Trie&Hash& 扩展 KMP&AC 自动机

V!oleT

2019 年 7 月 5 日

Outline

哈希

- 最常用来快速判断两个字符串是否相等
- 假设字符集是小写字母，那么可以将字符看作是 $0 \sim 25$ 的数字，字符串就变为一个 26 进制的数，判断两个字符串相等等价于判断这个 26 进制数是否相等。

哈希

- 数字太大无法存下，故我们将数字对一个数 p 取模，若取模后值相等我们就认为它们相等，模数 p 一般是一个大小适中的质数
- 有时我们需要用到任意子串的 $hash$ 值，而这个问题可以用类似前缀和的思路，递推得出每个前缀字符串的 $hash$ 值，再差分得出任意子串的 $hash$ 值

$$hash[i] \equiv hash[i-1] \times 26 + str[i] \pmod{p}$$

$$hash[l,r] \equiv hash[r] - hash[l-1] \times p^{r-l+1} \pmod{p}$$

- 已知两段字符串的长度与 hash 值，可以快速合并

哈希

- 若题目数据是随机的，可以直接用 unsigned 类型的自然溢出来代替取模步骤加快速度。
- 若题目数据人工构造，有时需要对多个质数取模，每个余数对应相等才认为原字符串真正相同。

哈希的一些应用

- Hash+ 二分求最长公共前缀
- 比较两个串的字典序
- 求后缀数组
- 判断循环节

一些题目

POJ 2758

给定一个字符串，要求维护两种操作

在字符串中插入一个字符

询问某两个位置开始的 LCP

插入操作次数 ≤ 200，字符串总长度 $\leq 5 \times 10^4$，查询操作次数 $\leq 2 \times 10^4$。

一些题目

Substring

给出两个字符串 A，B，现在可以修改 A 的一个字符，使得 A 的某个前缀 P 是 B 的子串，求这个前缀的最长长度。$1 \leq |A|, |B| \leq 10^5$。

一些题目

CodeForces 580E

给一段长度为 $N \leq 10^5$ 的数字串，有两种操作：一是将一段区间的数字都变成 c；二是问这个查询的区间对区间内所有 i 是否符合 $str(i) = str(i + d)$。保证任何时刻数字串中的数字不超过 9。

一些题目

BZOJ 2124

给一个 1 到 $N \leq 10000$ 的排列 A_i，询问是否存在 $1 \leq p_1 < p_2 < p_3 < p_4 < p_5 < ... < p_{len} \leq N(len \geq 3)$，使得 $A_{p_1}, A_{p_2}, A_{p_3}, ..., A_{p_{len}}$ 是一个等差序列。

Outline

KMP

- 最常用来求解模式 S 在目标串 T 中出现位置的问题
- 思想是在暴力基础上尽可能地复用匹配结果
- 令 $next[i]$ 表示对于 S 中长为 i 的前缀串，它最长的既是前缀又是后缀的前缀子串长度

KMP

- 考虑当 T 的当前位与 S 的第 $k+1$ 位匹配不上后，之后可以不从头开始匹配，而是从 $next[k]$ 开始进行匹配
- 求解 $next$ 的过程就是一个自己与自己匹配的过程，即 $S[j+1]$ 与 $S[i]$ 无法匹配则令 $j = next[j]$，满足条件后 $next[i] = j$

Outline

Trie

- Trie 也称字典树，它将一个字符串集合对应到一棵有根树上。
- 树上每条边代表一个字符，将从根到任意顶点的路径上边的字符连起来就是该顶点代表的字符串。
- Trie 中任意一个结点代表的字符串都是实际字符串集合中某些串的前缀

Trie

- Trie 还将前缀相等的字符串压缩到一起，节省了存储空间
- 结点处还可以存储额外信息，比如该结点的字符串是否是字符串集合中的某个串
- 假设字符集是小写字母，则可简单的将 Trie 认为是一个 26 叉树，那么它的插入与查询操作都十分方便
- 时间复杂度：建树 O(字符串总长)，查询：O(查询串总长)

Trie 的一些应用

- 查询字符串集合中是否含有某个字符串
- 将集合内字符串按照字典序排序
- 求集合内两字符串的 LCP

一些题目

POJ 2001

给定若干字符串，对于每个字符串求出一个最短前缀，使得这个前缀不是任何其他字符串的前缀。

一些题目

POJ 3764

给定一棵 N 个点的树，每条边上有一个权值 w_i，求一条路径使得路径上的边权异或和尽可能大。

Outline

AC 自动机

与 KMP 类似，AC 自动机也是用来处理字符串匹配问题，不同点在于 KMP 的模式串只有一个，而 AC 自动机处理多模式串，比如：给出 n 个单词，再给出一段包含 m 个字符的文章，问有多少个单词在文章中出现

AC 自动机

- 构建 AC 自动机并用于匹配需要三个步骤：将所有模式串建成一棵 Trie，对 Trie 上所有结点构造失败指针 fail(功能类似于 KMP 中的 next)，匹配则利用失败指针进行。
- fail 指针实际上与 KMP 中的 next 相似，故 AC 自动机可看做 Trie 上的 KMP。
- 一般称 Trie 的结点为 AC 自动机的状态，Trie 中的边称为转移，fail 指针对应的结点叫做当前结点的失配转移

