字符串算法

北京 提高组基础班

张若天 me@zrt.io

2018 年 2 月 9 日

清华大学 交叉信息研究院

大家好!

OI 中常用的字符串算法/数据结构

哈希 Hash

KMP

Manacher

Trie 树

AC 自动机 (Trie 图,*)

后缀数组 (SA,*)

后缀自动机 (SAM,*)

hash

字符串 hash 就是一个字符串到整数的映射。

主要用处是快速比较两个字符串是否相等。

比较整数 (O(1))，比较字符串 (O(L))。

字符串 hash 就是一个字符串到整数的映射。

主要用处是快速比较两个字符串是否相等。

比较整数 (O(1))，比较字符串 (O(L))。

hash 相等两字符串大概率相等。

hash 不等两字符串一定不等。

常用算法是 RK-hash。

RK-hash

RK-hash 就是把一个字符串看作一个 base 进制的大整数，然后对一个素数 p 取模。

RK-hash

RK-hash 就是把一个字符串看作一个 base 进制的大整数，然后对一个素数 p 取模。

hash[i]=(hash[i-1]*base+s[i])%p。

RK-hash

RK-hash 就是把一个字符串看作一个 base 进制的大整数，然后对一个素数 p 取模。

hash[i]=(hash[i-1]*base+s[i])%p。

base 可以取 31,131,13131 等，大于 $|\Sigma|$ 即可。

注意要把 a 看作 1 而不是 0。

p 一定要是 long long 范围的一个素数，比如 $2333333333333333, 10^{18}+9$。

RK-hash

RK-hash 就是把一个字符串看作一个 base 进制的大整数，然后对一个素数 p 取模。

hash[i]=(hash[i-1]*base+s[i])%p。

base 可以取 31,131,13131 等，大于 $|\Sigma|$ 即可。

注意要把 a 看作 1 而不是 0。

p 一定要是 long long 范围的一个素数，比如 2333333333333333,$10^{18}+9$。

unsigned long long 自然溢出可以看作是对 2^{64} 取模，但是可以被特殊构造卡掉。

不放心可以多找几个素数同时算。

hash[i]=(hash[i-1]*base+s[i])%p。

任意一段的 hash 值?

hash[i]=(hash[i-1]*base+s[i])%p。

任意一段的 hash 值?

两个字符串的 hash 值合并?

hash[i]=(hash[i-1]*base+s[i])%p。

任意一段的 hash 值?

两个字符串的 hash 值合并?

为何要用素数?

hash[i]=(hash[i-1]*base+s[i])%p。

任意一段的 hash 值?

两个字符串的 hash 值合并?

为何要用素数?

求两字符串的最长公共前缀 (LCP)?

hash[i]=(hash[i-1]*base+s[i])%p。

任意一段的 hash 值?

两个字符串的 hash 值合并?

为何要用素数?

求两字符串的最长公共前缀 (LCP)?

判断两字符串的大小 (字典序)?

hash[i]=(hash[i-1]*base+s[i])%p。

任意一段的 hash 值?

两个字符串的 hash 值合并?

为何要用素数?

求两字符串的最长公共前缀 (LCP)?

判断两字符串的大小 (字典序)?

哈希表?

XOR 版本的 RK-hash

hash[i]=(hash[i-1]*base xor s[i])%p。

无法利用前缀和。

可以避免被卡。

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最长公共前缀

插入操作 $<=200$, 字符串长度 $<=5w$, 查询操作 $<=2w$

BZOJ 3555

N 个长度为 L 的字符串，问有多少对只有一位不同。

KMP

KMP 是一个常用的单模式串匹配算法。

KMP 是一个常用的单模式串匹配算法。

朴素算法?

KMP 是一个常用的单模式串匹配算法。

朴素算法?

next 数组?

KMP 是一个常用的单模式串匹配算法。

朴素算法?

next 数组?最长的后缀等于前缀的位置。

KMP 是一个常用的单模式串匹配算法。

朴素算法?

next 数组?最长的后缀等于前缀的位置。

next 数组求法?

