ACMICPC Standard Code Library

Beijing University of Posts and Telecommunications

April 29, 2016

Contents

1 Math

1.1 fft

```
1  // Copyright [2017] <dmnsn7@gmail.com>
2
3  #include <bits/stdc++.h>
4
5  using std::swap;
6
7  const double PI = acos(-1);
8  struct Complex {
9    double x, y;
10   Complex() {
11     x = 0;
12     y = 0;
13   }
14   Complex(double _x, double _y) {
15     x = _x;
16     y = _y;
17   }
18   Complex operator-(const Complex &b) const {
19     return Complex(x - b.x, y - b.y);
20   }
21   Complex operator+(const Complex &b) const {
22     return Complex(x + b.x, y + b.y);
23   }
24   Complex operator*(const Complex &b) const {
25     return Complex(x * b.x - y * b.y, x * b.y + y * b.x);
26   }
27 };
28 void change(Complex y[], int len) {
29   for (int i = 1, j = len / 2; i < len - 1; i++) {
30     if (i < j) {
31       swap(y[i], y[j]);
32     }
33
34     int k = len / 2;
35
36     while (j >= k) {
37       j -= k;
38       k /= 2;
39     }
40
41     if (j < k) {
42       j += k;
43     }
44   }
45 }
46 void fft(Complex y[], int len, int on) {
47   change(y, len);
48
49   for (int h = 2; h <= len; h <<= 1) {
50     Complex wn(cos(-on * 2 * PI / h), sin(-on * 2 * PI / h));
51
52     for (int j = 0; j < len; j += h) {
53       Complex w(1, 0);
54
55       for (int k = j; k < j + h / 2; k++) {
56         Complex u = y[k];
57         Complex t = w * y[k + h / 2];
58         y[k] = u + t;
59         y[k + h / 2] = u - t;
60         w = w * wn;
61       }
62     }
63   }
64
65   if (on == -1) {
```

```
66      for (int i = 0; i < len; i++) {
67        y[i].x /= len;
68      }
69    }
70  }
71
72  int main() { return 0; }
```

2 String

2.1 sa

```
1   // Copyright [2017] <dmnsn7@gmail.com>
2
3   #include <bits/stdc++.h>
4
5   using std::swap;
6
7   const int maxn = 0;
8
9   int r[maxn], wa[maxn], wb[maxn], wv[maxn], _ws[maxn], sa[maxn];
10  int _rank[maxn], height[maxn];
11
12  /*
13   *       nn  -10
14   *         sasa[1]~sa[n]
15   *       rankrank[0]~rank[n  -1],
16   *    heightii       -12~  nsa
17   */
18
19  inline bool cmp(int *r, int a, int b, int l) {
20    return r[a] == r[b] && r[a + l] == r[b + l];
21  }
22
23  void da(int n, int m) {
24    int i, j, p, *x = wa, *y = wb;
25
26    for (i = 0; i < m; i++) {
27      _ws[i] = 0;
28    }
29
30    for (i = 0; i < n; i++) {
31      _ws[x[i] = r[i]]++;
32    }
33
34    for (i = 1; i < m; i++) {
35      _ws[i] += _ws[i - 1];
36    }
37
38    for (i = n - 1; i >= 0; i--) {
39      sa[--_ws[x[i]]] = i;
40    }
41
42    for (j = 1, p = 1; p < n; j <<= 1, m = p) {
43      for (p = 0, i = n - j; i < n; i++) {
44        y[p++] = i;
45      }
46
47      for (i = 0; i < n; i++)
48        if (sa[i] >= j) {
49          y[p++] = sa[i] - j;
50        }
51
52      for (i = 0; i < n; i++) {
53        wv[i] = x[y[i]];
54      }
55
56      for (i = 0; i < m; i++) {
57        _ws[i] = 0;
```

```
58      }
59
60      for (i = 0; i < n; i++) {
61        _ws[wv[i]]++;
62      }
63
64      for (i = 1; i < m; i++) {
65        _ws[i] += _ws[i - 1];
66      }
67
68      for (i = n - 1; i >= 0; i--) {
69        sa[--_ws[wv[i]]] = y[i];
70      }
71
72      swap(x, y);
73
74      for (p = 1, x[sa[0]] = 0, i = 1; i < n; i++) {
75        x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
76      }
77    }
78
79    return;
80  }
81
82  void calheitght(int n) {
83    int i, j, k = 0;
84
85    for (i = 1; i < n; i++) {
86      _rank[sa[i]] = i;
87    }
88
89    // print(_rank, n);
90    for (i = 0; i < n; height[_rank[i++]] = k)
91      for (k ? k-- : 0, j = sa[_rank[i] - 1]; r[i + k] == r[j + k]; k++)
92        {;}
93
94    return;
95  }
96
97  int main() {
98    return 0;
99  }
```

2.2 sam

```
1   // Copyright [2017] <dmnsn7@gmail.com>
2
3   #include <bits/stdc++.h>
4
5   const int MAXN = 0;
6
7   int tot, root, np, tail;
8   int cnt[MAXN], son[MAXN][MAXN], gao[MAXN], g[MAXN], b[MAXN], len[MAXN],
9       pre[MAXN];
10
11  void copy(int x, int y) {
12    pre[x] = pre[y];
13    len[x] = len[y];
14    memcpy(son[x], son[y], sizeof son[0]);
15  }
16
17  void insert(int c, int l) {
18    int p = tail, np = ++tot;
19    len[np] = l;
20    tail = np;
21
22    while (p && son[p][c] == 0) {
23      son[p][c] = np, p = pre[p];
24    }
25
```

```
26    if (p == 0) {
27      pre[np] = root;
28    } else {
29      int q = son[p][c];
30
31      if (len[p] + 1 == len[q]) {
32        pre[np] = q;
33      } else {
34        int nq = ++tot;
35        copy(nq, q);
36        len[nq] = len[p] + 1;
37        pre[np] = pre[q] = nq;
38
39        while (p && son[p][c] == q) {
40          son[p][c] = nq, p = pre[p];
41        }
42      }
43    }
44  }
45
46  void build(int n) {
47    for (int i = 1; i <= tot; i++) {
48      cnt[len[i]]++;
49    }
50
51    for (int i = 1; i <= n; i++) {
52      cnt[i] += cnt[i - 1];
53    }
54
55    for (int i = 1; i <= tot; i++) {
56      b[--cnt[len[i]]] = i;
57    }
58
59    for (int i = tot - 1; i >= 0; i--) {
60      int p = b[i], k = 0;
61      g[p] = 1;
62
63      for (int j = 0; j < 26; j++)
64        if (son[p][j]) {
65          int v = son[p][j];
66          g[p] += g[v];
67          son[p][k] = v;
68          gao[v] = j + 'a';
69          k++;
70        }
71
72      son[p][k] = 0;
73    }
74  }
75
76  int main() { return 0; }
```

