

Nanyang Technological University  
Joker  
Reference Book



Contents

|     |        |   |     |            |   |
|-----|--------|---|-----|------------|---|
| 1   | String | 3 | 1.2 | Z-function | 3 |
| 1.1 | KMP    | 3 | 1.3 | AC 自动机     | 3 |
|     |        |   | 1.4 | manachar   | 4 |

## 1 String

### 1.1 KMP

```

1 std::vector<int> kmp(std::string s) {
2     int n = s.length();
3     std::vector<int> pi(n);
4     for (int i = 1; i < n; ++i) {
5         int j = pi[i - 1];
6         while (j && s[i] != s[j]) {
7             j = pi[j - 1];
8         }
9         if (s[i] == s[j]) {
10             j++;
11         }
12         pi[i] = j;
13     }
14     return pi;
15 }

```

### 1.2 Z-function

```

16 std::vector<int> z_function(std::string s) {
17     int n = s.length();
18     std::vector<int> z(n);
19     z[0] = n;
20     for (int i = 1, l = 0, r = 0; i < n; ++i) {
21         if (i <= r && z[i - l] < r - i + 1) {
22             z[i] = z[i - l];
23         } else {
24             z[i] = std::max(0, r - i + 1);
25             while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
26                 z[i]++;
27             }
28         }
29         if (i + z[i] - 1 > r) {
30             l = i, r = i + z[i] - 1;
31         }
32     }
33     return z;
34 }

```

### 1.3 AC 自动机

```

const int maxn = 200005;

int ans[maxn];

struct Aho_Corasick {
    std::vector<int> id[maxn];
    int son[maxn][26];
    int fail[maxn];
    int val[maxn];
    int cnt;

    Aho_Corasick() {
        cnt = 0;
        memset(son, 0, sizeof(son));
        memset(fail, 0, sizeof(fail));
        memset(val, 0, sizeof(val));
    }

    void insert(std::string s, int _id) {
        int now = 0;
        for (auto c : s) {
            const int x = c - 'a';
            if (!son[now][x]) {
                son[now][x] = ++cnt;
            }
            now = son[now][x];
        }
        id[now].push_back(_id);
    }

    std::vector<int> fas[maxn];

    void build() {
        std::queue<int> q;
        for (int i = 0; i < 26; ++i) {
            if (son[0][i]) {
                q.push(son[0][i]);
            }
        }
        while (!q.empty()) {
            int now = q.front();
            q.pop();

```

35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76

```

77     for (int i = 0; i < 26; ++i) {
78         if (son[now][i]) {
79             fail[son[now][i]] = son[fail[now]][i];
80             q.push(son[now][i]);
81         } else {
82             son[now][i] = son[fail[now]][i];
83         }
84     }
85 }
86 }
87
88 void getval(std::string s) {
89     int now = 0;
90     for (auto c : s) {
91         now = son[now][c - 'a'];
92         val[now]++;
93     }
94 }
95
96 void build_fail_tree() {
97     for (int i = 1; i <= cnt; ++i) {
98         fas[fail[i]].push_back(i);
99     }
100 }
101
102 void dfs(int now = 0) {
103     for (auto x : fas[now]) {
104         dfs(x);
105         val[now] += val[x];
106     }
107     if (!id[now].empty()) {
108         for (auto x : id[now]) {
109             ans[x] = val[now];
110         }
111     }
112 }
113 };
114
115 Aho_Corasick ac;
116
117 int n;
118
119 int main() {
120     std::cin >> n;

```

```

    for (int i = 1; i <= n; ++i) {
        std::string s;
        std::cin >> s;
        ac.insert(s, i);
    }
    ac.build();
    std::string s;
    std::cin >> s;
    ac.getval(s);
    ac.build_fail_tree();
    ac.dfs();
    for (int i = 1; i <= n; ++i) {
        std::cout << ans[i] << std::endl;
    }
    return 0;
}

```

## 1.4 manacher

```

int n;
char s[N], ss[N];
int len[N];
void Manacher(char *s, int n) {
    For(i, 0, n+1)
        len[i] = 0;
    int mx = 1;
    For(i, 1, n) {
        len[i] = max(1, min(mx + len[mx] - i, len[mx * 2 - i]));
        while (s[i - len[i]] == s[i + len[i]])
            len[i]++;
        if (i + len[i] > mx + len[mx])
            mx = i;
    }
}
int solve(char *s, int n) {
    ss[0] = '#', ss[1] = '*';
    For(i, 1, n) {
        ss[i << 1] = s[i];
        ss[i << 1 | 1] = '*';
    }
    ss[n * 2 + 2] = '$';
    Manacher(ss, n * 2 + 1);
}

```

|     |  |                        |  |                    |  |     |
|-----|--|------------------------|--|--------------------|--|-----|
| 160 |  | <b>int</b> ans=0;      |  | <b>return</b> ans; |  | 163 |
| 161 |  | For(i,1,n*2+1)         |  | }                  |  | 164 |
| 162 |  | ans=max(ans,len[i]-1); |  |                    |  |     |