Aho-Corasick 自动机

邹家树

2025 年 11 月 22 日

Aho-Corasick 自动机，简称 AC 自动机，是一个用来做 multiple pattern matching 的数据结构。

有 k 个模式串 $p_1, \ldots, p_k$，总长度是 m。

AC 自动机 = 字典树 + fail 指针。

把边补满并不是必须的。在不补边的情况下，how much time does it take to actually build all the suffix links?

可以证明是 $O(m)$。考虑某个模式串 p_i。设对应于 p_i 的字典树节点是 x_i。在字典树里，考虑从根到 x_i 的路径上的每个节点的 suffix links 的总的计算时间，是 $O(|p_i|)$，分析方法和计算 border 数组总时间的分析方法是一样的。

设字典树有 n 个节点，也就是说 $p_1, \ldots, p_k$ 总共有 n 个不同的前缀。算出所有 suffix links 的时间不是 $O(n)$ 的。考虑 aaaaaaa, aaaaaab, aaaaaac, aaaaaad 这样的模式串。如果用 n 来表达计算所有 suffix links 的总时间，可以说是 $O(n|\Sigma|)$。

更常用的 AC 自动机 = 把边补满的字典树。

在只有 fail 指针的情况下，计算每个状态（即字典树的节点）的 fail 指针的总时间仍是 $O(N)$ 的（为什么?），N 是字典树的节点数量，对文本串 T 做 multiple pattern matching 的时间仍是 $O(|T|)$ 的。

其实，从自动机的角度来说，（把边补满的字典树）和（字典树 + fail 指针）是等价的。

应该把 fail 指针称作 suffix link。

模板一

```
template <int sigma_size, char alpha>
struct AhoCorasick {
```

```
3       vector<array<int, sigma_size>> go;
4       int new_node() {
5           go.push_back({});
6           return go.size() - 1;
7       }
8       AhoCorasick() { new_node(); }
9
10      int insert(string s) {
11          int p = 0;
12          for (char c : s) {
13              int i = c - alpha;
14              if (go[p][i] == 0)
15                  go[p][i] = new_node();
16              p = go[p][i];
17          }
18          return p;
19      }
20
21      vector<pair<int,int>> build() {
22          vector<int> fail(go.size());
23          queue<int> q;
24          for (int i = 0; i < sigma_size; i++)
25              if (go[0][i])
26                  q.push(go[0][i]);
27          vector<pair<int,int>> edges; // u --> fail[u]
28          while (!q.empty()) {
29              int u = q.front();
30              q.pop();
31              edges.push_back({u, fail[u]});
32              for (int i = 0; i < sigma_size; i++) {
33                  int& to = go[u][i];
34                  if (to) {
35                      q.push(to);
36                      fail[to] = go[fail[u]][i];
```

```
37                  } else
38                      to = go[fail[u]][i];
39              }
40          }
41          return edges;
42      }
43
44      vector<int> run(string s) {
45          int p = 0;
46          vector<int> ans;
47          for (char c : s) {
48              p = go[p][c - alpha];
49              if (p)
50                  ans.push_back(p);
51          }
52          return ans;
53      }
54  };
```

模板二

```
template <int sigma_size, int(*ctoi)(char)>
struct AhoCorasick {
    // 省略部分同模板一
    int insert(string s) {
        int p = 0;
        for (char c : s) {
            int i = ctoi(c);
            if (go[p][i] == 0)
                go[p][i] = new_node();
            p = go[p][i];
        }
        return p;
    }
```

```
    vector<int> run(string s) {
        int p = 0;
        vector<int> ans;
        for (char c : s) {
            p = go[p][ctoi(c)];
            if (p)
                ans.push_back(p);
        }
        return ans;
    }
};
```

AC 自动机的易错点：忘记调用 build()。

例题 1（玄武密码，P5231） 考虑字符集 E，S，W，N 上的字符串。给你一个长为 n 的字符串 s 和 m 个单词 $w_1, \ldots, w_m$。对每个单词 w_i，求既是 w_i 的前缀又是 s 的子串的字符串的最大长度。

限制：$1 \le n \le 10^7$，$1 \le m \le 10^5$，$1 \le |t| \le 100$。

```
const char* sigma = "ESWN";
int ctoi(char c) {
    for (int i = 0; i < 4; i++)
        if (c == sigma[i])
            return i;
    return -1;
}

int main() {
    int n, m;
    cin >> n >> m;
    string s;
    cin >> s;
    AhoCorasick<4, ctoi> ac;
    vector<string> w(m);
    for (int i = 0; i < m; i++) {
        cin >> w[i];
```

```
        ac.insert(w[i]);
    }
    auto suff = ac.build();
    vector<bool> vis(ac.go.size());
    for (int x : ac.run(s))
        vis[x] = true;
    reverse(suff.begin(), suff.end());
    for (auto [i, j] : suff)
        vis[j] = vis[j] || vis[i];
    vector<int> ans(m);
    for (int i = 0; i < m; i++) {
        int p = 0;
        for (int j = 0; j < w[i].size(); j++) {
            p = ac.go[p][ctoi(w[i][j])];
            if (vis[p])
                ans[i] = j + 1;
        }
    }
    for (int x : ans)
        cout << x << '\n';
}
```

