后缀自动机
Suffix Automaton

李顶龙 (PB13011077)

前言

•自动机的概念书上解释详尽，下面将介绍确定有限状态自动机的实例。

•有同学将举到 AC 自动机的例子，而后缀自动机 (简写为 SAM) 能做 AC 自动机能做的所有操作，更能做 AC 自动机不能做的许多操作。

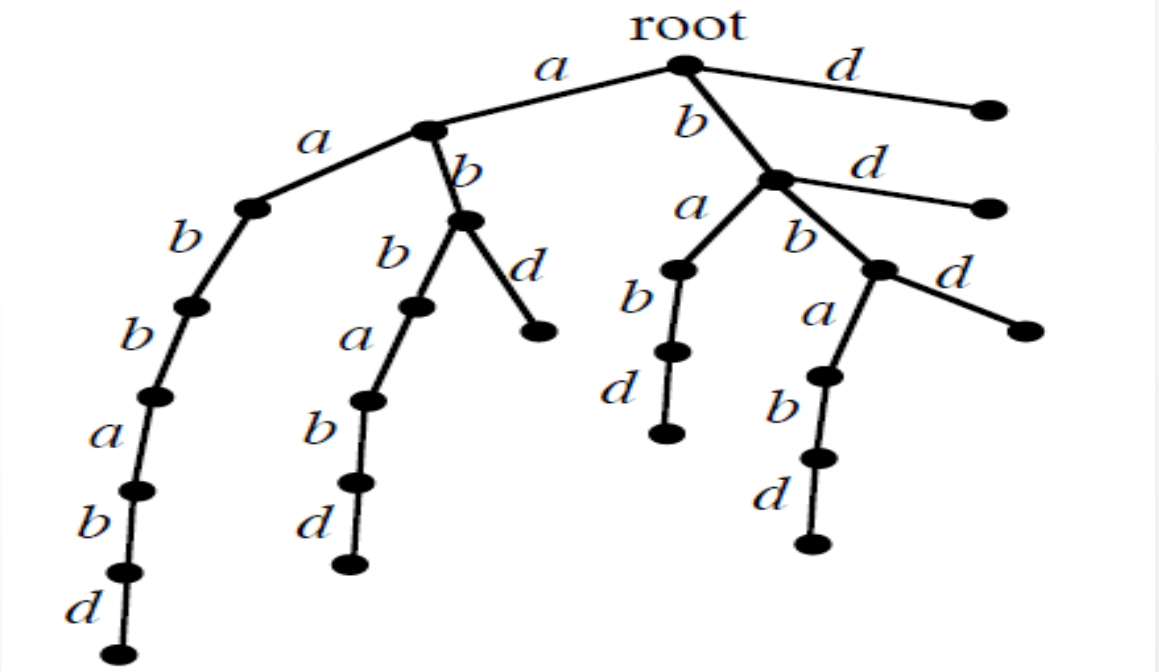
后缀自动机的定义

- 后缀自动机是一个能且仅能识别字符串 S 的所有后缀的自动机。

- 即 SAM(x)==True 当且仅当 x 是 S 的后缀

简单的实现方法

- 例如字符串" aabbabd" 。我们得出所有后缀：“ d”,”bd”,”abd”,”babd”,”bbabd”,“abbabd”,”aabbabd” ，直接构造如下字典树就可以得到最简单粗暴的后缀自动机。



But

- 总共 n 个后缀，我们的字典树上结点数可以达到 O(N^2) 级别。对于比较长的字符串，普通计算机的内存还是不够用的。

- 必须另辟蹊径。

另辟蹊径

- 我们需要找到最简状态 SAM.

- 设 ST(str) 表示通过转移函数 trans(init, str) 转移到的状态。 (init 代表初态， str 为输入字符串)

最简后缀自动机构造 (1)

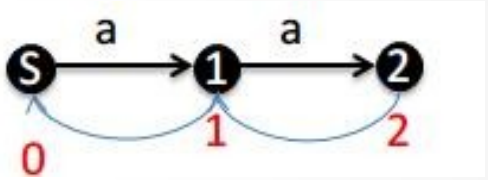
- 每次添加一个字符 x ，使得 SAM(T) 变为 SAM(Tx), 设 T 长度是 L
- 类似 AC 自动机添加转移边和 fail 指针
- 记录自动机中每个点最长可接受的后缀长度 L
- 设添加点时新建的结点为 np ，设上次加入点为 tail ，即 T 的最后一个字符， x 的前一个字符

最简后缀自动机构造 (2)

