AC 自动机

by 李翔

AC 自动机

- 不是 accept 自动机。

- 是 Aho-Corasick 造出来的。所以你懂的

用途

- 字符串的匹配问题
- 多串的匹配问题

- 例如给几个单词 acbs ， asf,dsef,
- 再给出一个 很长的文章， acbsdfgeasf
- 问在这个文章中，总共出现了多少个单词，或者是单词出现的总次数。

实现的原理

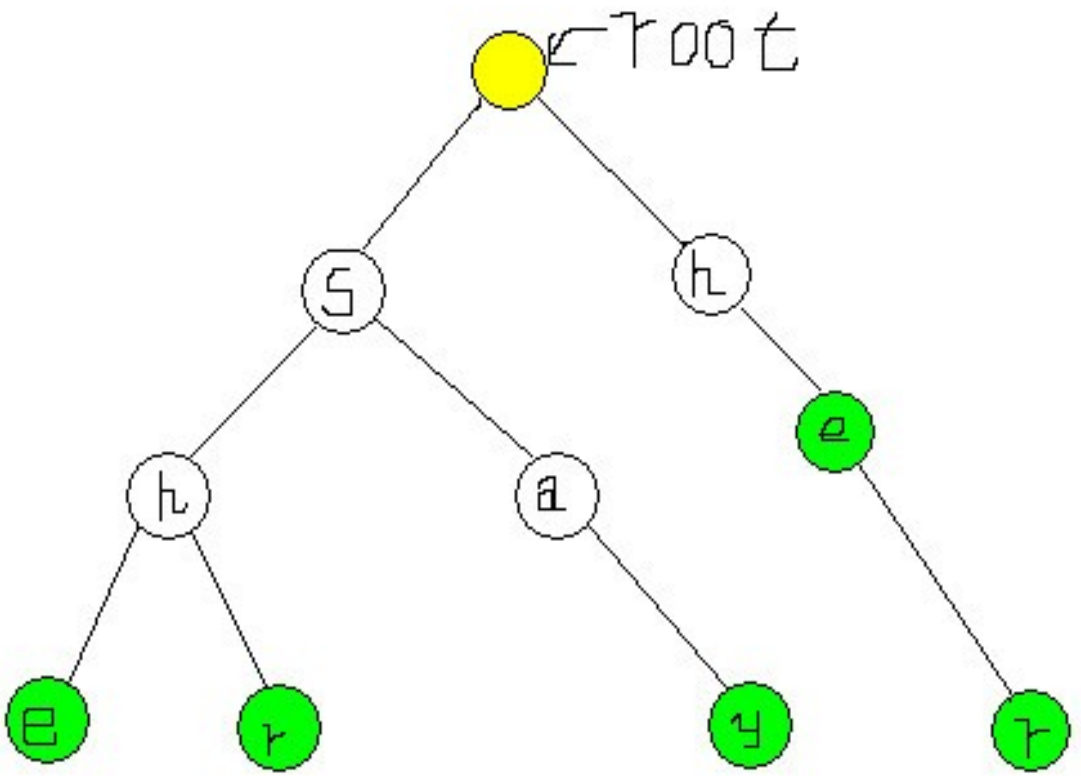
- 形象的说： KMP+trie 树（字典树）

  KMP ，之前学过的。为什么用到了这个呢?