2.3 kmp

```
1  // Copyright [2017] <dmnsn7@gmail.com>
2
3  #include <bits/stdc++.h>
4
5  char *s, *f;
6  void getFail() {
7    int m = strlen(s);
8    f[0] = f[1] = 0;
9
10   for (int i = 1; i < m; i++) {
11     int j = f[i];
12
13     while (j && s[i] != s[j]) {
14       j = f[j];
15     }
```

```
16
17      f[i + 1] = s[i] == s[j] ? j + 1 : 0;
18    }
19  }
20
21  int main() { return 0; }
```

2.4 ac

```
1   // Copyright [2017] <dmnsn7@gmail.com>
2
3   #include <bits/stdc++.h>
4
5   using std::queue;
6   using std::vector;
7
8   struct CH {
9     int fail, isend;
10    vector<int> next;
11    void init() {}
12  };
13
14  vector<CH> ch;
15
16  int sz = 0;
17  queue<int> q;
18  void init() {
19    sz = 1;
20    ch[0].init();
21  }
22
23  void build() {
24    queue<int> q;
25    ch[0].fail = 0;
26
27    for (int c = 0; c < 4; c++) {
28      int u = ch[0].next[c];
29
30      if (u) {
31        ch[u].fail = 0;
32        q.push(u);
33      }
34    }
35
36    while (!q.empty()) {
37      int r = q.front();
38      q.pop();
39
40      for (int c = 0; c < 4; c++) {
41        int u = ch[r].next[c];
42
43        if (!u) {
44          ch[r].next[c] = ch[ch[r].fail].next[c];
45          continue;
46        }
47
48        q.push(u);
49        int v = ch[r].fail;
50
51        while (v && !ch[v].next[c]) {
52          v = ch[v].fail;
53        }
54
55        ch[u].fail = ch[v].next[c];
56        ch[u].isend |= ch[ch[u].fail].isend;
57      }
58    }
59  }
60
61  int main() { return 0; }
```

3 Geometry

3.1 c2c

```
1  // Copyright [2017] <dmnsn7@gmail.com>
2
3  #include <bits/stdc++.h>
4
5  using std::max;
6
7  struct Point {
8    double x, y;
9    Point() {}
10   Point(double _x, double _y) {
11     x = _x;
12     y = _y;
13   }
14
15   double len() const { return sqrt(x * x + y * y); }
16   Point operator+(const Point &b) const { return Point(0, 0); }
17   Point operator-(const Point &b) const { return Point(0, 0); }
18   Point operator*(const double &b) const { return Point(0, 0); }
19 };
20
21 struct Circle {
22   Point o;
23   double r;
24 };
25
26 Point rotate(const Point &p, double cost, double sint) {
27   double x = p.x, y = p.y;
28   return Point(x * cost - y * sint, x * sint + y * cost);
29 }
30
31 void circle_cross_circle(Circle a, Circle b, Point cro[]) {
32   double d = (a.o - b.o).len();
33   double cost = (a.r * a.r + d * d - b.r * b.r) / (a.r * d * 2);
34   double sint = sqrt(max(1.0 - cost * cost, 0.0));
35   Point v = (b.o - a.o) * (a.r / d);
36   cro[0] = a.o + rotate(v, cost, -sint);
37   cro[1] = a.o + rotate(v, cost, sint);
38 }
39
40 int main() { return 0; }
```

3.2 c2l

```
1  // Copyright [2017] <dmnsn7@gmail.com>
2
3  #include <bits/stdc++.h>
4
5  using std::min;
6  using std::max;
7  using std::vector;
8
9  const double PI = 3.14;
10
11 struct Point {
12   double x, y;
13   Point() {}
14   Point(double _x, double _y) {
15     x = _x;
16     y = _y;
17   }
18
19   double len() const { return sqrt(x * x + y * y); }
20   Point operator-(const Point &b) const { return Point(0, 0); }
21   Point operator*(const double &b) const { return Point(0, 0); }
22   Point operator/(const double &b) const { return Point(0, 0); }
23   double operator*(const Point &b) const { return 0; }
```

```
24  };
25
26  int dlcmp(double x) { return 0; }
27
28  Point crosspt(const Point &a, const Point &b, const Point &p, const Point &q) {
29    double a1 = (b - a) * (p - a);
30    double a2 = (b - a) * (q - a);
31    return (p * a2 - q * a1) / (a2 - a1);
32  }
33
34  double r = 0;
35  double sector_area(const Point &a, const Point &b) {
36    double theta = atan2(a.y, a.x) - atan2(b.y, b.x);
37
38    while (theta <= 0) {
39      theta += 2 * PI;
40    }
41
42    while (theta > 2 * PI) {
43      theta -= 2 * PI;
44    }
45
46    theta = min(theta, 2 * PI - theta);
47    return r * r * theta / 2;
48  }
49
50  double sqr(double x) { return x * x; }
51  void circle_cross_line(Point a, Point b, Point o, double r, Point ret[],
52                         int num) {
53    double x0 = o.x, y0 = o.y;
54    double x1 = a.x, y1 = a.y;
55    double x2 = b.x, y2 = b.y;
56    double dx = x2 - x1, dy = y2 - y1;
57    double A = dx * dx + dy * dy;
58    double B = 2 * dx * (x1 - x0) + 2 * dy * (y1 - y0);
59    double C = sqr(x1 - x0) + sqr(y1 - y0) - sqr(r);
60    double delta = B * B - 4 * A * C;
61    num = 0;
62
63    if (dlcmp(delta) >= 0) {
64      double t1 = (-B - sqrt(max(delta, 0.0))) / (2 * A);
65      double t2 = (-B + sqrt(max(delta, 0.0))) / (2 * A);
66
67      if (dlcmp(t1 - 1) <= 0 && dlcmp(t1) >= 0) {
68        ret[num++] = Point(x1 + t1 * dx, y1 + t1 * dy);
69      }
70
71      if (dlcmp(t2 - 1) <= 0 && dlcmp(t2) >= 0) {
72        ret[num++] = Point(x1 + t2 * dx, y1 + t2 * dy);
73      }
74    }
75  }
76
77  double calc(const Point &a, const Point &b) {
78    Point p[2];
79    int num = 0;
80    int ina = dlcmp(a.len() - r) < 0;
81    int inb = dlcmp(b.len() - r) < 0;
82
83    if (ina) {
84      if (inb) {
85        return fabs(a * b) / 2;
86      } else {
87        circle_cross_line(a, b, Point(0, 0), r, p, num);
88        return sector_area(b, p[0]) + fabs(a * p[0]) / 2;
89      }
90    } else {
91      if (inb) {
92        circle_cross_line(a, b, Point(0, 0), r, p, num);
```

```
93        return sector_area(p[0], a) + fabs(p[0] * b) / 2;
94      } else {
95        circle_cross_line(a, b, Point(0, 0), r, p, num);
96
97        if (false) {
98          return sector_area(a, p[0]) + sector_area(p[1], b) +
99                 fabs(p[0] * p[1]) / 2;
100       } else {
101         return sector_area(a, b);
102       }
103     }
104   }
105 }
106
107 vector<Point> res;
108 int n;
109 double area() {
110   double ret = 0;
111
112   for (int i = 0; i < n; i++) {
113     int sgn = dlcmp(res[i] * res[i + 1]);
114
115     if (sgn != 0) {
116       ret += sgn * calc(res[i], res[i + 1]);
117     }
118   }
119
120   return ret;
121 }
122
123 int main() { return 0; }
```