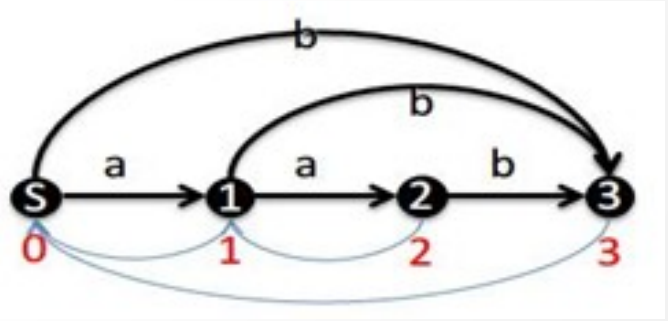
- 从 tail 向 np 链一条状态边 ;np 的 L 为 tail 的 L 再加 1.
- 沿着 tail 的 fail 指针走，走到的每个结点添加一条 tail 边。如果遍历到的点有了指向 x 这个字符的转移边，那么之前的符合要求的点都有 x 这个字符的转移边，就不需要继续遍历了。如果都没有，那么 fail 指针指向 init。
- 这个自动机的 end 状态是 np 以及 fail 链上所有点。

图文过程” aabb”(1)

添加完了两个 a 字符 (黑边为状态边，蓝色边为 fail 指正，红色数字为 L ，圈中数字为点标号)

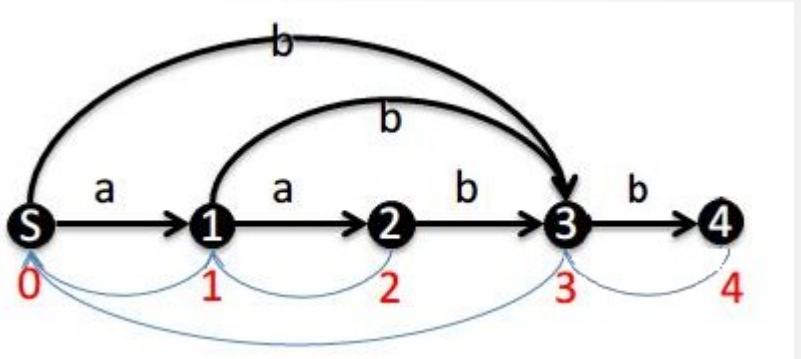


再添加一个 b 字符



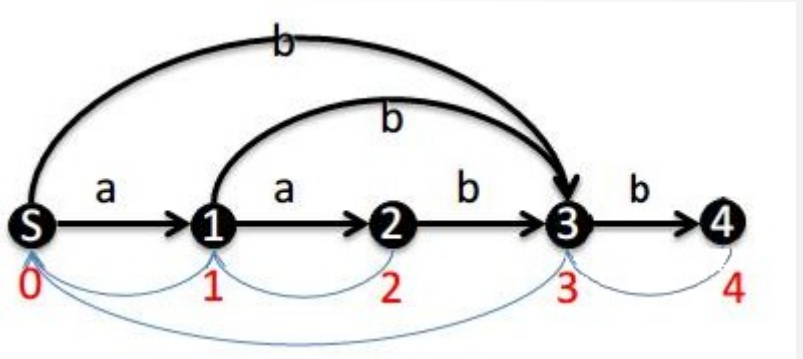
图文过程" aabb"(2)

添加最后一个 b。
现在问题来了！！！
挖掘机技术哪家强？
顺着转移边走，可以转移出" ab" 后缀走到终止态，显然" ab" 不是" aabb" 的后缀！
需要修正！



图文过程” aabb”(3)

