

Code Library for Lmxyy

Lmxyy

Shanghai Jiao Tong University

2017 年 10 月 29 日

目录

1	Algorithms	3
1.1	1D1D Dynamic Programming	3
1.2	Dynamic Minimal Spanning Tree	5
1.3	Plug-like Dynamic Programming	8
1.4	Slop Optimization	10
1.5	Three-dimension Partial Order	12
2	Computational Geometry	14
2.1	Circle Intersection	14
2.2	Common Formulas	18
2.3	Convex Hull	19
2.4	Cross Points of Circles	21
2.5	Cross Points of Line and Circle	21
2.6	Graham Scanning Algorithm	21
2.7	Half Plane Intersection	24
2.8	Intersecting Area of Circle and Polygon	25
2.9	Intersection of Line and Convex Hull	27
2.10	Minimal Product	28
2.11	Planar Graph	30
2.12	Polygon Class	36
2.13	Union Area of Circles	37
3	Data Structure	41
3.1	Divide and Conquer on Tree	41
3.2	Dynamicly Divide and Conquer on Tree	42
3.3	Heavy Path Decomposition	46
3.4	K-Dimension Tree	48
3.5	Leftlist Tree	51
3.6	Link Cut Tree	52
3.7	Merge Split Treap	56
3.8	Modui Algorithm on Tree	62
3.9	Modui Algorithm without Deletion	64
3.10	President Tree	67
3.11	Splay	67

4	Graph Theory	75
4.1	2-Sat	75
4.2	Bridges and Cut Vertices	76
4.3	Cost Flow	77
4.4	Difference Constraints	78
4.5	Dinic Algorithm	80
4.6	Dominator Tree	81
4.7	Hungary	85
4.8	Isap Algorithm	86
4.9	Kuhn-Munkres Algorithm	91
4.10	Maximal Matching in General Graphs	93
4.11	Maximal Weighted Matching in General Graphs	95
4.12	Maximum Cardinality Search	97
4.13	Network Flow with Lower Bound	99
4.14	Point Biconnected Component	99
4.15	Steiner Tree	102
4.16	Stoer Wagner Algorithm	103
4.17	Strongly Connected Component	104
4.18	Virtual Tree	105
4.19	Zhu-Liu Algorithm	106
4.20	ZKW Cost Flow	108
5	Number Theory	110
5.1	Baby Step Giant Step	110
5.2	Chinese Remainder Theorem	110
5.3	Extended Euclidean Algorithm	111
5.4	Linearly Sieve	112
5.5	N-Power Residue	112
5.6	Number Theoretic Transformation	113
5.7	Pollard Rho Algorithm	116
5.8	Primitive Root	118
5.9	Quadratic Residue	119
5.10	Single Variable Modulus Linear Equation	119
6	Numerical Algorithms	121
6.1	Counting Integral Points under Straight Line	121
6.2	Evaluation of Expression	121
6.3	Fast Fourier Transformation	123
6.4	Fast Input and Output	125
6.5	Fraction Class	126
6.6	Gray Code	126
6.7	Numerical Integration	126
6.8	Simplex	127

6.8.1	Description	127
6.8.2	Input	127
6.8.3	Output	127
6.8.4	Code	128
6.9	Solutions of Equation of Higher Order	129
7	String Algorithms	131
7.1	Aho-Corasick Automaton	131
7.2	Knuth-Morris-Pratt Algorithm	132
7.3	Manacher Algorithm	132
7.4	Palindrome Automaton	133
7.5	Suffix Array	134
7.6	Suffix Automaton	134
8	Others	137
8.1	Calculation of Date	137
8.2	Java Hints	138
8.3	Emacs Configuration-Full	142
8.4	Emacs Configuration-Competition	145

Chapter 1

Algorithms

1.1 1D1D Dynamic Programming

```
1 // noi2009 诗人小 G
2 #include<cstring>
3 #include<cstdio>
4 #include<cstdlib>
5 using namespace std;
6
7 #define limit (1e18)
8 #define maxn 100010
9 #define maxm 40
10 int N,L,P,pre[maxn],top;
11 char s[maxm];
12 long double f[maxn];
13 struct node { int l,r,key; }stack[maxn];
14
15 inline long double qsm(int a,int b)
16 {
17     long double ret = 1;
18     while (b--) ret *= 1.0*a;
19     return ret;
20 }
21
22 inline long double calc(int a,int b)
23 {
24     return f[b]+qsm(abs(pre[a]-pre[b]-L),P);
25 }
26
27 inline int find(int a)
28 {
29     int l = 1,r = top,mid;
30     while (l <= r)
31     {
32         mid = (l + r) >> 1;
33         if (stack[mid].l<=a&&stack[mid].r>=a) return stack[mid].key;
```

```

34         if (a < stack[mid].l) r = mid - 1;
35         else l = mid + 1;
36     }
37 }
38
39 inline void updata(int now)
40 {
41     int l = 1, r;
42     while (top)
43     {
44         if (calc(stack[top].l, stack[top].key) >= calc(stack[top].l, now))
45             --top;
46         else
47         {
48             l = stack[top].l, r = stack[top].r;
49             while (l <= r)
50             {
51                 int mid = (l + r) >> 1;
52                 if (calc(mid, stack[top].key) >= calc(mid, now)) r = mid - 1;
53                 else l = mid + 1;
54             }
55             stack[top].r = r;
56             break;
57         }
58     }
59     if (l <= N) stack[++top] = (node){l, N, now};
60 }
61
62 inline void dp()
63 {
64     f[0] = 0;
65     stack[top = 1] = (node) {1, N, 0};
66     for (int i = 1; i <= N; ++i)
67     {
68         int key = find(i);
69         f[i] = calc(i, key);
70         updata(i);
71     }
72 }
73
74 int main()
75 {
76     freopen("1563.in", "r", stdin);
77     freopen("1563.out", "w", stdout);
78     int T; scanf("%d", &T);
79     while (T--)
80     {
81         scanf("%d %d %d\n", &N, &L, &P);
82         L++;

```

```

83     for (int i = 1; i <= N; ++i)
84     {
85         scanf("%s", s);
86         pre[i] = strlen(s) + 1 + pre[i - 1];
87     }
88     dp();
89     if (f[N] > limit) printf("Too hard to arrange\n");
90     else printf("%.0Lf\n", f[N]);
91     printf("-----\n");
92 }
93 fclose(stdin); fclose(stdout);
94 return 0;
95 }

```

1.2 Dynamic Minimal Spanning Tree

```

1  // 每次修改一条边，每次修改一条边权值，求最小生成树
2  #include<algorithm>
3  #include<cstring>
4  #include<vector>
5  #include<iostream>
6  #include<cstdio>
7  #include<stdlib.h>
8  using namespace std;
9
10 typedef long long ll;
11 const int maxn = 100010; const ll inf = 1LL<<40;
12 int N, M, Q, father[maxn], cnt[maxn], reid[maxn]; ll ans[maxn];
13
14 inline int find(int a) { if (father[a] != a) return father[a] = find(father[a]); return
    ↪ father[a]; }
15
16 inline int gi()
17 {
18     char ch; int ret = 0, f = 1;
19     do ch = getchar(); while (!(ch >= '0' && ch <= '9') && ch != '-');
20     if (ch == '-') f = -1, ch = getchar();
21     do ret = ret * 10 + ch - '0', ch = getchar(); while (ch >= '0' && ch <= '9');
22     return ret * f;
23 }
24
25 struct Edge
26 {
27     int a, b, id; ll c;
28     inline Edge() = default;
29     inline Edge(int _a, int _b, int _id, ll _c): a(_a), b(_b), id(_id), c(_c) {}
30     inline void read(int i) { a = gi(), b = gi(), c = gi(); id = i; }
31     friend inline bool operator <(const Edge &x, const Edge &y) { return x.c < y.c; }
32 } edge[22][maxn], tmp[maxn], bac[maxn];

```

```

33
34 struct Operation
35 {
36     int x; ll y;
37     inline Operation() = default;
38     inline Operation(int _x,ll _y):x(_x),y(_y) {}
39     inline void read() { x = gi(),y = gi(); }
40 }opt[maxn];
41
42 inline void construct(int &tot,ll &sum)
43 {
44     sort(tmp+1,tmp+tot+1);
45     for (int i = 1;i <= tot;++i)
46         father[tmp[i].a] = tmp[i].a,father[tmp[i].b] = tmp[i].b;
47     vector <Edge> vec;
48     for (int i = 1;i <= tot;++i)
49     {
50         int u = find(tmp[i].a),v = find(tmp[i].b);
51         if (u != v) father[u] = v,vec.push_back(tmp[i]);
52     }
53     for (int i = 0;i < (int)vec.size();++i)
54         father[vec[i].a] = vec[i].a,father[vec[i].b] = vec[i].b;
55     for (int i = 0;i < (int)vec.size();++i)
56     {
57         Edge e = vec[i];
58         if (e.c != -inf) father[find(e.a)] = find(e.b),sum += e.c;
59     }
60     vec.clear();
61     for (int i = 1;i <= tot;++i)
62     {
63         int u = find(tmp[i].a),v = find(tmp[i].b);
64         if (u != v)
65         {
66             tmp[i].a = u,tmp[i].b = v;
67             vec.push_back(tmp[i]);
68         }
69     }
70     for (int i = 0;i < (int)vec.size();++i) tmp[i+1] = vec[i];
71     for (int i = 1;i <= tot;++i) reid[tmp[i].id] = i;
72     tot = (int)vec.size();
73 }
74
75 inline void destruct(int &tot)
76 {
77     sort(tmp+1,tmp+tot+1);
78     for (int i = 1;i <= tot;++i)
79         father[tmp[i].a] = tmp[i].a,father[tmp[i].b] = tmp[i].b;
80     vector <Edge> vec;
81     for (int i = 1;i <= tot;++i)

```



```

82     {
83         int u = find(tmp[i].a), v = find(tmp[i].b);
84         if (u != v) father[u] = v, vec.push_back(tmp[i]);
85         else if (tmp[i].c == inf) vec.push_back(tmp[i]);
86     }
87     for (int i = 0; i < (int)vec.size(); ++i) tmp[i+1] = vec[i];
88     tot = (int)vec.size();
89 }
90
91 inline void work(int l, int r, int dep, ll sum)
92 {
93     int tot = cnt[dep];
94     for (int i = 1; i <= tot; ++i) tmp[i] = edge[dep][i];
95     if (l == r)
96     {
97         bac[opt[l].x].c = opt[l].y;
98         for (int i = 1; i <= tot; ++i)
99         {
100             tmp[i].c = bac[tmp[i].id].c;
101             father[tmp[i].a] = tmp[i].a;
102             father[tmp[i].b] = tmp[i].b;
103         }
104         sort(tmp+1, tmp+tot+1);
105         for (int i = 1; i <= tot; ++i)
106         {
107             int u = find(tmp[i].a), v = find(tmp[i].b);
108             if (u != v) sum += tmp[i].c, father[u] = v;
109         }
110         ans[l] = sum; return;
111     }
112     for (int i = 1; i <= tot; ++i)
113         tmp[i].c = bac[tmp[i].id].c, reid[tmp[i].id] = i;
114     for (int i = 1; i <= r; ++i) tmp[reid[opt[i].x]].c = -inf;
115     construct(tot, sum);
116     for (int i = 1; i <= r; ++i)
117         tmp[reid[opt[i].x]].c = inf;
118     destruct(tot);
119     for (int i = 1; i <= tot; ++i) edge[dep+1][i] = tmp[i];
120     int mid = (l+r)>>1; cnt[dep+1] = tot;
121     work(l, mid, dep+1, sum); work(mid+1, r, dep+1, sum);
122 }
123
124 int main()
125 {
126     // freopen("B.in", "r", stdin);
127     N = gi(), M = gi(), Q = gi();
128     for (int i = 1; i <= M; ++i) bac[i].read(i), edge[0][i] = bac[i];
129     for (int i = 1; i <= Q; ++i) opt[i].read();
130     for (int i = 1; i <= N; ++i) father[i] = i;

```

```

131     cnt[0] = M; work(1,Q,0,0);
132     for (int i = 1;i <= Q;++i) printf("%lld\n",ans[i]);
133     return 0;
134 }

```

1.3 Plug-like Dynamic Programming

```

1  // ural 1519
2  #include<cstdio>
3  #include<cstdlib>
4  #include<cstring>
5  #include<iostream>
6  #include<algorithm>
7  using namespace std;
8
9  typedef long long ll;
10 const int maxn = 14,maxs = 300010;
11 int mp[maxn][maxn],N,M,cur,last,total[2];
12 int size,ex,ey,head[maxs],nxt[maxs],Hash[maxs];
13 ll f[2][maxs],state[2][maxs];
14
15 inline void init()
16 {
17     memset(mp,0,sizeof mp); ex = ey = 0;
18     size = cur = 0; last = 1;
19     total[cur] = 1;
20     state[cur][1] = 0;
21     f[cur][1] = 1;
22 }
23
24 inline void calc(ll s,ll inc)
25 {
26     int pos = s%maxs;
27     for (int i = head[pos];i;i = nxt[i])
28         if (state[cur][Hash[i]] == s)
29             {
30                 f[cur][Hash[i]] += inc;
31                 return;
32             }
33     ++total[cur];
34     state[cur][total[cur]] = s;
35     f[cur][total[cur]] = inc;
36     nxt[++size] = head[pos];
37     head[pos] = size;
38     Hash[size] = total[cur];
39 }
40
41 inline ll work()
42 {

```

```

43     ll ret = 0;
44     for (int i = 1; i <= N; ++i)
45     {
46         for (int k = 1; k <= total[cur]; ++k) state[cur][k] <= 2;
47         for (int j = 1; j <= M; ++j)
48         {
49             memset(head, 0, sizeof head);
50             size = 0; cur ^= 1, last ^= 1;
51             total[cur] = 0;
52             for (int k = 1; k <= total[last]; ++k)
53             {
54                 ll s = state[last][k], num = f[last][k];
55                 int p = (s >> ((j-1) << 1)) % 4, q = (s >> (j << 1)) % 4;
56                 if (!mp[i][j]) { if (!p && !q) calc(s, num); }
57                 else if (!p && !q)
58                 {
59                     if (mp[i+1][j] && mp[i][j+1])
60                         calc(s + (1 << ((j-1) << 1)) + 2 * (1 << (j << 1)), num);
61                 }
62                 else if (!p && q)
63                 {
64                     if (mp[i][j+1]) calc(s, num);
65                     if (mp[i+1][j]) calc(s - q * (1 << (j << 1)) + q * (1 << ((j-1) << 1)), num);
66                 }
67                 else if (p && !q)
68                 {
69                     if (mp[i+1][j]) calc(s, num);
70                     if (mp[i][j+1]) calc(s - p * (1 << ((j-1) << 1)) + p * (1 << (j << 1)), num);
71                 }
72                 else if (p == 1 && q == 1)
73                 {
74                     int b = 1;
75                     for (int t = j+1; t <= M; ++t)
76                     {
77                         int v = (s >> (t << 1)) % 4;
78                         if (v == 1) ++b; else if (v == 2) --b;
79                         if (b == 0) { s -= 1 * (1 << (t << 1)); break; }
80                     }
81                     calc(s - (1 << ((j-1) << 1)) - (1 << (j << 1)), num);
82                 }
83                 else if (p == 2 && q == 2)
84                 {
85                     int b = 1;
86                     for (int t = j-2; t >= 0; --t)
87                     {
88                         int v = (s >> (t << 1)) % 4;
89                         if (v == 2) ++b; else if (v == 1) --b;
90                         if (b == 0) { s += 1 * (1 << (t << 1)); break; }
91                     }

```

```

92         calc(s-2*(1<<((j-1)<<1))-2*(1<<(j<<1)),num);
93     }
94     else if (p == 1&&q == 2) { if (i == ex&&j == ey) ret += num; }
95     else if (p == 2&&q == 1)
96         calc(s-2*(1<<((j-1)<<1))-(1<<(j<<1)),num);
97     }
98 }
99 }
100 return ret;
101 }
102
103 int main()
104 {
105     freopen("1519.in","r",stdin);
106     while (scanf("%d%d",&N,&M) != EOF)
107     {
108         init();
109         for (int i = 1;i <= N;++i)
110             for (int j = 1;j <= M;++j)
111             {
112                 char ch; do ch = getchar(); while (ch != '.'&&ch != '*');
113                 if (ch == '.') ex = i,ey = j,mp[i][j] = 1;
114             }
115         cout << work() << endl;
116     }
117     return 0;
118 }

```

1.4 Slop Optimization

```

1  #include<algorithm>
2  #include<cstring>
3  #include<iostream>
4  #include<cstdio>
5  #include<cstdlib>
6  using namespace std;
7
8  typedef long long ll;
9  const int maxn = 500010; const ll inf = 1LL<<60;
10 int N,K,A[maxn]; ll pre[maxn],f[maxn];
11
12 struct Point
13 {
14     ll x,y;
15     inline Point() = default;
16     inline Point(ll _x,ll _y):x(_x),y(_y) {}
17     friend inline Point operator -(const Point &a,const Point &b) { return
↵ Point(a.x-b.x,a.y-b.y); }
18     friend inline ll operator /(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }

```

```

19  };
20
21  inline ll calc(const Point &a,int b) { return -a.x*b+a.y; }
22
23  struct Queue
24  {
25      Point array[maxn]; int h,t;
26      inline Queue() = default;
27      inline void init() { h = t = 0; }
28      inline void pop_front(int i) { while (t-h >= 2&&calc(array[h+1],i) > calc(array[h+2],i))
↪      ++h; }
29      inline void push(const Point &a,int i) { while (t-h >=
↪      2&&(a-array[t-1])/(array[t]-array[t-1]) >= 0) --t; array[++t] = a; }
30      inline Point front() const { return array[h+1]; }
31  }team;
32
33  inline int gi()
34  {
35      char ch; int ret = 0,f = 1;
36      do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
37      if (ch == '-') f = -1,ch = getchar();
38      do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
39      return ret*f;
40  }
41
42  int main()
43  {
44      // freopen("E.in","r",stdin);
45      for (int T = gi();T--;)
46      {
47          N = gi(),K = gi(); team.init();
48          for (int i = 1;i <= N;++i) A[i] = gi();
49          for (int i = 1;i <= N;++i) pre[i] = pre[i-1]+A[i];
50          for (int i = 1;i <= N;++i)
51          {
52              if (i >= K)
53              {
54                  if (f[i-K] != inf)
55                      team.push(Point(A[i-K+1],f[i-K]-pre[i-K]+(ll)(i-K)*A[i-K+1]),i);
56                  team.pop_front(i);
57                  f[i] = calc(team.front(),i)+pre[i];
58              }
59              else f[i] = inf;
60          }
61          cout << f[N] << endl;
62      }
63      return 0;
64  }

```

1.5 Three-dimension Partial Order

```

1  //三维偏序, CDQ 分治
2  #define lowbit(a) (a&-a)
3  int M,N,A,B,tree[maxn];
4
5  inline void ins(int a,int b) { for (;a < maxn;a += lowbit(a)) tree[a] = max(tree[a],b); }
6  inline void clear(int a) { for (;a < maxn;a += lowbit(a)) tree[a] = 0; }
7  inline int calc(int a) { int ret = 0; for (;a;a -= lowbit(a)) ret = max(tree[a],ret); return
   ↪ ret; }
8
9  struct Node
10 {
11     int x,y,z,res;
12     inline Node(int _x = 0,int _y = 0,int _z = 0,int _res = 0):x(_x),y(_y),z(_z),res(_res) {}
13     inline void update() { ++x,++y,++z; }
14 }E[maxn];
15
16 inline bool cmpx(const Node &a,const Node &b)
17 {
18     if (a.x != b.x) return a.x < b.x;
19     else if (a.y != b.y) return a.y > b.y;
20     else return a.z > b.z;
21 }
22 inline bool cmpy(const Node &a,const Node &b) { return a.y < b.y; }
23
24 inline void work(int l,int r)
25 {
26     if (l == r) { E[l].res = max(E[l].res,1); return; }
27     int mid = (l+r) >> 1,p = 1;
28     work(l,mid);
29     sort(E+l,E+mid+1,cmpy);
30     sort(E+mid+1,E+r+1,cmpy);
31     for (int i = mid+1;i <= r;++i)
32     {
33         for (;p <= mid&&E[p].y < E[i].y;++p) ins(E[p].z,E[p].res);
34         E[i].res = max(E[i].res,calc(E[i].z-1)+1);
35     }
36     while (p > 1) clear(E[--p].z);
37     sort(E+mid+1,E+r+1,cmpx);
38     work(mid+1,r);
39 }
40
41 inline int run()
42 {
43     for (int i = 1;i <= N+M;++i) E[i].update();
44     sort(E+1,E+N+M+1,cmpx); work(1,N+M);
45     int ret = 0;
46     for (int i = 1;i <= N+M;++i) ret = max(ret,E[i].res);
47     return ret;

```


Chapter 2

Computational Geometry

2.1 Circle Intersection

```
1  //modified
2  const double eps = 1e-7, pi = acos(-1.0);
3  int N,M; double area[maxn]; // area[k] -> area of intersections >= k.
4
5  inline int dcmp(double a)
6  {
7      if (-eps <= a && a <= eps) return 0;
8      else if (a > 0) return 1; else return -1;
9  }
10
11 struct Point
12 {
13     double x,y;
14     inline Point() = default;
15     inline Point(double _x,double _y):x(_x),y(_y) {}
16     inline void read() { x = gi(),y = gi(); }
17     inline double norm() const { return sqrt(x*x+y*y); }
18     inline double angle() const { return atan2(y,x); }
19     inline Point unit() const { double len = norm(); return Point(x/len,y/len); }
20     friend inline Point operator-(const Point &a,const Point &b) { return
↵ Point(a.x-b.x,a.y-b.y); }
21     friend inline Point operator+(const Point &a,const Point &b) { return
↵ Point(a.x+b.x,a.y+b.y); }
22     friend inline Point operator*(const Point &a,double b) { return Point(a.x*b,a.y*b); }
23     friend inline Point operator*(double b,const Point &a) { return Point(a.x*b,a.y*b); }
24     friend inline Point operator/(const Point &a,double b) { return Point(a.x/b,a.y/b); }
25     friend inline double operator/(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
26 };
27 struct Circle
28 {
29     Point C; double r; int sgn;
30     inline Circle() = default;
```



```

31     inline Circle(const Point &_C,double _r,int _sgn):C(_C),r(_r),sgn(_sgn) {}
    ↪ // sgn 代表该圆的权值, 默认 1
32     friend inline bool operator==(const Circle &a,const Circle &b)
33     {
34         if (dcmp(a.r-b.r)) return false;
35         if (dcmp(a.C.x-b.C.x)) return false;
36         if (dcmp(a.C.y-b.C.y)) return false;
37         if (a.sgn != b.sgn) return false;
38         return true;
39     }
40     friend inline bool operator!=(const Circle &a,const Circle &b) { return !(a == b); }
41 }cir[maxn];
42
43 inline Point rotate(const Point &p,double cost,double sint)
44 {
45     double x = p.x,y = p.y;
46     return Point(x*cost-y*sint,x*sint+y*cost);
47 }
48 inline pair <Point,Point> crosspoint(const Point &ap,double ar,const Point &bp,double br)
49 {
50     double d = (ap-bp).norm(),cost = (ar*ar+d*d-br*br)/(2*ar*d),sint = sqrt(1-cost*cost);
51     Point v = ((bp-ap).unit())*ar;
52     return make_pair(ap+rotate(v,cost,-sint),ap+rotate(v,cost,sint));
53 }
54 inline pair <Point,Point> crosspoint(const Circle &a,const Circle &b) { return
    ↪ crosspoint(a.C,a.r,b.C,b.r); }
55
56 inline bool overlap(const Circle &a,const Circle &b) { return dcmp(a.r-b.r-(a.C-b.C).norm()) >=
    ↪ 0; } // b 是不是在 a 里面
57 inline bool intersect(const Circle &a,const Circle &b)
58 {
59     if (overlap(a,b)) return false;
60     if (overlap(b,a)) return false;
61     return dcmp((a.C-b.C).norm()-a.r-b.r) < 0;
62 }
63
64 struct Event
65 {
66     Point p; double a; int d;
67     inline Event() = default;
68     inline Event(const Point &p,double _a,double _d):p(_p),a(_a),d(_d) {}
69     friend inline bool operator <(const Event &a,const Event &b) { return a.a < b.a; }
70 };
71
72 inline void solve()
73 {
74     for (int i = 1;i <= M;++i) area[i] = 0;
75     for (int i = 1;i <= M;++i)
76     {

```

```

77     int cnt = cir[i].sgn; if (cnt<0) cnt = 0; vector <Event> event;
78     for (int j = 1;j < i;++j) if (cir[i] == cir[j]) cnt += cir[j].sgn;
79     for (int j = 1;j <= M;++j)
80         if (j != i&&cir[i] != cir[j]&&overlap(cir[j],cir[i])) cnt += cir[j].sgn;
81     for (int j = 1;j <= M;++j)
82         if (j != i&&intersect(cir[i],cir[j]))
83         {
84             pair <Point,Point> res = crosspoint(cir[i],cir[j]); swap(res.first,res.second);
85             double alpha1 = (res.first-cir[i].C).angle(),alpha2 =
↪ (res.second-cir[i].C).angle();
86             event.push_back(Event(res.second,alpha2,cir[j].sgn));
87             event.push_back(Event(res.first,alpha1,-cir[j].sgn));
88             cnt += (alpha2 > alpha1)*cir[j].sgn;
89         }
90     if (!event.size()) area[cnt] += pi*cir[i].r*cir[i].r*cir[i].sgn;
91     else
92     {
93         sort(event.begin(),event.end());
94         event.push_back(event.front());
95         for (int j = 0;j+1 < (int)event.size();++j)
96         {
97             cnt += event[j].d;
98             area[cnt] += event[j].p/event[j+1].p/2*cir[i].sgn;
99             double alpha = event[j+1].a-event[j].a;
100             if (alpha < 0) alpha += 2*pi;
101             if (!dcmp(alpha)) continue;
102             area[cnt] += alpha*cir[i].r*cir[i].r/2*cir[i].sgn;
103             area[cnt] += -sin(alpha)*cir[i].r*cir[i].r/2*cir[i].sgn;
104         }
105     }
106 }
107 }
108
109 // origin
110 struct Event {
111     Point p;
112     double ang;
113     int delta;
114     Event (Point p = Point(0, 0), double ang = 0, double delta = 0) : p(p), ang(ang),
↪ delta(delta) {}
115 };
116 bool operator < (const Event &a, const Event &b) {
117     return a.ang < b.ang;
118 }
119 void addEvent(const Circle &a, const Circle &b, vector<Event> &evt, int &cnt) {
120     double d2 = (a.o - b.o).len2(),
121     dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
122     pRatio = sqrt(-(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r)) / (d2 * d2 * 4));
123     Point d = b.o - a.o, p = d.rotate(PI / 2),

```

```

124         q0 = a.o + d * dRatio + p * pRatio,
125         q1 = a.o + d * dRatio - p * pRatio;
126     double ang0 = (q0 - a.o).ang(),
127         ang1 = (q1 - a.o).ang();
128     evt.push_back(Event(q1, ang1, 1));
129     evt.push_back(Event(q0, ang0, -1));
130     cnt += ang1 > ang0;
131 }
132 bool issame(const Circle &a, const Circle &b) { return sign((a.o - b.o).len()) == 0 && sign(a.r
↪ - b.r) == 0; }
133 bool overlap(const Circle &a, const Circle &b) { return sign(a.r - b.r - (a.o - b.o).len()) >=
↪ 0; }
134 bool intersect(const Circle &a, const Circle &b) { return sign((a.o - b.o).len() - a.r - b.r) <
↪ 0; }
135 Circle c[N];
136 double area[N]; // area[k] -> area of intersections >= k.
137 Point centroid[N]; //k 次圆的质心
138 bool keep[N];
139 void add(int cnt, DB a, Point c) {
140     area[cnt] += a;
141     centroid[cnt] = centroid[cnt] + c * a;
142 }
143 void solve(int C) {
144     for (int i = 1; i <= C; ++i) {
145         area[i] = 0;
146         centroid[i] = Point(0, 0);
147     }
148     for (int i = 0; i < C; ++i) {
149         int cnt = 1;
150         vector<Event> evt;
151         for (int j = 0; j < i; ++j) if (issame(c[i], c[j])) ++cnt;
152         for (int j = 0; j < C; ++j) {
153             if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i])) {
154                 ++cnt;
155             }
156         }
157         for (int j = 0; j < C; ++j) {
158             if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) && intersect(c[i], c[j]))
↪ {
159                 addEvent(c[i], c[j], evt, cnt);
160             }
161         }
162         if (evt.size() == 0u) {
163             add(cnt, PI * c[i].r * c[i].r, c[i].o);
164         } else {
165             sort(evt.begin(), evt.end());
166             evt.push_back(evt.front());
167             for (int j = 0; j + 1 < (int)evt.size(); ++j) {
168                 cnt += evt[j].delta;

```

```

169         add(cnt, det(evt[j].p, evt[j + 1].p) / 2, (evt[j].p + evt[j + 1].p) / 3);
170         double ang = evt[j + 1].ang - evt[j].ang;
171         if (ang < 0) {
172             ang += PI * 2;
173         }
174         if (sign(ang) == 0) continue;
175         double ang0 = evt[j].a, ang1 = evt[j+1].a;
176         add(cnt, ang * c[i].r * c[i].r / 2, c[i].o +
177             Point(sin(ang1) - sin(ang0), -cos(ang1) + cos(ang0)) * (2 / (3 * ang) *
↪ c[i].r));
178         add(cnt, -sin(ang) * c[i].r * c[i].r / 2, (c[i].o + evt[j].p + evt[j + 1].p) /
↪ 3);
179     }
180 }
181 }
182 for (int i = 1; i <= C; ++ i)
183     if (sign(area[i])) {
184         centroid[i] = centroid[i] / area[i];
185     }
186 }

```

2.2 Common Formulas

```

1 //计算几何常用公式
2 inline int dcmp(double a)
3 {
4     if (fabs(a) <= eps) return 0;
5     else if (a > 0) return 1;
6     else return -1;
7 }
8 struct Point
9 {
10     double x,y;
11     inline Point() = default;
12     inline Point(double _x,double _y):x(_x),y(_y) {}
13     inline Point unit() const
14     {
15         double len = norm();
16         if (!dcmp(len)) return Point(1,0);
17         else return *this/len;
18     }
19     inline double norm() const { return sqrt(x*x+y*y); }
20     inline Point reflect(const Point &p) const
21     {
22         Point v = *this-p; double len = v.norm();
23         v = v/len; return p+v*(1/len);
24     }
25     inline void read() { scanf("%lf %lf",&x,&y); }
26     inline Point vertical() const { return Point(y,-x); }

```

```

27     inline double angle() const
28     {
29         double ret = atan2(y,x);
30         if (ret < 0) ret += 2*pi;
31         return ret;
32     }
33     friend inline bool operator ==(const Point &a,const Point &b) { return
↪ !dcmp(a.x-b.x)&&!dcmp(a.y-b.y); }
34     friend inline Point operator -(const Point &a,const Point &b) { return
↪ Point(a.x-b.x,a.y-b.y); }
35     friend inline Point operator +(const Point &a,const Point &b) { return
↪ Point(a.x+b.x,a.y+b.y); }
36     friend inline Point operator /(const Point &a,double b) { return Point(a.x/b,a.y/b); }
37     friend inline Point operator *(const Point &a,double b) { return Point(a.x*b,a.y*b); }
38     friend inline Point operator *(double b,const Point &a) { return Point(a.x*b,a.y*b); }
39     friend inline double operator /(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
40 };
41 struct Line
42 {
43     Point p,v; double slop;
44     inline Line() = default;
45     inline Line(const Point &p,const Point &v):p(_p),v(_v) {}
46     inline void update() { slop = v.alpha(); }
47     friend inline bool operator <(const Line &l1,const Line &l2)
48     { return l1.slop < l2.slop; }
49     inline double dis(const Point &a) { fabs((a-p)/v)/(v.len()); } //点到直线距离
50 };
51
52 inline bool OnLine(const Line &l,const Point &p) { return !dcmp(l.v/(p-l.p)); } //点在直线上
53
54 inline Point CrossPoint(const Line &a,const Line &b) //直线交点
55 {
56     Point u = a.p - b.p;
57     double t = (b.v/u)/(a.v/b.v);
58     return a.p+a.v*t;
59 }
60
61 inline bool parallel(const Line &a,const Line &b) { return !dcmp(a.v/b.v); } //直线平行

```

2.3 Convex Hull

```

1 struct Point
2 {
3     inline Point() = default;
4     inline Point(double _x,double _y):x(_x),y(_y) {}
5     inline Point unit() const
6     {
7         double len = norm();
8         if (!dcmp(len)) return Point(1,0);