3.3 halfplaneintersection

```
1  // Copyright [2017] <dmnsn7@gmail.com>
2
3  #include <bits/stdc++.h>
4
5  using std::cin;
6  using std::deque;
7  using std::swap;
8  using std::vector;
9
10 const double eps = 1e-8;
11
12 int sgn(double n) { return 0; }
13
14 struct Point {
15   double x, y;
16   Point() {}
17   Point(double _x, double _y) {
18     x = _x;
19     y = _y;
20   }
21
22   void input() { cin >> x >> y; }
23
24   double z() const { return 0; }
25   Point rev() const { return Point(0, 0); }
26   double len() const { return sqrt(x * x + y * y); }
27   double arg() const { return 0; }
28   Point operator+(const Point &b) const { return Point(0, 0); }
29   Point operator-(const Point &b) const { return Point(0, 0); }
30   double operator*(const Point &b) const { return 0; }
31   Point operator*(const double &b) const { return Point(0, 0); }
32   Point operator/(const double &b) const { return Point(0, 0); }
33 };
34
35 struct Halfplane {
36   Point a, b;
```

```
37    Halfplane() {}
38    Halfplane(Point a, Point b) : a(a), b(b) {}
39
40    bool satisfy(const Point &rhs) const { return sgn((rhs - a) * (b - a)) <= 0; }
41    bool operator<(const Halfplane &rhs) const {
42      int res = sgn((b - a).arg() - (rhs.b - rhs.a).arg());
43      return res == 0 ? rhs.satisfy(a) : res < 0;
44    }
45  };
46
47  Point crosspoint(const Halfplane &a, const Halfplane &b) {
48    double k = (b.a - b.b) * (a.a - b.b);
49    k = k / (k - ((b.a - b.b) * (a.b - b.b)));
50    return a.a + (a.b - a.a) * k;
51  }
52
53  vector<Point> halfplaneIntersection(vector<Halfplane> v) {
54    sort(v.begin(), v.end());
55    deque<Halfplane> q;
56    deque<Point> ans;
57    q.push_back(v[0]);
58
59    for (int i = 1; i < v.size(); i++) {
60      if (sgn((v[i - 1].b - v[i - 1].a) * (v[i].b - v[i].a)) == 0) {
61        continue;
62      }
63
64      while (ans.size() > 0 && !v[i].satisfy(ans.back())) {
65        ans.pop_back();
66        q.pop_back();
67      }
68
69      while (ans.size() > 0 && !v[i].satisfy(ans.front())) {
70        ans.pop_front();
71        q.pop_front();
72      }
73
74      ans.push_back(crosspoint(q.back(), v[i]));
75      q.push_back(v[i]);
76    }
77
78    while (ans.size() > 0 && !q.front().satisfy(ans.back())) {
79      ans.pop_back();
80      q.pop_back();
81    }
82
83    while (ans.size() > 0 && !q.back().satisfy(ans.front())) {
84      ans.pop_front();
85      q.pop_front();
86    }
87
88    ans.push_back(crosspoint(q.back(), q.front()));
89    return vector<Point>(ans.begin(), ans.end());
90  }
91
92  double area(const vector<Point> &p, int ansi) {
93    double res = 0;
94
95    for (int i = ansi; i + 1 < p.size(); i++) {
96      res += p[i] * p[i + 1];
97    }
98
99    res += p.back() * p[ansi];
100   return fabs(res) / 2;
101 }
102
103 double ptol(Point a, Point b, Point c) {
104   double are = fabs((b - a) * (c - a));
```

```
105   return are / (b - c).len();
106 }
107
108 vector<Point> p;
109 int main() {
110   int T_T, n, nc = 0;
111   cin >> T_T;
112   Point __0(0, 0), __1(1, 0), __2(1, 1), __3(0, 1);
113
114   while (T_T--) {
115     printf("Case #%d:\n", ++nc);
116     scanf("%d", &n);
117
118     for (int i = 0; i < n; i++) {
119       p[i].input();
120     }
121
122     for (int i = 0; i < n; i++) {
123       vector<Halfplane> v;
124       v.push_back(Halfplane(__0, __1));
125       v.push_back(Halfplane(__1, __2));
126       v.push_back(Halfplane(__2, __3));
127       v.push_back(Halfplane(__3, __0));
128
129       for (int j = 0; j < n; j++)
130         if (i != j) {
131           Point a = (p[i] + p[j]) / 2;
132           Point b = a + (p[i] - p[j]).rev();
133
134           if (!Halfplane(a, b).satisfy(p[i])) {
135             swap(a, b);
136           }
137
138           v.push_back(Halfplane(a, b));
139         }
140
141       vector<Point> ans = halfplaneIntersection(v);
142
143       double ret = 0, low = 1e100;
144       int ansi = 0;
145
146       for (int j = 0; j < ans.size(); j++)
147         if (ans[j].z() < low) {
148           low = ans[j].z(), ansi = j;
149         }
150
151       for (int j = 0; j < ansi; j++) {
152         ans.push_back(ans[j]);
153       }
154
155       ret = area(ans, ansi) * low;
156
157       for (int j = ansi + 1; j + 1 < ans.size(); j++) {
158         double ll = (ans[j] - ans[j + 1]).len();
159
160         if (ll < eps) {
161           continue;
162         }
163
164         double s = (ans[j].z() + ans[j + 1].z() - low * 2) * ll / 2;
165         double h = ptol(ans[ansi], ans[j], ans[j + 1]);
166         ret += s * h / 3;
167       }
168
169       printf("%.6f\n", ret);
170     }
171   }
172
173   return 0;
```

```
174 }
```

