# 有限状态自动机DFA：

Ac自动机是Aho-Corasick 提出的一种dfa

概念去维基看，简单来讲DFA是一张图，有一个开始节点，代表空串，其他各个点并没有实际意义(准确来讲部分自动机，但是不是所有的自动机都要求节点有意义)，点之间有边。

DFA的图的边是有实际意义，一条边代表了**一种字符**，比如在纯英文小写字符构建的dfa中，每个节点有26个分叉，但分叉不一定指向某个实际的点，也可能指向空，如果从起点出发，走出一条路径，那么这个路径可以代表一个字符串。

Ac自动机解决了多模式串匹配的问题。

给定6个模式串字符串

6

beta

alpha

haha

delta

dede

tata

主串是：dedeltalphahahahototatalpha

利用模式串构建ac自动机，完了用主串在ac自动机里跑一边即可。

下面说怎么构建ac自动机，每个节点维护三个属性，son[26]代表26个叉，fail代表失败指针，cnt是代表以这个点为结尾的字符串数量初始是0，注意ac自动机一个点只可能是一个字符串的结尾，所有构建的新节点son和fail指向NULL，事先构建好根节点root，其他节点构建分为三步：

1. 构建字典树，比较容易，每加入一个字符串，就遍历一遍每个字符，字典树从根开始如果没有这个叉，就建立一个新节点，并且连上，向这个方向走；如果有这个叉，就直接向这个方向走即可。走完后最后一个点的cnt+=1
2. 建立fail指针，从root利用广度优先搜索顺序建立，首先处理root，root->fail=root,遍历根节点所有存在的叉，把它们入队，并且把它们的fail连到root。处理完root，对于非root节点，开始广搜的循环，每次取出一个点，找它的所有叉now->son[i]：如果now->son[i]存在, now->son[i]->fail= now->fail->son[i],这里如果now->fail->son[i]不存在，now->son[i]->fail就连到root；如果now->son[i]不存在，就now->son[i]= now->fail->son[i]
3. 对于主串str，从i=0开始，ac自动机从root开始，匹配一遍：如果now->son[str[i]]存在说明匹配成功，正常的前进一步；

如果now->son[str[i]]不存在说明匹配未成功，沿着now->fail反复跳，跳几次取决于问题本身，不同问题有着不同思路，列举几个

1.求模式串在主串中出现几个，同一个多次出现也记录：

沿着fail一直跳，直到跳到根为止，每次加上cnt的值，总和就是答案。

2.求模式串在主串中出现几个，同一个多次出现不记录：

沿着fail一直跳，途中访问的节点的cnt做上标记，比如让他是-1，跳到root了或者跳到是-1的节点就退出，每次加上cnt的值

* Ac自动机指向空的指针，在代码实现时候都视为指向根，有助于简化代码