
大连理工大学

ACM 代码册

EDITED BY

PUFANYI

目录

第一章 搜索	1
1.1 Dancing Links	1
1.2 $\alpha - \beta$ 剪枝	3
第二章 动态规划	5
第三章 字符串	7
3.1 KMP	7
3.2 Z-function	7
3.3 AC 自动机	8
第四章 数学	11
4.1 快速幂	11
4.2 位运算	11
4.2.1 Gray 码	11
4.3 数论	11
4.3.1 最大公约数	11
4.3.2 欧几里得算法	11
4.3.3 筛法	12
第五章 数据结构	13
5.1 动态树	13
5.1.1 Link-Cut Tree	13
第六章 图论	17
第七章 计算几何	19
第八章 其他	21
8.1 读入输出优化	21

第一章 搜索

1.1 Dancing Links

```
1  const int maxn = 505;
2  const int maxm = 6005;
3
4  struct Dancing_Links {
5      int n, m, total, ans;
6
7      struct Node {
8          int up, down, left, right, row, column;
9      } no[maxm];
10
11     int siz[maxn];
12     int first[maxn];
13     int stk[maxn];
14
15     void init(int n, int m) {
16         ans = 0;
17         this->n = n, this->m = m;
18         memset(first, 0, sizeof(first));
19         memset(siz, 0, sizeof(siz));
20         for (int i = 0; i <= m; ++i) {
21             no[i].left = i - 1, no[i].right = i + 1;
22             no[i].up = no[i].down = i;
23         }
24         no[0].left = m, no[m].right = 0, total = m;
25     }
26
27     void insert(int row, int col) {
28         total++, siz[col]++;
29         no[total].row = row, no[total].column = col;
30         no[total].down = col, no[total].up = no[col].up;
31         no[col].up = total, no[no[total].up].down = total;
32         if (!first[row]) {
33             first[row] = no[total].left = no[total].right = total;
34         } else {
35             no[total].right = first[row], no[total].left = no[first[row]].left;
```

```

36     no[no[total].left].right = no[first[row]].left = total;
37 }
38 }
39
40 void remove(int col) {
41     no[no[col].left].right = no[col].right;
42     no[no[col].right].left = no[col].left;
43     for (int i = no[col].down; i != col; i = no[i].down) {
44         for (int j = no[i].right; j != i; j = no[j].right) {
45             no[no[j].up].down = no[j].down;
46             no[no[j].down].up = no[j].up;
47             siz[no[j].column]--;
48         }
49     }
50 }
51
52 void recover(int col) {
53     for (int i = no[col].up; i != col; i = no[i].up) {
54         for (int j = no[i].left; j != i; j = no[j].left) {
55             no[no[j].up].down = no[no[j].down].up = j;
56             siz[no[j].column]++;
57         }
58     }
59     no[no[col].left].right = no[no[col].right].left = col;
60 }
61
62 bool dance(int dep) {
63     if (!no[0].right) {
64         ans = dep - 1;
65         return true;
66     }
67     int col = no[0].right;
68     for (int i = no[0].right; i; i = no[i].right) {
69         if (siz[i] < siz[col]) {
70             col = i;
71         }
72     }
73     remove(col);
74     for (int i = no[col].down; i != col; i = no[i].down) {
75         stk[dep] = no[i].row;
76         for (int j = no[i].right; j != i; j = no[j].right) {
77             remove(no[j].column);
78         }
79         if (dance(dep + 1)) {
80             return true;
81         }
82         for (int j = no[i].left; j != i; j = no[j].left) {

```

```
83         recover(no[j].column);
84     }
85 }
86     recover(col);
87     return false;
88 }
89 } dlx;
90
91 int main() {
92     int n, m, x;
93     read(n), read(m);
94     dlx.init(n, m);
95     for (int i = 1; i <= n; ++i) {
96         for (int j = 1; j <= m; ++j) {
97             if (read(x) && x) {
98                 dlx.insert(i, j);
99             }
100         }
101     }
102     if (dlx.dance(1)) {
103         for (int i = 1; i <= dlx.ans; ++i) {
104             writesp(dlx.stk[i]);
105         }
106         puts("");
107     } else {
108         puts("No Solution!");
109     }
110     return 0;
111 }
```

1.2 $\alpha - \beta$ 剪枝

第二章 动态规划

第三章 字符串

3.1 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 }
```

3.2 Z-function

```
1 std::vector<int> z_function(std::string s) {
2     int n = s.length();
3     std::vector<int> z(n);
4     z[0] = n;
5     for (int i = 1, l = 0, r = 0; i < n; ++i) {
6         if (i <= r && z[i - l] < r - i + 1) {
7             z[i] = z[i - l];
8         } else {
9             z[i] = std::max(0, r - i + 1);
10            while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
11                z[i]++;
12            }
13        }
14        if (i + z[i] - 1 > r) {
15            l = i, r = i + z[i] - 1;
16        }
17    }
```