4 DataStruct

4.1 lct

```
1  // Copyright [2017] <dmnsn7@gmail.com>
2
3  #include <bits/stdc++.h>
4
5  using std::max;
6  using std::swap;
7  using std::vector;
8
9  const int MAXN = 0;
10 const int INF = 0;
11
12 int ch[MAXN][2], pre[MAXN], key[MAXN];
13 int add[MAXN], Max[MAXN], rev[MAXN], n;
14 bool rt[MAXN];
15 void update_add(int r, int d) {
16   if (!r) {
17     return;
18   }
19
20   key[r] += d;
21   add[r] += d;
22   Max[r] += d;
23 }
24 void update_rev(int r) {
25   if (!r) {
26     return;
27   }
28
29   swap(ch[r][0], ch[r][1]);
30   rev[r] ^= 1;
31 }
32 void push_down(int r) {
33   if (add[r]) {
34     update_add(ch[r][0], add[r]);
35     update_add(ch[r][1], add[r]);
36     add[r] = 0;
37   }
38
39   if (rev[r]) {
40     update_rev(ch[r][0]);
41     update_rev(ch[r][1]);
42     rev[r] = 0;
43   }
44 }
45 void display() {
46   for (int i = 1; i <= n; i++) {
47     printf("%d %d %d %d <%d>  ", i, ch[i][0], ch[i][1], pre[i], rt[i]);
48     printf("%d %d %d\n", add[i], key[i], Max[i]);
49   }
50 }
51 void push_up(int r) { Max[r] = max(max(Max[ch[r][0]], Max[ch[r][1]]), key[r]); }
52 void rotate(int x) {
53   int y = pre[x], kind = ch[y][1] == x;
54   ch[y][kind] = ch[x][!kind];
55   pre[ch[y][kind]] = y;
56   pre[x] = pre[y];
57   pre[y] = x;
58   ch[x][!kind] = y;
59
60   if (rt[y]) {
61     rt[y] = 0, rt[x] = 1;
62   } else {
63     ch[pre[x]][ch[pre[x]][1] == y] = x;
```

```
64    }
65
66    push_up(y);
67  }
68  void P(int r) {
69    if (!rt[r]) {
70      P(pre[r]);
71    }
72
73    push_down(r);
74  }
75  void splay(int r) {
76    P(r);
77
78    while (!rt[r]) {
79      int f = pre[r], ff = pre[f];
80
81      if (rt[f]) {
82        rotate(r);
83      } else if ((ch[ff][1] == f) == (ch[f][1] == r)) {
84        rotate(f), rotate(r);
85      } else {
86        rotate(r), rotate(r);
87      }
88    }
89
90    push_up(r);
91  }
92  int access(int x) {
93    int y = 0;
94
95    for (; x; x = pre[y = x]) {
96      splay(x);
97      rt[ch[x][1]] = 1, rt[ch[x][1] = y] = 0;
98      push_up(x);
99    }
100
101   return y;
102 }
103 bool judge(int u, int v) {
104   while (pre[u]) {
105     u = pre[u];
106   }
107
108   while (pre[v]) {
109     v = pre[v];
110   }
111
112   return u == v;
113 }
114 void mroot(int r) {
115   access(r);
116   splay(r);
117   update_rev(r);
118 }
119 void lca(int u, int v) {
120   access(v), v = 0;
121
122   while (u) {
123     splay(u);
124
125     if (!pre[u]) {
126       return;
127     }
128
129     rt[ch[u][1]] = 1;
130     rt[ch[u][1] = v] = 0;
131     push_up(u);
```

```
132       u = pre[v = u];
133     }
134   }
135   void link(int u, int v) {
136     if (judge(u, v)) {
137       puts("-1");
138       return;
139     }
140
141     mroot(u);
142     pre[u] = v;
143   }
144   void cut(int u, int v) {
145     if (u == v || !judge(u, v)) {
146       puts("-1");
147       return;
148     }
149
150     mroot(u);
151     splay(v);
152     pre[ch[v][0]] = pre[v];
153     pre[v] = 0;
154     rt[ch[v][0]] = 1;
155     ch[v][0] = 0;
156     push_up(v);
157   }
158   void ADD(int u, int v, int w) {
159     if (!judge(u, v)) {
160       puts("-1");
161       return;
162     }
163
164     lca(u, v);
165     update_add(ch[u][1], w);
166     update_add(v, w);
167     key[u] += w;
168     push_up(u);
169   }
170   void query(int u, int v) {
171     if (!judge(u, v)) {
172       puts("-1");
173       return;
174     }
175
176     lca(u, v);
177     printf("%d\n", max(max(Max[v], Max[ch[u][1]]), key[u]));
178   }
179   vector<int> G[MAXN];
180   int que[MAXN];
181   void bfs() {
182     int front = 0, rear = 0;
183     que[rear++] = 1;
184     pre[1] = 0;
185
186     while (front < rear) {
187       int u = que[front++];
188
189       for (int i = 0; i < G[u].size(); i++) {
190         int v = G[u][i];
191
192         if (v == pre[u]) {
193           continue;
194         }
195
196         pre[v] = u;
197         que[rear++] = v;
198       }
199     }
```

```
}
int main() {
  int q, u, v;

  while (~scanf("%d", &n)) {
    memset(add, 0, sizeof add);
    memset(pre, 0, sizeof pre);
    memset(rev, 0, sizeof rev);
    memset(ch, 0, sizeof ch);

    for (int i = 0; i <= n; i++) {
      G[i].clear();
      rt[i] = 1;
    }

    Max[0] = -INF;

    for (int i = 1; i < n; i++) {
      scanf("%d%d", &u, &v);
      G[u].push_back(v);
      G[v].push_back(u);
    }

    for (int i = 1; i <= n; i++) {
      scanf("%d", &key[i]);
      Max[i] = key[i];
    }

    scanf("%d", &q);
    bfs();

    int op, x, y, w;

    while (q--) {
      scanf("%d", &op);

      if (op == 1) {
        scanf("%d%d", &x, &y);
        link(x, y);
      } else if (op == 2) {
        scanf("%d%d", &x, &y);
        cut(x, y);
      } else if (op == 3) {
        scanf("%d%d%d", &w, &x, &y);
        ADD(x, y, w);
      } else {
        scanf("%d%d", &x, &y);
        query(x, y);
      }
    }

    puts("");
  }

  return 0;
}
```