```

```

9         else return *this/len;
10    }
11    inline double norm() const { return sqrt(x*x+y*y); }
12    inline Point reflect(const Point &p) const
13    {
14        Point v = *this-p; double len = v.norm();
15        v = v/len; return p+v*(1/len);
16    }
17    inline void read() { scanf("%lf %lf",&x,&y); }
18    inline Point vertical() const { return Point(y,-x); }
19    inline double angle() const
20    {
21        double ret = atan2(y,x);
22        if (ret < 0) ret += 2*pi;
23        return ret;
24    }
25    friend inline bool operator ==(const Point &a,const Point &b) { return
↪ !dcmp(a.x-b.x)&&!dcmp(a.y-b.y); }
26    friend inline Point operator -(const Point &a,const Point &b) { return
↪ Point(a.x-b.x,a.y-b.y); }
27    friend inline Point operator +(const Point &a,const Point &b) { return
↪ Point(a.x+b.x,a.y+b.y); }
28    friend inline Point operator /(const Point &a,double b) { return Point(a.x/b,a.y/b); }
29    friend inline Point operator *(const Point &a,double b) { return Point(a.x*b,a.y*b); }
30    friend inline Point operator *(double b,const Point &a) { return Point(a.x*b,a.y*b); }
31    friend inline double operator /(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
32    friend inline bool operator <(const Point &a,const Point &b)
33    {
34        if (a.x != b.x) return a.x < b.x;
35        else return a.y < b.y;
36    }
37 }P[maxn],convex[maxn];
38
39 inline void ConvexHull()
40 {
41     sort(P+1,P+N+1); //x 第一关键字, y 第二关键字从小到大排序
42     for (int i = 1;i <= N;++i)
43     {
44         while (m > 1&&(convex[m]-convex[m-1])/(P[i]-convex[m-1]) <= 0) --m;
45         convex[++m] = P[i];
46     }
47     int k = m;
48     for (int i = N-1;i-->0)
49     {
50         while (m > k&&(convex[m]-convex[m-1])/(P[i]-convex[m-1]) <= 0) --m;
51         convex[++m] = P[i];
52     }
53     if (N > 1) m--;
54 }

```

2.4 Cross Points of Circles

```

1 //圓圓求交，需先判定兩圓有交
2 inline Point rotate(const Point &p,double cost,double sint)
3 {
4     double x = p.x,y = p.y;
5     return Point(x*cost-y*sint,x*sint+y*cost);
6 }
7
8 inline pair <Point,Point> CrossPoint(const Point &ap,double ar,const Point &bp,double br)
9 {
10     double d = (ap-bp).norm();
11     double cost = (ar*ar+d*d-br*br)/(2*ar*d),sint = sqrt(1-cost*cost);
12     Point v = ((bp-ap)/(bp-ap).norm())*ar;
13     return make_pair(ap+rotate(v,cost,-sint),ap+rotate(v,cost,sint));
14 }

```

2.5 Cross Points of Line and Circle

```

1 //a b 直线两点, o 圆心
2 //若 a b 为线段, 则 0 <= t1,t2 <= 1
3 inline void CrossPoint(const Point &a,const Point &b,const Point &o,double r,Point *ret,int
4     ↪ &num)
5 {
6     double X0 = o.x,Y0 = o.y;
7     double X1 = a.x,Y1 = a.y;
8     double X2 = b.x,Y2 = b.y;
9     double dx = X2-X1,dy = Y2-Y1;
10    double A = dx*dx+dy*dy;
11    double B = 2*dx*(X1-X0)+2*dy*(Y1-Y0);
12    double C = (X1-X0)*(X1-X0)+(Y1-Y0)*(Y1-Y0)-r*r;
13    double delta = B*B-4*A*C+eps;
14    num = 0;
15    if (delta >= 0)
16    {
17        double t1 = (-B-sqrt(delta))/(2*A);
18        double t2 = (-B+sqrt(delta))/(2*A);
19        ret[++num] = Point(X1+t1*dx,Y1+t1*dy);
20        ret[++num] = Point(X1+t2*dx,Y1+t2*dy);
21    }
22 }

```

2.6 Graham Scanning Algorithm

```

1 //凸包上最大四边形面积
2 #include<cmath>
3 #include<algorithm>
4 #include<cstring>
5 #include<iostream>

```

```

6  #include<cstdio>
7  #include<cstdlib>
8  using namespace std;
9
10 const int maxn = 2010;
11 int N,M; double ans;
12
13 struct Point
14 {
15     double x,y;
16     Point() = default;
17     Point(double _x,double _y):x(_x),y(_y) {}
18     inline void read() { scanf("%lf %lf",&x,&y); }
19     friend inline Point operator -(const Point &a,const Point &b) { return
↪ Point(a.x-b.x,a.y-b.y); }
20     friend inline double operator /(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
21     friend inline double operator <(const Point &a,const Point &b)
22     {
23         if (a.x != b.x) return a.x < b.x;
24         else return a.y < b.y;
25     }
26 }P[maxn],convex[maxn];
27
28 inline int gi()
29 {
30     char ch; int ret = 0,f = 1;
31     do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
32     if (ch == '-') f = -1,ch = getchar();
33     do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
34     return ret*f;
35 }
36
37 inline void ConvexHull()
38 {
39     int m = 0;
40     sort(P+1,P+N+1); //x 第一关键字, y 第二关键字从小到大排序
41     for (int i = 1;i <= N;++i)
42     {
43         while (m > 1&&(convex[m]-convex[m-1])/(P[i]-convex[m-1]) <= 0) --m;
44         convex[++m] = P[i];
45     }
46     int k = m;
47     for (int i = N-1;i-->0)
48     {
49         while (m > k&&(convex[m]-convex[m-1])/(P[i]-convex[m-1]) <= 0) --m;
50         convex[++m] = P[i];
51     }
52     if (N > 1) m--; M = m;
53 }

```



```

54
55 inline void Graham()
56 {
57     for (int i = 1; i <= M; ++i) convex[i+M] = convex[i];
58     int p1, p2, p3, p4;
59     for (p1 = 1; p1 <= M; ++p1)
60     {
61         p2 = p1+1;
62         p3 = p2+1;
63         p4 = p3+1;
64         for (; p3 < p1+M-1; ++p3)
65         {
66             Point v = convex[p3]-convex[p1];
67             while (p2 < p3 && fabs((convex[p2]-convex[p1])/v) <
↪ fabs((convex[p2+1]-convex[p1])/v)) ++p2;
68             while (p4 < p1+M && fabs((convex[p4]-convex[p1])/v) <
↪ fabs((convex[p4+1]-convex[p1])/v)) ++p4;
69             ans = max(ans, fabs((convex[p2]-convex[p1])/v)+fabs((convex[p4]-convex[p1])/v));
70         }
71     }
72     ans = ans/2;
73 }
74
75 int main()
76 {
77     N = gi();
78     for (int i = 1; i <= N; ++i) P[i].read();
79     ConvexHull();
80     Graham();
81     printf("%.3f\n", ans);
82     return 0;
83 }
84 ///////////////////////////////////////////////////////////////////
85 inline void jam() //凸包上最大四边形面积
86 {
87     for (int i = 1; i <= m; ++i) ch[i+m] = ch[i]; //凸包倍长
88     for (int p1 = 1, p2, p3, p4; p1 <= m; ++p1)
89     {
90         p2 = p1 + 1;
91         p3 = p2 + 1;
92         p4 = p3 + 1;
93         for (; p3 < p1 + m - 1; ++p3)
94         {
95             Line l = ((SEG) { ch[p1], ch[p3] }).extend(); //枚举对角线，线段变成直线
96             while (p2 < p3 && l.dis(ch[p2]) < l.dis(ch[p2 + 1])) ++p2; //点到直线距离
97             while (p4 < p1 + m && l.dis(ch[p4]) < l.dis(ch[p4 + 1])) ++p4;
98             ans = max(ans, (l.dis(ch[p2])+l.dis(ch[p4]))*(ch[p1] - ch[p3]).len()/2); //更新答案
99         }
100     }

```

101 }

2.7 Half Plane Intersection

```

1 //半平面交，直线左侧半平面，注意最后是 tail-head <= 0 还是 tail-head <= 1
2 inline int dcmp(double a)
3 {
4     if (-eps <= a && a <= eps) return 0;
5     else if (a > 0) return 1; else return -1;
6 }
7
8 struct Point
9 {
10     double x,y;
11     inline Point() = default;
12     inline Point(double _x,double _y):x(_x),y(_y) {}
13     inline void read() { x = gi(),y = gi(); }
14     inline Point vertical() const { return Point(-y,x); }
15     inline Point unit() const
16     {
17         double len = norm();
18         if (!dcmp(len)) return Point(1,0);
19         else return *this/len;
20     }
21     inline double norm() const { return sqrt(x*x+y*y); }
22     inline double angle() const { return atan2(y,x); }
23     friend inline Point operator+(const Point &a,const Point &b) { return
↪ Point(a.x+b.x,a.y+b.y); }
24     friend inline Point operator-(const Point &a,const Point &b) { return
↪ Point(a.x-b.x,a.y-b.y); }
25     friend inline Point operator*(const Point &a,double b) { return Point(a.x*b,a.y*b); }
26     friend inline Point operator*(double b,const Point &a) { return Point(a.x*b,a.y*b); }
27     friend inline double operator/(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
28 }P[maxn],pp[maxn],pol[maxn];
29
30 struct Line
31 {
32     Point p,v;
33     inline Line(const Point _p = Point(),const Point _v = Point()):p(_p),v(_v) {}
34     inline double slop() const { return v.angle(); }
35     friend inline bool operator<(const Line &a,const Line &b) { return a.slop() < b.slop(); }
36 }line[maxn],qq[maxn];
37
38 inline bool onleft(const Line &L,const Point &p)
39 {
40     return dcmp(L.v/(p-L.p)) > 0;
41 }
42 inline bool parallel(const Line &a,const Line &b) { return !dcmp(a.v/b.v); }
43 inline Point crosspoint(const Line &a,const Line &b)

```

```

44 {
45     Point u = a.p-b.p;
46     double t = (b.v/u)/(a.v/b.v);
47     return a.p+(a.v*t);
48 }
49
50 inline int half_plane_intersection()
51 {
52     sort(lines+1,lines+tot+1); //直线按斜率排序
53     int head,tail;
54     qq[head = tail = 1] = lines[1];
55     for (int i = 2;i <= tot;++i)
56     {
57         while (head < tail&&!onleft(lines[i],pp[tail-1])) --tail;
58         while (head < tail&&!onleft(lines[i],pp[head])) ++head;
59         qq[++tail] = lines[i];
60         if (parallel(qq[tail],qq[tail-1]))
61         {
62             tail--;
63             if (onleft(qq[tail],lines[i].p)) qq[tail] = lines[i];
64         }
65         if (head < tail) pp[tail-1] = crosspoint(qq[tail],qq[tail-1]);
66     }
67     while (head < tail && !onleft(qq[head],pp[tail-1])) --tail;
68     if (tail-head <= 0) return 0;
69     pp[tail] = crosspoint(qq[tail],qq[head]);
70     for (int i = head;i <= tail;++i) pol[++m] = pp[i]; //半平面交点
71     pol[0] = pol[m];
72     return m;
73 }

```

2.8 Intersecting Area of Circle and Polygon

```

1  const int maxn = 510;
2  const double eps = 1e-9;
3
4  inline int dcmp(double a)
5  {
6      if (a > eps) return 1;
7      else if (a < -eps) return -1;
8      else return 0;
9  }
10
11 struct Point
12 {
13     double x,y;
14     Point() = default;
15     Point(double _x,double _y):x(_x),y(_y) {}
16     inline double norm() const { return sqrt(x*x+y*y); }

```

```

17     inline Point unit() const { double len = norm(); return Point(x/len,y/len); }
18     friend Point operator +(const Point &a,const Point &b) { return Point(a.x+b.x,a.y+b.y); }
19     friend Point operator -(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
20     friend Point operator *(const Point &a,double b) { return Point(a.x*b,a.y*b); }
21     friend Point operator *(double b,const Point &a) { return Point(a.x*b,a.y*b); }
22     friend Point operator /(const Point &a,double b) { return Point(a.x/b,a.y/b); }
23     friend double operator /(const Point &a,const Point &b) { return a.x*b.y-b.x*a.y; }
24     friend double operator *(const Point &a,const Point &b) { return a.x*b.x+a.y*b.y; }
25     inline void read() { scanf("%lf %lf",&x,&y); }
26 }P[maxn],A,B;
27 int N; double K;
28
29 inline double getSectorArea(const Point &a,const Point &b,double r)
30 {
31     double c = (2*r*r-((a-b)*(a-b)))/(2*r*r);
32     double alpha = acos(c);
33     return r*r*alpha/2.0;
34 }
35
36 inline pair <double,double> getSolution(double a,double b,double c)
37 {
38     double delta = b*b-4*a*c;
39     if (dcmp(delta) < 0) return make_pair(0,0);
40     else return make_pair((-b-sqrt(delta))/(2*a),(-b+sqrt(delta))/(2*a));
41 }
42
43 inline pair <Point,Point> getIntersection(const Point &a,const Point &b,double r)
44 {
45     Point d = b-a;
46     double A = d*d,B = 2*(d*a),C = (a*a)-r*r;
47     pair <double,double> s = getSolution(A,B,C);
48     return make_pair(a+(d*s.first),a+(d*s.second));
49 }
50
51 inline double getPointDist(const Point &a,const Point &b)
52 {
53     Point d = b-a;
54     int sA = dcmp(a*d),sB = dcmp(b*d);
55     if (sA*sB <= 0) return (a/b)/((a-b).norm());
56     else return min(a.norm(),b.norm());
57 }
58
59 double getArea(const Point &a,const Point &b,double r)
60 {
61     double dA = a*a,dB = b*b,dC = getPointDist(a,b),ans = 0;
62     if (dcmp(dA-r*r) <= 0&&dcmp(dB-r*r) <= 0) return (a/b)/2;
63     Point tA = a.unit()*r,tB = b.unit()*r;
64     if (dcmp(dC-r) > 0) return getSectorArea(tA,tB,r);
65     pair <Point,Point> ret = getIntersection(a,b,r);

```

```

66     if (dcmp(dA-r*r) > 0&&dcmp(dB-r*r) > 0)
67     {
68         ans += getSectorArea(tA,ret.first,r);
69         ans += (ret.first/ret.second)/2;
70         ans += getSectorArea(ret.second,tB,r);
71         return ans;
72     }
73     if (dcmp(dA-r*r) > 0) return (ret.first/b)/2+getSectorArea(tA,ret.first,r);
74     else return (a/ret.second)/2.0+getSectorArea(ret.second,tB,r);
75 }
76
77 double getArea(int n,Point *p,const Point &c,double r)
78 {
79     double ret = 0;
80     for (int i = 0;i < n;++i)
81     {
82         int sgn = dcmp((p[i]-c)/(p[(i+1)%n]-c));
83         if (sgn > 0) ret += getArea(p[i]-c,p[(i+1)%n]-c,r);
84         else ret -= getArea(p[(i+1)%n]-c,p[i]-c,r);
85     }
86     return fabs(ret);
87 }

```

2.9 Intersection of Line and Convex Hull

```

1  //O(logN)
2  inline double getA(const Node &a)
3  {
4      double ret = atan2(a.y,a.x);
5      if (ret <= -pi/2) ret += 2*pi;
6      return ret;
7  }
8
9  inline int find(double x)
10 {
11     if (x <= w[1]||x >= w[m]) return 1;
12     return upper_bound(w+1,w+m+1,x)-w;
13 }
14
15 inline bool intersect(const Node &a,const Node &b)
16 {
17     int i = find(getA(b-a)),j = find(getA(a-b));
18     if (dcmp((b-a)/(hull[i]-a))*dcmp((b-a)/(hull[j]-a)) > 0) return false;
19     else return true;
20 }
21
22 inline void convex()
23 {
24     for (int i = 1;i <= N;++i)

```

```

25     {
26         while (m > 1 && (hull[m]-hull[m-1])/(P[i]-hull[m-1]) <= 0) --m;
27         hull[++m] = P[i];
28     }
29     int k = m;
30     for (int i = N-1; i; --i)
31     {
32         while (m > k && (hull[m]-hull[m-1])/(P[i]-hull[m-1]) <= 0) --m;
33         hull[++m] = P[i];
34     }
35     if (N > 1) m--;
36     for (int i = 1; i <= m; ++i)
37         w[i] = getA(hull[i+1]-hull[i]);
38 }

```

2.10 Minimal Product

```

1  // 最小乘积匹配
2  #include<algorithm>
3  #include<cstring>
4  #include<iostream>
5  #include<cstdio>
6  #include<cstdlib>
7  using namespace std;
8
9  const int maxn = 80, inf = 1<<29;
10 int N, ans, A[maxn][maxn], B[maxn][maxn];
11
12 inline int gi()
13 {
14     char ch; int ret = 0, f = 1;
15     do ch = getchar(); while (!(ch >= '0' && ch <= '9') && ch != '-');
16     if (ch == '-') f = -1, ch = getchar();
17     do ret = ret*10+ch-'0', ch = getchar(); while (ch >= '0' && ch <= '9');
18     return ret*f;
19 }
20
21 struct KM
22 {
23     int w[maxn][maxn], lx[maxn], ly[maxn], match[maxn], way[maxn], slack[maxn]; bool used[maxn];
24
25     inline void init()
26     {
27         for (int i = 1; i <= N; ++i)
28             match[i] = lx[i] = ly[i] = way[i] = 0;
29     }
30
31     inline void hungary(int x)
32     {

