

Algorithm__Template\String__Algorithm.cpp

```

1  /**
2   * 常规字符串处理      STRING
3   * KMP                  KMP
4   * 字符串哈希          hash
5   * trie      字典树  trie
6   * manacher  马拉车  manacher
7   * 序列自动机    fakeac
8  */
9
10 #include <vector>
11 #include <iostream>
12 #include <algorithm>
13 #include <cstring>
14 using namespace std;
15
16 namespace golitter {
17 namespace STRING {
18     /**
19     * 字符串输入
20     */
21 void String_Input() {
22     // 1. string
23     string str; getline(cin, str); // 一行
24     // 2. 从下标1开始
25     char ph[33]; cin >> ph + 1;
26 }
27     /**
28     * 子字符串
29     */
30 void subString() {
31     string str, sub; int pos, length;
32     // 获取字符串 从str的pos下标开始获取，子字符串的长度为length
33     sub = str.substr(pos, length);
34     // 获取substring在string中存在的下标位置，如果不存在为-1
35     int pos = str.find(sub); // 注意：如果直接判断 str.find() == ... ，可能错误，因为
    str.find返回为size_t，没有进行隐式转换。
36 }
37     /**
38     * 字符串转换
39     */
40 void String_Transform() {
41     string str;
42     // 全部转换为大写字母
43     transform(str.begin(), str.end(), str.begin(), ::toupper);
44     // 全部转换为小写字母
45     transform(str.begin(), str.end(), str.begin(), ::tolower);
46     // 翻转
47     reverse(str.begin(), str.end());
48     /**
49     * 字符串和数字之间的转换
50     */
51     int i; double d; float f;
52     // 数字转字符串 -- to_string
53     str = to_string(i); str = to_string(d); str = to_string(f);
54     // 字符串转数字 -- stoi / 数据类型前缀
55     i = stoi(str); d = stod(str); f = stof(str);

```

```

56     long long ll = stoll(str);
57 }
58 }}
59
60 namespace golitter {
61 namespace KMP { // https://ac.nowcoder.com/acm/contest/57358/A
62                 // https://oi-wiki.org/string/kmp/
63 /**
64  *
65  * 给定一个长度为 n 的字符串 s，其 前缀函数 被定义为一个长度为 n 的数组 nxt。其中 nxt[i] 的
    定义是：
66  * 如果子串 s[i] 有一对相等的真前缀与真后缀：s[k-1] 和 s[i - (k - 1)i]，那么 nxt[i] 就是这个
    相等的真前缀（或者真后缀，因为它们相等）的长度，也就是 nxt[i]=k；
67  * 如果不止有一对相等的，那么 nxt[i] 就是其中最长的那一对的长度；
68  * 如果没有相等的，那么 nxt[i]=0。
69  * 简单来说 nxt[i] 就是，子串 s[i] 最长的相等的真前缀与真后缀的长度。
70 */
71 vector<int> prefix_function(string s) {
72     int n = (int)s.length();
73     vector<int> nxt(n);
74     for (int i = 1; i < n; i++) {
75         int j = nxt[i - 1];
76         while (j > 0 && s[i] != s[j]) j = nxt[j - 1];
77         if (s[i] == s[j]) j++;
78         nxt[i] = j;
79     }
80     return nxt;
81 }
82 // 在字符串text中查找pattern字符串
83 vector<int> find_occurrences(string text, string pattern) {
84     string cur = pattern + '#' + text; // 加一个两个字符串中不存在的字符，表示最长前缀为n略
85     int sz1 = text.size(), sz2 = pattern.size();
86     vector<int> v;
87     vector<int> lps = prefix_function(cur);
88     for (int i = sz2 + 1; i <= sz1 + sz2; i++) {
89         if (lps[i] == sz2)
90             v.push_back(i - 2 * sz2);
91     }
92     return v;
93 }
94
95 }}
96
97 namespace golitter {
98 namespace hash_ {
99 /**
100     核心思想：将字符串看成P进制数，P的经验值是131或13331，取这两个值的冲突概率低
101     小技巧：取模的数用2^64，这样直接用unsigned long long存储，溢出的结果就是取模的结果
102 */
103 typedef unsigned long long ULL;
104 ULL h[N], p[N], n; // h[k]存储字符串前k个字母的哈希值，p[k]存储 P^k mod 2^64
105 ULL P = 111451; // 质数即可
106 char str[N];
107 void init() {
108     // 初始化
109     p[0] = 1;
110     for (int i = 1; i <= n; i++) {
111         h[i] = h[i - 1] * P + str[i];
112         p[i] = p[i - 1] * P;
113     }