4.2 kdt

```
// Copyright [2017] <dmnsn7@gmail.com>

#include <bits/stdc++.h>

using std::min;
using std::min_element;
using std::max_element;
using std::nth_element;
using std::swap;
using std::vector;

```

```
12 struct Node {
13   int64_t x, y;
14 };
15
16 vector<Node>::iterator p;
17 vector<bool> d;
18
19 bool cmpx(const Node &a, const Node &b) { return a.x < b.x; }
20 bool cmpy(const Node &a, const Node &b) { return a.y < b.y; }
21
22 int64_t sqr(int64_t a) { return a * a; }
23
24 int64_t dis(const Node &a, const Node &b) {
25   return sqr(a.x - b.x) + sqr(a.y - b.y);
26 }
27
28 void build(int l, int r) {
29   if (l > r) {
30     return;
31   }
32
33   int64_t minx = min_element(p + l, p + r + 1, cmpx)->x;
34   int64_t maxx = max_element(p + l, p + r + 1, cmpx)->x;
35   int64_t miny = min_element(p + l, p + r + 1, cmpy)->y;
36   int64_t maxy = max_element(p + l, p + r + 1, cmpy)->y;
37   int mid = l + (r - l) / 2;
38   d[mid] = maxx - minx > maxy - miny;
39   nth_element(p + l, p + mid, p + r + 1, d[mid] ? cmpx : cmpy);
40
41   build(l, mid - 1);
42   build(mid + 1, r);
43 }
44
45 void query(int l, int r, const Node &a) {
46   if (l > r) {
47     return;
48   }
49
50   int mid = l + (r - l) / 2;
51   int64_t dist = dis(a, p[mid]), res;
52   int64_t d1 = d[mid] ? a.x - p[mid].x : a.y - p[mid].y;
53
54   if (dist > 0) {
55     res = min(res, dist);
56   }
57
58   int l1 = l, r1 = mid - 1;
59   int l2 = mid + 1, r2 = r;
60
61   if (d1 > 0) {
62     swap(l1, l2);
63     swap(r1, r2);
64   }
65
66   query(l1, r1, a);
67
68   if (d1 * d1 < res) {
69     query(l2, r2, a);
70   }
71 }
72
73 int main() { return 0; }
```

5 Graph

5.1 targan point connecting

```
1 // Copyright [2017] <dmnsn7@gmail.com>
2
```

```
3  #include <bits/stdc++.h>
4
5  const int MAXN = 0;
6
7  struct EDGE {
8    int to, next;
9  };
10 EDGE edge[MAXN];
11
12 int top = 0, Index = 0;
13 int head[MAXN], Instack[MAXN], Low[MAXN], Stack[MAXN], DFN[MAXN];
14
15 void Tarjan(int u, int pre) {
16   Low[u] = DFN[u] = ++Index;
17   Stack[top++] = u;
18   Instack[u] = true;
19
20   for (int i = head[u]; i != -1; i = edge[i].next) {
21     int v = edge[i].to;
22
23     if (v == pre) {
24       continue;    //
25     }
26
27     if (!DFN[v]) {
28       Tarjan(v, u);
29
30       if (Low[u] > Low[v]) {
31         Low[u] = Low[v];
32       }
33       /*
34       if (Low[v] >= DFN[u]) {
35         block++;
36         int vn;
37         cc = 0;
38         memset(ok, false, sizeof(ok));
39
40         do {
41           vn = Stack[--top];
42           Belong[vn] = block;
43           Instack[vn] = false;
44           ok[vn] = true;
45           tmp[cc++] = vn;
46         } while (vn != v);
47
48         ok[u] = 1;
49         memset(color, -1, sizeof(color));
50
51         if (!dfs(u, 0)) {
52           can[u] = true;
53
54           while (cc--) {
55             can[tmp[cc]] = true;
56           }
57         }
58       }
59       */
60     } else if (Instack[v] && Low[u] > DFN[v]) {
61       Low[u] = DFN[v];
62     }
63   }
64
65   /*  targan
66   if (Low[u] == DFN[u]) {
67     scc++;
68
69     do {
70       v = Stack[--top];m
71       Instack[v] = false;
```

```
72        Belong[v] = scc;
73        num[scc]++;
74      } while (v != u);
75    }
76    */
77  }
78
79  int main() { return 0; }
```

5.2 cut point bridge

```
1   // Copyright [2017] <dmnsn7@gmail.com>
2
3   #include <bits/stdc++.h>
4
5   const int MAXN = 10010;
6   const int MAXM = 100010;
7   struct Edge {
8     int to, next;
9     bool cut;  //
10  } edge[MAXM];
11  int head[MAXN], tot;
12  int Low[MAXN], DFN[MAXN], Stack[MAXN];
13  int Index, top;
14  bool Instack[MAXN];
15  bool cut[MAXN];
16  int add_block[MAXN];  //
17  int bridge;
18  void addedge(int u, int v) {
19    edge[tot].to = v;
20    edge[tot].next = head[u];
21    edge[tot].cut = false;
22    head[u] = tot++;
23  }
24  void Tarjan(int u, int pre) {
25    Low[u] = DFN[u] = ++Index;
26    Stack[top++] = u;
27    Instack[u] = true;
28    int son = 0;
29
30    for (int i = head[u]; i != -1; i = edge[i].next) {
31      int v = edge[i].to;
32
33      if (v == pre) {
34        continue;
35      }
36
37      if (!DFN[v]) {
38        son++;
39        Tarjan(v, u);
40
41        if (Low[u] > Low[v]) {
42          Low[u] = Low[v];
43        }
44
45        if (Low[v] > DFN[u]) {  //
46          bridge++;
47          edge[i].cut = true;
48          edge[i ^ 1].cut = true;
49        }
50
51        if (u != pre && Low[v] >= DFN[u]) {  //
52          cut[u] = true;
53          add_block[u]++;
54        }
55      } else if (Low[u] > DFN[v]) {
56        Low[u] = DFN[v];
57      }
58
```

```
59      /*
60       * else if( Instack[v] && Low[u] > DFN[v] )
61       *     Low[u] = DFN[v];
62       * }
63       * if(Low[u] == DFN[u]){
64       *     block++;
65       *     do
66       *     {
67       *         v = Stack[--top];
68       *         Instack[v] = false;
69       *         Belong[v] = block;
70       *     }while( v!=u );
71       * }
72       */
73    }
74
75    if (u == pre && son > 1) {
76      cut[u] = true;  //     ,1
77    }
78
79    if (u == pre) {
80      add_block[u] = son - 1;
81    }
82
83    Instack[u] = false;
84    top--;
85  }
86  void solve(int N) {
87    memset(DFN, 0, sizeof(DFN));
88    memset(Instack, false, sizeof(Instack));
89    memset(add_block, 0, sizeof(add_block));
90    memset(cut, false, sizeof(cut));
91    Index = top = 0;
92    bridge = 0;
93
94    for (int i = 1; i <= N; i++)
95      if (!DFN[i]) {
96        Tarjan(i, i);
97      }
98
99    printf("%d critical links\n", bridge);
100 }
101 void init() {
102   tot = 0;
103   memset(head, -1, sizeof(head));
104 }
105
106 int main() {
107   return 0;
108 }
```