```

```

33     match[0] = x; int j0 = 0;
34     for (int j = 0; j <= N; ++j)
35         slack[j] = -inf, used[j] = false;
36     do
37     {
38         used[j0] = true;
39         int i0 = match[j0], delta = -inf, j1 = 0;
40         for (int j = 1; j <= N; ++j)
41             if (!used[j])
42             {
43                 int cur = -w[i0][j] - lx[i0] - ly[j];
44                 if (cur > slack[j]) slack[j] = cur, way[j] = j0;
45                 if (slack[j] > delta) delta = slack[j], j1 = j;
46             }
47         for (int j = 0; j <= N; ++j)
48         {
49             if (used[j]) lx[match[j]] += delta, ly[j] -= delta;
50             else slack[j] -= delta;
51         }
52         j0 = j1;
53     }
54     while (match[j0]);
55     do
56     {
57         int j1 = way[j0];
58         match[j0] = match[j1];
59         j0 = j1;
60     }
61     while (j0);
62 }
63
64 inline void work() { for (int i = 1; i <= N; ++i) hungary(i); }
65
66 inline int get_ans()
67 {
68     int sum = 0;
69     for (int i = 1; i <= N; ++i)
70     {
71         // if (w[match[i]][i] == inf) ; // 无解
72         if (match[i] > 0) sum += w[match[i]][i];
73     }
74     return sum;
75 }
76
77 inline void getp(int &x, int &y)
78 {
79     x = y = 0;
80     for (int i = 1; i <= N; ++i)
81         x += A[match[i]][i], y += B[match[i]][i];

```

```

82     }
83 }km;
84
85 inline void work(int X1,int Y1,int X2,int Y2)
86 {
87     km.init();
88     for (int i = 1;i <= N;++i)
89         for (int j = 1;j <= N;++j)
90             km.w[i][j] = (X2-X1)*B[i][j]+(Y1-Y2)*A[i][j];
91     km.work();
92     if (km.get_ans() >= X2*Y1-X1*Y2) return;
93     int x,y; km.getp(x,y);
94     ans = min(ans,x*y);
95     work(X1,Y1,x,y); work(x,y,X2,Y2);
96 }
97
98 int main()
99 {
100     // freopen("B.in","r",stdin);
101     for (int T = gi();T--;)
102     {
103         N = gi(); ans = inf;
104         for (int i = 1;i <= N;++i) for (int j = 1;j <= N;++j) A[i][j] = gi();
105         for (int i = 1;i <= N;++i) for (int j = 1;j <= N;++j) B[i][j] = gi();
106         int X1,Y1,X2,Y2;
107         km.init();
108         for (int i = 1;i <= N;++i)
109             for (int j = 1;j <= N;++j)
110                 km.w[i][j] = A[i][j];
111         km.work(); km.getp(X1,Y1);
112         km.init();
113         for (int i = 1;i <= N;++i)
114             for (int j = 1;j <= N;++j)
115                 km.w[i][j] = B[i][j];
116         km.work(); km.getp(X2,Y2);
117         ans = min(X1*Y1,X2*Y2);
118         work(X1,Y1,X2,Y2);
119         cout << ans << endl;
120     }
121     fclose(stdin); fclose(stdout);
122     return 0;
123 }

```

2.11 Planar Graph

```

1 // 包括平面图转对偶图
2 inline int dcmp(double a)
3 {
4     if (fabs(a) <= eps) return 0;

```



```

5     else if (a > 0) return 1;
6     else return -1;
7 }
8 struct Point
9 {
10     double x,y;
11     inline Point(double _x = 0,double _y = 0):x(_x),y(_y) {}
12     inline void read() { x = gi(),y = gi(); }
13     friend inline Point operator-(const Point &a,const Point &b) { return
↪ Point(a.x-b.x,a.y-b.y); }
14     friend inline double operator/(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
15     inline double angle() { return atan2(y,x); }
16 }pp[maxn];
17 struct Segment
18 {
19     int from,to,h,id,sur; // from 号点到 to 号点, h 为边权,suf 为这条有向边维出来的平面编号。
20     inline Segment(int _from = 0,int _to = 0,int _h = 0,int _id = 0,int _sur =
↪ 0):from(_from),to(_to),h(_h),id(_id),sur(_sur) {}
21     friend inline bool operator<(const Segment &a,const Segment &b) { return
↪ (pp[a.to]-pp[a.from]).angle() < (pp[b.to]-pp[b.from]).angle(); }
22 }edge[maxm*2];
23 vector <int> G[maxn];
24
25 inline void nadd(int u,int v,int h) { ++ncnt; G[u].push_back(ncnt); edge[ncnt] = Segment(u,v,h);
↪ }
26 inline void nins(int u,int v,int h) { nadd(u,v,h); nadd(v,u,h); }
27
28 inline bool cmp(int a,int b) { return edge[a] < edge[b]; }
29
30 inline void find_surface()
31 {
32     for (int i = 1;i <= N;++i) sort(G[i].begin(),G[i].end(),cmp);
33     for (int i = 1;i <= N;++i)
34     {
35         int nn = G[i].size();
36         for (int j = 0;j < nn;++j)
37             edge[G[i][j]].id = j;
38     }
39     for (int i = 2;i <= ncnt;++i)
40         if (!edge[i].sur)
41         {
42             ++tot; int j = i,p,nn; vector <Point> vec;
43             while (!edge[j].sur)
44             {
45                 edge[j].sur = tot; vec.push_back(pp[edge[j].from]);
46                 p = edge[j].to; nn = G[p].size();
47                 j ^= 1; j = G[p][(edge[j].id+1)%nn];
48             }
49             double res = 0; nn = vec.size();

```

```

50         for (j = 0; j < nn; ++j)
51             res += (vec[j]-vec[0])/(vec[(j+1)%nn]-vec[0]);
52         res /= 2; space[tot] = res;
53     }
54     // 开始建边, 以 mst 为例
55     // for (int i = 2; i <= cnt; i += 2)
56     // {
57     //     if (space[edge[i].sur]<0&&space[edge[i^1].sur]<0)
58     //         arr[++all] = (ARR) { edge[i].sur, edge[i^1].sur, edge[i].h };
59     //     else arr[++all] = (ARR) { edge[i].sur, edge[i^1].sur, inf };
60     // }
61 }
62
63 // 点定位
64 struct Scan
65 {
66     double x,y; int bel,sign;
67     inline Scan(double _x = 0, double _y = 0, int _bel = 0, int _sign =
68     ↪ 0):x(_x),y(_y),bel(_bel),sign(_sign) {}
69     friend inline bool operator < (const Scan &a, const Scan &b)
70     {
71         if (a.x != b.x) return a.x < b.x;
72         else return a.sign > b.sign;
73     }
74 }bac[maxn*4];
75
76 struct Splay
77 {
78     int num,root,ch[maxn][2],fa[maxn],key[maxn]; queue <int> team;
79
80     inline int newnode()
81     {
82         int ret;
83         if (team.empty()) ret = ++num;
84         else ret = team.front(),team.pop();
85         fa[ret] = ch[ret][0] = ch[ret][1] = 0;
86         return ret;
87     }
88
89     inline void init() { num = 0; root = newnode(); key[root] = cnt; }
90
91     inline void rotate(int x)
92     {
93         int y = fa[x], z = fa[y], l = ch[y][1] == x, r = l^1;
94         if (z != 0) ch[z][ch[z][1] == y] = x;
95         fa[x] = z; fa[y] = x; fa[ch[x][r]] = y;
96         ch[y][1] = ch[x][r]; ch[x][r] = y;
97     }

```

```

97
98     inline void splay(int x)
99     {
100         while (fa[x] != 0)
101         {
102             int y = fa[x], z = fa[y];
103             if (fa[y] != 0)
104             {
105                 if ((ch[y][0] == x) ^ (ch[z][0] == y)) rotate(x);
106                 else rotate(y);
107             }
108             rotate(x);
109         }
110         root = x;
111     }
112
113     inline int lower_bound(const Point &p)
114     {
115         int now = root, ret = 0;
116         while (now)
117         {
118             int k = key[now];
119             if ((p - pp[edge[k].from]) / (pp[edge[k].to] - pp[edge[k].from]) >= 0)
120                 ret = k, now = ch[now][0];
121             else now = ch[now][1];
122         }
123         return ret;
124     }
125
126     inline int find(int w)
127     {
128         int now = root;
129         double x = pp[edge[w].to].x, y = pp[edge[w].to].y;
130         double ang = (pp[edge[w].to] - pp[edge[w].from]).angle();
131         while (now)
132         {
133             int k = key[now];
134             if (k == w) return now;
135             NODE p = pp[edge[k].to] - pp[edge[k].from], q = pp[edge[k].from];
136             double xx = x - q.x, yy = q.y + xx / p.x * p.y;
137             if (equal(yy, y))
138             {
139                 double t = p.angle();
140                 now = ch[now][ang < t];
141             }
142             else now = ch[now][y > yy];
143         }
144     }
145

```

```

146 inline void erase(int w)
147 {
148     int p = find(w);
149     while (ch[p][0] || ch[p][1])
150     {
151         if (ch[p][0])
152         {
153             rotate(ch[p][0]);
154             if (p == root) root = fa[p];
155         }
156         else
157         {
158             rotate(ch[p][1]);
159             if (p == root) root = fa[p];
160         }
161     }
162     team.push(p);
163     ch[fa[p]][ch[fa[p]][1] == p] = 0;
164     fa[p] = 0;
165 }
166
167 inline void insert(int w)
168 {
169     int now = root, pre;
170     double x = pp[edge[w].from].x, y = pp[edge[w].from].y;
171     double ang = (pp[edge[w].to] - pp[edge[w].from]).angle();
172     double xx, yy;
173     while (true)
174     {
175         int k = key[now];
176         NODE p = pp[edge[k].to] - pp[edge[k].from], q = pp[edge[k].from];
177         xx = x - q.x, yy = q.y + xx / p.x * p.y;
178         if (equal(yy, y))
179         {
180             double t = p.angle();
181             pre = now, now = ch[now][ang > t];
182             if (!now)
183             {
184                 now = newnode();
185                 fa[now] = pre; ch[pre][ang > t] = now; key[now] = w;
186                 break;
187             }
188         }
189         else
190         {
191             pre = now, now = ch[now][y > yy];
192             if (!now)
193             {
194                 now = newnode();

```

```

195         fa[now] = pre; ch[pre][y>yy] = now; key[now] = w;
196         break;
197     }
198 }
199 }
200 splay(now);
201 }
202 }S;
203
204 inline void locate()
205 {
206     int nn = 0;
207     for (int i = 2; i <= cnt; i += 2)
208     {
209         if (!dcmp(pp[edge[i].from].x - pp[edge[i].to].x)) continue;
210         bac[++nn] = Scan(pp[edge[i].from].x, pp[edge[i].from].y, i, 2);
211         bac[++nn] = Scan(pp[edge[i].to].x, pp[edge[i].to].y, i, 3);
212     }
213     scanf("%d", &T); double x, y;
214     // 查询 (x,y) 所在平面
215     for (int i = 1; i <= T; ++i)
216     {
217         scanf("%lf %lf", &x, &y);
218         bac[++nn] = Scan(x, y, i, 0);
219         scanf("%lf %lf", &x, &y);
220         bac[++nn] = Scan(x, y, i, 1);
221     }
222     sort(bac+1, bac+nn+1);
223     pp[++n] = Point(-oo, -oo); pp[++n] = (oo, oo);
224     edge[++cnt] = Edge(n-1, n);
225     S.init(); int p;
226     for (int i = 1; i <= nn; ++i)
227     {
228         if (bac[i].sign == 2 || bac[i].sign == 3)
229         {
230             if (bac[i].sign == 2) S.insert(bac[i].bel);
231             else S.erase(bac[i].bel);
232         }
233         else
234         {
235             p = S.lower_bound(Point(bac[i].x, bac[i].y));
236             query[bac[i].bel][bac[i].sign] = edge[p].sur;
237         }
238     }
239 }

```

2.12 Polygon Class

```

1  inline bool PointOnSegment(const Point &t,const Point &a,const Point &b)
2  {
3      if (dcmp((t-a)/(b-a))) return false;
4      if (dcmp((t-a)*(t-b)) > 0) return false;
5      return true;
6  }
7
8  inline bool in(const Point &a,const Point &b,const Point &c)
9  {
10     double alpha = a.angle(),beta = b.angle(),gamma = c.angle(); // angle 返回 [0,2pi]
11     if (alpha <= beta) return dcmp(gamma-alpha) > 0&&dcmp(beta-gamma) > 0;
12     else return dcmp(gamma-alpha) > 0||dcmp(beta-gamma) > 0;
13 }
14
15 struct Polygon
16 {
17     int n; Point a[maxn];
18     inline Polygon() {}
19     inline void read()
20     {
21         n = gi();
22         for (int i = 0;i < n;++i) a[i].read();
23         a[n] = a[0];
24     }
25     // 点是否在多边形内部, 内部为 1, 外部为 0, 边界为 2, 不管顺逆时针
26     inline int Point_In(const Point &t) const
27     {
28         int num = 0;
29         for (int i = 0;i < n;++i)
30         {
31             if (PointOnSegment(t,a[i],a[i+1])) return 2;
32             int k = dcmp((a[i+1]-a[i])/(t-a[i]));
33             int d1 = dcmp(a[i].y-t.y),d2 = dcmp(a[i+1].y-t.y);
34             if (k > 0&&d1 <= 0&&d2 > 0) ++num;
35             if (k < 0&&d2 <= 0&&d1 > 0) --num;
36         }
37         return num != 0;
38     }
39     // 判断多边形的方向, true 为逆时针, false 为顺时针, 用叉积判断哪个多
40     inline bool CalculateClockDirection()
41     {
42         int res = 0;
43         for (int i = 0;i < n;++i)
44         {
45             int p = i-1,s = i+1,sgn;
46             if (p < 0) p += n; if (s >= n) s -= n;
47             sgn = dcmp((a[i]-a[p])/(a[s]-a[i]));
48             if (sgn) { if (sgn > 0) ++res; else --res; }

```

```

49     }
50     return res > 0;
51 }
52 // 判断多边形方向, true 为逆时针, false 为顺时针, 用 Green 公式
53 inline bool CalculateClockDirection()
54 {
55     double res = 0;
56     for (int i = 0; i < n; ++i)
57         res -= 0.5*(a[i+1].y+a[i].y)*(a[i+1].x-a[i].x);
58     return res > 0;
59 }
60
61 // 线段 ab 是否有点严格在多边形内部, 先判断线段是否与多边形边界有交, 再判断 ab 是否与多边形有交, 内部 false, 外
62 inline bool can(int ia, int ib)
63 {
64     Point a = P[ia], b = P[ib], v = b-a;
65     if (in(P[ia+1]-a, P[ia-1]-a, b-a) || in(P[ib+1]-b, P[ib-1]-b, a-b)) return false;
66     for (register int i = 0; i < N; ++i)
67     {
68         if (dcmp(v/(P[i]-a))*dcmp(v/(P[i+1]-a)) <
        ↪ 0 && dcmp(vec[i]/(a-P[i]))*dcmp(vec[i]/(b-P[i])) < 0)
69             return false;
70         if (PointOnSegment(a, P[i], P[i+1]) || PointOnSegment(b, P[i], P[i+1])) return false;
71         if (PointOnSegment(P[i], a, b) || PointOnSegment(P[i+1], a, b)) return false;
72     }
73     return true;
74 }
75 }poly;

```

2.13 Union Area of Circles

```

1 //N 为开始圆的个数, M 为离散化后圆的个数, cnt 为去包含后圆的个数
2 int N, M, cnt;
3
4 struct Node
5 {
6     double x, y;
7     inline Node(double _x = 0, double _y = 0):x(_x), y(_y) {}
8     inline void read() { x = gi(), y = gi(); }
9     inline double norm() const { return sqrt(x*x+y*y); }
10    inline double angle() const { return atan2(y, x); }
11    inline Node unit() const { double len = norm(); return Node(x/len, y/len); }
12    friend inline Node operator-(const Node &a, const Node &b) { return Node(a.x-b.x, a.y-b.y); }
13    friend inline Node operator+(const Node &a, const Node &b) { return Node(a.x+b.x, a.y+b.y); }
14    friend inline Node operator*(const Node &a, double b) { return Node(a.x*b, a.y*b); }
15    friend inline Node operator*(double b, const Node &a) { return Node(a.x*b, a.y*b); }
16    friend inline double operator/(const Node &a, const Node &b) { return a.x*b.y-a.y*b.x; }
17 };
18 struct Circle

