字符串选讲

fadeawayQAQ

August 18, 2024

符号与约定

符号与约定

- S 表示字符串，c 小写字母表示字符。
- $|S|$ 表示 S 的长度，$S[i]$ 表示 S 的第 i 个字符，下标从 1 开始。
- $|\Sigma|$ 表示字符集大小。
- $S[l:r]$ 表示字符串第 l 个字符到第 r 个字符组成的子串。
- AB 表示把字符串 A 和字符串 B 拼起来。
- A^k 表示把 k 个 A 拼起来。
- $suf_S[i]$ 表示 $S[i:|S|]$，$pre_S[i]$ 表示 $S[1:i]$。

Hash

Hash

这里可以简单地把 hash 理解为建立一个字符串到数的映射关系。

一般将映射关系定义为：

$$f(S)=\sum_{i=1}^{|S|} S[i]\cdot t^{|S|-i} \mod P$$

这样就可以在 $O(n)$ 预处理之后 $O(1)$ 实现字符串之间的拼接、取出子串、判断是否相同等操作。

Hash

解决冲突

不难发现这个映射关系不是单射的，可能会有很多字符串有同一个 *hash* 值导致发生一些奇妙的错误.

为了避免这一情况的发生，我们一般会取多个优质模数（大质数）以提高它的正确性。

Hash

解决冲突

不难发现这个映射关系不是单射的，可能会有很多字符串有同一个 *hash* 值导致发生一些奇妙的错误.

为了避免这一情况的发生，我们一般会取多个优质模数（大质数）以提高它的正确性。

- 2^{64} 自然溢出是非常不优秀的，很容易构造冲突。

Hash

解决冲突

不难发现这个映射关系不是单射的，可能会有很多字符串有同一个 *hash* 值导致发生一些奇妙的错误.

为了避免这一情况的发生，我们一般会取多个优质模数（大质数）以提高它的正确性。

- 2^{64} 自然溢出是非常不优秀的，很容易构造冲突。
- 链表法

Hash

字符串排序

- 给定 n 个字符串 S_i ，将 S_i 按字典序排序。
  $\sum |S_i| \leq 10^6$

Hash

字符串排序

- 给定 n 个字符串 S_i ，将 S_i 按字典序排序。
  $\sum |S_i| \leq 10^6$
- 考虑用 *hash* 优化字符串大小的比较。
  二分两个字符串的第一个不相等的位置，用 *hash* 比较两个字符串的前缀是否相等。
  时间复杂度 $O(n \log n \log |S|)$。

Hash

字符串排序（更优的做法）

考虑对字符串做 random_shuffle，然后直接 sort 的时间复杂度。

每一个串都和其他串比较 $O(logn)$ 次，一次比较最坏花费 $O(|S_i|)$。

故均摊时间复杂度为 $O(|S|\log n)$

KMP

Border

Definition 1

对于一个长为 n 的字符串 S，当 $S[1:i]=S[n-i+1:n]$ 时，称 $S[1:i]$ 为 S 的一个 border。

Definition 2

对于一个长为 n 的字符串 S 和正整数 p，若对于所有 $i\in\{1,2,...,n-p+1\}$，都有 $S[i]=S[i+p]$，称 p 为 S 的一个周期。

KMP

Border

Theorem 1

$S[1:i]$ 为 S 的一个 border 等价于 $n-i$ 为 S 的一个周期。

Theorem 2

字符串 S 的 border 可以被划分为 $O(\log n)$ 个等差数列。

KMP

KMP

- 基本的字符串匹配算法。
  我们用 $Fail_i$ 表示 $pre_S[i]$ 的最长 $border$ 的长度。
  这样我们用 T 在 S 上匹配的时候，若 $T+c$ 失配了，就跳转到 $fail[T]$，看看 $fail[T]+c$ 是否可以匹配，直到 T 为空或者能够匹配。
  求出 $Fail[i]$ 的过程是类似的，相当于自己和自己做匹配。
  时间复杂度 $O(n)$。

KMP

[NOI2014] 动物园

- 题意：给定一个字符串 S，对于其每个前缀 $pre_S[i]$，求有多少个长度不超过 $\lfloor \frac{i}{2} \rfloor$ 的 $border$，其中 $|S| \le 10^6$。