5.3 hungary

```
1  // Copyright [2017] <dmnsn7@gmail.com>
2
3  #include <bits/stdc++.h>
4
5  const int MAXN = 0;
6
7  int uN;
8  int used[MAXN], head[MAXN], linker[MAXN];
9
10 struct EDGE {
11   int next, to;
12 };
13 EDGE edge[MAXN];
14
15 bool dfs(int u) {
16   for (int i = head[u]; i != -1; i = edge[i].next) {
```

```
17      int v = edge[i].to;
18
19      if (!used[v]) {
20        used[v] = true;
21
22        if (linker[v] == -1 || dfs(linker[v])) {
23          linker[v] = u;
24          return true;
25        }
26      }
27    }
28
29    return false;
30  }
31
32  int res = 0;
33  int hungary() {
34    memset(linker, -1, sizeof(linker));
35
36    for (int u = 0; u < uN; u++) {  //   0~uN-1
37      memset(used, false, sizeof(used));
38
39      if (dfs(u)) {
40        res++;
41      }
42    }
43
44    return res;
45  }
46
47  int main() { return 0; }
```

5.4 maxflow

```
1   // Copyright [2017] <dmnsn7@gmail.com>
2
3   #include <bits/stdc++.h>
4
5   const int MAXN = 100010;  //
6   const int MAXM = 400010;  //
7   const int oo = 0x3f3f3f3f;
8   struct Edge {
9     int to, next, cap, flow;
10  } edge[MAXM];  //  MAXM
11  int tol;
12  int head[MAXN];
13  int gap[MAXN], dep[MAXN], cur[MAXN];
14  void init() {
15    tol = 0;
16    memset(head, -1, sizeof(head));
17  }
18  void addedge(int u, int v, int w, int rw = 0) {
19    edge[tol].to = v;
20    edge[tol].cap = w;
21    edge[tol].flow = 0;
22    edge[tol].next = head[u];
23    head[u] = tol++;
24    edge[tol].to = u;
25    edge[tol].cap = rw;
26    edge[tol].flow = 0;
27    edge[tol].next = head[v];
28    head[v] = tol++;
29  }
30  int Q[MAXN];
31  void BFS(int ss, int tt) {
32    memset(dep, -1, sizeof(dep));
33    memset(gap, 0, sizeof(gap));
34    gap[0] = 1;
35    int front = 0, rear = 0;
```

```
36    dep[tt] = 0;
37    Q[rear++] = tt;
38
39    while (front != rear) {
40      int u = Q[front++];
41
42      for (int i = head[u]; i != -1; i = edge[i].next) {
43        int v = edge[i].to;
44
45        if (dep[v] != -1) {
46          continue;
47        }
48
49        Q[rear++] = v;
50        dep[v] = dep[u] + 1;
51        gap[dep[v]]++;
52      }
53    }
54  }
55  int S[MAXN];
56  int sap(int ss, int tt, int N) {
57    BFS(ss, tt);
58    memcpy(cur, head, sizeof(head));
59    int top = 0;
60    int u = ss;
61    int ans = 0;
62
63    while (dep[ss] < N) {
64      if (u == tt) {
65        int mi = oo;
66        int inser;
67
68        for (int i = 0; i < top; i++)
69          if (mi > edge[S[i]].cap - edge[S[i]].flow) {
70            mi = edge[S[i]].cap - edge[S[i]].flow;
71            inser = i;
72          }
73
74        for (int i = 0; i < top; i++) {
75          edge[S[i]].flow += mi;
76          edge[S[i] ^ 1].flow -= mi;
77        }
78
79        ans += mi;
80        top = inser;
81        u = edge[S[top] ^ 1].to;
82        continue;
83      }
84
85      bool flag = false;
86      int v;
87
88      for (int i = cur[u]; i != -1; i = edge[i].next) {
89        v = edge[i].to;
90
91        if (edge[i].cap - edge[i].flow && dep[v] + 1 == dep[u]) {
92          flag = true;
93          cur[u] = i;
94          break;
95        }
96      }
97
98      if (flag) {
99        S[top++] = cur[u];
100       u = v;
101       continue;
102     }
103
104     int mi = N;
105
```

```
106       for (int i = head[u]; i != -1; i = edge[i].next)
107         if (edge[i].cap - edge[i].flow && dep[edge[i].to] < mi) {
108           mi = dep[edge[i].to];
109           cur[u] = i;
110         }
111
112       gap[dep[u]]--;
113
114       if (!gap[dep[u]]) {
115         return ans;
116       }
117
118       dep[u] = mi + 1;
119       gap[dep[u]]++;
120
121       if (u != ss) {
122         u = edge[S[--top] ^ 1].to;
123       }
124     }
125
126     return ans;
127 }
128
129 int main() {
130     return 0;
131 }
```

