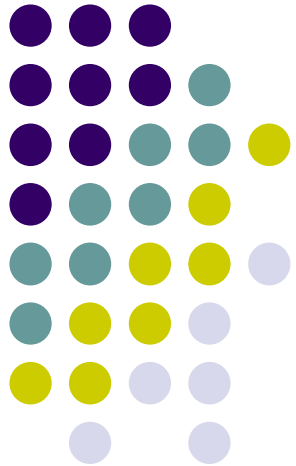
TRIE和AC自动机

吉林大学计算机学院
谷方明
fmgu2002@sina.com

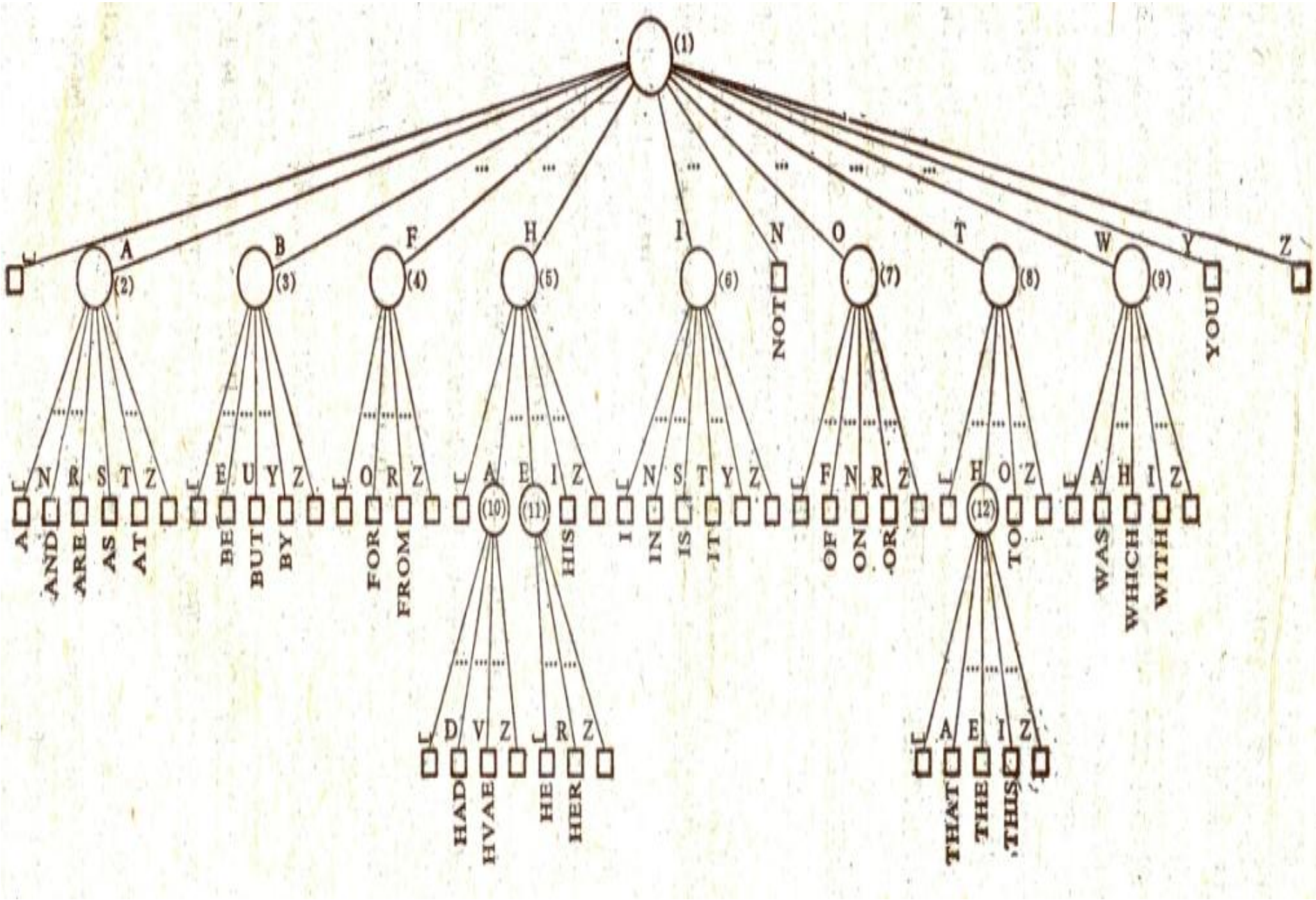


字符串查找

- 例：有 31 个最常用的英文单词的集合如下

A , AND , ARE , AS , AT , BE , BUT , BY , FOR , FROM , HAD , HAVE , HE , HER , HIS , I , IN , IS , IT , NOT , OF , ON , OR , THAT , THE , THIS , TO , WAS , WHICH , WITH , YOU