例题 2 (Count Substring Query，abc362_g) 给定字符串 S 和 Q 个询问。第 i 个询问是：给你字符串 T_i，求 S 有多少个子串等于 T_i。两子串不同若它们取自不同位置。S 和 T_i 都由小写英文字母构成。

$1 \le |S|, Q \le 5 \times 10^5$，$1 \le |T_i| \le |S|$，$\sum_{i=1}^{Q} |T_i| \le 5 \times 10^5$。

```
    string s;
    int q;
    cin >> s >> q;
    AhoCorasick<26, 'a'> ac;
    vector<int> id(q);
    for (int i = 0; i < q; i++) {
        string t;
        cin >> t;
```

```
        id[i] = ac.insert(t);
    }
    auto fail = ac.build();
    reverse(fail.begin(), fail.end());
    vector<int> cnt(ac.go.size());
    for (int x : ac.run(s))
        cnt[x]++;
    for (auto [i, j] : fail)
        cnt[j] += cnt[i];
    for (int x : id)
        cout << cnt[x] << '\n';
```

例题 3 (P13352，腊肠披萨) 对于字符串 x, y，定义 $\mathrm{LCPS}(x, y)$ 为既是 x 的前缀又是 y 的后缀的字符串的最大长度。

给你 N 个字符串 $s_1, \ldots, s_N$ 和正整数 C, P。求

$$\sum_{i=1}^{N}\sum_{j=1}^{N} C^{\mathrm{LCPS}(s_i, s_j)} \bmod P.$$

限制：$1 \le N, |s_i|, \sum_{i=1}^{N} |s_i| \le 3 \times 10^6$，$2 \le C < P < 2^{30}$。字符集是小写英文字母。

解法一：枚举 i。枚举 s_i 的前缀，用前缀的长度 j 表示这个前缀。我们可以数出 $s_i[1..j]$ 是多少个单词的后缀。

对于字典树的节点 x，我们可以算出 x 是多少个单词的后缀，也可以算出 x 是多少个单词的前缀。于是我们可以算出有多少对 (i, j) 满足 x 是 s_i 前缀又是 s_j 的后缀。如果还有一个节点 y 满足 y 是 s_i 的前缀又是 s_j 的后缀，且 $|y| < |x|$，那么 y 是 x 的一个 border。对于每个非根节点 x，把 x 的最长 border 对应的节点记作 $\mathrm{Border}(x)$。

1 关键词查找问题

例 4 给你 N 个单词 $s_1, s_2, \ldots, s_N$ 和一个文本串 t。对每个 $i = 1, 2, \ldots, N$，

- 判断 s_i 是否在 t 里出现过；
- 若出现过，找出 s_i 在 t 里首次出现的位置。

- 求 s_i 在 t 里出现的次数。
- 求在 t 里出现过的 s_i 的最长前缀的长度。

有四个单词 he, she, his, hers。给你一个文本串 x，找出单词在 x 里出现的位置。例如 x 是 ushers，那么 he 在位置 2 出现，she 在位置 1 出现，hers 在位置 2 出现。

2 子串计数类问题

例题 5 给你 n 个单词 $w_1, \ldots, w_n$。回答 m 个询问。每个询问给你一个文本串 t，求 t 里含有多少个单词。

这个问题比较难，重复计算不好处理。

例题 6 给你 n 个单词 $w_1, \ldots, w_n$。回答 m 个询问。每个询问给你一个文本串 t，求单词在 t 里出现的总次数。

这个问题比较简单。

例题 7 给你 n 个文本串 $t_1, \ldots, t_n$。回答 m 个询问。每个询问给你一个单词 w，求 w 在多少个文本串里出现过。

这个题我还不会。

例题 8 给你 n 个字符串 $s_1, \ldots, s_n$。回答 m 个询问。每个询问给你两个字符串 p, q，求 $s_1, \ldots, s_n$ 中有多少个串满足 p 是它的前缀且 q 是它的后缀。

这是一个典型题。

命题 9 设 s, p, q 是字符串，$\#$ 是 s, p, q 里都没有的一个字符。那么

$$p \text{ 是 } s \text{ 的前缀且 } q \text{ 是 } s \text{ 的后缀} \iff q\#p \text{ 是 } s\#s \text{ 的子串.}$$

由于 s, p, q 三者都不含字符 $\#$，$\#$ 在 $q\#p$ 和 $s\#s$ 里都只出现了一次。

推论 10 如果 $q\#p$ 在 $s\#s$ 里出现了，那么它只出现了一次。

推论 11 设 $q_1\#p_1$ 和 $q_2\#p_2$ 都在 $s\#s$ 里出现了，并且 $q_1\#p_1$ 是 $q_2\#p_2$ 的后缀。那么 $p_1 = p_2$ 且 q_1 是 q_2 的前缀；如果 q_1 是 q_2 的严格后缀，那么在 AC 自动机上跑 $s\#s$ 走不到 $q_1\#p_1$。

例题 12 (QOJ 5551，Selling RNA Strands)

3 状态的权值

例 13 有 N 个单词 $s_1, s_2, \ldots, s_N$ 和 M 个文本串 $t_1, t_2, \ldots, t_M$。求每个文本串能由单词凑成的最长前缀的长度。单词和文本串都只含有小写英文字母。

限制：

- $1 \leq N \leq 20$
- $1 \leq M \leq 50$
- $1 \leq |s| \leq 20$
- $1 \leq |t| \leq 2 \times 10^6$