5.5 costflow

```
1  // Copyright [2017] <dmnsn7@gmail.com>
2
3  #include <bits/stdc++.h>
4
5  using std::queue;
6
7  const int MAXN = 10000;
8  const int MAXM = 100000;
9  const int INF = 0x3f3f3f3f;
10 struct Edge {
11   int to, next, cap, flow, cost;
12 } edge[MAXM];
13 int head[MAXN], tol;
14 int pre[MAXN], dis[MAXN];
15 bool vis[MAXN];
16 int N;  //      ,0~N-1
17 void init(int n) {
18   N = n;
19   tol = 0;
20   memset(head, -1, sizeof(head));
21 }
22 void addedge(int u, int v, int cap, int cost) {
23   edge[tol].to = v;
24   edge[tol].cap = cap;
25   edge[tol].cost = cost;
26   edge[tol].flow = 0;
27   edge[tol].next = head[u];
28   head[u] = tol++;
29   edge[tol].to = u;
30   edge[tol].cap = 0;
31   edge[tol].cost = -cost;
32   edge[tol].flow = 0;
33   edge[tol].next = head[v];
34   head[v] = tol++;
35 }
36
37 queue<int> q;
38 bool spfa(int s, int t) {
39   queue<int> q;
40
41   for (int i = 0; i < N; i++) {
```

```
42      dis[i] = INF;
43      vis[i] = false;
44      pre[i] = -1;
45    }
46
47    dis[s] = 0;
48    vis[s] = true;
49    q.push(s);
50
51    while (!q.empty()) {
52      int u = q.front();
53      q.pop();
54      vis[u] = false;
55
56      for (int i = head[u]; i != -1; i = edge[i].next) {
57        int v = edge[i].to;
58
59        if (edge[i].cap > edge[i].flow && dis[v] > dis[u] + edge[i].cost) {
60          dis[v] = dis[u] + edge[i].cost;
61          pre[v] = i;
62
63          if (!vis[v]) {
64            vis[v] = true;
65            q.push(v);
66          }
67        }
68      }
69    }
70
71    if (pre[t] == -1) {
72      return false;
73    } else {
74      return true;
75    }
76  }
77  //      ,    cost
78  int minCostMaxflow(int s, int t, int cost) {
79    int flow = 0;
80    cost = 0;
81
82    while (spfa(s, t)) {
83      int Min = INF;
84
85      for (int i = pre[t]; i != -1; i = pre[edge[i ^ 1].to]) {
86        if (Min > edge[i].cap - edge[i].flow) {
87          Min = edge[i].cap - edge[i].flow;
88        }
89      }
90
91      for (int i = pre[t]; i != -1; i = pre[edge[i ^ 1].to]) {
92        edge[i].flow += Min;
93        edge[i ^ 1].flow -= Min;
94        cost += edge[i].cost * Min;
95      }
96
97      flow += Min;
98    }
99
100   return flow;
101 }
102
103 int main() {
104   return 0;
105 }
```

5.6 min tree graph

```
1  // Copyright [2017] <dmnsn7@gmail.com>
2
```

```
3  #include <bits/stdc++.h>
4
5  using std::swap;
6
7  const int INF = 0x3f3f3f3f;
8  const int MAXN = 1010;
9  const int MAXM = 40010;
10 struct Edge {
11   int u, v, cost;
12 };
13 Edge edge[MAXM];
14 int pre[MAXN], id[MAXN], visit[MAXN], in[MAXN];
15 int zhuliu(int root, int n, int m, Edge edge[]) {
16   int res = 0, u, v;
17
18   while (1) {
19     for (int i = 0; i < n; i++) {
20       in[i] = INF;
21     }
22
23     for (int i = 0; i < m; i++)
24       if (edge[i].u != edge[i].v && edge[i].cost < in[edge[i].v]) {
25         pre[edge[i].v] = edge[i].u;
26         in[edge[i].v] = edge[i].cost;
27       }
28
29     for (int i = 0; i < n; i++)
30       if (i != root && in[i] == INF) {
31         return -1;      //
32       }
33
34     int tn = 0;
35     memset(id, -1, sizeof(id));
36     memset(visit, -1, sizeof(visit));
37     in[root] = 0;
38
39     for (int i = 0; i < n; i++) {
40       res += in[i];
41       v = i;
42
43       while (visit[v] != i && id[v] == -1 && v != root) {
44         visit[v] = i;
45         v = pre[v];
46       }
47
48       if (v != root && id[v] == -1) {
49         for (int u = pre[v]; u != v; u = pre[u]) {
50           id[u] = tn;
51         }
52
53         id[v] = tn++;
54       }
55     }
56
57     if (tn == 0) {
58       break;     //
59     }
60
61     for (int i = 0; i < n; i++)
62       if (id[i] == -1) {
63         id[i] = tn++;
64       }
65
66     for (int i = 0; i < m;) {
67       v = edge[i].v;
68       edge[i].u = id[edge[i].u];
69       edge[i].v = id[edge[i].v];
70
71       if (edge[i].u != edge[i].v) {
```

```
72          edge[i++].cost -= in[v];
73        } else {
74          swap(edge[i], edge[--m]);
75        }
76      }
77
78      n = tn;
79      root = id[root];
80    }
81
82    return res;
83  }
84
85  int main() { return 0; }
```

5.7 flowertree

```
1   // Copyright [2017] <dmnsn7@gmail.com>
2
3   #include <bits/stdc++.h>
4
5   const int MAXN = 250;
6   int N;  //       ,1N
7   bool Graph[MAXN][MAXN];
8   int Match[MAXN];
9   bool InQueue[MAXN], InPath[MAXN], InBlossom[MAXN];
10  int Head, Tail;
11  int Queue[MAXN];
12  int Start, Finish;
13  int NewBase;
14  int Father[MAXN], Base[MAXN];
15  int Count;  //      ,Count/2
16  void CreateGraph() {
17    int u, v;
18    memset(Graph, false, sizeof(Graph));
19    scanf("%d", &N);
20
21    while (scanf("%d%d", &u, &v) == 2) {
22      Graph[u][v] = Graph[v][u] = true;
23    }
24  }
25  void Push(int u) {
26    Queue[Tail] = u;
27    Tail++;
28    InQueue[u] = true;
29  }
30  int Pop() {
31    int res = Queue[Head];
32    Head++;
33    return res;
34  }
35  int FindCommonAncestor(int u, int v) {
36    memset(InPath, false, sizeof(InPath));
37
38    while (true) {
39      u = Base[u];
40      InPath[u] = true;
41
42      if (u == Start) {
43        break;
44      }
45
46      u = Father[Match[u]];
47    }
48
49    while (true) {
50      v = Base[v];
51
52      if (InPath[v]) {
53        break;
```

```
54      }
55
56      v = Father[Match[v]];
57    }
58
59    return v;
60  }
61  void ResetTrace(int u) {
62    while (Base[u] != NewBase) {
63      int v = Match[u];
64      InBlossom[Base[u]] = InBlossom[Base[v]] = true;
65      u = Father[v];
66
67      if (Base[u] != NewBase) {
68        Father[u] = v;
69      }
70    }
71  }
72  void BloosomContract(int u, int v) {
73    NewBase = FindCommonAncestor(u, v);
74    memset(InBlossom, false, sizeof(InBlossom));
75    ResetTrace(u);
76    ResetTrace(v);
77
78    if (Base[u] != NewBase) {
79      Father[u] = v;
80    }
81
82    if (Base[v] != NewBase) {
83      Father[v] = u;
84    }
85
86    for (int tu = 1; tu <= N; tu++)
87      if (InBlossom[Base[tu]]) {
88        Base[tu] = NewBase;
89
90        if (!InQueue[tu]) {
91          Push(tu);
92        }
93      }
94  }
95
96  void FindAugmentingPath() {
97    memset(InQueue, false, sizeof(InQueue));
98    memset(Father, 0, sizeof(Father));
99
100   for (int i = 1; i <= N; i++) {
101     Base[i] = i;
102   }
103
104   Head = Tail = 1;
105   Push(Start);
106   Finish = 0;
107
108   while (Head < Tail) {
109     int u = Pop();
110
111     for (int v = 1; v <= N; v++)
112       if (Graph[u][v] && (Base[u] != Base[v]) && (Match[u] != v)) {
113         if ((v == Start) || ((Match[v] > 0) && Father[Match[v]] > 0)) {
114           BloosomContract(u, v);
115         } else if (Father[v] == 0) {
116           Father[v] = u;
117
118           if (Match[v] > 0) {
119             Push(Match[v]);
120           } else {
121             Finish = v;
```

```
122                 return;
123               }
124             }
125           }
126     }
127   }
128   void AugmentPath() {
129     int u = Finish;
130
131     while (u > 0) {
132       int v = Father[u];
133       int w = Match[v];
134       Match[v] = u;
135       Match[u] = v;
136       u = w;
137     }
138   }
139   void Edmonds() {
140     memset(Match, 0, sizeof(Match));
141
142     for (int u = 1; u <= N; u++)
143       if (Match[u] == 0) {
144         Start = u;
145         FindAugmentingPath();
146
147         if (Finish > 0) {
148           AugmentPath();
149         }
150       }
151   }
152   void PrintMatch() {
153     Count = 0;
154
155     for (int u = 1; u <= N; u++)
156       if (Match[u] > 0) {
157         Count++;
158       }
159
160     printf("%d\n", Count);
161
162     for (int u = 1; u <= N; u++)
163       if (u < Match[u]) {
164         printf("%d %d\n", u, Match[u]);
165       }
166   }
167   int main() {
168     CreateGraph();  //
169     Edmonds();      //
170     PrintMatch();   //
171     return 0;
172   }
```