给定一个单词，查询是否在最常用单词集合中

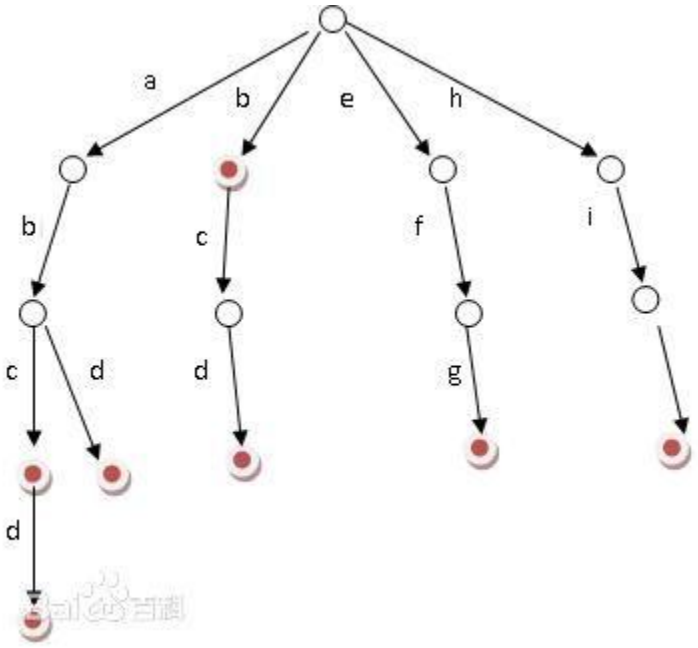

(1)
A
(2)
B
(3)
F
(4)
H
(5)
I
(6)
NOT
O
(7)
T
(8)
W
(9)
YOU
Z
N R S T Z
E U Y Z
O R Z
A E I Z
N S T Y Z
F N R Z
H O Z
A H I Z
A
AND
ARE
AS
AT
BE
BUT
BY
FOR
FROM
(10)
(11)
HIS
I
IN
IS
IT
OF
ON
OR
(12)
TO
WAS
WHICH
WITH
D V Z
R Z
HAD
HVAE
HE
HER
A E I Z
THAT
THE
THIS

