字符串匹配

主讲：姜添翼Ausert
复核：李梁裕、应懿

2022-2023学年 下半学期 重要事件

	日期	时间	备注
ICPC Shanghai	2023/3	/	/
Google Code Jam	2023/3/25	1:00 ~ 3:30	Round 1A
	2023/4/2	16:00 ~ 18:30	Round 1B
	2023/4/22	9:00 ~ 11:30	Round 1C

内容

A. 字符串，树
B. Trie树
C. KMP算法

字符串

字符串是多个字符组成的序列。

C++中可以直接用string 类型，比如：

“114514”

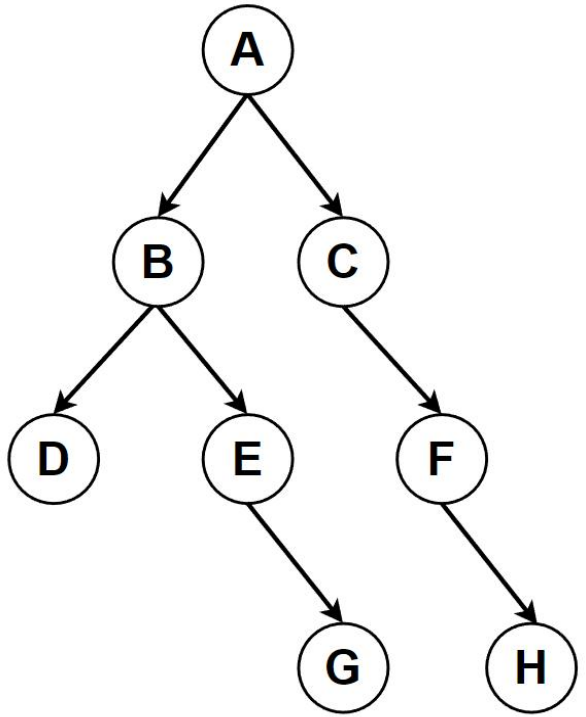
“codeforces”