  因为也是一种没有多余回溯的算法。

  什么是 trie 树（字典树）？

Trie 树



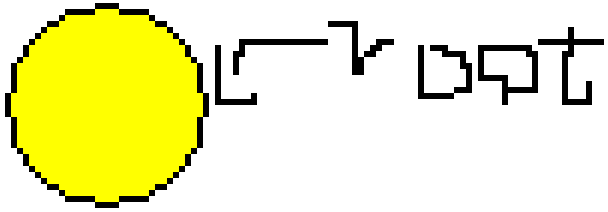
有神马特点

- 有一个空的根节点。

- 对于一般的 trie 数是解决用英文字母组成文章。所以每个节点会有 26 个子节点（指针）。

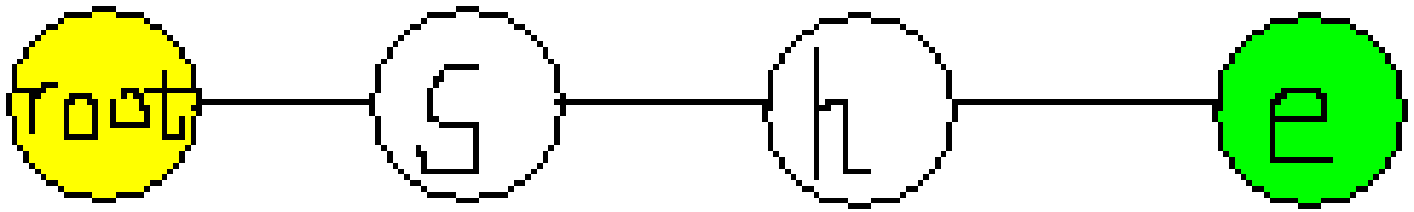
构造的过程

- 开始的时候有一个 root