KMP 是一个常用的单模式串匹配算法。

朴素算法?

next 数组?最长的后缀等于前缀的位置。

next 数组求法?自己与稍短一些的自己匹配。

KMP 是一个常用的单模式串匹配算法。

朴素算法?

next 数组?最长的后缀等于前缀的位置。

next 数组求法?自己与稍短一些的自己匹配。

代码?

```
int j=0;
for(int i=2;i<=m;i++){
    while(j&&s[i]!=s[j+1]) j=nxt[j];
    if(s[i]==s[j+1]) j++;
    nxt[i]=j;
}

j=0;
for(int i=2;i<=L;i++){
    while(j&&s[j+1]!=s[i]) j=nxt[j];
    if(s[j+1]==s[i]) j++;
    if(j==m){
        //
        j=nxt[j];
    }
}
```

应用

单模式串匹配。

求串的循环节。

POJ 2752

求所有 x 使得串的长度为 x 的前缀和 x 的后缀相等。

BZOJ 3670

求 num 数组一一对于字符串 S 的前 i 个字符构成的子串，既是它的后缀同时又是它的前缀，并且该后缀与该前缀不重叠，将这种字符串的数量记作 num[i]。

Manacher/exKMP

Manacher 及 exKMP

Manacher 是一个处理回文串的算法。

Manacher 及 exKMP

Manacher 是一个处理回文串的算法。

hash?O(nlogn)

Manacher 及 exKMP

Manacher 是一个处理回文串的算法。

hash?O(nlogn)

能 O(n) 求出每个位置的回文半径。

通过加字符，只考虑奇数长度回文串。

Manacher 及 exKMP

Manacher 是一个处理回文串的算法。

hash?O(nlogn)

能 O(n) 求出每个位置的回文半径。

通过加字符，只考虑奇数长度回文串。

mx,id?

Manacher 及 exKMP

Manacher 是一个处理回文串的算法。

hash?O(nlogn)

能 O(n) 求出每个位置的回文半径。

通过加字符，只考虑奇数长度回文串。

mx,id? mx 表示当前扩展过的最远位置，id 表示达到最原位置的时候的回文串中心。

Manacher 及 exKMP

Manacher 是一个处理回文串的算法。

hash?O(nlogn)

能 O(n) 求出每个位置的回文半径。

通过加字符，只考虑奇数长度回文串。

mx,id? mx 表示当前扩展过的最远位置，id 表示达到最原位置的时候的回文串中心。

一个串的本质不同的回文串个数不超过 O(n) 个。

Manacher 及 exKMP

Manacher 是一个处理回文串的算法。

hash?O(nlogn)

能 O(n) 求出每个位置的回文半径。

通过加字符，只考虑奇数长度回文串。

mx,id? mx 表示当前扩展过的最远位置，id 表示达到最原位置的时候的回文串中心。

一个串的本质不同的回文串个数不超过 O(n) 个。

BZOJ 2342 双倍回文

http://www.lydsy.com/JudgeOnline/problem.php?id=2342

BZOJ 3676 回文串

http://www.lydsy.com/JudgeOnline/problem.php?id=3676

课后练习。

TYVJ 1068

扩展 KMP，思想和 manacher 一致。

Trie

字典树 (Trie 树)。

每个节点代表一个字符串。

每个节点有 $|\Sigma|$ 条出边。

相同前缀共享相同节点。

字典树 (Trie 树)。

每个节点代表一个字符串。

每个节点有 $|\Sigma|$ 条出边。

相同前缀共享相同节点。

插入操作?

查询操作?

Trie 树 DFS 序为这些串的字典序。

LCP–> LCA 深度。

因为相同前缀共享相同节点。

题目大多是应用相同前缀相同节点这个特点出的。

POJ 2001

POJ 3630

BZOJ 2251

POJ 3764(XOR 最大)

Questions?

谢谢大家！

Email: me@zrt.io

QQ: 401794301

https://zrt.io

LaTeX