```

```

114 }
115
116 // 计算子串 str[l ~ r] 的哈希值
117 ULL get(int l, int r) {
118     return h[r] - h[l - 1] * p[r - l + 1];
119 }
120 // C++ lambda
121 auto get_pre = [&](int l, int r) -> ULL {
122     return h[r] - h[l - 1] * p[r - l + 1];
123 }; // lambda注意 逗号
124
125 // 封装
126 // https://ac.nowcoder.com/acm/contest/view-submission?submissionId=62962241
127 class strHash {
128     typedef unsigned long long ULL;
129 public:
130     strHash(const string& s) {
131         this->str = "^" + s;
132         dispose();
133     }
134     ULL get(int l, int r) {
135         return h[r] - h[l - 1] * p[r - l + 1];
136     }
137 private:
138     void dispose() {
139         len = str.size();
140         h.assign(len + 1, 0); p.assign(len + 1, 0);
141         h[0] = p[0] = 1;
142         for(int i = 1; i <= len; ++i) {
143             h[i] = h[i-1] * P + str[i];
144             p[i] = p[i - 1] * P;
145         }
146     }
147
148     const ULL P = 11451;
149     string str;
150     ULL len;
151     vector<ULL> h,p;
152 };
153
154 typedef long long LL;
155 /**
156  * 二维哈希板子
157  * 注意：一定记得初始化 init()
158  */
159 const int B[] = {223333333, 773333333}; // P1 P2
160 const int P = 1000002233;
161
162 LL *p[2];
163
164 void init(int N) { // 初始化
165     p[0] = new LL [N];
166     p[1] = new LL [N];
167     p[0][0] = p[1][0] = 1;
168     for (int i = 1; i < N; i++) {
169         p[0][i] = p[0][i - 1] * B[0] % P;
170         p[1][i] = p[1][i - 1] * B[1] % P;
171     }
172 }
173

```

```

174 struct StringHash2D { // 字符串二维哈希 板子来源: https://ac.nowcoder.com/acm/contest/view-
    submission?submissionId=63032157
175     using LL = long long;
176     int n, m; // n * m
177     vector<vector<LL>> h;
178     StringHash2D(const vector<string> &a) {
179         n = a.size();
180         m = (n == 0 ? 0 : a[0].size());
181         h.assign(n + 1, {});
182         for (int i = 0; i <= n; i++) { // 分配 n * m
183             h[i].assign(m + 1, 0);
184         }
185         for (int i = 0; i < n; i++) {
186             for (int j = 0; j < m; j++) { // 二维哈希
187                 h[i + 1][j + 1] = (h[i][j + 1] * B[0] % P + h[i + 1][j] * B[1] % P +
188                     (P - h[i][j]) * B[0] % P * B[1] % P + a[i][j]) % P;
189             }
190         }
191     }
192
193     LL get(int x1, int y1, int x2, int y2) { // p1 = (x1, y1), p2 = (x2, y2) [p1, p2)
194         return (h[x2][y2] + h[x1][y1] * p[0][x2 - x1] % P * p[1][y2 - y1] % P +
195             (P - h[x1][y2]) * p[0][x2 - x1] % P + (P - h[x2][y1]) * p[1][y2 - y1] % P) %
196         P;
197     };
198
199 }
200
201 namespace golitter {
202     namespace trie {
203
204         const int N = 2e5 + 21;
205         int tr[N][26], idx;
206         int cnt[N];
207         void insert(string str) {
208             int p = 0;
209             for(auto t: str) {
210                 int u = t - '0';
211                 if(!tr[p][u]) tr[p][u] = ++ idx;
212                 p = tr[p][u];
213             }
214             cnt[p]++;
215         }
216         int query(string str) {
217             int p = 0;
218             for(auto t: str) {
219                 int u = t - '0';
220                 if(!tr[p][u]) return 0;
221                 p = tr[p][u];
222             }
223             return cnt[p];
224         }
225     }
226 }
227 namespace trie_01 { // XOR https://zhuanlan.zhihu.com/p/373477543?utm_id=0
228
229     const int N = 1e5 + 21;
230     const int M = N * 31;
231     int tr[M][2], a[N], idx;