```

```

19 {
20     Node C; double r;
21     inline Circle(const Node &_C = Node(), double _r = 0):C(_C),r(_r) {}
22     friend inline bool operator<(const Circle &a, const Circle &b)
23     {
24         if (dcmp(a.r-b.r)) return dcmp(a.r-b.r) < 0;
25         else if (dcmp(a.C.x-b.C.x)) return dcmp(a.C.x-b.C.x) < 0;
26         else return dcmp(a.C.y-b.C.y) < 0;
27     }
28     friend inline bool operator==(const Circle &a, const Circle &b)
29     {
30         if (dcmp(a.r-b.r)) return false;
31         if (dcmp(a.C.x-b.C.x)) return false;
32         if (dcmp(a.C.y-b.C.y)) return false;
33         return true;
34     }
35 }tc[maxn], cir[maxn];
36
37 inline Node rotate(const Node &p, double cost, double sint)
38 {
39     double x = p.x, y = p.y;
40     return Node(x*cost-y*sint, x*sint+y*cost);
41 }
42 inline pair <Node, Node> crosspoint(const Node &ap, double ar, const Node &bp, double br)
43 {
44     double d = (ap-bp).norm(), cost = (ar*ar+d*d-br*br)/(2*ar*d), sint = sqrt(1-cost*cost);
45     Node v = ((bp-ap).unit())*ar;
46     return make_pair(ap+rotate(v, cost, -sint), ap+rotate(v, cost, sint));
47 }
48 inline pair <Node, Node> crosspoint(const Circle &a, const Circle &b) { return
    ↪ crosspoint(a.C, a.r, b.C, b.r); }
49
50 struct Event
51 {
52     Node p; double a; int d;
53     inline Event(const Node &_p = Node(), double _a = 0, double _d = 0):p(_p),a(_a),d(_d) {}
54     friend inline bool operator <(const Event &a, const Event &b) { return a.a < b.a; }
55 };
56
57 inline double work()
58 {
59     sort(tc+1, tc+M+1); M = unique(tc+1, tc+M+1)-tc-1;
60     for (int i = M; i--;)
61     {
62         bool ok = true;
63         for (int j = i+1; j <= M; ++j)
64         {
65             double d = (tc[i].C-tc[j].C).norm();
66             if (dcmp(d-fabs(tc[i].r-tc[j].r)) <= 0) { ok = false; break; }

```



```

67     }
68     if (ok) cir[++cnt] = tc[i];
69 }
70 // for (int i = M;i;--i) cir[++cnt] = tc[i];
71 double ret = 0;
72 for (int i = 1;i <= cnt;++i)
73 {
74     vector <Event> event;
75     Node boundary = cir[i].C+Node(cir[i].r,0);
76     event.push_back(Event(boundary,-pi,0));
77     event.push_back(Event(boundary,pi,0));
78     for (int j = 1;j <= cnt;++j)
79     {
80         if (i == j) continue;
81         double d = (cir[i].C-cir[j].C).norm();
82         if (dcmp(d-(cir[i].r+cir[j].r)) < 0)
83         {
84             pair <Node,Node> res = crosspoint(cir[i],cir[j]);
85             double x = (res.first-cir[i].C).angle(),y = (res.second-cir[i].C).angle();
86             if (dcmp(x-y) > 0)
87             {
88                 event.push_back(Event(res.first,x,1));
89                 event.push_back(Event(boundary,pi,-1));
90                 event.push_back(Event(boundary,-pi,1));
91                 event.push_back(Event(res.second,y,-1));
92             }
93             else
94             {
95                 event.push_back(Event(res.first,x,1));
96                 event.push_back(Event(res.second,y,-1));
97             }
98         }
99     }
100     sort(event.begin(),event.end());
101     int sum = event[0].d;
102     for (int j = 1;j < (int)event.size();++j)
103     {
104         if (!sum)
105         {
106             ret += (event[j-1].p/event[j].p)/2;
107             double x = event[j-1].a,y = event[j].a;
108             double area = cir[i].r*cir[i].r*(y-x)/2;
109             Node v1 = event[j-1].p-cir[i].C,v2 = event[j].p-cir[i].C;
110             area -= (v1/v2)/2; ret += area;
111         }
112         sum += event[j].d;
113     }
114 }
115 return ret;

```


Chapter 3

Data Structure

3.1 Divide and Conquer on Tree

```
1  #include<cstring>
2  #include<iostream>
3  #include<cstdio>
4  #include<cstdlib>
5  using namespace std;
6
7  #define maxn (100010)
8  int best,cnt = 1,side[maxn],toit[maxn],next[maxn],large[maxn];
9  int sd[maxn],d[maxn],ns,nd,ans,N,K,size[maxn]; bool vis[maxn];
10
11 inline void add(int a,int b)
12 { next[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; }
13 inline void ins(int a,int b)
14 { add(a,b); add(b,a); }
15
16 inline void getroot(int now,int fa,int rest)
17 {
18     size[now] = 1; large[now] = 0;
19     for (int i = side[now];i;i = next[i])
20     {
21         if (toit[i] == fa||vis[toit[i]]) continue;
22         getroot(toit[i],now,rest);
23         size[now] += size[toit[i]];
24         large[now] = max(large[now],size[toit[i]]);
25     }
26     large[now] = max(large[now],rest-size[now]);
27     if (large[now] < large[best]) best = now;
28 }
29 inline int find_root(int now,int rest)
30 { best = 0; getroot(now,0,rest); return best; }
31
32 inline void dfs(int now,int fa,int dep)
33 {
```

```

34     size[now] = 1; nd = max(dep,nd); ++d[dep];
35     for (int i = side[now];i;i = next[i])
36         if (toit[i] != fa&&!vis[toit[i]])
37             dfs(toit[i],now,dep+1),size[now] += size[toit[i]];
38 }
39
40 inline void subdivide(int now)
41 {
42     vis[now] = true;
43     for (int i = side[now];i;i = next[i])
44     {
45         if (vis[toit[i]]) continue;
46         dfs(toit[i],now,1); ans += d[K];
47         for (int j = 1;j < K;++j) ans += d[j]*sd[K-j];
48         for (int j = 1;j <= nd;++j) sd[j] += d[j],d[j] = 0;
49         ns = max(nd,ns); nd = 0;
50     }
51     memset(sd,0,4*(ns+1)); ns = 0;
52     for (int i = side[now];i;i = next[i])
53         if (!vis[toit[i]])
54             subdivide(find_root(toit[i],size[toit[i]]));
55 }
56
57 int main()
58 {
59     freopen("D.in","r",stdin);
60     freopen("D.out","w",stdout);
61     scanf("%d %d",&N,&K);
62     for (int i = 1,a,b;i < N;++i) scanf("%d %d",&a,&b),ins(a,b);
63     large[0] = 1<<30; subdivide(find_root(1,N));
64     printf("%d",ans);
65     fclose(stdin); fclose(stdout);
66     return 0;
67 }

```

3.2 Dynamicly Divide and Conquer on Tree

```

1  // N 个点的树，每个点点权 0/1，询问两个 0 点之间最远距离，每次可以 flip 一个点的点权
2  #include<set>
3  #include<vector>
4  #include<algorithm>
5  #include<cstring>
6  #include<iostream>
7  #include<cstdio>
8  #include<cstdlib>
9  using namespace std;
10
11 const int maxn = 200010,inf = 1<<29,lhh = 4000037;

```

```

12  int
    ↪ N,cnt,nlight,tot,best,Root,side[maxn],toit[maxn],nxt[maxn],size[maxn],large[maxn],optimal[maxn];
13  int father[maxn],L[maxn],R[maxn],leaf[maxn],rechain[lhh],depth[lhh]; bool off[maxn],vis[maxn];
14  vector <int> son[maxn]; pair <int,int> Hash[lhh]; multiset <int> mx[maxn],S[maxn];
15
16  struct Value
17  {
18      int a,b;
19      inline Value() {}
20      inline Value(int _a,int _b):a(_a),b(_b) {}
21      friend inline Value operator +(const Value &x,const Value &y)
22      {
23          Value ret;
24          if (x.a > y.a)
25          {
26              ret.a = x.a;
27              if (x.b > y.a) ret.b = x.b;
28              else ret.b = y.a;
29          }
30          else
31          {
32              ret.a = y.a;
33              if (y.b > x.a) ret.b = y.b;
34              else ret.b = x.a;
35          }
36          return ret;
37      }
38  }tree[maxn*2];
39
40  inline void add(int a,int b) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; }
41  inline void ins(int a,int b) { add(a,b); add(b,a); }
42
43  inline int gi()
44  {
45      char ch; int ret = 0,f = 1;
46      do ch = getchar(); while (!(ch >= '0' && ch <= '9') && ch != '-');
47      if (ch == '-') f = -1, ch = getchar();
48      do ret = ret*10+ch-'0', ch = getchar(); while (ch >= '0' && ch <= '9');
49      return ret*f;
50  }
51
52  inline int find(const pair <int,int> &key)
53  {
54      int now = (2333*key.first+5003*key.second)%lhh;
55      while (true)
56      {
57          if (Hash[now].first == 0 || Hash[now] == key) return now;
58          else ++now;
59          if (now >= lhh) now -= lhh;

```

```

60     }
61 }
62
63 inline void getroot(int now,int rest,int fa)
64 {
65     size[now] = 1; large[now] = 0;
66     for (int i = side[now];i;i = nxt[i])
67     {
68         if (vis[toit[i]]||toit[i] == fa) continue;
69         getroot(toit[i],rest,now);
70         size[now] += size[toit[i]];
71         large[now] = max(large[now],size[toit[i]]);
72     }
73     large[now] = max(large[now],rest-size[now]);
74     if (large[now] < large[best]) best = now;
75 }
76 inline int find_root(int rest,int now) { best = 0; getroot(now,rest,0); return best; }
77
78 inline void dfs(int id,int root,int now,int fa,int dep)
79 {
80     S[id].insert(dep);
81     pair <int,int> key = make_pair(root,now); int pos = find(key);
82     rechain[pos] = id; depth[pos] = dep; Hash[pos] = key;
83     size[now] = 1;
84     for (int i = side[now];i;i = nxt[i])
85     {
86         if (vis[toit[i]]||toit[i] == fa) continue;
87         dfs(id,root,toit[i],now,dep+1); size[now] += size[toit[i]];
88     }
89 }
90
91 inline void subdivide(int root)
92 {
93     optimal[root] = -inf; mx[root].insert(-inf);
94     L[root] = tot+1;
95     for (int i = side[root];i;i = nxt[i])
96     {
97         if (vis[toit[i]]) continue;
98         ++tot; dfs(tot,root,toit[i],root,1);
99         S[tot].insert(-inf);
100     }
101     R[root] = tot; vis[root] = true;
102     for (int i = side[root];i;i = nxt[i])
103     {
104         if (vis[toit[i]]) continue;
105         int tmp = find_root(size[toit[i]],toit[i]);
106         father[tmp] = root; son[root].push_back(tmp);
107         subdivide(tmp);
108     }

```

```

109 }
110
111 inline void build(int now,int l,int r)
112 {
113     if (l == r)
114     {
115         tree[now] = Value(*S[l].rbegin(),-inf);
116         leaf[l] = now; return;
117     }
118     int mid = (l+r)>>1;
119     build(now<<1,l,mid); build(now<<1|1,mid+1,r);
120     tree[now] = tree[now<<1]+tree[now<<1|1];
121 }
122
123 inline Value query(int now,int l,int r,int ql,int qr)
124 {
125     if (l == ql&&qr == r) return tree[now];
126     int mid = (l+r)>>1;
127     if (qr <= mid) return query(now<<1,l,mid,ql,qr);
128     else if (ql > mid) return query(now<<1|1,mid+1,r,ql,qr);
129     else return query(now<<1,l,mid,ql,mid)+query(now<<1|1,mid+1,r,mid+1,qr);
130 }
131
132 inline void upd(int &a,int b) { if (a < b) a = b; }
133
134 inline void modify(int pos,int dep,bool sign)
135 {
136     if (sign) S[pos].insert(dep); else S[pos].erase(S[pos].find(dep));
137     tree[leaf[pos]] = Value(*S[pos].rbegin(),-inf);
138     for (int now = leaf[pos]>>1;now;now >>= 1)
139         tree[now] = tree[now<<1]+tree[now<<1|1];
140 }
141 inline void modify(int pos)
142 {
143     int c = 0;
144     if (father[pos]) mx[father[pos]].erase(mx[father[pos]].find(optimal[pos]));
145     off[pos] ^= 1; if (off[pos]) nlight++; else nlight--;
146     if (L[pos] <= R[pos])
147     {
148         Value res = query(1,1,tot,L[pos],R[pos]);
149         optimal[pos] = max(res.a+res.b,*mx[pos].rbegin());
150         if (off[pos]) upd(optimal[pos],res.a);
151     }
152     if (father[pos]) mx[father[pos]].insert(optimal[pos]);
153     for (int now = father[pos];now;now = father[now])
154     {
155         int t = find(make_pair(now,pos));
156         int id = rechain[t],dep = depth[t];
157         modify(id,dep,off[pos]);

```

```

158         if (father[now]) mx[father[now]].erase(mx[father[now]].find(optimal[now]));
159         Value res = query(1,1,tot,L[now],R[now]);
160         optimal[now] = max(res.a+res.b,*mx[now].rbegin());
161         if (off[now]) upd(optimal[now],res.a);
162         if (father[now]) mx[father[now]].insert(optimal[now]);
163         ++c;
164     }
165 }
166
167 inline void redfs(int now)
168 {
169     for (int i = 0; i < (int)son[now].size(); ++i)
170         redfs(son[now][i]), mx[now].insert(optimal[son[now][i]]);
171     if (L[now] <= R[now])
172     {
173         Value res = query(1,1,tot,L[now],R[now]);
174         optimal[now] = max(res.a+res.b,*mx[now].rbegin());
175         if (off[now]) upd(optimal[now],res.a);
176     }
177 }
178
179 int main()
180 {
181     // freopen("A.in", "r", stdin);
182     memset(off, true, sizeof off);
183     nlight = N = gi();
184     for (int i = 1; i < N; ++i) ins(gi(), gi());
185     large[0] = inf;
186     subdivide(Root = find_root(N, 1));
187     build(1, 1, tot); redfs(Root);
188     for (int Q = gi(); Q--;)
189     {
190         char opt; do opt = getchar(); while (opt != 'G' && opt != 'C');
191         if (opt == 'G')
192         {
193             if (!nlight) puts("-1");
194             else if (nlight == 1) puts("0");
195             else printf("%d\n", optimal[Root]);
196         }
197         else modify(gi());
198     }
199     return 0;
200 }

```

3.3 Heavy Path Decomposition

```

1  int side[maxn], toit[maxn<<1], nxt[maxn<<1], nxt[maxn<<1];
2  int timestamp, father[maxn], dfn[maxn], redfn[maxn], top[maxn], size[maxn];
3

```



```

4  void decompose(int now,int tp)
5  {
6      redfn[dfn[now] = ++timestamp] = now;
7      top[now] = tp; int heavy = 0;
8      for (int i = side[now];i;i = nxt[i])
9          if (toit[i] != father[now]&&size[toit[i]] > size[heavy]) heavy = toit[i];
10     if (!heavy) return; decompose(heavy,tp);
11     for (int i = side[now];i;i = nxt[i])
12         if (toit[i] != father[now]&&toit[i] != heavy) decompose(toit[i],toit[i]);
13 }
14
15 void dfs(int now)
16 {
17     size[now] = 1;
18     for (int i = side[now];i;i = nxt[i])
19         if (toit[i] != father[now])
20             {
21                 father[toit[i]] = now,con[toit[i]] = i;
22                 dep[toit[i]] = dep[now]+1,dfs(toit[i]);
23                 size[now] += size[toit[i]] ;
24             }
25 }
26
27 // 对点操作
28 inline int query(int a,int b)
29 {
30     int ret = -inf;
31     while (top[a] != top[b])
32     {
33         if (dep[top[a]] < dep[top[b]]) swap(a,b);
34         ret = max(ret,ask(1,1,N,dfn[top[a]],dfn[a]));
35         a = father[top[a]];
36     }
37     if (dep[a] < dep[b]) swap(a,b);
38     ret = max(ret,query(1,1,N,dfn[b],dfn[a]));
39     return ret;
40 }
41
42 // 对边操作
43 inline int query(int a,int b)
44 {
45     int ret = -inf;
46     while (top[a] != top[b])
47     {
48         if (dep[top[a]] < dep[top[b]]) swap(a,b);
49         ret = max(ret,ask(1,1,N,dfn[top[a]],dfn[a]));
50         a = father[top[a]];
51     }
52     if (a == b) return ret;

```

```

53     if (dep[a] < dep[b]) swap(a,b);
54     ret = max(ret,ask(1,1,N,dfn[b]+1,dfn[a]));
55     return ret;
56 }