KMP

[NOI2014] 动物园

- 题意：给定一个字符串 S ，对于其每个前缀 $pre_S[i]$ ，求有多少个长度不超过 $\lfloor \frac{i}{2} \rfloor$ 的 $border$ ，其中 $|S| \leq 10^6$ 。
- 一个简单的方法是倍增，时间复杂度为 $O(n \log n)$ 。
维护一个 $\leq len/2$ 的指针，类似 KMP 做，时间复杂度是线性的。

KMP

QOJ2273 Suffixes may Contain Prefixes

- 题意：给定一个串 S 和正整数 n，构造一个长度为 n 的最优的字符串 T，使得 $\sum_i LCP(Suf_T[i], S)$ 最大。其中 $|S|, x \leq 2000$。

KMP

QOJ2273 Suffixes may Contain Prefixes

- 题意：给定一个串 S 和正整数 n，构造一个长度为 n 的最优的字符串 T，使得 $\sum_i LCP(Suf_T[i], S)$ 最大。其中 $|S|, x \leq 2000$ 。
- 考虑从后往前确定 T 中字符的过程，$dp_{i,j}$ 表示从当前 T 长度为 i，且与 S 的 LCP 为 j，且之后不会存在贡献的右端点比当前的右端点更右。
  直接枚举 j, k 表示从前缀 j 转移到前缀 k 并计算贡献（先去尾，再加头）是 $O(n^3)$ 的。

KMP

QOJ2273 Suffixes may Contain Prefixes

- 题意：给定一个串 S 和正整数 n，构造一个长度为 n 的最优的字符串 T，使得 $\sum_i LCP(Suf_T[i], S)$ 最大。其中 $|S|, x \leq 2000$。
- 考虑从后往前确定 T 中字符的过程，$dp_{i,j}$ 表示从当前 T 长度为 i，且与 S 的 LCP 为 j，且之后不会存在贡献的右端点比当前的右端点更右。
  直接枚举 j, k 表示从前缀 j 转移到前缀 k 并计算贡献（先去尾，再加头）是 $O(n^3)$ 的。
- 更优的，一方面 $dp_{i,j}$ 可以从它的 $border$ 转移过来，即 $dp_{i,j} = \max\{dp_{i-(j-fail[j]),fail[j]+j+g[j-fail[j]]}\}$
  另一方面 $dp_{i,j}$ 可以由它的一个更长的 LCP 转移过来，即 $dp_{i,j} = \max_{k>j}\{dp_{i,k}\}$
  其中 $g[i]$ 为 $\sum_{j=2}^{i} LCP(S, suf_S[j])$

trie 树

trie 树

- 可以当成把一堆字符串 $S_1, S_2, S_3 \ldots S_m$ 插入到树上弄出来的数据结构。
- 树上每个结点都表示某一个或多个字符串的前缀。

AC 自动机

AC 自动机

- 用 P_x 表示 trie 树上 x 结点表示的字符串。
- AC 自动机就是把 trie 树拉下来，然后对于每个结点 x，求出 $fail[x]$ 和 $ch[x][c]$。
- $fail[x]$ 为 trie 树上最深 (即长度最大) 的且为 P_x 的严格后缀的结点。
- $ch[x][c]$ 为 trie 树上最深 (即长度最大) 的且为 $P_x + c$ 的严格后缀的结点。
- 构建的过程是个 bfs。
- 这个东西可以帮助我们进行多串匹配，即求出在某个串在 $\{S_1, S_2, \dots, S_m\}$ 的最大匹配位置。

AC 自动机

经典题

- 题意：给定 n 个串 $S_1 \ldots S_n$，q 次询问，给定 T_i，询问 T_i 是否有子串与 S_k 相等。长度 10^6 级别。
- 记录到根路径是否有危险节点即可。