```

```

232 int s[N];
233 void insert(int x) {
234     int p = 0;
235     for(int i = 30; i >= 0; --i) {
236         int u = x >> i & 1; // 获得x二进制表示的第i位数
237         if(!tr[p][u]) tr[p][u] = ++idx;
238         p = tr[p][u];
239     }
240 }
241 int query(int x) {
242     int p = 0, res = 0;
243     for(int i = 30; i >= 0; --i) {
244         int u = x >> i & 1;
245         if(tr[p][!u]) {
246             p = tr[p][!u];
247             res += 1<<i;
248         } else {
249             p = tr[p][u];
250             // res = res << 1 + u;
251         }
252     }
253     return res;
254 }
255 void solve() {
256     int n; //idx = 0;
257     for(int i = 0; i <= n; ++i) {
258         tr[i][1] = 0; tr[i][0] = 0;
259         s[i] = 0;
260     }
261     cin>>n;
262     int res = 0;
263     for(int i = 1; i <= n; ++i) cin>>a[i];
264     for(int i = 1; i <= n; ++i) {
265         s[i] = s[i-1] ^ a[i];
266     }
267     for(int i = 0; i <= n; ++i) {
268         insert(s[i]);
269         int t = query(s[i]);
270         res = max(res, t);
271     }
272     cout<<res<<'\n';
273 }
274 }
275 }
276
277
278 namespace golitter {
279 namespace manacher {
280 // https://zhuanlan.zhihu.com/p/549242325
281
282 vector<int> manacher(string &a) { // max(vector<int> P.size() ) - 1;
283     string b = "$|";
284     for(auto t: a) {
285         b += t;
286         b += '|';
287     }
288     int len = b.size();
289     vector<int> hw(len);
290     int maxright(0), mid(0);
291     for(int i = 1; i < len; ++i) {

```

```
292     if(i < maxright) hw[i] = min(hw[mid*2 - i], hw[mid] + mid - i);
293     else hw[i] = 1;
294     while(b[i - hw[i]] == b[i + hw[i]]) hw[i]++;
295     if(i + hw[i] > maxright) {
296         maxright = i + hw[i];
297         mid = i;
298     }
299 }
300 a = b;
301 return hw;
302 }
303
304 }}
305
306 namespace golitter {
307 namespace fakeac { // 序列自动机
308
309 int fake[N][27];
310 char s[N], t[N];
311 void NC23053() { // https://ac.nowcoder.com/acm/problem/23053
312     cin>>s + 1;
313     int n; cin>>n;
314     int len = strlen(s + 1);
315     for(int i = len; i >= 0; --i) {
316         for(int j = 0; j < 26; ++j) fake[i][j] = fake[i+1][j];
317         if(i < len) fake[i][s[i+1] - 'a'] = i + 1;
318     }
319     while(n--) {
320         cin>>t + 1;
321         int tlen = strlen(t + 1);
322         bool fg = false;
323         int pos = 0;
324         for(int i = 1; i <= tlen; ++i) {
325             if(fake[pos][t[i] - 'a']) pos = fake[pos][t[i] - 'a'];
326             else {
327                 fg = true;
328                 break;
329             }
330         }
331         puts(!fg ? "Yes" : "No");
332     }
333 }
334
335 }}
```