```

3.4 K-Dimension Tree

```

1  struct Point
2  {
3      double x,y; int id;
4      inline Point() = default;
5      inline Point(double _x,double _y,int _id):x(_x),y(_y),id(_id) {}
6      inline void read(int i = 0) { scanf("%lf %lf",&x,&y); id = i; }
7      inline double norm() { return sqrt(x*x+y*y); }
8      friend inline Point operator+(const Point &a,const Point &b) { return
↪ Point(a.x+b.x,a.y+b.y); }
9      friend inline Point operator-(const Point &a,const Point &b) { return
↪ Point(a.x-b.x,a.y-b.y); }
10     friend inline double operator*(const Point &a,const Point &b) { return a.x*b.x+a.y*b.y; }
11     friend inline double operator/(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
12 }P[maxn];
13
14 struct Rectangle
15 {
16     double lx,rx,ly,ry;
17     inline Rectangle() = default;
18     inline Rectangle(double _lx,double _rx,double _ly,double
↪ _ry):lx(_lx),rx(_rx),ly(_ly),ry(_ry) {}
19     inline void set(const Point &p) { lx = rx = p.x; ly = ry = p.y; }
20     inline void merge(const Point &p)
21     {
22         lx = min(lx,p.x); rx = max(rx,p.x);
23         ly = min(ly,p.y); ry = max(ry,p.y);
24     }
25     inline void merge(const Rectangle &r)
26     {
27         lx = min(lx,r.lx); rx = max(rx,r.rx);
28         ly = min(ly,r.ly); ry = max(ry,r.ry);
29     }
30     // 最小距离, 到 4 个角和 4 条边距离
31     inline double dist(const Point &p)
32     {
33         if (p.x <= lx&&p.y <= ly) return (p-Point(lx,ly)).norm();
34         else if (p.x <= rx&&p.y <= ly) return p.y-ly;
35         else if (p.x >= rx&&p.y <= ly) return (p-Point(rx,ly)).norm();
36         else if (p.x >= rx&&p.y <= ry) return p.x-rx;
37         else if (p.x >= rx&&p.y >= ry) return (p-Point(rx,ry)).norm();
38         else if (p.x >= lx&&p.y >= ry) return p.y-ry;
39         else if (p.x <= lx&&p.y >= ry) return (p-Point(lx,ry)).norm();

```

```

40         else if (p.x <= lx&& p.y >= ly) return p.x-lx;
41         return 0;
42     }
43     // 最大距离, 到 4 个角的距离
44     inline double dist(const Point &p)
45     {
46         double ret = 0;
47         ret += max((rx-p.x)*(rx-p.x), (lx-p.x)*(lx-p.x));
48         ret += max((ry-p.y)*(ry-p.y), (ly-p.y)*(ly-p.y));
49         return ret;
50     }
51 };
52
53 struct Node
54 {
55     int child[2]; Point p; Rectangle r;
56     inline Node() = default;
57     inline Node(const Point &p, const Rectangle &r): p(_p), r(_r) { r.set(p); memset(child, 0, 8);
↪ }
58     inline void set(const Point &p) { p = _p; r.set(p); memset(child, 0, 8); }
59 }tree[maxn];
60
61 inline bool cmpx(const Point &a, const Point &b)
62 {
63     if (a.x != b.x) return a.x < b.x;
64     else return a.y < b.y;
65 }
66 inline bool cmpy(const Point &a, const Point &b)
67 {
68     if (a.y != b.y) return a.y < b.y;
69     else return a.x < b.x;
70 }
71
72 inline bool cmp(pair <double, int> a, pair <double, int> b)
73 {
74     int sgn = dcmp(a.first-b.first);
75     if (sgn) return sgn < 0;
76     else return a.second < b.second;
77 }
78
79 // 查询 k 大/小
80 inline void query(int now, const Point &p, int k, pair <double, int> ret[], bool dim = false)
81 {
82     if (dcmp(tree[now].r.dist(p)-ret[k].first) > 0) return;
83     pair <double, int> val = make_pair((p-tree[now].p).norm(), tree[now].p.id);
84     for (int i = 1; i <= k; ++i)
85         if (cmp(val, ret[i]))
86             {
87                 for (int j = k+1; j > i; --j) ret[j] = ret[j-1];

```

```

88         ret[i] = val; break;
89     }
90     if ((dim&&cmpx(p,tree[now].p))||(!dim&&cmpy(p,tree[now].p)))
91     {
92         if (tree[now].child[0]) query(tree[now].child[0],p,k,ret,dim^1);
93         if (tree[now].child[1]) query(tree[now].child[1],p,k,ret,dim^1);
94     }
95     else
96     {
97         if (tree[now].child[1]) query(tree[now].child[1],p,k,ret,dim^1);
98         if (tree[now].child[0]) query(tree[now].child[0],p,k,ret,dim^1);
99     }
100 }
101
102 // 查询最小/大
103 inline void query(int x,const Point &p,pair <double,int> ret,bool dim = false)
104 {
105     if (dcmp(tree[now].r.disp(p)-ret.first) > 0) return;
106     pair <double,int> val = make_pair((p-tree[now].p).norm(),tree[now].p.id);
107     if (cmp(val,ret)) ret = val;
108     if ((dim&&cmpx(p,tree[now].p))||(!dim&&cmpy(p,tree[now].p)))
109     {
110         if (tree[now].child[0]) query(tree[now].child[0],p,ret,dim^1);
111         if (tree[now].child[1]) query(tree[now].child[1],p,ret,dim^1);
112     }
113     else
114     {
115         if (tree[now].child[1]) query(tree[now].child[1],p,ret,dim^1);
116         if (tree[now].child[0]) query(tree[now].child[0],p,ret,dim^1);
117     }
118 }
119
120 inline int build(int l,int r,bool dim)
121 {
122     int now = ++size,mid = (l+r)>>1;
123     nth_element(vec.begin()+l-1,vec.begin()+mid-1,vec.begin()+r,dim?cmpx:cmpy);
124     tree[now].set(vec[mid-1]);
125     if (l < mid)
126     {
127         tree[now].child[0] = build(l,mid-1,dim^1);
128         tree[now].r.merge(tree[tree[now].child[0]].r);
129     }
130     if (r > mid)
131     {
132         tree[now].child[1] = build(mid+1,r,dim^1);
133         tree[now].r.merge(tree[tree[now].child[1]].r);
134     }
135     return now;
136 }

```

3.5 Leftlist Tree

```

1  // It's correct, but it needs be rewritten.
2  #include<iostream>
3  #include<cstdio>
4  #include<cstdlib>
5  using namespace std;
6
7  #define maxn (600010)
8  int N,M,root[maxn],size[maxn],v[maxn],dep[maxn],l[maxn],r[maxn],tot;
9
10 inline int Merge(int x,int y)
11 {
12     if (!x||!y) return x+y;
13     if (v[x]>v[y]) swap(x,y);
14     r[x] = Merge(r[x],y);
15     if (dep[l[x]] < dep[r[x]]) swap(l[x],r[x]);
16     dep[x] = dep[r[x]]+1;
17     return x;
18 }
19 inline int Init(int x) { v[++tot] = x; l[tot] = r[tot] = dep[tot] = 0; return tot;}
20 inline int Insert(int x,int y) { return Merge(x,Init(y)); }
21 inline int pop(int x) { return Merge(l[x],r[x]); }
22
23 inline int read()
24 {
25     char ch; int f = 1,ret = 0;
26     do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
27     if (ch == '-') f = -1,ch = getchar();
28     do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
29     return ret*f;
30 }
31
32 int main()
33 {
34     freopen("1050.in","r",stdin);
35     freopen("1050.out","w",stdout);
36     scanf("%d %d",&N,&M);
37     for (int i = 1;i <= N;++i) root[i] = Init(read()),size[i] = 1;
38     while (M--)
39     {
40         int opt = read();
41         if (!opt)
42         {
43             int a = read()+1,b = read()+1;
44             if (size[b]) root[a] = Merge(root[a],root[b]);
45             size[a] += size[b]; size[b] = 0;
46         }
47         else if (opt == 1)
48         {

```

```

49         int a = read()+1;
50         if (!size[a]) puts("-1");
51         else printf("%d\n",v[root[a]]),root[a] = pop(root[a]),--size[a];
52     }
53     else
54     {
55         int a = read()+1; ++size[a];
56         root[a] = Insert(root[a],read());
57     }
58 }
59 fclose(stdin); fclose(stdout);
60 return 0;
61 }

```

3.6 Link Cut Tree

```

1  inline bool isroot(int a) { return ch[fa[a]][0] != a&&ch[fa[a]][1] != a; }
2
3  inline void update(int x) { val[x] = (val[ch[x][0]]+val[ch[x][1]]).merge(x); }
4  inline void pushdown(int x)
5  {
6      if (rev[x])
7      {
8          int &lc = ch[x][0],&rc = ch[x][1];
9          swap(lc,rc);
10         if (lc) rev[lc] ^= 1;
11         if (rc) rev[rc] ^= 1;
12         rev[x] = false;
13     }
14 }
15
16 inline void rotate(int x)
17 {
18     int y = fa[x],z = fa[y],l = ch[y][1] == x,r = l^1;
19     if (!isroot(y)) ch[z][ch[z][1] == y] = x; fa[x] = z;
20     if (ch[x][r]) fa[ch[x][r]] = y; ch[y][1] = ch[x][r];
21     fa[y] = x; ch[x][r] = y; update(y); update(x);
22 }
23 inline void splay(int x)
24 {
25     int top = 0,i;
26     for (i = x;!isroot(i);i = fa[i]) stk[++top] = i; stk[++top] = i;
27     while (top) pushdown(stk[top--]);
28     while (!isroot(x))
29     {
30         int y = fa[x],z = fa[y];
31         if (!isroot(y))
32         {
33             if ((ch[y][0] == x)^(ch[z][0] == y)) rotate(x);

```

```

34         else rotate(y);
35     }
36     rotate(x);
37 }
38 }
39
40 inline int access(int x)
41 {
42     int t = 0;
43     for (t = 0; x; t = x, x = fa[x])
44         splay(x), ch[x][1] = t, update(x);
45     return t;
46 }
47 inline int evert(int x) { int t; rev[t = access(x)] ^= 1; return t; }
48 inline int find(int x)
49 {
50     x = access(x);
51     while (pushdown(x), ch[x][0]) x = ch[x][0];
52     return x;
53 }
54 inline void cut(int x, int y)
55 {
56     evert(x); access(y); splay(y);
57     if (ch[y][0] != x || ch[x][1] != 0) return;
58     ch[y][0] = fa[x] = 0; update(x); update(y);
59 }
60 inline void link(int x, int y) { fa[evert(x)] = y; }
61
62
63 // Magic Forest
64 #include<algorithm>
65 #include<cstring>
66 #include<iostream>
67 #include<cstdio>
68 #include<cstdlib>
69 using namespace std;
70
71 const int maxn = 200010, inf = 1<<29;
72 int N, M, A[maxn], B[maxn], fa[maxn], ch[maxn][2];
73 int stk[maxn], ans = inf; bool rev[maxn];
74
75 struct Value
76 {
77     int ma, mb, id;
78     inline Value(int _ma = 0, int _mb = 0, int _id = 0) : ma(_ma), mb(_mb), id(_id) {}
79     friend inline Value operator +(const Value &a, const Value &b)
80     {
81         Value ret = Value(max(a.ma, b.ma), max(a.mb, b.mb), a.id);
82         if (B[a.id] < B[b.id]) ret.id = b.id;

```

```

83         return ret;
84     }
85     inline Value merge(int i)
86     {
87         Value ret = Value(max(ma,A[i]),max(mb,B[i]),id);
88         if (B[i] > B[id]) ret.id = i;
89         return ret;
90     }
91 }val[maxn];
92
93 inline int gi()
94 {
95     char ch; int ret = 0,f = 1;
96     do ch = getchar(); while (!(ch >= '0' && ch <= '9') && ch != '-');
97     if (ch == '-') f = -1, ch = getchar();
98     do ret = ret*10+ch-'0', ch = getchar(); while (ch >= '0' && ch <= '9');
99     return ret*f;
100 }
101
102 struct Edge
103 {
104     int x,y,a,b;
105     inline Edge(int _x = 0,int _y = 0,int _a = 0,int _b = 0):x(_x),y(_y),a(_a),b(_b) {}
106     inline void read() { x = gi(),y = gi(),a = gi(),b = gi(); }
107     friend inline bool operator <(const Edge &s,const Edge &t) { return s.a < t.a; }
108 }edge[maxn];
109
110 inline bool isroot(int a) { return ch[fa[a]][0] != a && ch[fa[a]][1] != a; }
111
112 inline void update(int x) { val[x] = (val[ch[x][0]]+val[ch[x][1]]).merge(x); }
113 inline void pushdown(int x)
114 {
115     if (rev[x])
116     {
117         int &lc = ch[x][0],&rc = ch[x][1];
118         swap(lc,rc);
119         if (lc) rev[lc] ^= 1;
120         if (rc) rev[rc] ^= 1;
121         rev[x] = false;
122     }
123 }
124
125 inline void rotate(int x)
126 {
127     int y = fa[x],z = fa[y],l = ch[y][1] == x,r = l^1;
128     if (!isroot(y)) ch[z][ch[z][1] == y] = x; fa[x] = z;
129     if (ch[x][r]) fa[ch[x][r]] = y; ch[y][l] = ch[x][r];
130     fa[y] = x; ch[x][r] = y; update(y); update(x);
131 }

```



```

132 inline void splay(int x)
133 {
134     int top = 0,i;
135     for (i = x;!isroot(i);i = fa[i]) stk[++top] = i; stk[++top] = i;
136     while (top) pushdown(stk[top--]);
137     while (!isroot(x))
138     {
139         int y = fa[x],z = fa[y];
140         if (!isroot(y))
141         {
142             if ((ch[y][0] == x)^(ch[z][0] == y)) rotate(x);
143             else rotate(y);
144         }
145         rotate(x);
146     }
147 }
148
149 inline int access(int x)
150 {
151     int t = 0;
152     for (t = 0;x;t = x,x = fa[x])
153         splay(x),ch[x][1] = t,update(x);
154     return t;
155 }
156 inline int evert(int x) { int t; rev[t = access(x)] ^= 1; return t; }
157 inline int find(int x)
158 {
159     x = access(x);
160     while (pushdown(x),ch[x][0]) x = ch[x][0];
161     return x;
162 }
163 inline void cut(int x,int y)
164 {
165     evert(x); access(y); splay(y);
166     if (ch[y][0] != x||ch[x][1] != 0) return;
167     ch[y][0] = fa[x] = 0; update(x); update(y);
168 }
169 inline void link(int x,int y) { fa[evert(x)] = y; }
170
171 inline Value query(int x,int y) { evert(x); return val[access(y)]; }
172
173 int main()
174 {
175     // freopen("D.in","r",stdin);
176     N = gi(),M = gi();
177     for (int i = 1;i <= M;++i) edge[i].read();
178     sort(edge+1,edge+M+1);
179     for (int i = 0;i <= N;++i)
180         A[i] = B[i] = -inf,val[i] = Value(A[i],B[i],i);

```

```

181     for (int i = 1; i <= M; ++i)
182         A[i+N] = edge[i].a, B[i+N] = edge[i].b, val[i+N] = Value(A[i+N], B[i+N], i+N);
183     for (int i = 1; i <= M; ++i)
184     {
185         if (edge[i].x == edge[i].y) continue;
186         if (find(edge[i].x) == find(edge[i].y))
187         {
188             Value res = query(edge[i].x, edge[i].y); int id = res.id - N;
189             if (edge[i].b < edge[id].b)
190             {
191                 cut(edge[id].x, id+N), cut(edge[id].y, id+N);
192                 link(edge[i].x, i+N), link(edge[i].y, i+N);
193             }
194         }
195         else link(edge[i].x, i+N), link(i+N, edge[i].y);
196         if (find(1) == find(N))
197         {
198             Value res = query(1, N);
199             ans = min(ans, res.ma + res.mb);
200         }
201     }
202     if (ans == inf) ans = -1;
203     printf("%d\n", ans);
204     return 0;
205 }

```

3.7 Merge Split Treap

```

1  // jisuanke17123
2  // Warning: 给指针赋值时, 不要赋 this, 因为 this 是临时变量的地址
3  #include <queue>
4  #include <algorithm>
5  #include <cstring>
6  #include <iostream>
7  #include <cstdio>
8  #include <cstdlib>
9  #include <set>
10 using namespace std;
11
12 typedef long long ll;
13 const int maxn = 1000010;
14 int N;
15
16 inline int rnd() { static int x = 1; return x = (x*23333+233); }
17 inline int rnd(int n) { int x = rnd(); if (x < 0) x = -x; return x%n+1; }
18
19 struct Node
20 {
21     int size, key, val; Node *mn, *ch[2];

