大连理工大学

ACM 代码册

EDITED BY

PUFANYI

目录

第一章　搜索

1.1　Dancing Links

```
const int maxn = 505;
const int maxm = 6005;

struct Dancing_Links {
  int n, m, total, ans;

  struct Node {
    int up, down, left, right, row, column;
  } no[maxm];

  int siz[maxn];
  int first[maxn];
  int stk[maxn];

  void init(int n, int m) {
    ans = 0;
    this->n = n, this->m = m;
    memset(first, 0, sizeof(first));
    memset(siz, 0, sizeof(siz));
    for (int i = 0; i <= m; ++i) {
      no[i].left = i - 1, no[i].right = i + 1;
      no[i].up = no[i].down = i;
    }
    no[0].left = m, no[m].right = 0, total = m;
  }

  void insert(int row, int col) {
    total++, siz[col]++;
    no[total].row = row, no[total].column = col;
    no[total].down = col, no[total].up = no[col].up;
    no[col].up = total, no[no[total].up].down = total;
    if (!first[row]) {
      first[row] = no[total].left = no[total].right = total;
    } else {
      no[total].right = first[row], no[total].left = no[first[row]].left;
    no[no[total].left].right = no[first[row]].left = total;
  }
}

void remove(int col) {
  no[no[col].left].right = no[col].right;
  no[no[col].right].left = no[col].left;
  for (int i = no[col].down; i != col; i = no[i].down) {
    for (int j = no[i].right; j != i; j = no[j].right) {
      no[no[j].up].down = no[j].down;
      no[no[j].down].up = no[j].up;
      siz[no[j].column]--;
    }
  }
}

void recover(int col) {
  for (int i = no[col].up; i != col; i = no[i].up) {
    for (int j = no[i].left; j != i; j = no[j].left) {
      no[no[j].up].down = no[no[j].down].up = j;
      siz[no[j].column]++;
    }
  }
  no[no[col].left].right = no[no[col].right].left = col;
}

bool dance(int dep) {
  if (!no[0].right) {
    ans = dep - 1;
    return true;
  }
  int col = no[0].right;
  for (int i = no[0].right; i; i = no[i].right) {
    if (siz[i] < siz[col]) {
      col = i;
    }
  }
  remove(col);
  for (int i = no[col].down; i != col; i = no[i].down) {
    stk[dep] = no[i].row;
    for (int j = no[i].right; j != i; j = no[j].right) {
      remove(no[j].column);
    }
    if (dance(dep + 1)) {
      return true;
    }
    for (int j = no[i].left; j != i; j = no[j].left) {
        recover(no[j].column);
      }
    }
    recover(col);
    return false;
  }
} dlx;

int main() {
  int n, m, x;
  read(n), read(m);
  dlx.init(n, m);
  for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= m; ++j) {
      if (read(x) && x) {
        dlx.insert(i, j);
      }
    }
  }
  if (dlx.dance(1)) {
    for (int i = 1; i <= dlx.ans; ++i) {
      writesp(dlx.stk[i]);
    }
    puts("");
  } else {
    puts("No Solution!");
  }
  return 0;
}
```

1.2　$\alpha - \beta$ 剪枝

第二章　动态规划

第三章　字符串

3.1　KMP

```cpp
std::vector<int> kmp(std::string s) {
  int n = s.length();
  std::vector<int> pi(n);
  for (int i = 1; i < n; ++i) {
    int j = pi[i - 1];
    while (j && s[i] != s[j]) {
      j = pi[j - 1];
    }
    if (s[i] == s[j]) {
      j++;
    }
    pi[i] = j;
  }
  return pi;
}
```

3.2　Z-function

```cpp
std::vector<int> z_function(std::string s) {
  int n = s.length();
  std::vector<int> z(n);
  z[0] = n;
  for (int i = 1, l = 0, r = 0; i < n; ++i) {
    if (i <= r && z[i - l] < r - i + 1) {
      z[i] = z[i - l];
    } else {
      z[i] = std::max(0, r - i + 1);
      while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
        z[i]++;
      }
    }
    if (i + z[i] - 1 > r) {
      l = i, r = i + z[i] - 1;
    }
  }
  return z;
}
```

3.3 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()) {
```

```cpp
      int now = q.front();
      q.pop();
      for (int i = 0; i < 26; ++i) {
        if (son[now][i]) {
          fail[son[now][i]] = son[fail[now]][i];
          q.push(son[now][i]);
        } else {
          son[now][i] = son[fail[now]][i];
        }
      }
    }
  }

  void getval(std::string s) {
    int now = 0;
    for (auto c : s) {
      now = son[now][c - 'a'];
      val[now]++;
    }
  }

  void build_fail_tree() {
    for (int i = 1; i <= cnt; ++i) {
      fas[fail[i]].push_back(i);
    }
  }

  void dfs(int now = 0) {
    for (auto x : fas[now]) {
      dfs(x);
      val[now] += val[x];
    }
    if (!id[now].empty()) {
      for (auto x : id[now]) {
        ans[x] = val[now];
      }
    }
  }
};

Aho_Corasick ac;

int n;

int main() {
  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;
}
```

第四章　数学

4.1　快速幂

```
template <class T>
T ksm(T a, T b, T mod) {
  T ans = 1;
  for (; b; b >>= 1, a = (LL) a * a % mod) {
    if (b & 1) {
      ans = (LL) ans * a % mod;
    }
  }
  return ans;
}
```

4.2　位运算

4.2.1　Gray 码

```
int g(int n) {
  return n ^ (n >> 1);
}

int rev_g(int g) {
  int n = 0;
  for (; g; g >>= 1) {
    n ^= g;
  }
  return n;
}
```

4.3　数论

4.3.1　最大公约数

4.3.2　欧几里得算法