```
17 | }
18 |     return z;
19 | }
```

3.3 AC 自动机

```
1 | const int maxn = 200005;
2 |
3 | int ans[maxn];
4 |
5 | struct Aho_Corasick {
6 |     std::vector<int> id[maxn];
7 |     int son[maxn][26];
8 |     int fail[maxn];
9 |     int val[maxn];
10 |    int cnt;
11 |
12 |    Aho_Corasick() {
13 |        cnt = 0;
14 |        memset(son, 0, sizeof(son));
15 |        memset(fail, 0, sizeof(fail));
16 |        memset(val, 0, sizeof(val));
17 |    }
18 |
19 |    void insert(std::string s, int _id) {
20 |        int now = 0;
21 |        for (auto c : s) {
22 |            const int x = c - 'a';
23 |            if (!son[now][x]) {
24 |                son[now][x] = ++cnt;
25 |            }
26 |            now = son[now][x];
27 |        }
28 |        id[now].push_back(_id);
29 |    }
30 |
31 |    std::vector<int> fas[maxn];
32 |
33 |    void build() {
34 |        std::queue<int> q;
35 |        for (int i = 0; i < 26; ++i) {
36 |            if (son[0][i]) {
37 |                q.push(son[0][i]);
38 |            }
39 |        }
40 |        while (!q.empty()) {
```

```
41     int now = q.front();
42     q.pop();
43     for (int i = 0; i < 26; ++i) {
44         if (son[now][i]) {
45             fail[son[now][i]] = son[fail[now]][i];
46             q.push(son[now][i]);
47         } else {
48             son[now][i] = son[fail[now]][i];
49         }
50     }
51 }
52 }
53
54 void getval(std::string s) {
55     int now = 0;
56     for (auto c : s) {
57         now = son[now][c - 'a'];
58         val[now]++;
59     }
60 }
61
62 void build_fail_tree() {
63     for (int i = 1; i <= cnt; ++i) {
64         fas[fail[i]].push_back(i);
65     }
66 }
67
68 void dfs(int now = 0) {
69     for (auto x : fas[now]) {
70         dfs(x);
71         val[now] += val[x];
72     }
73     if (!id[now].empty()) {
74         for (auto x : id[now]) {
75             ans[x] = val[now];
76         }
77     }
78 }
79 };
80
81 Aho_Corasick ac;
82
83 int n;
84
85 int main() {
86     std::cin >> n;
87     for (int i = 1; i <= n; ++i) {
```

```
88     std::string s;
89     std::cin >> s;
90     ac.insert(s, i);
91 }
92 ac.build();
93 std::string s;
94 std::cin >> s;
95 ac.getval(s);
96 ac.build_fail_tree();
97 ac.dfs();
98 for (int i = 1; i <= n; ++i) {
99     std::cout << ans[i] << std::endl;
100 }
101 return 0;
102 }
```

第四章 数学

4.1 快速幂

```
1 template <class T>
2 T ksm(T a, T b, T mod) {
3     T ans = 1;
4     for (; b >>= 1, a = (LL) a * a % mod) {
5         if (b & 1) {
6             ans = (LL) ans * a % mod;
7         }
8     }
9     return ans;
10 }
```

4.2 位运算

4.2.1 Gray 码

```
1 int g(int n) {
2     return n ^ (n >> 1);
3 }
4
5 int rev_g(int g) {
6     int n = 0;
7     for (; g; g >>= 1) {
8         n ^= g;
9     }
10    return n;
11 }
```

4.3 数论

4.3.1 最大公约数

4.3.2 欧几里得算法

```
1 template <class T>
2 T gcd(T a, T b) {
3     while (b) {
4         int t = a % b;
5         a = b;
6         b = t;
7     }
8     return a;
9 }
```

4.3.3 筛法

Eratosthenes 筛法

Euler 筛法

```
1 void oula(const int n = 100000) {
2     np[1] = true;
3     int cnt = 0;
4     for (int i = 2; i <= n; ++i) {
5         if (!np[i]) {
6             prime[++cnt] = i;
7         }
8         for (int j = 1; j <= cnt && (LL) i * prime[j] <= n; ++j) {
9             np[i * prime[j]] = true;
10            if (!(i % prime[j])) {
11                break;
12            }
13        }
14    }
15 }
```


第五章 数据结构

5.1 动态树

5.1.1 Link-Cut Tree

```
1  #include <cstdio>
2  #include <iostream>
3  #include <algorithm>
4
5  using namespace std;
6
7  const int maxn = 300005;
8
9  class LCT {
10     // node
11
12     public:
13         int sum[maxn], val[maxn];
14         int s[maxn][2], fa[maxn];
15
16     private:
17         bool lzy_fan[maxn];
18
19         void push_up(int x) {
20             sum[x] = val[x] ^ sum[s[x][0]] ^ sum[s[x][1]];
21         }
22
23         bool nrt(int x) {
24             return s[fa[x]][0] == x || s[fa[x]][1] == x;
25         }
26
27         void fan(int x) {
28             swap(s[x][0], s[x][1]);
29             lzy_fan[x] ^= 1;
30         }
31
32         void push_down(int x) {
33             if (lzy_fan[x]) {
```