M叉检索树

字典树

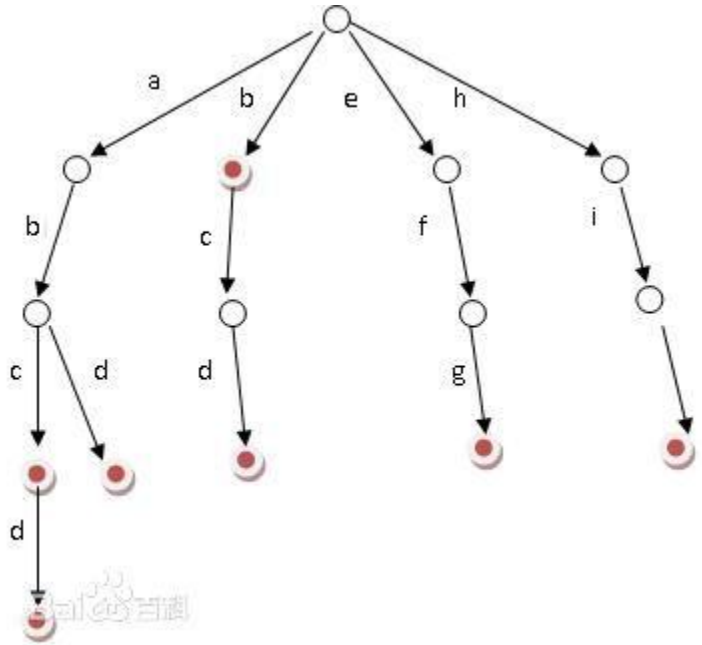
- 单词查找树、 TRIE树等
- 字典树建立在字符集Σ上，

  $|\Sigma| = M$
- 字典树是一种M叉树
  - ✓ 根节点不包含字符，根节点外的每一个节点都包含一个字符
  - ✓ 每个节点的所有子节点包含的字符都不相同。
  - ✓ 根节点到某一节点的路径上的字符连起来，为该节点对应的字符串；

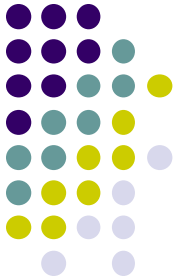


存储结构

- 每个结点存储**M**个儿子即可
- 顺序存储
  - ✓ $O(M^{MAXL})$， 浪费空间
- 链接存储
  - ✓ O(N*MAXL*M)
  - ✓ 共用前缀，一般达不到。
  - ✓ 可用静态链表实现。



静态链表（结构体数组）



```
struct Node{
    int  son [ 26 ];
    int flag; //单词结束标志
};

Node trie[ MAXN ];
int  sp;
```

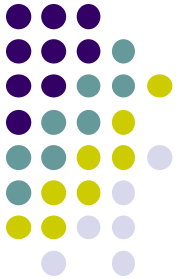
插入操作

```
//设根节点为0
void insertWord(char s[]) {
    int i,p=0,len=strlen(s),c;
    for(i=0; i<len; i++) {
        c=s[i]-'a';
        if(!trie[p].son[c]) trie[p].son[c] = ++sp;
        p = trie[p].son[c];
    }
   trie[p].flag = 1;
}
```

查询操作

```
//设根节点为0
int query(char s[]){
        int i,p=0,len=strlen(s),c;
        for(i=0;i<len;i++){
                c=s[i]-'a';
                if(!trie[p].son[c]) return 0;
                p = trie[p].son[c];
        }
        return trie[p].flag == 1;
}
```

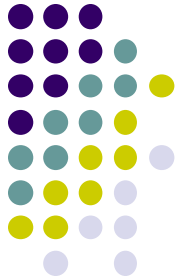
分析



- 时间复杂度
  - ✓ O(N*MAXL)
- 空间复杂度
  - ✓ O(MAXN*MAXL)