AC 自动机

QOJ9111 Zayin and String

- 题意：给定一个带权字典，即 n 个字符串 s_i，每个字符串有一个价值 a_i。定义一个串 S 的价值为 $\frac{\sum_{l=1}^{|S|}\sum_{r=l}^{|S|} C(S[l:r])}{|S|}$，

$$C(S) = a_i, when\ S = S_i$$

$$C(S) = 0, otherwise$$

再给定一个字符串 T，求 T 的最大价值子序列，数据规模 2000。

AC 自动机

QOJ9111 Zayin and String

- 题意：给定一个带权字典，即 n 个字符串 s_i，每个字符串有一个价值 a_i。定义一个串 S 的价值为 $\frac{\sum_{l=1}^{|S|}\sum_{r=l}^{|S|} C(S[l:r])}{|S|}$，

$$C(S) = a_i, when\ S = S_i$$

$$C(S) = 0, otherwise$$

再给定一个字符串 T，求 T 的最大价值子序列，数据规模 2000。

- 首先分数规划，二分答案 t，每次加一个字符的贡献为当前串 T 的 $\sum_i C(suf_T[i]) - t$。
然后 dp 即可。

AC 自动机

QOJ2273 Suffixes may Contain Prefixes

- 题意：给定一个串 S 和正整数 n，构造一个长度为 n 的最优的字符串 T，使得 $\sum_i LCP(Suf_T[i], S)$ 最大。其中 $|S|, x \leq 2000$。

AC 自动机

QOJ2273 Suffixes may Contain Prefixes

- 题意：给定一个串 S 和正整数 n，构造一个长度为 n 的最优的字符串 T，使得 $\sum_i LCP(Suf_T[i], S)$ 最大。其中 $|S|, x \leq 2000$。
- 我们发现所有后缀的 LCP 的和相当于所有子串是否出现在 S 的前缀，也就相当于 T 所有前缀的后缀是否出现在 S 的前缀里。
  这样我们对 S 的所有前缀 (即 S 本身) 建 AC 自动机，然后考虑从前往后确定 T 中字符的过程。
  $dp_{i,j}$ 表示当前 T 长度为 i，与 S 的 LCP 为 j。
  转移为 $dp_{i,j} + g[ch[j][c]] \to dp_{i+1,ch[j][c]}$。
  g_i 为 $pre_S[i]$ 的所有后缀与 S 前缀的匹配次数。
  这里的 AC 自动机实际上可以看成是 S 做 KMP 产生的 border 树。

后缀数组

P3809 【模板】后缀排序

- 考虑倍增，每次对 2^i 位做基数排序，就可以 $O(n\log n)$ 求出后缀排序。
- 令 $sa[i]$ 为排名 i 的后缀的起始位置，$rank[i]$ 为起始位置为 i 的后缀的排名，$height[i] = LCP(sa[i], sa[i+1])$。
- 性质：假设 $rank[i] < rank[j]$，则 $LCP(i,j) = \min_{k=rank[i]}^{rank[j]-1} height[k]$。
- 于是后缀 LCP 转化为 RMQ。
- 另 $H[i] = height[rank[i]]$
- 性质：$H[i+1] \geq H[i] - 1$，因此可以 $O(n)$ 求出 H。

后缀数组

UOJ#131. 【NOI2015】品酒大会

- 题面：对于每个 $r \in [0, n)$，求有多少对 i, j 满足 $LCP_{i,j} \geq r$。

后缀数组

UOJ#131. 【NOI2015】品酒大会

- 题面：对于每个 $r \in [0, n)$，求有多少对 i, j 满足 $LCP_{i,j} \geq r$。
- 建后缀数组之后从大到小枚举 r 维护连续段。

后缀数组

UOJ#219. 【NOI2016】优秀的拆分

- 题面：给定长度为 n 的串 S，计算其子串有多少优秀的拆分。其中 S 为优秀的拆分等价于 S 能被表示为 $AABB$ 的形式，且 A, B 非空，不同拆分方法计算多次。$n \leq 30000$，多组数据。

后缀数组

UOJ#219. 【NOI2016】优秀的拆分

- 题面：给定长度为 n 的串 S，计算其子串有多少优秀的拆分。其中 S 为优秀的拆分等价于 S 能被表示为 $AABB$ 的形式，且 A, B 非空，不同拆分方法计算多次。$n \leq 30000$，多组数据。
- 枚举 $|A|$，隔 $|A|$ 个位置标记关键点，求出相邻关键点的 $LCS(suffix)$ 和 LCP，维护区间加。

后缀数组

缺位匹配

- 题面：给定 S, T，问 T 在 S 中的相似匹配位置（相似匹配指相差不超过 k 位)。其中 $n \leq 10^5, k \leq 20, |\Sigma|$ *is large*。