```

```

22     inline Node *update()
23     {
24         mn = this; size = 1;
25         if (ch[0])
26         {
27             size += ch[0]->size;
28             if (ch[0]->mn->val < mn->val) mn = ch[0]->mn;
29         }
30         if (ch[1])
31         {
32             size += ch[1]->size;
33             if (ch[1]->mn->val < mn->val) mn = ch[1]->mn;
34         }
35         return this;
36     }
37     inline Node() = default;
38     inline Node(int v, Node *_mn):size(1),key(rnd()),val(v),mn(_mn) { ch[0] = ch[1] = NULL; }
39 }pool[maxn*50/4],*root[maxn],*cur;
40 struct Status
41 {
42     int l,r; ll val;
43     inline Status() = default;
44     inline Status(int _l,int _r,ll _val):l(_l),r(_r),val(_val) {}
45     friend inline bool operator <(const Status &a,const Status &b) { return a.val > b.val; }
46 };
47
48 inline int sz(const Node *x) { if (x == NULL) return 0; else return x->size; }
49
50 inline Node *newnode(int v = 0) { *cur = Node(v,cur); return cur++; }
51
52 Node *insert(Node *p,Node *q)
53 {
54     if (p == NULL&&q == NULL) return NULL;
55     if (p == NULL||q == NULL) return p?p:q;
56     Node *u = NULL;
57     if (rnd(sz(p)+sz(q)) < sz(p))
58         u = p,u->ch[1] = insert(u->ch[1],q);
59     else u = q,u->ch[0] = insert(p,u->ch[0]);
60     return u->update();
61 }
62
63 Node *merge(Node *p,Node *q)
64 {
65     if (p == NULL&&q == NULL) return NULL;
66     Node *u = newnode();
67     if (p == NULL||q == NULL) return p?p:q;
68     if (rnd(sz(p)+sz(q)) < sz(p))
69         *u = *p,u->ch[1] = merge(u->ch[1],q);
70     else *u = *q,u->ch[0] = merge(p,u->ch[0]);

```

```

71     return u->update();
72 }
73
74 Node *split(Node *u, int l, int r)
75 {
76     if (l > r || u == NULL) return 0;
77     Node *x = NULL;
78     if (l == 1 && r == sz(u))
79     {
80         x = newnode(); *x = *u;
81         return x->update();
82     }
83     int lsz = sz(u->ch[0]);
84     if (r <= lsz) return split(u->ch[0], l, r);
85     if (l > lsz+1) return split(u->ch[1], l-lsz-1, r-lsz-1);
86     x = newnode(); *x = *u;
87     x->ch[0] = split(u->ch[0], l, lsz);
88     x->ch[1] = split(u->ch[1], 1, r-lsz-1);
89     return x->update();
90 }
91
92 inline int gi()
93 {
94     char ch; int ret = 0, f = 1;
95     do ch = getchar(); while (!(ch >= '0' && ch <= '9') && ch != '-');
96     if (ch == '-') f = -1, ch = getchar();
97     do ret = ret*10+ch-'0', ch = getchar(); while (ch >= '0' && ch <= '9');
98     return ret*f;
99 }
100
101 int get_pos(Node *rt, Node *mn)
102 {
103     if (rt == mn) return sz(rt->ch[0]);
104     else if (rt->ch[0] && rt->ch[0]->mn == mn)
105         return get_pos(rt->ch[0], mn);
106     else return sz(rt->ch[0])+1+get_pos(rt->ch[1], mn);
107 }
108 inline pair <int, int> Qmin(Node *rt, int l, int r)
109 {
110     if (l > r) return make_pair(-1, -1);
111     Node *v = split(rt, l, r);
112     auto ret = make_pair(v->mn->val, get_pos(v, v->mn)+1);
113     return ret;
114 }
115 inline int get(Node *u, int x) { return split(u, x, x)->val; }
116
117 inline void work(Node *rt, int k)
118 {
119     int n = sz(rt);

```

```

120     set <int> S; S.insert(0); S.insert(n+1);
121     priority_queue <Status> heap;
122     auto tmp = Qmin(rt,1,n);
123     heap.push(Status(tmp.second,tmp.second,tmp.first));
124     while (k--)
125     {
126         auto now = heap.top(); heap.pop();
127         printf("%lld\n",now.val);
128         if (now.l == now.r)
129         {
130             S.insert(now.l);
131             auto it = S.find(now.l);
132             int pre = *(--it);
133             int nxt = *++(++it);
134             auto ls = Qmin(rt,pre+1,now.l-1);
135             auto rs = Qmin(rt,now.l+1,nxt-1);
136             if (ls.first != -1)
137                 heap.push(Status(ls.second,ls.second,ls.first));
138             if (rs.first != -1)
139                 heap.push(Status(rs.second,rs.second,rs.first));
140         }
141         if (now.r < n)
142         {
143             int inc = get(rt,now.r+1);
144             ++now.r; now.val += (ll)inc;
145             heap.push(now);
146         }
147     }
148 }
149
150 inline void init() { N = 0; cur = pool; }
151
152 int main()
153 {
154     freopen("J.in","r",stdin);
155     for (int T = gi();T--;)
156     {
157         init();
158         for (int Q = gi();Q--;)
159         {
160             int opt = gi();
161             if (opt == 1)
162             {
163                 root[++N] = NULL;
164                 for (int n = gi();n--;)
165                     root[N] = insert(root[N],newnode(gi()));
166             }
167             else if (opt == 2)
168             {

```

```

169         root[++N] = NULL;
170         int x = gi(), l1 = gi(), r1 = gi(), y = gi(), l2 = gi(), r2 = gi();
171         Node *ls = split(root[x], l1, r1);
172         Node *rs = split(root[y], l2, r2);
173         root[N] = merge(ls, rs);
174     }
175     else
176     {
177         int x = gi(), k = gi();
178         work(root[x], k);
179     }
180 }
181 }
182 return 0;
183 }
184
185 // By zky. To be rewritten.
186 const int mo=1e9+7;
187 int rnd(){
188     static int x=1;
189     return x=(x*23333+233);
190 }
191 int rnd(int n){
192     int x=rnd();
193     if(x<0)x=-x;
194     return x%n+1;
195 }
196 struct node{
197     int siz, key;
198     int val;
199     LL sum;
200     node *c[2];
201     node* rz(){
202         sum=val; siz=1;
203         if(c[0])sum+=c[0]->sum, siz+=c[0]->siz;
204         if(c[1])sum+=c[1]->sum, siz+=c[1]->siz;
205         return this;
206     }
207     node(){ }
208     node(int v){
209         siz=1; key=rnd();
210         val=v; sum=v;
211         c[0]=c[1]=0;
212     }
213
214 }pool[maxn*8], *root, *cur=pool, *old_root, *stop;
215 node *newnode(int v=0){
216     *cur=node(v);
217     return cur++;

```

```

218 }
219 node *old_merge(node *p,node *q){
220     if(!p&&!q)return 0;
221     node *u=0;
222     if(!p||!q)return u=p?p->rz():(q?q->rz():0);
223     if(rnd(sz(p)+sz(q))<sz(p)){
224         u=p;
225         u->c[1]=old_merge(u->c[1],q);
226     }else{
227         u=q;
228         u->c[0]=old_merge(p,u->c[0]);
229     }
230     return u->rz();
231 }
232 node *merge(node *p,node *q){
233     if(!p&&!q)return 0;
234     node *u=newnode();
235     if(!p||!q)return u=p?p->rz():(q?q->rz():0);
236     if(rnd(sz(p)+sz(q))<sz(p)){
237         *u=*p;
238         u->c[1]=merge(u->c[1],q);
239     }else{
240         *u=*q;
241         u->c[0]=merge(p,u->c[0]);
242     }
243     return u->rz();
244 }
245 node *split(node *u,int l,int r){
246     if(l>r||!u)return 0;
247     node *x=0;
248     if(l==1&&r==sz(u)){
249         x=newnode();
250         *x=*u;
251         return x->rz();
252     }
253     int lsz=sz(u->c[0]);
254     if(r<=lsz)
255         return split(u->c[0],l,r);
256     if(l>lsz+1)
257         return split(u->c[1],l-lsz-1,r-lsz-1);
258     x=newnode();
259     *x=*u;
260     x->c[0]=split(u->c[0],l,lsz);
261     x->c[1]=split(u->c[1],1,r-lsz-1);
262     return x->rz();
263 }

```

3.8 Modui Algorithm on Tree

```

1  // 询问树上路径元素 mex, inc dec 复杂度不对, 需要用线段树/set(带 log) 或者分块 (修改 O(1))
2  // 若包括 lca, 每组询问需要把 lca 补 (inc) 上去。
3  #include<cstdio>
4  #include<cstdlib>
5  #include<algorithm>
6  #include<cstring>
7  #include<iostream>
8  using namespace std;
9
10 const int Size = 337,maxn = 200010;
11 int N,Q,cnt,nxt[maxn],side[maxn],len[maxn],toit[maxn],f[maxn][20],key[maxn],timestamp;
12 int dep[maxn],L[maxn],R[maxn],dfn[maxn],ans[maxn],exist[maxn],show[maxn],res;
13
14 inline void add(int a,int b,int c) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b;
    ↪ len[cnt] = c; }
15 inline void ins(int a,int b,int c) { add(a,b,c); add(b,a,c); }
16
17 void dfs(int now)
18 {
19     dfn[L[now] = ++timestamp] = now;
20     for (int i = 1;(1<<i) <= dep[now];++i) f[now][i] = f[f[now][i-1]][i-1];
21     for (int i = side[now];i;i = nxt[i])
22     {
23         if (toit[i] == f[now][0]) continue;
24         f[toit[i]][0] = now; key[toit[i]] = len[i];
25         dep[toit[i]] = dep[now]+1;
26         dfs(toit[i]);
27     }
28     dfn[R[now] = ++timestamp] = now;
29 }
30
31 inline int jump(int a,int b) { for (int i = 0;b;b >= 1,++i) if (b&1) a = f[a][i]; return a; }
32 inline int lca(int a,int b)
33 {
34     if (dep[a] < dep[b]) swap(a,b);
35     a = jump(a,dep[a]-dep[b]);
36     if (a == b) return a;
37     for (int i = 0;i >= 0;)
38     {
39         if (f[a][i] == f[b][i]) --i;
40         else a = f[a][i],b = f[b][i],++i;
41     }
42     return f[a][0];
43 }
44
45 struct Node
46 {
47     int a,b,c,id;

```



```

48     Node() = default;
49     Node(int _a,int _b,int _c = 0,int _id = 0):a(_a),b(_b),c(_c),id(_id) {}
50     inline void read(int i)
51     {
52         id = i; scanf("%d %d",&a,&b); c = lca(a,b);
53         if (c == a||c == b) { if (a != c) swap(a,b); a = L[c]+1; b = L[b]; }
54         else { if (L[a] > L[b]) swap(a,b); a = R[a]; b = L[b]; }
55     }
56     friend inline bool operator <(const Node &x,const Node &y) { return x.b < y.b; }
57 }query[maxn];
58
59 inline bool cmp(const Node &x,const Node &y) { return x.a < y.a; }
60
61 inline void inc(int id)
62 {
63     if (key[id] >= maxn) return;
64     ++exist[key[id]];
65     while (exist[res]) ++res;
66 }
67 inline void dec(int id)
68 {
69     if (key[id] >= maxn) return;
70     --exist[key[id]];
71     if (key[id] < res&&!exist[key[id]]) res = key[id];
72 }
73
74 inline void work()
75 {
76     int l = 1,r = 0;
77     for (int i = 1;i <= Q;++i)
78     {
79         while (r < query[i].b)
80         {
81             show[dfn[++r]]++;
82             if (show[dfn[r]] == 2) dec(dfn[r]); else inc(dfn[r]);
83         }
84         while (l > query[i].a)
85         {
86             show[dfn[--l]]++;
87             if (show[dfn[l]] == 2) dec(dfn[l]); else inc(dfn[l]);
88         }
89         while (r > query[i].b)
90         {
91             if (show[dfn[r]] == 1) dec(dfn[r]); else inc(dfn[r]);
92             show[dfn[r--]]--;
93         }
94         while (l < query[i].a)
95         {
96             if (show[dfn[l]] == 1) dec(dfn[l]); else inc(dfn[l]);

```

```

97         show[dfn[l++]]--;
98     }
99     ans[query[i].id] = res;
100 }
101 }
102
103 int main()
104 {
105     freopen("F.in", "r", stdin);
106     scanf("%d %d", &N, &Q);
107     for (int i = 1; i < N; ++i)
108     {
109         int a, b, c;
110         scanf("%d %d %d", &a, &b, &c);
111         ins(a, b, c);
112     }
113     dfs(1);
114     for (int i = 1; i <= Q; ++i) query[i].read(i);
115     sort(query+1, query+Q+1, cmp);
116     for (int i = 1, j; i <= Q; i = j)
117     {
118         for (j = i; j <= Q && query[j].a - query[i].a <= Size; ++j);
119         sort(query+i+1, query+j);
120     }
121     work();
122     for (int i = 1; i <= Q; ++i) printf("%d\n", ans[i]);
123     return 0;
124 }

```

3.9 Modui Algorithm without Deletion

```

1  // r 单调右移, l 只会在  $\sqrt{N}$  中移动, 保证每次 undo 的复杂度可行即可。
2  // CodeForces 620F
3  #include<vector>
4  #include<algorithm>
5  #include<cstring>
6  #include<iostream>
7  #include<cstdio>
8  #include<cstdlib>
9  using namespace std;
10
11 const int maxn = 1000010, len = 200, inf = 1<<29;
12 int N, M, pre[maxn], A[maxn], ans[maxn];
13
14 inline int gi()
15 {
16     char ch; int ret = 0, f = 1;
17     do ch = getchar(); while (!(ch >= '0' && ch <= '9') && ch != '-');
18     if (ch == '-') f = -1, ch = getchar();

```

```

19     do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
20     return ret*f;
21 }
22
23 inline void upd(int &a,int b) { if (a < b) a = b; }
24
25 struct Trie
26 {
27     int nxt[maxn][2],val[maxn],root,cnt; vector < pair<int,int> > vec;
28     inline int newnode() { val[++cnt] = inf; memset(nxt[cnt],0,8); return cnt; }
29     inline void init() { val[0] = inf; cnt = 0; root = newnode(); }
30
31     inline int query(int key,int num)
32     {
33         int now = root,ret = 0;
34         for (int i = 19;i >= 0;--i)
35         {
36             int dir = !(num&(1<<i));
37             if (val[nxt[now][dir]] <= key)
38                 ret |= (1<<i),now = nxt[now][dir];
39             else now = nxt[now][dir^1];
40         }
41         return ret;
42     }
43
44     inline void insert(int key,int num,int mode) { insert(root,19,key,num,mode); }
45     inline void insert(int &now,int dep,int key,int num,int mode)
46     {
47         if (!now) now = newnode();
48         if (dep < 0)
49         {
50             if (mode) vec.push_back(make_pair(num,val[now]));
51             val[now] = min(val[now],key); return;
52         }
53         insert(nxt[now][(num&(1<<dep)) > 0],dep-1,key,num,mode);
54         val[now] = min(val[nxt[now][0]],val[nxt[now][1]]);
55     }
56
57     inline void change(int now,int dep,int num,int v)
58     {
59         if (dep < 0) { val[now] = v; return; }
60         change(nxt[now][(num&(1<<dep)) > 0],dep-1,num,v);
61         val[now] = min(val[nxt[now][0]],val[nxt[now][1]]);
62     }
63
64     inline void undo()
65     {
66         reverse(vec.begin(),vec.end());
67         for (auto x:vec) change(root,19,x.first,x.second);

```

```

68         vec.clear();
69     }
70 }tree1,tree2;
71
72 struct Query
73 {
74     int l,r,id;
75     inline void read(int i) { l = gi(),r = gi(),id = i; }
76     friend inline bool operator <(const Query &a,const Query &b) { return a.l < b.l; }
77 }query[maxn];
78 inline bool cmp(const Query &a,const Query &b) { return a.r < b.r; }
79
80 inline void work(int l,int r)
81 {
82     int lim = query[r].l;
83     sort(query+l,query+r+1,cmp);
84     tree1.init(); tree2.init();
85     for (int i = l;i <= r;++i)
86     {
87         if (query[i].r <= lim)
88         {
89             for (int j = query[i].l;j <= query[i].r;++j)
90             {
91                 tree1.insert(A[j],pre[A[j]-1],false);
92                 tree2.insert(-A[j],pre[A[j]],false);
93                 upd(ans[query[i].id],tree1.query(A[j],pre[A[j]]));
94                 upd(ans[query[i].id],tree2.query(-A[j],pre[A[j]-1]));
95             }
96             tree1.init(),tree2.init();
97         }
98         else
99         {
100             int pos = lim,mx = 0;
101             for (;i <= r;++i)
102             {
103                 while (pos < query[i].r)
104                 {
105                     ++pos;
106                     tree1.insert(A[pos],pre[A[pos]-1],false);
107                     tree2.insert(-A[pos],pre[A[pos]],false);
108                     upd(mx,tree1.query(A[pos],pre[A[pos]]));
109                     upd(mx,tree2.query(-A[pos],pre[A[pos]-1]));
110                 }
111                 upd(ans[query[i].id],mx);
112                 for (int j = lim;j >= query[i].l;--j)
113                 {
114                     tree1.insert(A[j],pre[A[j]-1],true);
115                     tree2.insert(-A[j],pre[A[j]],true);
116                     upd(ans[query[i].id],tree1.query(A[j],pre[A[j]]));

