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

目录

第一章 搜索

1.1 Dancing Links

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

1.1 DANCING LINKS 第一章 搜索

```
no[no[total].left].right = no[first[row]].left = total;
36
37
        }
38
     }
39
40
     void remove(int col) {
        no[no[col].left].right = no[col].right;
41
42
        no[no[col].right].left = no[col].left;
        for (int i = no[col].down; i != col; i = no[i].down) {
43
          for (int j = no[i].right; j != i; j = no[j].right) {
44
45
            no[no[j].up].down = no[j].down;
46
            no[no[j].down].up = no[j].up;
            siz[no[j].column]--;
47
          }
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) {
            no[no[j].up].down = no[no[j].down].up = j;
55
            siz[no[j].column]++;
56
          }
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;
        for (int i = no[0].right; i; i = no[i].right) {
68
          if (siz[i] < siz[col]) {</pre>
69
70
            col = i;
          }
71
72
73
        remove(col);
        for (int i = no[col].down; i != col; i = no[i].down) {
74
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
          }
          if (dance(dep + 1)) {
79
80
            return true;
81
82
          for (int j = no[i].left; j != i; j = no[j].left) {
```

第一章 搜索 $1.2 \alpha - \beta$ 剪枝

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

1.2 $\alpha - \beta$ 剪枝

 $1.2 \alpha - \beta$ 剪枝 第一章 搜索

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);
     for (int i = 1; i < n; ++i) {</pre>
4
        int j = pi[i - 1];
5
6
        while (j && s[i] != s[j]) {
7
          j = pi[j - 1];
8
        if (s[i] == s[j]) {
9
10
          j++;
11
        }
12
       pi[i] = j;
13
     return pi;
14
15 }
```

3.2 Z-function

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

3.3 AC 自动机 第三章 字符串

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

3.3 AC 自动机

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

第三章 字符串 3.3 AC 自动机

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

3.3 AC 自动机 第三章 字符串

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

第四章 数学

4.1 快速幂

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

4.2 位运算

4.2.1 Gray 码

```
1 | int g(int n) {
    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

4.3 数论 第四章 数学

```
1 | template <class T>
  T gcd(T a, T b) {
2
    while (b) \{
3
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

```
void oula(const int n = 100000) {
 2
     np[1] = true;
 3
     int cnt = 0;
     for (int i = 2; i <= n; ++i) {</pre>
 5
        if (!np[i]) {
 6
          prime[++cnt] = i;
 7
        }
 8
        for (int j = 1; j <= cnt && (LL) i * prime[j] <= n; ++j) {</pre>
          np[i * prime[j]] = true;
9
          if (!(i % prime[j])) {
10
            break;
11
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>
   #include <iostream>
   #include <algorithm>
3
4
5
   using namespace std;
6
7
   const int maxn = 300005;
8
9
   class LCT {
     // node
10
11
12
    public:
13
     int sum[maxn], val[maxn];
14
     int s[maxn][2], fa[maxn];
15
16
    private:
     bool lzy_fan[maxn];
17
18
19
     void push_up(int x) {
       sum[x] = val[x] ^ sum[s[x][0]] ^ sum[s[x][1]];
20
21
22
     bool nrt(int x) {
23
24
        return s[fa[x]][0] == x || s[fa[x]][1] == x;
25
     }
26
     void fan(int x) {
27
28
       swap(s[x][0], s[x][1]);
       lzy_fan[x] ^= 1;
29
30
     }
31
32
     void push_down(int x) {
       if (lzy_fan[x]) {
33
```

5.1 动态树 第五章 数据结构

```
if (s[x][0]) {
34
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
     // splay
44
45
     private:
     void rotate(int x) {
46
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;
        s[y][k] = ss;
55
56
        if (ss) {
57
          fa[ss] = y;
58
        }
59
        push_up(y);
60
        push_up(x);
61
62
63
      int sta[maxn];
     void splay(int x) {
64
65
        int K = x, top = 0;
66
        sta[++top] = K;
67
        while (nrt(K)) {
          sta[++top] = K = fa[K];
68
69
        }
70
        while (top) {
71
          push_down(sta[top--]);
72
        while (nrt(x)) {
73
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
     }
```

第五章 数据结构 5.1 动态树

```
81
 82
      // LCT
 83
     private:
       void access(int x) {
 84
         for (int y = 0; x; x = fa[y = x]) {
 85
 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
       int find_root(int x) {
 98
 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);
```

5.1 动态树 第五章 数据结构

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

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

8.1 读入输出优化 第八章 其他

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