

AC 自动机

Aho-Corasick Automaton

Idea: 在 Trie 树上做 fail 匹配。两个关键：1. 子节点的 `fail` 是 `fail` 的对应子节点；2. 若无子节点，则子节点设为 `fail` 的对应子节点（Trie 树化为 **Trie** 图，这是一种对 AC 自动机的优化）。

ATT: 在 `getfail` 时，先预处理第二层的 `fail[]` 作为初始条件。

Complexity: $O(N + M)$ ，其中 N 是模式串长度和， M 是主串长度。

What's more: **fail 树**——由 `fail` 指针构成的树形结构，在统计模式串出现次数时可以拿出来拓扑或 dfs 统计以减少复杂度。

Code:

```
1  struct Trie{
2      int ch[26], cntEnd; // cntEnd can be changed according to different problem
3  }tr[NODEN];
4  #define root 0
5  int id;
6  void insertTrie(char s[]){
7      int len = (int)strlen(s);
8      int now = root;
9      for(int i = 0; i < len; i++){
10         if(!tr[now].ch[s[i]-'a'])
11             tr[now].ch[s[i]-'a'] = ++id;
12         now = tr[now].ch[s[i]-'a'];
13     }
14     tr[now].cntEnd++;
15 }
16
17 int fail[NODEN];
18 void getFail(){
19     queue<int> q;
20     for(int i = 0; i < 26; i++){
21         if(tr[root].ch[i]){
22             fail[tr[root].ch[i]] = root;
23             q.push(tr[root].ch[i]);
24         }
25     }
26     while(!q.empty()){
27         int cur = q.front(); q.pop();
28         for(int i = 0; i < 26; i++){
29             if(!tr[cur].ch[i])
30                 tr[cur].ch[i] = tr[fail[cur]].ch[i];
31             else{
32                 fail[tr[cur].ch[i]] = tr[fail[cur]].ch[i];
33                 q.push(tr[cur].ch[i]);
34             }
35         }
36     }
37 }
38
39 void ACauto(char s[]){
40     int len = (int)strlen(s);
41     int now = root;
42     for(int i = 0; i < len; i++){
43         now = tr[now].ch[s[i]-'a'];
44         for(int t = now; t && tr[t].cntEnd != -1; t = fail[t]){
45             ans += tr[t].cntEnd;
46             tr[t].cntEnd = -1;
47         }
48     }
49 }
```

```
47 // in this example, if a sting appears many times, we only count it once
48     }
49 }
50 }
```