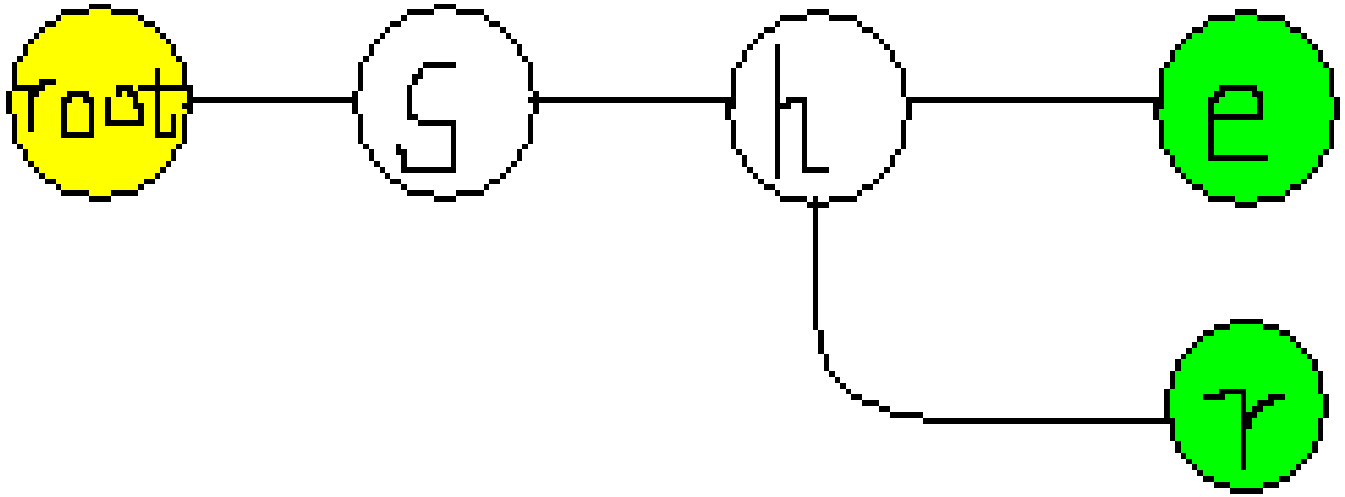
过程

- 要在该树中插入一个单词 she



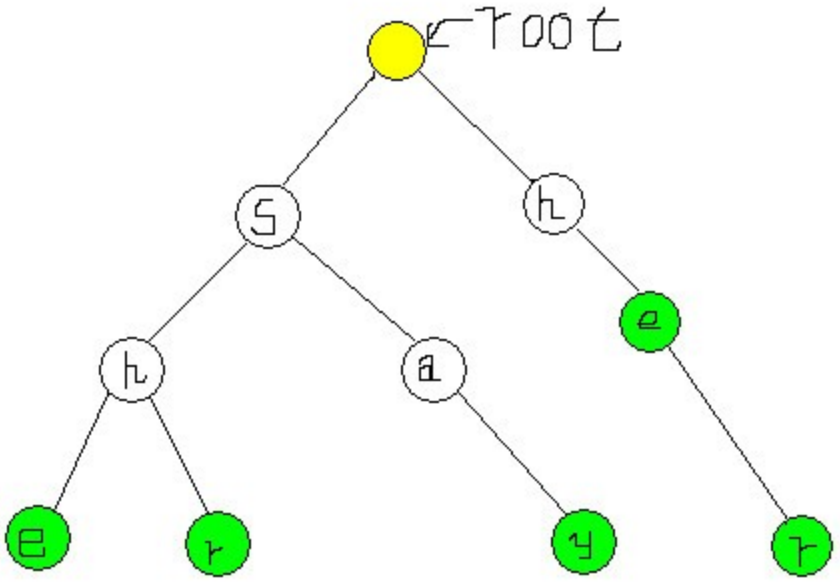
过程

- 再加一个单词 shr 。



过程

- 再插入 say 和 her 等，这样一个 trie 树就搞定了



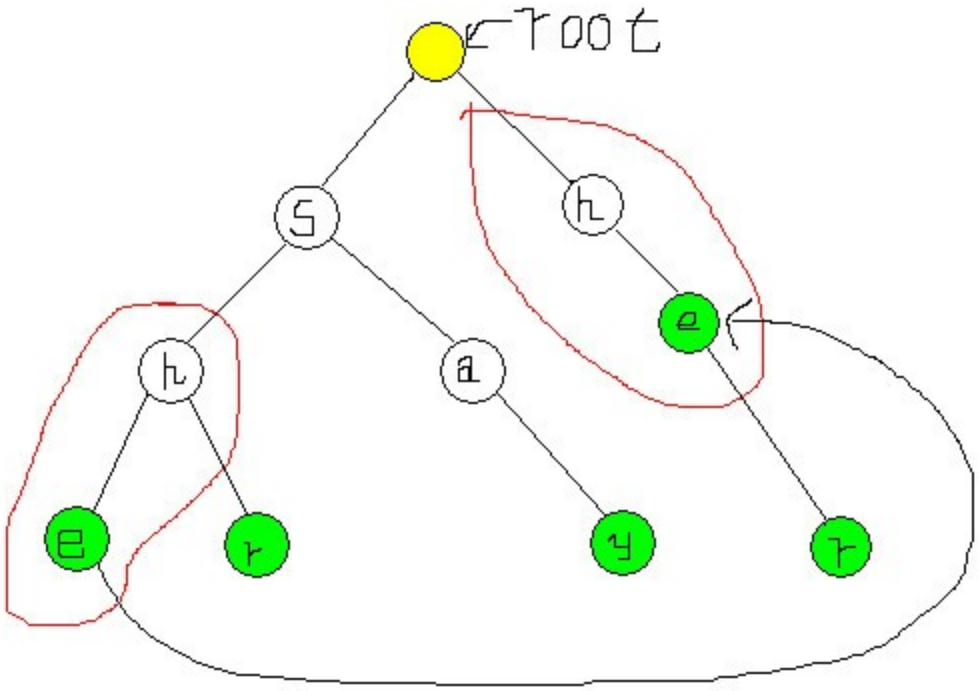
如何与 kmp 联系在一起?