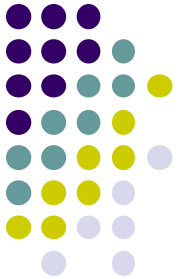
- 速度比**Hash**快，但浪费空间

Trie树的应用



- 串的快速检索
- 最长公共前缀
- “串”排序
- **辅助结构**

多模匹配



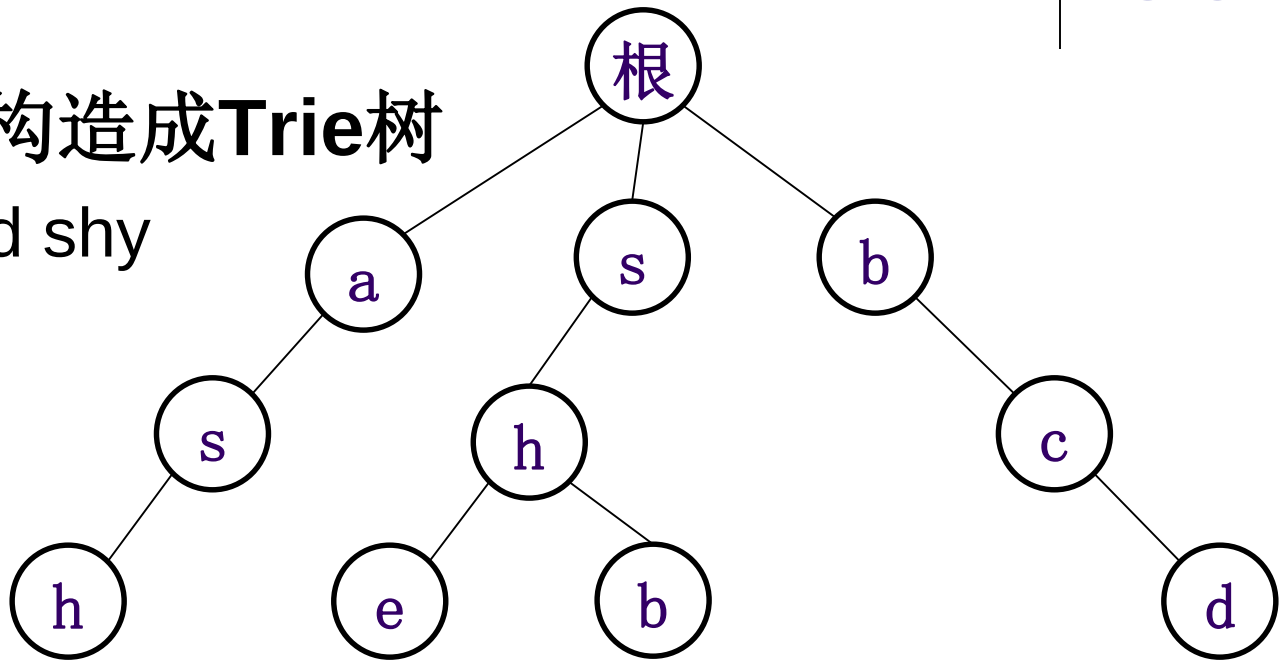
- 文本串：**ashecd**
- 模式串：**ash she bcd shb**