5.8 2-sat

```
1  // Copyright [2017] <dmnsn7@gmail.com>
2
3  #include <bits/stdc++.h>
4
5  const int MAXN = 20020;
6  const int MAXM = 100010;
7  struct Edge {
8    int to, next;
9  } edge[MAXM];
10 int head[MAXN], tot;
11 void init() {
12   tot = 0;
13   memset(head, -1, sizeof(head));
14 }
15 void addedge(int u, int v) {
```

```
16    edge[tot].to = v;
17    edge[tot].next = head[u];
18    head[u] = tot++;
19  }
20  bool vis[MAXN];     //     ,  true
21  int S[MAXN], top;   //
22  bool dfs(int u) {
23    if (vis[u ^ 1]) {
24      return false;
25    }
26
27    if (vis[u]) {
28      return true;
29    }
30
31    vis[u] = true;
32    S[top++] = u;
33
34    for (int i = head[u]; i != -1; i = edge[i].next)
35      if (!dfs(edge[i].to)) {
36        return false;
37      }
38
39    return true;
40  }
41  bool Twosat(int n) {
42    memset(vis, false, sizeof(vis));
43
44    for (int i = 0; i < n; i += 2) {
45      if (vis[i] || vis[i ^ 1]) {
46        continue;
47      }
48
49      top = 0;
50
51      if (!dfs(i)) {
52        while (top) {
53          vis[S[--top]] = false;
54        }
55
56        if (!dfs(i ^ 1)) {
57          return false;
58        }
59      }
60    }
61
62    return true;
63  }
64  int main() {
65    int n, m;
66    int u, v;
67
68    while (scanf("%d%d", &n, &m) == 2) {
69      init();
70
71      while (m--) {
72        scanf("%d%d", &u, &v);
73        u--;
74        v--;
75        addedge(u, v ^ 1);
76        addedge(v, u ^ 1);
77      }
78
79      if (Twosat(2 * n)) {
80        for (int i = 0; i < 2 * n; i++)
81          if (vis[i]) {
82            printf("%d\n", i + 1);
83          }
84      } else {
```

```
85        printf("NIE\n");
86      }
87    }
88
89    return 0;
90  }
```

5.9 km

```
1   // Copyright [2017] <dmnsn7@gmail.com>
2
3   #include <bits/stdc++.h>
4
5   const int MAXN = 0;
6   const int INF = 0;
7
8   int ny, nx;
9   int g[MAXN][MAXN], slack[MAXN], linker[MAXN], lx[MAXN], ly[MAXN], visx[MAXN],
10      visy[MAXN];
11
12  bool DFS(int x) {
13    visx[x] = true;
14
15    for (int y = 0; y < ny; y++) {
16      if (visy[y]) {
17        continue;
18      }
19
20      int tmp = lx[x] + ly[y] - g[x][y];
21
22      if (tmp == 0) {
23        visy[y] = true;
24
25        if (linker[y] == -1 || DFS(linker[y])) {
26          linker[y] = x;
27          return true;
28        }
29      } else if (slack[y] > tmp) {
30        slack[y] = tmp;
31      }
32    }
33
34    return false;
35  }
36  int KM() {
37    memset(linker, -1, sizeof(linker));
38    memset(ly, 0, sizeof(ly));
39
40    for (int i = 0; i < nx; i++) {
41      lx[i] = -INF;
42
43      for (int j = 0; j < ny; j++)
44        if (g[i][j] > lx[i]) {
45          lx[i] = g[i][j];
46        }
47    }
48
49    for (int x = 0; x < nx; x++) {
50      for (int i = 0; i < ny; i++) {
51        slack[i] = INF;
52      }
53
54      while (true) {
55        memset(visx, false, sizeof(visx));
56        memset(visy, false, sizeof(visy));
57
58        if (DFS(x)) {
59          break;
60        }
61
```

```
62        int d = INF;
63
64        for (int i = 0; i < ny; i++)
65          if (!visy[i] && d > slack[i]) {
66            d = slack[i];
67          }
68
69        for (int i = 0; i < nx; i++)
70          if (visx[i]) {
71            lx[i] -= d;
72          }
73
74        for (int i = 0; i < ny; i++) {
75          if (visy[i]) {
76            ly[i] += d;
77          } else {
78            slack[i] -= d;
79          }
80        }
81      }
82    }
83
84    int res = 0;
85
86    for (int i = 0; i < ny; i++)
87      if (linker[i] != -1) {
88        res += g[linker[i]][i];
89      }
90
91    return res;
92  }
93
94  int main() { return 0; }
```