# Standard Code Library

ONGLU

North Eastern University

August 2021

# Contents

初始化

数据结构

数学

图论

计算几何

字符串

### 字串哈希

```
1   namespace String {
2       const int x = 135;
3       const int p1 = 1e9 + 7, p2 = 1e9 + 9;
4       ull xp1[N], xp2[N], xp[N];
5       void init_xp() {
6           xp1[0] = xp2[0] = xp[0] = 1;
7           for(int i = 1; i < N; i++) {
8               xp1[i] = xp1[i - 1] * x % p1;
9               xp2[i] = xp2[i - 1] * x % p2;
10              xp[i] = xp[i - 1] * x;
11          }
12      }
13      struct HashString {
14          char s[N];
15          int length, subsize;
16          bool sorted;
17          ull h[N], hl[N];
18          ull init(const char *t) {
19              if(xp[0] != 1) init_xp();
20              length = strlen(t);
21              strcpy(s, t);
22              ull res1 = 0, res2 = 0;
23              h[length] = 0;
24              for(int j = length - 1; j >= 0; j--) {
25              #ifdef ENABLE_DOUBLE_HASH
26                  res1 = (res1 * x + s[j]) % p1;
27                  res2 = (res2 * x + s[j]) % p2;
28                  h[j] = (res1 << 32) | res2;
29              #else
30                  res1 = res1 * x + s[j];
31                  h[j] = res1;
32              #endif
33              }
34              return h[0];
35          }
36          //获取子串哈希，左闭右开
37          ull get_substring_hash(int left, int right)  {
38              int len = right - left;
39          #ifdef ENABLE_DOUBLE_HASH
40              unsigned int mask32 = ~(0u);
41              ull left1 = h[left] >> 32, right1 = h[right] >> 32;
42              ull left2 = h[left] & mask32, right2 = h[right] & mask32;
43              return ((((left1 - right1 * xp1[len] % p1 + p1) % p1) << 32) |
44                      (((left2 - right2 * xp2[len] % p2 + p2) % p2)));
45          #else
46              return h[left] - h[right] * xp[len];
47          #endif
48          }
49          void get_all_subs_hash(int sublen) {
50              subsize = length - sublen + 1;
51              for (int i = 0; i < subsize; ++i)
52                  hl[i] = get_substring_hash(i, i + sublen);
```

```
53          sorted = 0;
54        }
55
56        void sort_substring_hash() {
57            sort(hl, hl + subsize);
58            sorted = 1;
59        }
60
61        bool match(ull key) const {
62            if (!sorted) assert (0);
63            if (!subsize) return false;
64            return binary_search(hl, hl + subsize, key);
65        }
66    };
67 }
```

## Trie

```
1  namespace trie {
2      int t[N][26], sz, ed[N];
3      int _new() {
4          sz++;
5          memset(t[sz], 0, sizeof(t[sz]));
6          return sz;
7      }
8      void init() {
9          sz = 0;
10         _new();
11         memset(ed, 0, sizeof(ed));
12     }
13     void Insert(char *s, int n) {
14         int u = 1;
15         for(int i = 0; i < n; i++) {
16             int c = s[i] - 'a';
17             if(!t[u][c]) t[u][c] = _new();
18             u = t[u][c];
19         }
20         ed[u]++;
21     }
22     int find(char *s, int n) {
23         int u = 1;
24         for(int i = 0; i < n; i++) {
25             int c = s[i] - 'a';
26             if(!t[u][c]) return -1;
27             u = t[u][c];
28         }
29         return u;
30     }
31 }
```

## KMP 算法

```
1  namespace KMP {
2      void get_next(char *t, int m, int *nxt) {
3          int j = nxt[0] = 0;
4          for(int i = 1; i < m; i++) {
5              while(j && t[i] != t[j]) j = nxt[j - 1];
6              nxt[i] = j += (t[i] == t[j]);
7          }
8      }
9      vector<int> find(char *t, int m, int *nxt, char *s, int n) {
10         vector<int> ans;
11         int j = 0;
12         for(int i = 0; i < n; i++) {
13             while(j && s[i] != t[j]) j = nxt[j - 1];
14             j += s[i] == t[j];
15             if(j == m) {
16                 ans.push_back(i - m + 1);
17                 j = nxt[j - 1];
18             }
```

```
19            }
20            return ans;
21        }
22    }
```