- 询问：模式串在文本串中出现的次数

- **K遍KMP算法？**
  - ✓ 效率低

AC自动机

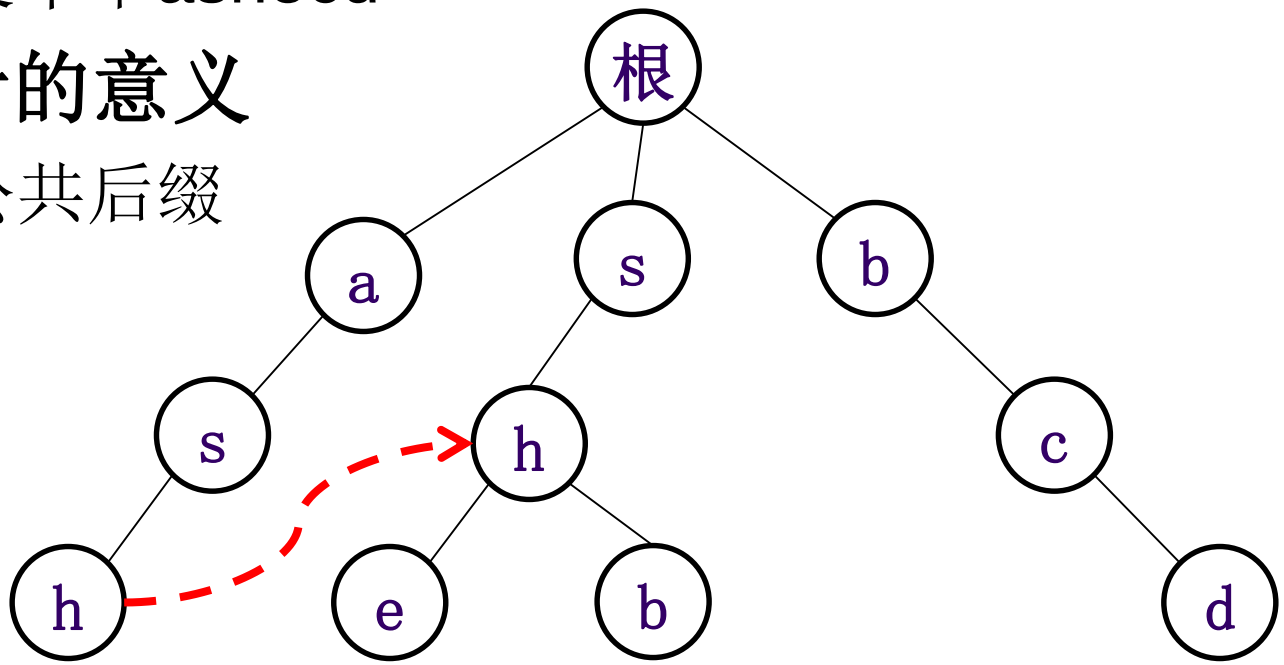
- 多个模式串构造成Trie树
  - ash she bcd shy



- 文本串在Trie树上进行匹配
  - ashecd

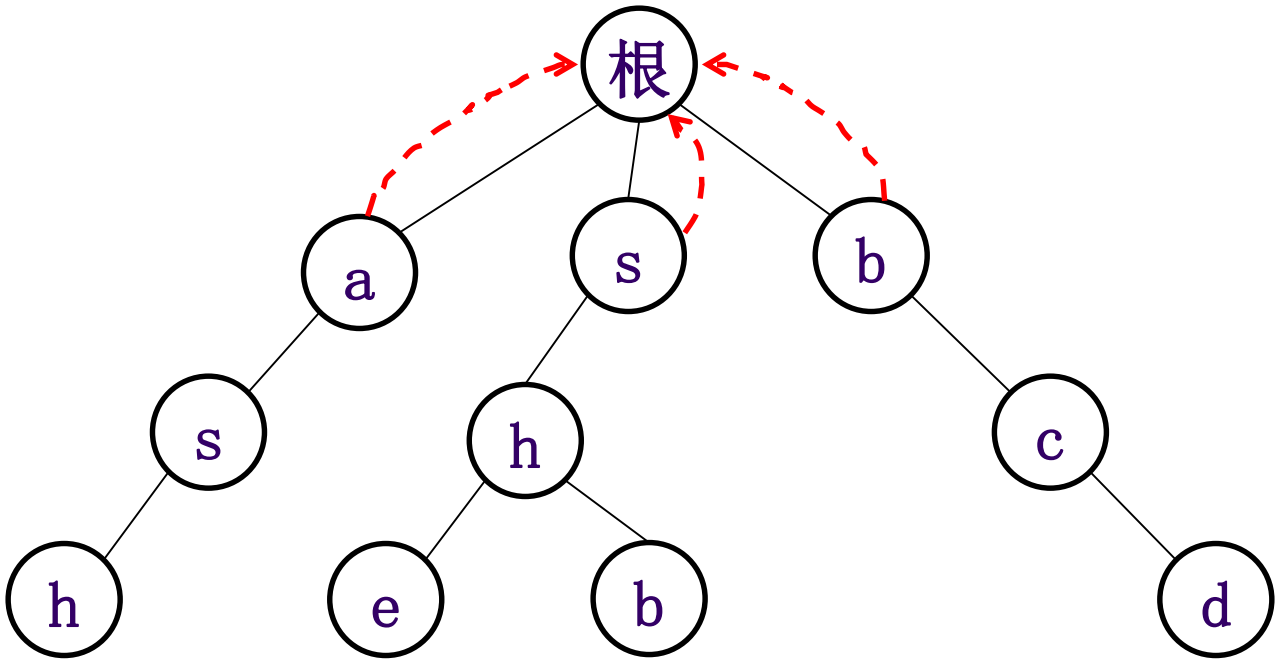
fail指针

- 类似KMP，引入fail指针
  - 例：文本串ashecd
- fail指针的意义
  - 最长公共后缀



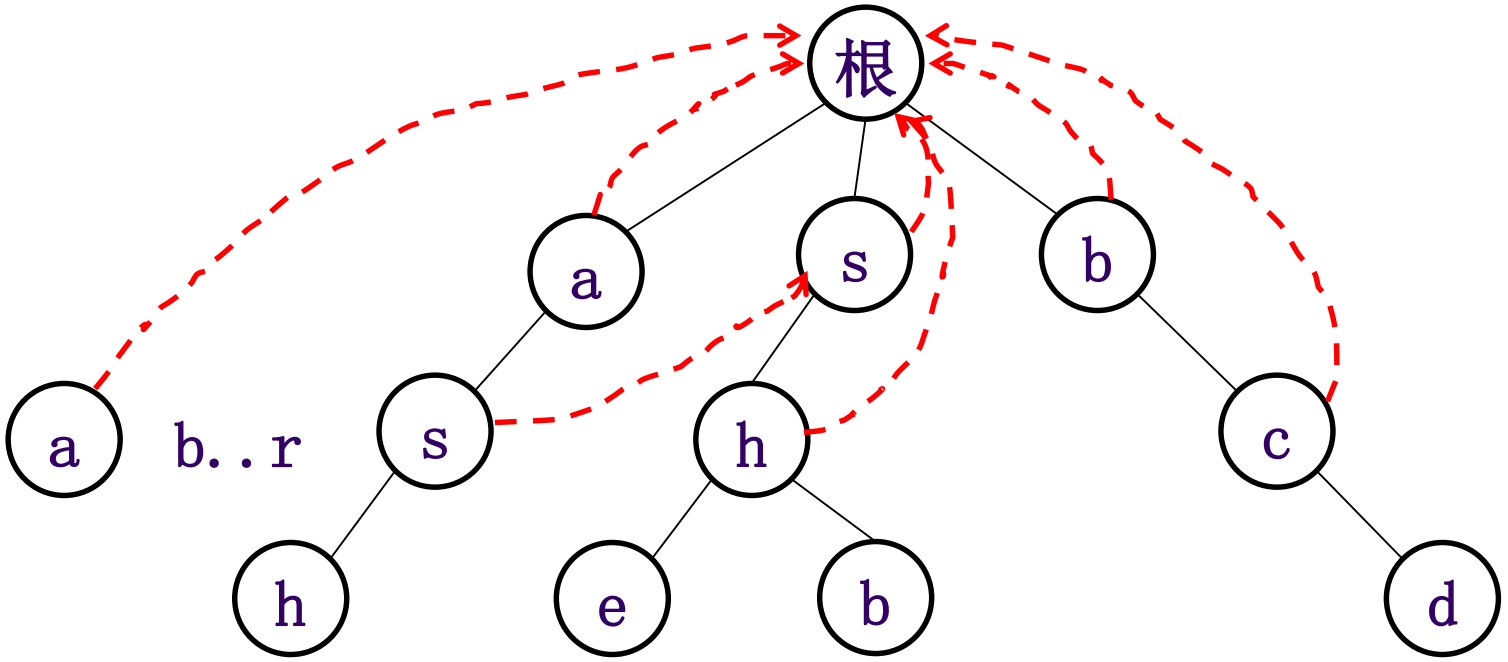
fail指针的建立

- 第一层



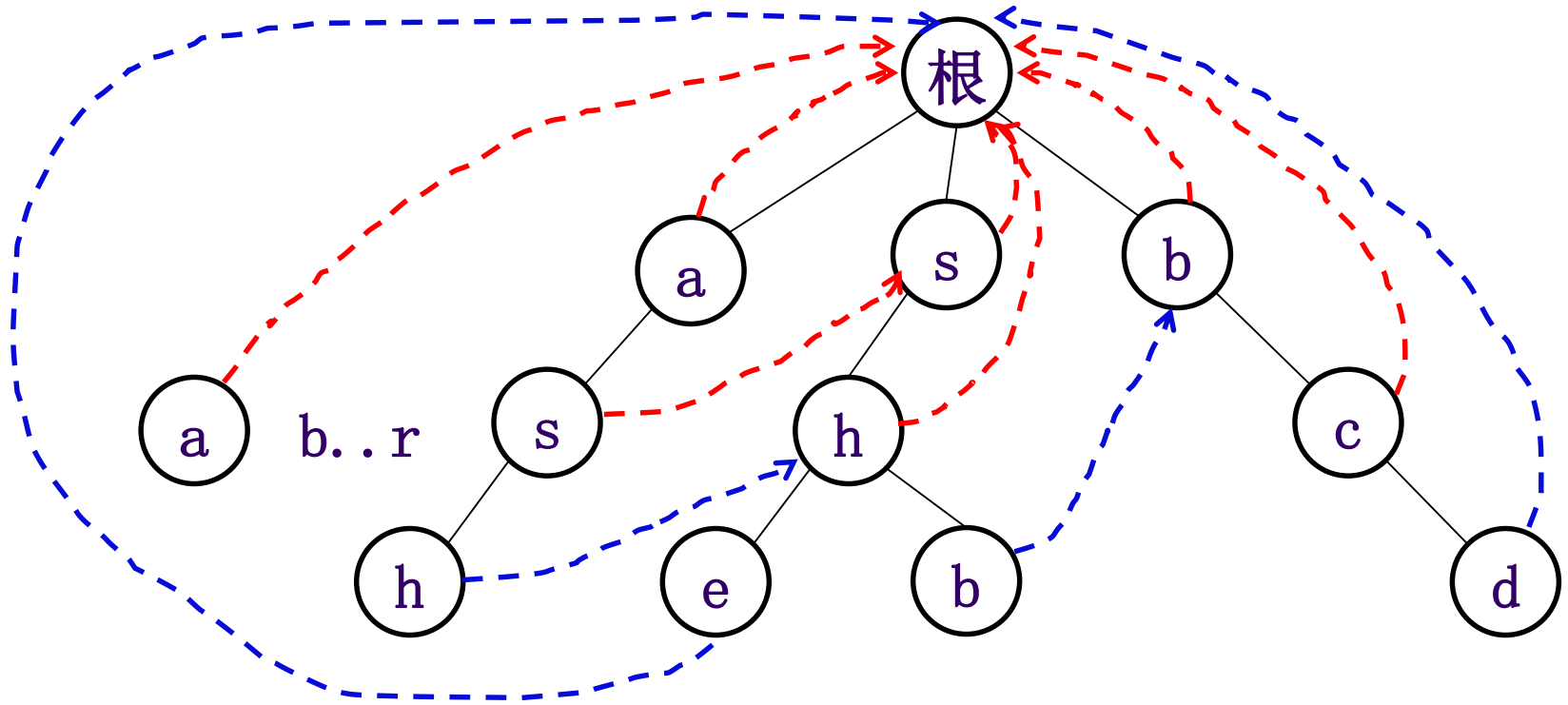
fail指针的建立

- 第二层
- **BFS: a 和 fail(a) 具有相同的儿子s**