“”（*空串*）

树

树是一种数据结构，它是由 n 个有限节点组成一个具有层次关系的集合。

右图是树的示意图



Trie树

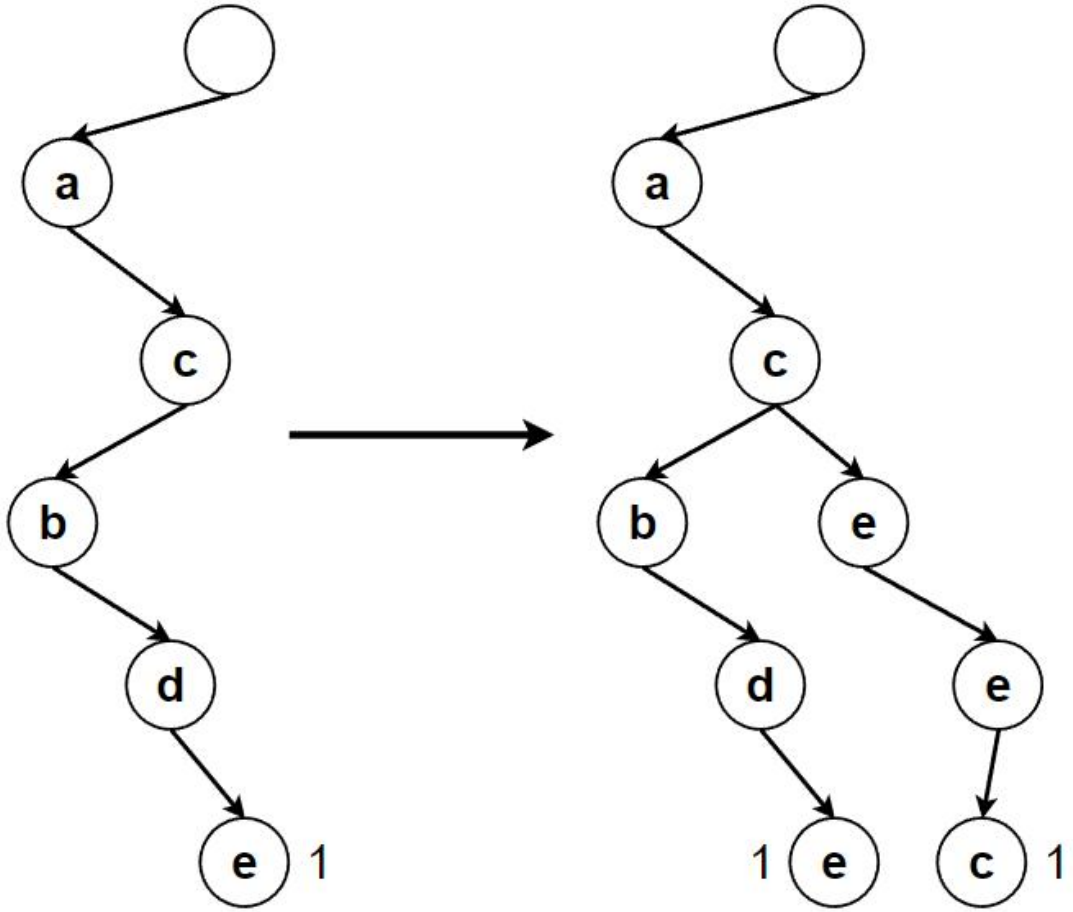
Trie树是一种能够高效存储和查询字符串的数据结构。
也叫字典树。

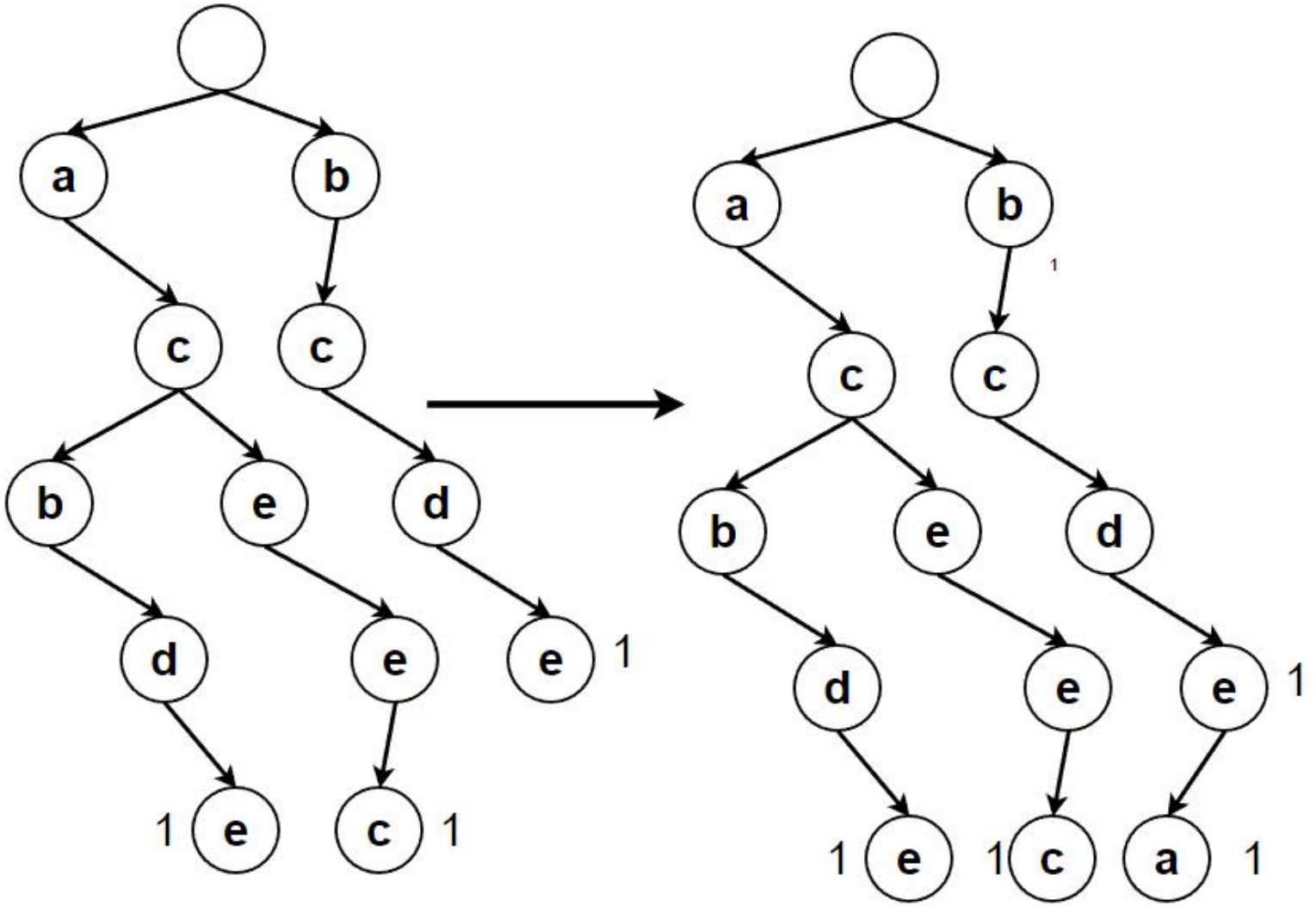
Trie树的例子

依次往Trie树中插入以下字符：

1. “acbde”
2. “aceec”
3. “bcde”
4. “bcdea”