```
template <class T>
T gcd(T a, T b) {
  while (b) {
    int t = a % b;
    a = b;
    b = t;
  }
  return a;
}
```

4.3.3 筛法

Eratosthenes 筛法

Euler 筛法

```
void oula(const int n = 100000) {
  np[1] = true;
  int cnt = 0;
  for (int i = 2; i <= n; ++i) {
    if (!np[i]) {
      prime[++cnt] = i;
    }
    for (int j = 1; j <= cnt && (LL) i * prime[j] <= n; ++j) {
      np[i * prime[j]] = true;
      if (!(i % prime[j])) {
        break;
      }
    }
  }
}
```

第五章　数据结构

5.1　动态树

5.1.1　Link-Cut Tree

```cpp
#include <cstdio>
#include <iostream>
#include <algorithm>

using namespace std;

const int maxn = 300005;

class LCT {
  // node

 public:
  int sum[maxn], val[maxn];
  int s[maxn][2], fa[maxn];

 private:
  bool lzy_fan[maxn];

  void push_up(int x) {
    sum[x] = val[x] ^ sum[s[x][0]] ^ sum[s[x][1]];
  }

  bool nrt(int x) {
    return s[fa[x]][0] == x || s[fa[x]][1] == x;
  }

  void fan(int x) {
    swap(s[x][0], s[x][1]);
    lzy_fan[x] ^= 1;
  }

  void push_down(int x) {
    if (lzy_fan[x]) {
      if (s[x][0]) {
        fan(s[x][0]);
      }
      if (s[x][1]) {
        fan(s[x][1]);
      }
      lzy_fan[x] = 0;
    }
  }

  // splay
 private:
  void rotate(int x) {
    int y = fa[x], z = fa[y];
    int k = (s[y][1] == x), ss = s[x][!k];
    if (nrt(y)) {
      s[z][s[z][1] == y] = x;
    }
    fa[x] = z;
    s[x][!k] = y;
    fa[y] = x;
    s[y][k] = ss;
    if (ss) {
      fa[ss] = y;
    }
    push_up(y);
    push_up(x);
  }

  int sta[maxn];
  void splay(int x) {
    int K = x, top = 0;
    sta[++top] = K;
    while (nrt(K)) {
      sta[++top] = K = fa[K];
    }
    while (top) {
      push_down(sta[top--]);
    }
    while (nrt(x)) {
      int y = fa[x], z = fa[y];
      if (nrt(y)) {
        rotate(((s[y][0] == x) ^ (s[z][0] == y)) ? x : y);
      }
      rotate(x);
    }
  }

 // LCT
private:
 void access(int x) {
   for (int y = 0; x; x = fa[y = x]) {
     splay(x);
     s[x][1] = y;
     push_up(x);
   }
 }

 void make_root(int x) {
   access(x);
   splay(x);
   fan(x);
 }

 int find_root(int x) {
   access(x);
   splay(x);
   while (s[x][0]) {
     push_down(x);
     x = s[x][0];
   }
   splay(x);
   return x;
 }

 void split(int x, int y) {
   make_root(x);
   access(y);
   splay(y);
 }

public:
 void link(int x, int y) {
   make_root(x);
   if (find_root(y) != x) {
     fa[x] = y;
   }
 }

 void cut(int x, int y) {
   make_root(x);
   if (find_root(y) == x && fa[y] == x && !s[y][0]) {
     fa[y] = s[x][1] = 0;
     push_up(x);
```

```
    }
  }

  void change(int x, int y) {
    splay(x);
    val[x] = y;
    push_up(x);
  }

  int ask(int x, int y) {
    split(x, y);
    return sum[y];
  }
} tr;

int main() {
  int n, m;
  scanf("%d%d", &n, &m);
  for (int i = 1; i <= n; ++i) {
    scanf("%d", &tr.val[i]);
    tr.sum[i] = tr.val[i];
  }
  while (m--) {
    int cmd, x, y;
    scanf("%d%d%d", &cmd, &x, &y);
    switch (cmd) {
      case 0:
        printf("%d\n", tr.ask(x, y));
        break;
      case 1:
        tr.link(x, y);
        break;
      case 2:
        tr.cut(x, y);
        break;
      case 3:
        tr.change(x, y);
    }
  }
  return 0;
}
```

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

```cpp
inline char gc() {
  static const int L = 23333;
  static char sxd[L], *sss = sxd, *ttt = sxd;
  if (sss == ttt) {
    ttt = (sss = sxd) + fread(sxd, 1, L, stdin);
    if (sss == ttt) {
      return EOF;
    }
  }
  return *sss++;
}

#ifdef Debug
#define dd c = getchar()
#else
#define dd c = gc()
#endif
template <class T>
inline bool read(T& x) {
  x = 0;
  char dd;
  bool flg = false;
  for (; !isdigit(c); dd) {
    if (c == '-') {
      flg = true;
    } else if (c == EOF) {
      return false;
    }
  }
  for (; isdigit(c); dd) {
    x = (x * 10) + (c ^ 48);
  }
  if (flg) {
    x = -x;
  }
  return true;
}
#undef dd

template <class T>
inline void write(T x) {
  if (x < 0) {
    x = -x;
    putchar('-');
  }
  if (x > 9) {
    write(x / 10);
    x %= 10;
  }
  putchar(x | 48);
}

template <class T>
inline void writeln(T x) {
  write(x);
  puts("");
}

template <class T>
inline void writesp(T x) {
  write(x);
  putchar(' ');
}
```