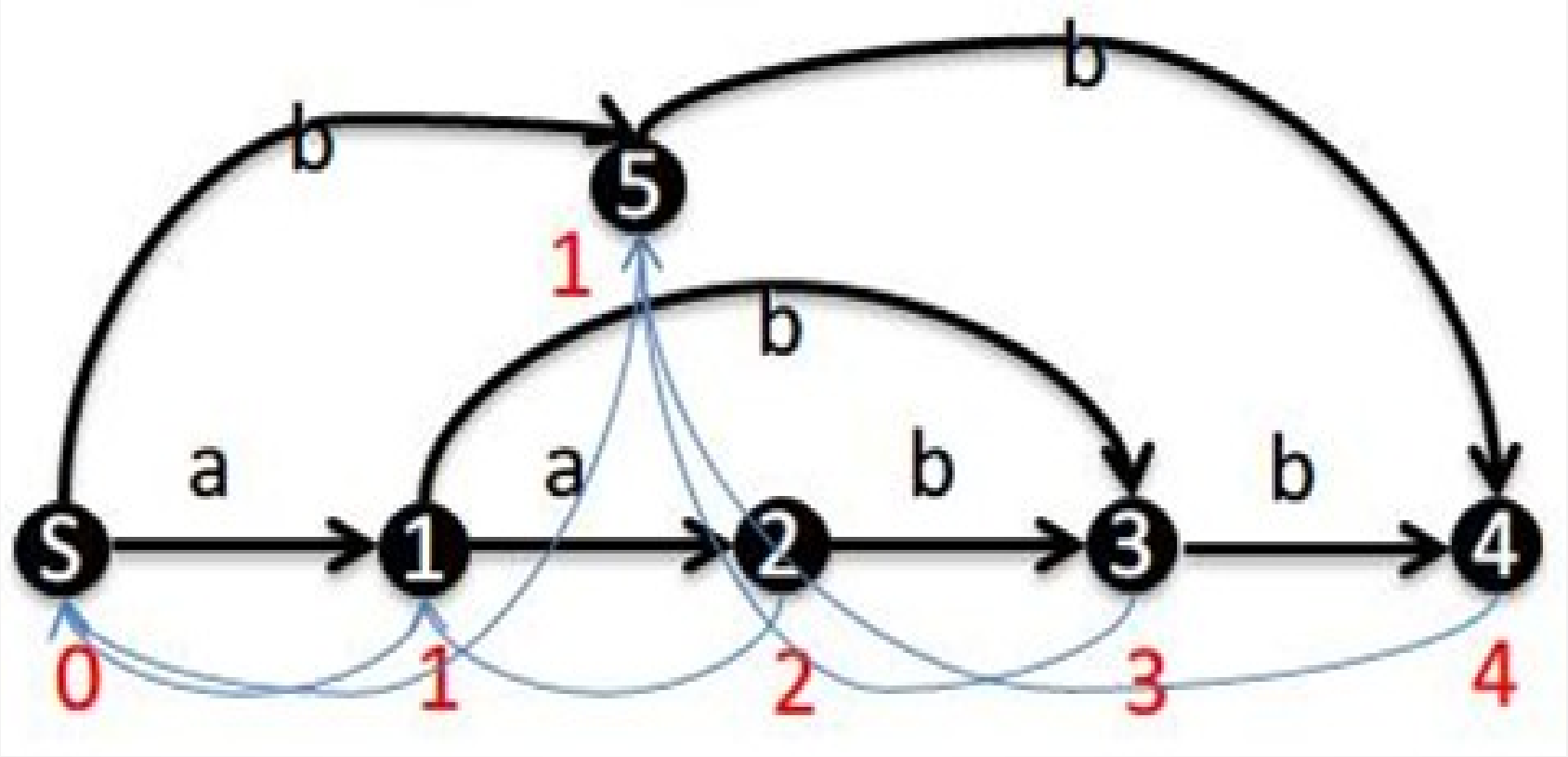
可以看到，罪魁祸首就是 3 号点！所以我们需要拆点，需要保留以 3 号点终止的正常状态，又不产生多余后缀！



图文过程” aabb”(4)

- 假设 a 为第一个有 x 的转移点，转移到 q
- 若 a.L+1==x.L, 将 np 的 fail 指向 b 即可。否则造成了上述新增不合法后缀的请款，那么需要拆点，新建 r 点，将 q 的信息复制到 r ，并将 q 和 np 的 fail 都指向 r。然后就的到了后一张 ppt 的最终版本的后缀自动机。

图文过程” aabb”(5)



代码实现

add 即为每次添加一个字符的过程。 else 中的一大串就是拆点，新建 r 。

```
struct node{
    int len;
    node *f, *ch[MaxM];
}pool[MaxN<<1], *init, *tail;
int tot;
void add(int c)
{
    node *p = tail, *np = &pool[++tot];
    np->len = p->len + 1;
    for (; p && !p->ch[c]; p = p->f) p->ch[c] = np;
    tail = np;
    if (!p) np->f = init;
    else
      if (p->ch[c]->len == p->len + 1) np->f = p->ch[c];
      else {
          node *q = p->ch[c], *r = &pool[++tot];
          *r = *q;
          r->len = p->len + 1;
          q->f = np->f = r;
          for (; p && p->ch[c] == q; p = p->f) p->ch[c] = r;
      }
}
```

复杂度分析

- 每次添加一个结点时最多新增一个 r 结点，所以最后最多有 2n 个结点。所以空间复杂度是 O(n) 级别。

- 沿着 fail 指针走，走过一次之后就不会再走，所以构建的时间复杂度也是 O(n)

应用

- 很好的性质就是所有能接受的状态就是后缀。加上 fail 指针，利用这些性质，可以在极佳的时间空间复杂度内实现 AC 自动机， tire 树，构造后缀树等常见字符串数据结构的功能。
- 经过巧妙的转化还可以轻松的求公共子串，求 k 大子串。限于篇幅不再赘述。
- 总之，拥有了后缀自动机，算法课和数据结构课接触到的所有字符串处理基本都可以用它解决。

更多有限状态自动机

- 很好的性质就是所有能接受的状态就是后缀。加上 fail 指针，利用这些性质，可以在极佳的时间空间复杂度内实现 AC 自动机， tire 树，构造后缀树等常见字符串数据结构的功能。
- 经过巧妙的转化还可以轻松的求公共子串，求 k 大子串。限于篇幅不再赘述。
- 总之，拥有了后缀自动机，算法课和数据结构课接触到的所有字符串处理基本都可以用它解决。

Thanks!