构造 fail 指针

- 当我们在 Trie 树中匹配到某个深度为 i 的结点 u 时，它代表着某些串的前缀。
- 当 u 无法继续匹配下去后，我们希望和 KMP 里一样，找到另一些串，使得它们的前缀等于当前串的后缀且长度尽量长。
- 由于串的前缀可以用树中的某个结点 v 来表示，所以我们令 v 是 u 的 fail 指针，当 u 匹配失败后将匹配结点移动为 v。

构造 fail 指针

- 显然 v 的深度小于 u，所以我们考虑从根结点 BFS 构造 fail 指针，求 fail 的过程也是自我匹配的过程，与 KMP 构造方式类似。
- 假 u 的 fail 指针是 v，现在考虑构造 u 的字符 c 方向的儿子 w 的 fail 指针，若 v 不存在 c 方向的边，则令 $v = fail[v]$，直到存在后 v 就是 w 的 fail 指针
- 这里有一个优化就是如果 c 方向儿子 w 不存在，则可以直接令这个儿子是它的 fail 指针

构造 fail 指针

```
for (int i = 0; i < 26; ++i) {
  if (root->nxt[i]) {
    node *p = root->nxt[i];
    p->fail = root;
    que.push(p);
  } else {
    root->nxt[i] = root;
  }
}
while (!que.empty()) {
  node *p = que.front();
  que.pop();
  for (int i = 0; i < 26; ++i) {
    if (p->nxt[i] == NULL) {
      p->nxt[i] = p->fail->nxt[i];
    } else {
      p->nxt[i]->fail = p->fail->nxt[i];
      que.push(p->nxt[i]);
    }
  }
}
```

一些题目

文本生成器

给出 n 个单词，求有多少个长度为 m 的文章包含至少一个单词。$n \leq 60$, 单词总长 ≤ 100,$m \leq 100$。

一些题目

文本生成器 2

给出 n 个单词，求包含所有单词的文章最短是多长。$n \leq 10$, 单词总长 ≤ 1000。

fail 树

- 把所有 fail 指针逆向, 这样就得到了一棵树
- 在 fail 树中根到点 z 的路径上的点都是 z 的后缀

一些题目

NOI 阿狸的打字机

给给定一棵 $trie$ 树，询问根到 x 的字符串在根到 y 的字符串中出现了多少次。支持离线。trie 树节点数 $\leq 10^5$，询问数 $\leq 10^5$

Outline

Manacher

- 用来解决一类回文串问题，它可以求出以每个字符为中心的最长回文串的半径
- 首先在每个字符之间加入一个不在原串中出现的字符，比如井字符，这样统一了奇回文和偶回文串
- 令 p_i 表示以 i 为中心的最长回文半径，考虑按顺序递推求解

Manacher

- 首先考虑两个辅助变量 $rmax$ 和 c，分别表示已求解的回文中心所拓展到的最右边界和对应的中心位置
- 计算 p_i 时，令 i 关于 c 的对称点为 $j = 2 \times c - i$，则 p_i 的初始值可以为 $min(p_j, rmax - i)$
- 对于剩余的部分我们可以通过尝试不断令 $p_i + 1$ 来求
- 匹配成功会导致 $rmax$ 右移，而 $rmax$ 右移不超过 n 次所以总时间复杂度 $O(n)$

Manacher

```
for (int i = 0; i < len; ++i) {
  p[i] = i < rmax ? min(p[2 * c - i], rmax - i) : 1;
  while (i - p[i] >= 0 && i + p[i] < len && str[i - p[i]] == str[i + p[i]]) {
    ++p[i];
  }
  if (i + p[i] > rmax) {
    rmax = i + p[i];
    c = i;
  }
}
```

一些题目

回文串

给定字符串 $S, |S| \leq 10^6$，求它最长的满足回文部分是偶回文串的偶回文子串。

一些题目

回文串

根据 Manacher 的构造过程我们可以看出，本质不同的回文子串只有 $O(n)$ 个。

每次 p_i 被拓展时我们就找到了一个新的回文串，我们可以利用四分之一处的 p_j 暴力判断新的回文串是否满足条件并更新答案。

Outline

EXKMP

- 求主串每个后缀与模式串的 LCP
- 类比 KMP 算法，同样先求出模式串每个后缀与模式串的 LCP，然后再求主串每个后缀与模式串的 LCP
- 设 $p[i]$ 表示模式串中 $suf[i]$ 与模式串的 LCP
- 类比 manacher 的思路，设 $rmax$ 是最远匹配到的位置，设 $c + p[c] - 1 = mx$

EXKMP

- 若 $rmax \geq i$，$p[i] = min(p[i-c+1], rmax-i+1)$，否则 $p[i] = 0$
- 然后比较 $i+p[i]$ 与 $p[i]$ 之后的字符，如果相等则 $p[i]++$，并用 $i+p[i]$ 更新 rmax,i 更新 c

一些题目

POI2005 Template

给定一个字符串，求一个长度最小的前缀，使得重复写下这个前缀可以恰好变成原字符串。写的过程中前缀可以重叠，但重叠部分的字符必须相等。$|S| \leq 500000$