右图每一层按照按照字典序排序。




a
b
c
c
b
e
d
d
e
e
1
1
e
c
1
a
b
1
c
c
b
e
d
d
e
e
1
1
e
1
c
a
1

例题

维护一个字符串集合，支持两种操作：

I x 向集合中插入一个字符串 x；

Q x 询问一个字符串在集合中出现了多少次。

共有 N 个操作，所有输入的字符串总长度不超过 10^5，字符串仅包含小写英文字母。

输入格式

第一行包含整数 N，表示操作数。接下来 N行，每行包含一个操作指令，指令为 I x 或 Q x 中的一种。

输出格式

对于每个询问指令 Q x，都要输出一个整数作为结果，表示 x 在集合中出现的次数。

每个结果占一行。

数据范围

$1 \leq N \leq 2 * 10^4$

例题

输入样例：

5

I abc

Q abc

Q ab

I ab

Q ab

输出样例：

1

0

1

KMP算法

1.朴素字符串匹配算法
2.KMP算法

朴素字符串匹配

原字符串

“codoefodoces”

模式字符串

“odoc”

朴素字符串匹配的问题

重复匹配导致时间的浪费。

设主串的长度为n，模式串的长度为m。
朴素算法的时间复杂度为$O(n*m)$

当匹配到这种情况的时候

codoefodoces

  odoc

e和c不匹配，指向模式串的指针应该完全回退至起点位置吗？

odo是已经匹配好的部分，如果我们不是从o开始匹配，而是从d开始匹配是不是也是可以的？

主串的最后一个字符o和模式串的第一个字符o已经匹配了，KMP利用了这个信息，构建next数组，减少重复匹配。

什么是next数组

next数组提供了当匹配失败的时候，指向模板串的下标应该回退至哪里。从而避免了朴素算法中的重复匹配的情况。

例如：

主串：　aacdccbb

模板串：　cdcb

next数组的使用

我们用下标i指向主串中待匹配的位置，j指向模式串中已经匹配好的最后一个位置，也就是说，我们每次进行的匹配是判断主串中i指向的字符和模式串中j+1指向的字符是否相同。

next数组的下标是对于模式串而言的。比如，当模式串匹配到j+1的时候不匹配了，j就跳转到next[j]的地方，然后重复匹配的过程。

事实上，我们发现，KMP匹配中，主串的下标i是不会回退的。

求next数组

实际上，next[i]的值就是主串1～i这部分子串的后缀和模式串1～i这部分子串的前缀相同的部分，同时使得这部分尽可能的长。

注意到，因为j及以前的都是已经匹配好的，所以next[i]的值也就是模式串的后缀和模式串的前缀相同的部分，同时尽可能的长。

说简洁点，就是模式串的最长非平凡相同前后缀。

（也就是说排除掉前后缀就是子串自己的情况）

对于模式串每一个1～i的子串如何求最长相同前后缀？

方法与KMP类似。

首先，容易得到next[1]=0的，因为就一个字符是不存在相同非平凡前后缀的。

然后，我们遍历一边模式串，如果模式串的i与模式串的j+1不匹配，我们不断使用next[j]回退j，直到不能回退或者成功匹配。循环退出后，如果是成功匹配的，j需要自加一下，然后将j的值存入next数组中去。

求next数组例子

主串：

codeforcesaabaab

模式串：

aabaab

next数组结果

下标i	0	1	2	3	4	5	6
数组值	0	0	1	0	1	2	3
实际字符	/	a	a	b	a	a	b

注意：此处主串和模式串的下标都是从1开始的。

模式串：
aabaab

KMP算法总结

KMP分为两步：

1. 求next数组（核心）；
2. 在匹配过程中，使用next数组略过不必要的尝试。

KMP的时间复杂度是$O(n+m)$

例题

给定一个字符串S，以及一个模式串P，所有字符串中只包含大小写英文字母以及阿拉伯数字。

模式串P在字符串S中多次作为子串出现。求出模式串P在字符串S中所有出现的位置的起始下标。

例题

输入格式

第一行输入整数 N，表示字符串 P 的长度。

第二行输入字符串 P。

第三行输入整数 M，表示字符串 S的长度。

第四行输入字符串 S。

输出格式

共一行，输出所有出现位置的起始下标（下标从0开始计数），整数之间用空格隔开。

数据范围

$1 \leq N \leq 105$
$1 \leq M \leq 106$

例题

输入样例：

3

aba

5

ababa

输出样例：

0 2

代码

例题的代码可以去我主页上找，搜索标题Trie树和KMP就可以了。

URL: https://ausertdream.github.io/

谢谢