```

34     if (s[x][0]) {
35         fan(s[x][0]);
36     }
37     if (s[x][1]) {
38         fan(s[x][1]);
39     }
40     lzy_fan[x] = 0;
41 }
42 }
43
44 // splay
45 private:
46 void rotate(int x) {
47     int y = fa[x], z = fa[y];
48     int k = (s[y][1] == x), ss = s[x][!k];
49     if (nrt(y)) {
50         s[z][s[z][1] == y] = x;
51     }
52     fa[x] = z;
53     s[x][!k] = y;
54     fa[y] = x;
55     s[y][k] = ss;
56     if (ss) {
57         fa[ss] = y;
58     }
59     push_up(y);
60     push_up(x);
61 }
62
63 int sta[maxn];
64 void splay(int x) {
65     int K = x, top = 0;
66     sta[++top] = K;
67     while (nrt(K)) {
68         sta[++top] = K = fa[K];
69     }
70     while (top) {
71         push_down(sta[top--]);
72     }
73     while (nrt(x)) {
74         int y = fa[x], z = fa[y];
75         if (nrt(y)) {
76             rotate(((s[y][0] == x) ^ (s[z][0] == y)) ? x : y);
77         }
78         rotate(x);
79     }
80 }

```

```
81
82 // LCT
83 private:
84 void access(int x) {
85     for (int y = 0; x; x = fa[y = x]) {
86         splay(x);
87         s[x][1] = y;
88         push_up(x);
89     }
90 }
91
92 void make_root(int x) {
93     access(x);
94     splay(x);
95     fan(x);
96 }
97
98 int find_root(int x) {
99     access(x);
100    splay(x);
101    while (s[x][0]) {
102        push_down(x);
103        x = s[x][0];
104    }
105    splay(x);
106    return x;
107 }
108
109 void split(int x, int y) {
110     make_root(x);
111     access(y);
112     splay(y);
113 }
114
115 public:
116 void link(int x, int y) {
117     make_root(x);
118     if (find_root(y) != x) {
119         fa[x] = y;
120     }
121 }
122
123 void cut(int x, int y) {
124     make_root(x);
125     if (find_root(y) == x && fa[y] == x && !s[y][0]) {
126         fa[y] = s[x][1] = 0;
127         push_up(x);
```

```
128     }
129 }
130
131 void change(int x, int y) {
132     splay(x);
133     val[x] = y;
134     push_up(x);
135 }
136
137 int ask(int x, int y) {
138     split(x, y);
139     return sum[y];
140 }
141 } tr;
142
143 int main() {
144     int n, m;
145     scanf("%d%d", &n, &m);
146     for (int i = 1; i <= n; ++i) {
147         scanf("%d", &tr.val[i]);
148         tr.sum[i] = tr.val[i];
149     }
150     while (m--) {
151         int cmd, x, y;
152         scanf("%d%d%d", &cmd, &x, &y);
153         switch (cmd) {
154             case 0:
155                 printf("%d\n", tr.ask(x, y));
156                 break;
157             case 1:
158                 tr.link(x, y);
159                 break;
160             case 2:
161                 tr.cut(x, y);
162                 break;
163             case 3:
164                 tr.change(x, y);
165         }
166     }
167     return 0;
168 }
```

第六章 图论

第七章 计算几何

第八章 其他

8.1 读入输出优化

```
1 inline char gc() {
2     static const int L = 23333;
3     static char sxd[L], *sss = sxd, *ttt = sxd;
4     if (sss == ttt) {
5         ttt = (sss = sxd) + fread(sxd, 1, L, stdin);
6         if (sss == ttt) {
7             return EOF;
8         }
9     }
10    return *sss++;
11 }
12
13 #ifndef Debug
14 #define dd c = getchar()
15 #else
16 #define dd c = gc()
17 #endif
18 template <class T>
19 inline bool read(T& x) {
20     x = 0;
21     char dd;
22     bool flg = false;
23     for (; !isdigit(c); dd) {
24         if (c == '-') {
25             flg = true;
26         } else if (c == EOF) {
27             return false;
28         }
29     }
30     for (; isdigit(c); dd) {
31         x = (x * 10) + (c ^ 48);
32     }
33     if (flg) {
34         x = -x;
35     }
```

```
36     return true;
37 }
38 #undef dd
39
40 template <class T>
41 inline void write(T x) {
42     if (x < 0) {
43         x = -x;
44         putchar('-');
45     }
46     if (x > 9) {
47         write(x / 10);
48         x %= 10;
49     }
50     putchar(x | 48);
51 }
52
53 template <class T>
54 inline void writeln(T x) {
55     write(x);
56     puts("");
57 }
58
59 template <class T>
60 inline void writesp(T x) {
61     write(x);
62     putchar(' ');
63 }
```