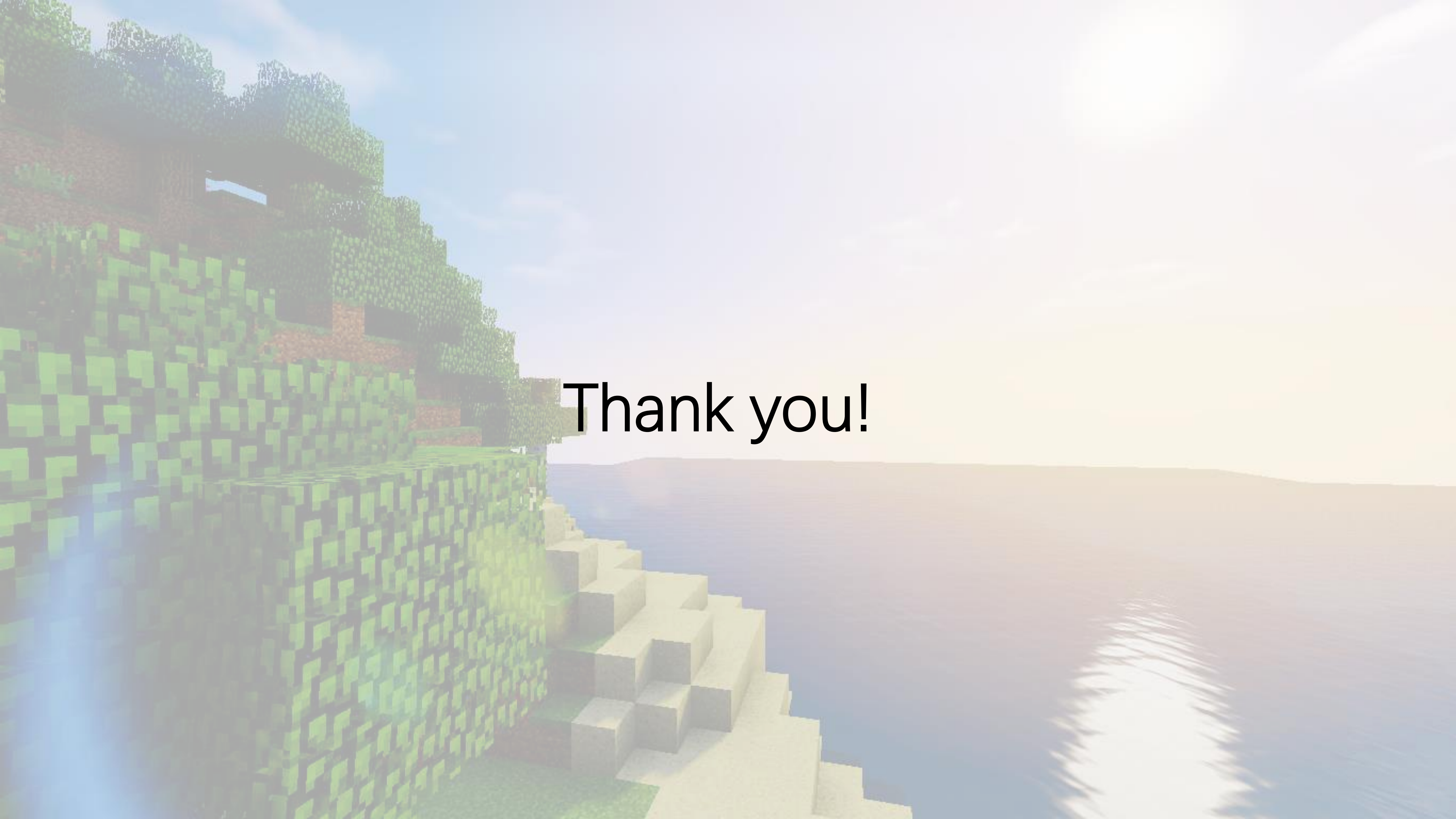
- 先对所有的模板串建一个广义SAM
- 因为查询的是模板串的区间，我们使用权值线段树，下标是串的id
- 我们为SAM上每个节点建立一个动态开点线段树，记录区间的最大值，以及最大值所在的下标
- 建完广义SAM后跑一个线段树合并
- 预处理出S的前缀$S[1..r]$ 在SAM里能匹配上的最长后缀的长度以及它匹配到的位置

Solution

- 每次查询的时候从后缀$S[1..r]$ 匹配到的位置开始，在parent树上倍增，找到R最小的节点，使得$R_u \geq r - l + 1$

Solution

- 每次查询的时候从后缀$S[1..r]$ 匹配到的位置开始，在parent树上倍增，找到R最小的节点，使得$R_u \geq r-l+1$
- 可以发现，倍增到的点正好包含了$S[l..r]$ 这个状态，在这个节点的线段树上查询$p_l \dots p_r$之间的最大值，就是答案



Thank you!