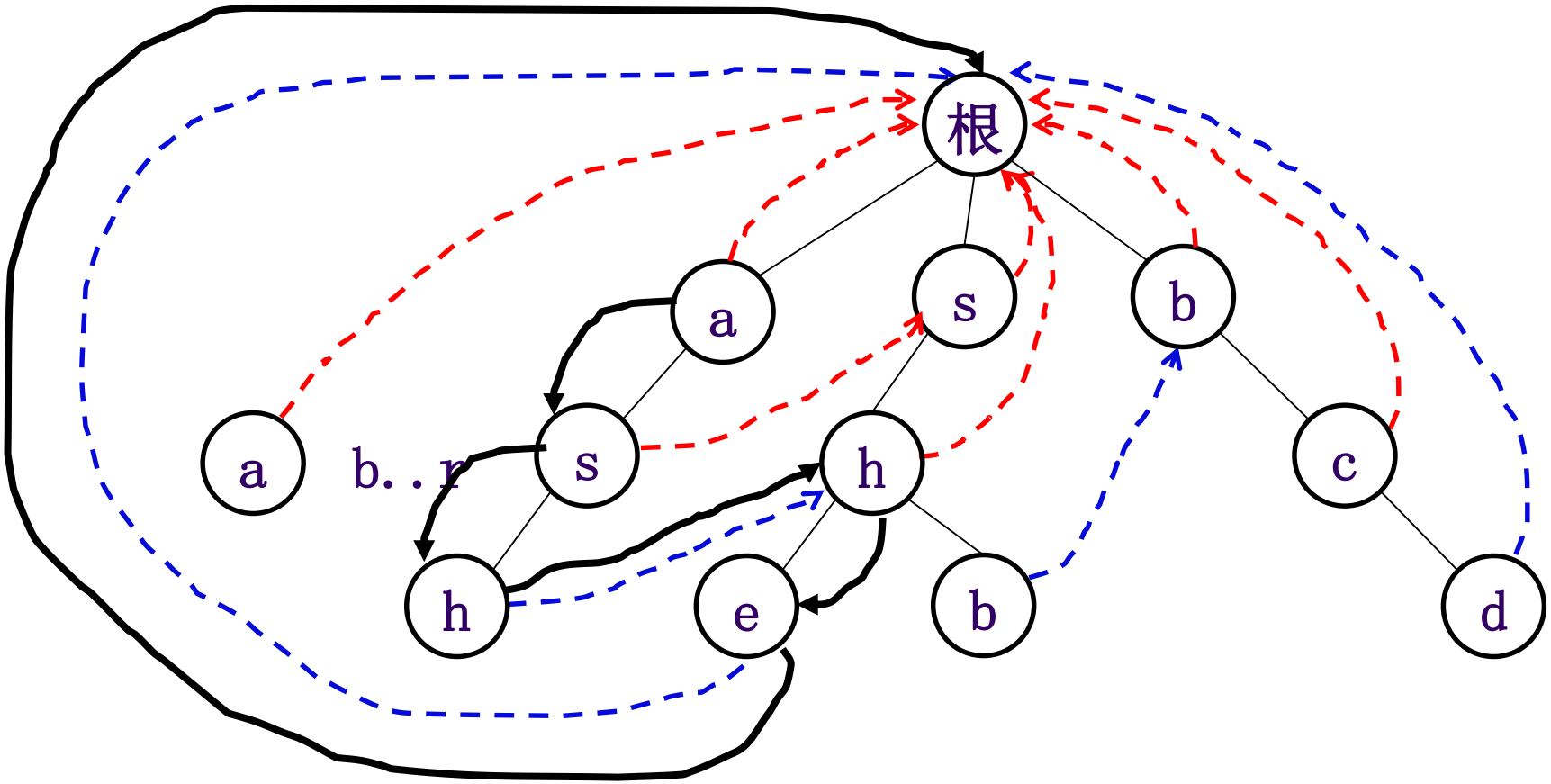
fail指针的建立

- 第三层

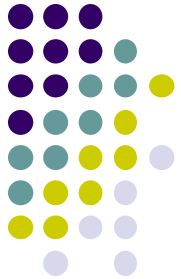


匹配过程

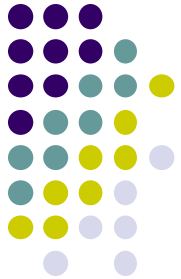
- 文本串ashecd



建立fail指针参考实现（朴素版）

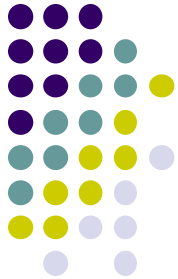


```
void getFail()
{
        int i,now;
        queue<int> q;
        for(i=0;i<26;i++)
                if(trie[0][i]){
                        fail[trie[0][i]]=0;
                        q.push(trie[0][i]);
                }
```



```
        while(!q.empty()){
                now=q.front();q.pop();
                for(i=0;i<26;i++)
                    if(trie[now][i]){
                        fail[trie[now][i]]=trie[fail[now]][i];
                        q.push(trie[now][i]);
                    }else{
                        trie[now][i]=trie[fail[now]][i];
                    }
        }
}//时间复杂度O(|S|), |S| =Σ |si|
```

多模匹配参考代码（朴素版）



```
int query(char s[]) {
        int now=0,len=strlen(s),i,j,ans=0;
        for(i=0;i<len;i++){
                now = trie[now][s[i]-'a'];
                for(j=now; j&&cntWord[j]!=-1;j=fail[j]){
                        ans += cntWord[j];
                        cntWord[j] = -1;
                }
        }
        return ans;
}// 时间复杂度O(|T|)
```