- 关键是在 trie 树上 加了一种 fail 指针。

- Fail 指针的用途：就像是 kmp 中的 next 的数组。
- 在字符串失配的时候确定转移的节点。

先看到底是什么样的

- 这只显示了 e 的失配指针。
- 例如匹配文章： sher



构造 fail 指针的原理

- 根据父亲节点的 fail 指针来构造子节点的 fail 指针。

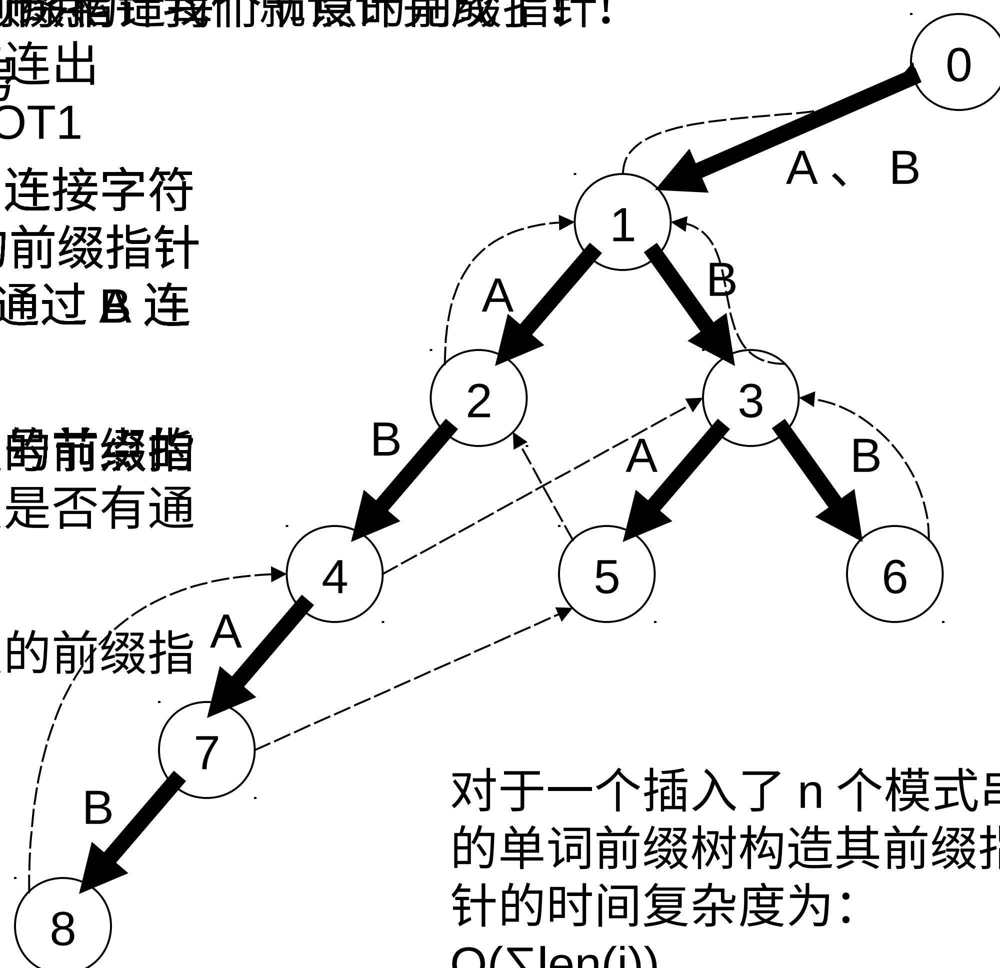
- 动态的过程：下面是拷贝人家的 ppt！ 编号为 0 的节点可以忽略之。

如何高效的构造前缀指针

父亲是 4 号节点，连接字符为 B，查找父亲的前缀指针 3 号节点，是否有通过 B 连接的儿子。

没有，于是继续查找 3 号节点的前缀指针指向的 1 号节点是否有通过 B 连接的儿子。

有！于是 8 号节点的前缀指针指向 4 号节点



对于一个插入了 n 个模式串的单词前缀树构造其前缀指针的时间复杂度为：O(∑len(i))

如何解决开始给的问题

在所有的绿色节点上做上标记。一旦访问到这个节点，就记录下。这样就能解决上面的问题。

hdu2222

5 // 单词数
she // 单词
he
say
shr
her
Yasherhs// 文章

问有多少单词在文章中出现

代码实现

```
struct node
{
    int next[26];
    int fail;
    int count;
    void init()
    {
        memset(next, -1, sizeof(next));
        fail = 0;
        count = 0;
    }
}s[500005];
```

在树中插入单词

```
void ins()
{
    int len = strlen(str);
    int i, j, ind;
    for(i = ind = 0; i < len; i++)
    {
        j = str[i] - 'a';
        if(s[ind].next[j] == -1)
        {
            s[sind].init();
            s[ind].next[j] = sind++;
        }
        ind = s[ind].next[j];
    }
    s[ind].count++;
}
```

在这里的操作对于不同的题目一般有 3 种不同的操作。

1： s[ind].count++;

这个是在解决出现总次数的时候是这样处理的。

2： s[ind],count=1;

这个是在 ac 自动机上进行 dp 的时候经常用的。

3. 新加一个标记 id 。

这个是在处理有哪些单词出现过。

```
void make_fail()
{
    qin = qout = 0;
    int i, ind, ind_f;
    for(i = 0; i < 26; i++)  {
        if(s[0].next[i] != -1) {
            q[qin++] = s[0].next[i];
        }
    }
    while(qin != qout)   {
        ind = q[qout++];
        for(i = 0; i < 26; i++)   {   // 找之后的子节点
            if(s[ind].next[i] != -1)   {
                q[qin++] = s[ind].next[i];
                ind_f = s[ind].fail;
                while(ind_f > 0 && s[ind_f].next[i] == -1)
                    ind_f = s[ind_f].fail;
                if(s[ind_f].next[i] != -1)
                    ind_f = s[ind_f].next[i];
                s[s[ind].next[i]].fail = ind_f;// 子节点的 fail 根据父节点 fail 指针的搞定
            }
        }
    }
}
```

```
int fd()  {
    int ct = 0;
    int di, i, ind, p;
    int len = strlen(des);// 这个是文章
    for(di = ind = 0; di < len; di++)   {
        i = des[di] - 'a';
        while(ind > 0 && s[ind].next[i] == -1)
            ind = s[ind].fail;

        if(s[ind].next[i] != -1) {// 等于 -1 的时候就已经是找打了根节点。
            ind = s[ind].next[i];
            p = ind;
            while(p > 0 && s[p].count != -1) {// 这里是精髓。在找过某个有标记的节点的时候
                ct += s[p].count;  // 答案          // 会把该位的标记标记为 -1 ，在下次经过有 -1
                s[p].count = -1;                    // 标记的时候，说明之后的都被计算过，不用
                p = s[p].fail;                      // 再重复计算了。
            }
        }
    }
    return ct;
}
```

拓展

在自动机上进行 dp

要大家自己去理解

题目

Poj
1204
 4052 （题目在 4044 上下载）
Hdu
2222
3065