后缀数组

缺位匹配

- 题面：给定 S, T，问 T 在 S 中的相似匹配位置（相似匹配指相差不超过 k 位)。其中 $n \le 10^5, k \le 20, |\Sigma|$ *is large*。
- 模拟，后缀数组优化求 LCP。

回文串

一些性质

Definition 1. 回文串

正着看反着看一样的串叫回文串，即满足 $S[i] = S[|S| - i + 1]$ 的串。

Theorem 1

长度大于 2 的回文串去掉最开头和最末尾的字符依然是回文串。

Theorem 2

若 T 是回文串 S 的回文后缀，则 T 是 S 的一个 border。

Theorem 3

一个字符串的本质不同回文子串个数为 $O(n)$。

Manacher

Manacher

- 用来求一个串在每个位置的最长回文半径。
- 奇偶长度回文串不一样，添加特殊字符，全转化为奇回文；边界处理麻烦，添加特殊字符。
- 从左到右求出 p_i 表示以 i 为中心的最长回文半径，同时维护当前延伸最右的回文串的延伸位置 $right$ 及其回文中心 mid。
- 求 p_i 时，可以直接令 $p_i = min(p_{2mid-i,right-i})$，再暴力拓展。
- 时间复杂度 $O(n)$。

回文自动机

回文自动机

- 回文自动机每个结点代表一个回文串，包含两棵树分别表示奇回文和偶回文。
- 令 P_x 为 x 结点表示的字符串。
- $Fail[x]$ 表示 P_x 的最长严格回文后缀所在的结点。
- $ch[x][c]$ 表示 cP_xc 所在的结点（不存在就为 0）。
- $len[x]$ 表示 $|P_x|$。
- 增量法 build。

回文自动机

P5496 【模板】回文自动机（PAM）

- 题意：给定长为 n 的字符串 S，对于每个 $i \le n$ 求以 i 为结尾的回文串个数。

回文自动机

[CERC2014]Virus synthesis

- 初始有一个空串，利用下面的操作构造给定串 S 。
- 1. 串开头或末尾加一个字符
- 2. 串开头或末尾加一个该串的逆串
- 求最小操作数，其中 $|S| \leq 10^5, |\Sigma| = 4$

回文自动机

[CERC2014]Virus synthesis

- 建出 PAM，并求出 $trans[x]$ 表示 x 的长度不超过 $len[x]/2$ 的最长回文后缀。
- 然后就可以在 PAM 上 dp 了。

其他字符串匹配方法

CF754E. Dasha and cyclic table

- 题意：给定一个 $n * m$ 的字符串矩阵（上下连通，左右连通），再给定一个 $r * c$ 的矩阵。
- 你需要求出每个位置是否能匹配。
- 其中 $n, m, r, c \leq 400, \Sigma = \{a, b, ..., z, ?\}$。
- 6s,512MB

其他字符串匹配方法

bitset

- 考虑一维的情况：我们要匹配 S, T。
- 令 f_i 表示以 $S[i]$ 为左端点的长度为 $lenT$ 的串是否匹配 T。
- 对于 $i \in [1, lenS], j \in [1, lenT]$，我们把 $f_i \& = (S[i+j-1] = S[j])$。
- 转化为对于 $j \in [1, lenT], i \in [1, lenS]$，$f_{i-j+1} \& = (S[i] = S[j])$。
- 预处理 bitset$B[c][i] = (T[i] == c)$，相当于 $F \& = B[S[j]] >> (j-1)$
- 时间复杂度为 $O(\frac{n^2}{w})$。

其他字符串匹配方法

bitset

- 二维的情况是相同的，直接一行一行做即可。
- 时间复杂度为 $O(\frac{nmrc}{w})$
- 注意循环位移即可。

其他字符串匹配方法

FFT

- 还是考虑一维的情况，匹配 S, T。
- 令 f_i 表示 i 作为 T 匹配到的右端点是否可行。
- $f_i = \sum_{j=1}^{k}(a_{i-k+j} - b_j)^2$
- 化成卷积形式跑 $FFT(NTT)$。

其他字符串匹配方法

FFT

- 再考虑刚才那道题。
- 相当于把它扩展到二维。
- 那么我们就做二维离散型傅里叶变换。
- 时间复杂度 $O(nm(\log n + \log m))$。