## manacher 算法

```cpp
1    namespace manacher {
2        char s[N];
3        int p[N], len;
4        void getp(string tmp) {
5            len = 0;
6            for(auto x : tmp) {
7                s[len++] = '#';
8                s[len++] = x;
9            }
10           s[len++] = '#';
11           memset(p, 0, sizeof(int) * (len + 10));
12           int c = 0, r = 0;
13           for(int i = 0; i < len; i++) {
14               if(i <= r) p[i] = min(p[2 * c - i], r - i);
15               else p[i] = 1;
16               while(i - p[i] >= 0 && i + p[i] < len && s[i - p[i]] == s[i + p[i]])
17                   p[i]++;
18               if(i + p[i] - 1 > r) {
19                   r = i + p[i] - 1;
20                   c = i;
21               }
22           }
23           for(int i = 0; i < len; i++) p[i]--;
24       }
25       void getp(char *tmp, int n) {
26           len = 0;
27           for(int i = 0; i < n; i++) {
28               s[len++] = '#';
29               s[len++] = tmp[i];
30           }
31           s[len++] = '#';
32           memset(p, 0, sizeof(int) * (len + 10));
33           int c = 0, r = 0;
34           for(int i = 0; i < len; i++) {
35               if(i <= r) p[i] = min(p[2 * c - i], r - i);
36               else p[i] = 1;
37               while(i - p[i] >= 0 && i + p[i] < len && s[i - p[i]] == s[i + p[i]])
38                   p[i]++;
39               if(i + p[i] - 1 > r) {
40                   r = i + p[i] - 1;
41                   c = i;
42               }
43           }
44           for(int i = 0; i < len; i++) p[i]--;
45       }
46       int getlen() {
47           return *max_element(p, p + len);
48       }
49       int getlen(string s) {
50           getp(s);
51           return getlen();
52       }
53   }
```

## AC 自动机

```cpp
1    struct ac_automaton {
2        int t[N][26], danger[N], tot, fail[N];
3        int dp[N][N];
4        void init() {
5            tot = -1;
6            _new();
7        }
```

```
8       int _new() {
9           tot++;
10          memset(t[tot], 0, sizeof(t[tot]));
11          danger[tot] = 0;
12          fail[tot] = 0;
13          return tot;
14      }
15      void Insert(const char *s) {
16          int u = 0;
17          for(int i = 0; s[i]; i++) {
18              if(!t[u][mp[s[i]]]) t[u][s[i] - 'a'] = _new();
19              u = t[u][mp[s[i]]];
20          }
21          danger[u] = 1;
22      }
23      void build() {
24          queue<int> q;
25          for(int i = 0; i < 26; i++) {
26              if(t[0][i]) {
27                  fail[i] = 0;
28                  q.push(t[0][i]);
29              }
30          }
31          while(q.size()) {
32              int u = q.front(); q.pop();
33              danger[u] |= danger[fail[u]];
34              for(int i = 0; i < 26; i++) {
35                  if(t[u][i]) {
36                      fail[t[u][i]] = t[fail[u]][i];
37                      q.push(t[u][i]);
38                  } else t[u][i] = t[fail[u]][i];
39              }
40          }
41      }
42      int query(const char *s) {
43          memset(dp, 0x3f, sizeof(dp));
44          int n = strlen(s);
45          dp[0][0] = 0;
46          for(int i = 0; i < n; i++) {
47              for(int j = 0; j <= tot; j++) if(!danger[j]) {
48                  for(int k = 0; k < 26; k++) if(!danger[t[j][k]]) {
49                      dp[i + 1][t[j][k]] = min(dp[i + 1][t[j][k]], dp[i][j] + (s[i] - 'a' != k));
50                  }
51              }
52          }
53          int ans = 0x3f3f3f3f;
54          for(int i = 0; i <= tot; i++) if(!danger[i]) {
55              ans = min(ans, dp[n][i]);
56          }
57          return ans == 0x3f3f3f3f ? -1 : ans;
58      }
59  };
```

# 杂项

## int128

```
1   typedef __uint128_t u128;
2   inline u128 read() {
3       static char buf[100];
4       scanf("%s", buf);
5       // std::cin >> buf;
6       u128 res = 0;
7       for(int i = 0;buf[i];++i) {
8           res = res << 4 | (buf[i] <= '9' ? buf[i] - '0' : buf[i] - 'a' + 10);
9       }
10      return res;
11  }
12  inline void output(u128 res) {
13      if(res >= 16)
```

```
14          output(res / 16);
15      putchar(res % 16 >= 10 ? 'a' + res % 16 - 10 : '0' + res % 16);
16      //std::cout.put(res % 16 >= 10 ? 'a' + res % 16 - 10 : '0' + res % 16);
17  }
```