```

```

117         upd(ans[query[i].id],tree2.query(-A[j],pre[A[j]-1]));
118     }
119     tree1.undo(); tree2.undo();
120 }
121     break;
122 }
123 }
124 }
125
126 int main()
127 {
128     // freopen("A.in","r",stdin);
129     for (int i = 1;i <= 1000000;++i) pre[i] = pre[i-1]^i;
130     N = gi(); M = gi();
131     for (int i = 1;i <= N;++i) A[i] = gi();
132     for (int i = 1;i <= M;++i) query[i].read(i);
133     sort(query+1,query+M+1);
134     for (int i = 1,j;i <= M;i = j)
135     {
136         for (j = i;j <= M&&query[j].l-query[i].l <= len;++j);
137         work(i,j-1);
138     }
139     for (int i = 1;i <= M;++i) printf("%d\n",ans[i]);
140     return 0;
141 }

```

3.10 President Tree

```

1  inline void build(int &now,int l,int r)
2  {
3      now = ++cnt; if (l == r) return;
4      int mid = (l+r)>>1;
5      build(tree[now].ch[0],l,mid); build(tree[now].ch[1],mid+1,r);
6  }
7
8  inline void ins(int &now,int ref,int l,int r,int key)
9  {
10     now = ++cnt; tree[now] = tree[ref];
11     if (l == r) { ++tree[now].sum; return; }
12     int mid = (l+r) >> 1;
13     if (key <= mid) ins(tree[now].ch[0],tree[ref].ch[0],l,mid,key);
14     else ins(tree[now].ch[1],tree[ref].ch[1],mid+1,r,key);
15     tree[now].sum = tree[tree[now].ch[0]].sum+tree[tree[now].ch[1]].sum;
16 }

```

3.11 Splay

```

1  //splay
2

```

```

3  inline int find(int rk)
4  {
5      for (int now = root;;)
6      {
7          if (rk == size[ch[now][0]]+1) return now;
8          else if (rk > size[ch[now][0]]+1)
9              rk -= size[ch[now][1]]+1, now = ch[now][1];
10         else now = ch[now][0];
11     }
12     return 0;
13 }
14
15 inline int upperbound(int x)
16 {
17     int ret = 0;
18     for (int now = root; now;)
19     {
20         if (key[now] > x) ret = now, now = ch[now][0];
21         else now = ch[now][1];
22     }
23     return ret;
24 }
25 inline int lowerbound(int x)
26 {
27     int ret = 0;
28     for (int now = root; now;)
29     {
30         if (key[now] >= x) ret = now, now = ch[now][0];
31         else now = ch[now][1];
32     }
33     return ret;
34 }
35
36 inline void rotate(int x)
37 {
38     int y = fa[x], z = fa[y], l = ch[y][0] != x, r = l^1;
39     if (z) ch[z][ch[z][0] != y] = x; fa[x] = z;
40     if (ch[x][r]) fa[ch[x][r]] = y;
41     ch[y][l] = ch[x][r]; fa[y] = x; ch[x][r] = y;
42     update(y); update(x);
43 }
44 inline void splay(int x, int aim) //aim is x's father.
45 {
46     int top = 0;
47     for (int i = x; i; i = fa[i]) stack[++top] = i;
48     while (top) pushdown(stack[top--]);
49     while (fa[x] != aim)
50     {
51         int y = fa[x], z = fa[y];

```

```

52         if (z != aim)
53         {
54             if ((ch[y][0] == x)^(ch[z][0] == y)) rotate(x);
55             else rotate(y);
56         }
57         rotate(x);
58     }
59     if (!aim) root = x;
60 }
61
62 // 维修数列
63 #include<cassert>
64 #include<queue>
65 #include<algorithm>
66 #include<cstring>
67 #include<iostream>
68 #include<cstdio>
69 #include<cstdlib>
70 using namespace std;
71
72 const int maxn = 500010, inf = 1<<29;
73 int N,M,root,cnt,arr[maxn],tag[maxn],key[maxn],fa[maxn],ch[maxn][2],lb[maxn],rb[maxn];
74 int wb[maxn],sum[maxn],size[maxn],stk[maxn]; bool rev[maxn]; char cmd[20]; queue<int> team;
75
76 inline int gi()
77 {
78     char ch; int ret = 0, f = 1;
79     do ch = getchar(); while (!(ch >= '0' && ch <= '9') && ch != '-');
80     if (ch == '-') f = -1, ch = getchar();
81     do ret = ret*10+ch-'0', ch = getchar(); while (ch >= '0' && ch <= '9');
82     return ret*f;
83 }
84
85 inline int newnode(int x = 0)
86 {
87     int ret;
88     if (!team.empty())
89         ret = team.front(), team.pop();
90     else ret = ++cnt;
91     key[ret] = sum[ret] = lb[ret] = rb[ret] = wb[ret] = x;
92     rev[ret] = false; tag[ret] = inf; size[ret] = 1;
93     return ret;
94 }
95
96 inline void pushdown(int now)
97 {
98     int lc = ch[now][0], rc = ch[now][1];
99     if (rev[now])
100     {

```

```

101     if (lc)
102     {
103         swap(ch[lc][0],ch[lc][1]);
104         swap(lb[lc],rb[lc]); rev[lc] ^= 1;
105     }
106     if (rc)
107     {
108         swap(ch[rc][0],ch[rc][1]);
109         swap(lb[rc],rb[rc]); rev[rc] ^= 1;
110     }
111     rev[now] = false;
112 }
113 if (tag[now] != inf)
114 {
115     if (lc)
116     {
117         key[lc] = tag[lc] = tag[now]; sum[lc] = tag[lc]*size[lc];
118         if (tag[lc] > 0) lb[lc] = rb[lc] = wb[lc] = sum[lc];
119         else lb[lc] = rb[lc] = wb[lc] = tag[lc];
120     }
121     if (rc)
122     {
123         key[rc] = tag[rc] = tag[now]; sum[rc] = tag[rc]*size[rc];
124         if (tag[rc] > 0) lb[rc] = rb[rc] = wb[rc] = sum[rc];
125         else lb[rc] = rb[rc] = wb[rc] = tag[rc];
126     }
127     tag[now] = inf;
128 }
129 }
130
131 inline void update(int now)
132 {
133     // pushdown(now);
134     int lc = ch[now][0],rc = ch[now][1];
135     size[now] = size[lc]+size[rc]+1;
136     sum[now] = sum[lc]+sum[rc]+key[now];
137     if (lc&&rc)
138     {
139         lb[now] = max(lb[lc],max(sum[lc]+key[now],sum[lc]+key[now]+lb[rc]));
140         rb[now] = max(rb[rc],max(sum[rc]+key[now],sum[rc]+key[now]+rb[lc]));
141         wb[now] = max(wb[lc],wb[rc]); wb[now] = max(wb[now],key[now]);
142         wb[now] = max(wb[now],rb[lc]+key[now]); wb[now] = max(wb[now],lb[rc]+key[now]);
143         wb[now] = max(wb[now],rb[lc]+key[now]+lb[rc]);
144     }
145     else if (lc)
146     {
147         lb[now] = max(lb[lc],sum[lc]+key[now]);
148         rb[now] = max(key[now],key[now]+rb[lc]);
149         wb[now] = max(wb[lc],key[now]);

```



```

150         wb[now] = max(wb[now],rb[lc]+key[now]);
151     }
152     else if (rc)
153     {
154         rb[now] = max(rb[rc],sum[rc]+key[now]);
155         lb[now] = max(key[now],key[now]+lb[rc]);
156         wb[now] = max(wb[rc],key[now]);
157         wb[now] = max(wb[now],lb[rc]+key[now]);
158     }
159     else sum[now] = lb[now] = rb[now] = wb[now] = key[now];
160 }
161
162 inline int build(int l,int r)
163 {
164     int mid = (l+r) >> 1,ret = newnode(arr[mid]);
165     if (l < mid) ch[ret][0] = build(l,mid-1),fa[ch[ret][0]] = ret;
166     if (r > mid) ch[ret][1] = build(mid+1,r),fa[ch[ret][1]] = ret;
167     update(ret); return ret;
168 }
169
170 inline void init()
171 {
172     root = newnode(); ch[root][1] = newnode(); fa[2] = 1;
173     for (int i = 1;i <= N;++i) arr[i] = gi();
174     ch[2][0] = build(1,N); fa[ch[2][0]] = 2;
175     update(2); update(1);
176 }
177
178 inline int find(int rk)
179 {
180     for (int now = root;;)
181     {
182         pushdown(now);
183         if (rk == size[ch[now][0]]+1) return now;
184         else if (rk > size[ch[now][0]]+1)
185             rk -= size[ch[now][0]]+1,now = ch[now][1];
186         else now = ch[now][0];
187     }
188     return 0;
189 }
190
191 inline void rotate(int x)
192 {
193     int y = fa[x],z = fa[y],l = ch[y][0] != x,r = l^1;
194     if (z) ch[z][ch[z][0] != y] = x;
195     fa[x] = z; fa[y] = x; fa[ch[x][r]] = y;
196     ch[y][l] = ch[x][r]; ch[x][r] = y;
197     update(y); update(x);
198 }

```

```

199 inline void splay(int x,int aim)
200 {
201     int top = 0;
202     for (int i = x;i;i = fa[i]) stk[++top] = i;
203     while (top) pushdown(stk[top--]);
204     while (fa[x] != aim)
205     {
206         int y = fa[x],z = fa[y];
207         if (z != aim)
208         {
209             if ((ch[y][0] == x)^(ch[z][0] == y)) rotate(x);
210             else rotate(y);
211         }
212         rotate(x);
213     }
214     if (!aim) root = x;
215 }
216
217 inline void Delete(int &now)
218 {
219     if (!now) return;
220     Delete(ch[now][0]);
221     Delete(ch[now][1]);
222     team.push(now); now = 0;
223 }
224
225 inline void print()
226 {
227     for (int i = 1;i <= cnt;++i)
228         printf("%d:%d %d\n",i,ch[i][0],ch[i][1]);
229     for (int i = 1;i <= cnt;++i)
230         printf("%d:%d\n",i,fa[i]);
231 }
232 }
233
234 inline void laydown(int now)
235 {
236     if (!now) return;
237     pushdown(now);
238     laydown(ch[now][0]);
239     printf("%d ",key[now]);
240     laydown(ch[now][1]);
241     update(now);
242 }
243
244 int main()
245 {
246     //freopen("C.in","r",stdin);
247     N = gi(); M = gi(); init();

```

```

248     while (M--)
249     {
250         scanf("%s",cmd);
251         if (cmd[0] == 'I')
252         {
253             int pos = gi(),a = find(pos+1),b = find(pos+2); N = gi();
254             for (int i = 1;i <= N;++i) arr[i] = gi();
255             splay(a,0); splay(b,a);
256             ch[b][0] = build(1,N); fa[ch[b][0]] = b;
257             update(b); update(a);
258         }
259         else if (cmd[0] == 'D')
260         {
261             int pos = gi(); N = gi();
262             int a = find(pos),b = find(pos+N+1);
263             splay(a,0); splay(b,a);
264             Delete(ch[b][0]); update(b); update(a);
265         }
266         else if (cmd[0] == 'M' && cmd[2] == 'K')
267         {
268             int pos = gi(); N = gi();
269             int a = find(pos),b = find(pos+N+1);
270             splay(a,0); splay(b,a);
271             key[ch[b][0]] = tag[ch[b][0]] = gi(); sum[ch[b][0]] = tag[ch[b][0]]*size[ch[b][0]];
272             if (tag[ch[b][0]] > 0) lb[ch[b][0]] = rb[ch[b][0]] = wb[ch[b][0]] = sum[ch[b][0]];
273             else lb[ch[b][0]] = rb[ch[b][0]] = wb[ch[b][0]] = tag[ch[b][0]];
274             update(b); update(a);
275         }
276         else if (cmd[0] == 'R')
277         {
278             int pos = gi(); N = gi();
279             int a = find(pos),b = find(pos+N+1);
280             splay(a,0); splay(b,a);
281             rev[ch[b][0]] ^= 1;
282             swap(ch[ch[b][0]][0],ch[ch[b][0]][1]);
283             swap(lb[ch[b][0]],rb[ch[b][0]]);
284             update(b); update(a);
285         }
286         else if (cmd[0] == 'G')
287         {
288             int pos = gi(); N = gi();
289             int a = find(pos),b = find(pos+N+1);
290             splay(a,0); splay(b,a);
291             printf("%d\n",sum[ch[b][0]]);
292         }
293         else
294         {
295             splay(1,0); splay(2,1);
296             printf("%d\n",wb[ch[2][0]]);

```

```
297         }  
298     }  
299     return 0;  
300 }
```

Chapter 4

Graph Theory

4.1 2-Sat

```
1  // bzoj 1823
2  #include<stack>
3  #include<iostream>
4  #include<cstring>
5  #include<cstdio>
6  #include<cstdlib>
7  using namespace std;
8
9  #define maxn 210
10 #define maxm 2010
11 int n,m,cnt,side[maxn],next[maxm],toit[maxm],dfn[maxn],id[maxn];
12 int tot,low[maxn],d[maxn],DFN;
13 stack <int> S; bool vis[maxn];
14
15 inline void init()
16 {
17     DFN = tot = cnt = 0; memset(vis,false,2*(n+4));
18     memset(side,0,8*(n+4)); memset(dfn,0,8*(n+4));
19 }
20
21 inline void add(int a,int b) { next[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; }
22
23 inline void dfs(int now)
24 {
25     S.push(now); dfn[now] = low[now] = ++DFN;
26     for (int i = side[now];i;i = next[i])
27     {
28         if (vis[toit[i]]) continue;
29         if (!dfn[toit[i]]) dfs(toit[i]);
30         low[now] = min(low[toit[i]],low[now]);
31     }
32     if (low[now] == dfn[now])
33     {
```

```

34         ++tot;
35         while (S.top() != now) id[S.top()] = tot, vis[S.top()] = true, S.pop();
36         id[S.top()] = tot, vis[S.top()] = true, S.pop();
37     }
38 }
39
40 int main()
41 {
42     freopen("1823.in", "r", stdin);
43     freopen("1823.out", "w", stdout);
44     int T; scanf("%d", &T);
45     while (T--)
46     {
47         scanf("%d %d\n", &n, &m);
48         init();
49         while (m--)
50         {
51             char c1, c2; int a, b; bool o1, o2;
52             scanf("%c%d %c%d\n", &c1, &a, &c2, &b);
53             o1 = c1 == 'h'; o2 = c2 == 'h';
54             add((o1^1)*n+a, o2*n+b);
55             add((o2^1)*n+b, o1*n+a);
56         }
57         int i;
58         for (i = 1; i <= n << 1; ++i) if (!dfn[i]) dfs(i);
59         for (i = 1; i <= n; ++i) if (id[i] == id[i+n]) { printf("BAD\n"); break; }
60         if (i <= n) continue;
61         printf("GOOD\n");
62     }
63     fclose(stdin); fclose(stdout);
64     return 0;
65 }

```

4.2 Bridges and Cut Vertices

```

1 // 求割边和割点
2 const int maxn = 200010;
3 int N, M, cnt, Ts, dfn[maxn], low[maxn], nxt[maxn];
4 int toIt[maxn], side[maxn];
5 bool bridge[maxn], cut[maxn];
6
7 inline void dfs(int now, int fa)
8 {
9     dfn[now] = low[now] = ++Ts; int child = 0;
10    for (int i = side[now]; i; i = nxt[i])
11    {
12        if (toIt[i] == fa) continue;
13        if (!dfn[toIt[i]])
14        {

```

```

15         dfs(toit[i],now); ++child;
16         low[now] = min(low[now],low[toit[i]]);
17         if (low[toit[i]] > dfn[now]) bridge[i] = true;
18         if (low[toit[i]] >= dfn[now]) cut[now] = true;
19     }
20     else low[now] = min(low[now],dfn[toit[i]]);
21 }
22 if (!fa&&child == 1) cut[now] = false;
23 }

```

4.3 Cost Flow

```

1  int side[maxv],nxt[maxe],toit[maxe],cost[maxe],pre[maxv];
2  int cap[maxv],arr[maxv],dis[maxv]; bool in[maxv];
3  int source,sink;
4
5  inline void add(int a,int b,int c,int d) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b;
   ⇐ cap[cnt] = c; cost[cnt] = d; }
6  inline void ins(int a,int b,int c,int d) { add(a,b,c,d); add(b,a,0,-d); }
7
8  inline bool spfa(int &Flow,int &Cost)
9  {
10     queue<int> team; team.push(source);
11     memset(dis,0x7f,4*(sink+5));
12     dis[source] = 0; in[source] = true;
13     arr[source] = inf; arr[sink] = 0;
14     while (!team.empty())
15     {
16         int now = team.front(); team.pop();
17         for (int i = side[now];i;i = nxt[i])
18         {
19             if (!cap[i]) continue;
20             if (dis[toit[i]] > dis[now]+cost[i])
21             {
22                 arr[toit[i]] = min(cap[i],arr[now]); pre[toit[i]] = i;
23                 dis[toit[i]] = dis[now]+cost[i];
24                 if (!in[toit[i]]) in[toit[i]] = true,team.push(toit[i]);
25             }
26         }
27         in[now] = false;
28     }
29     if (!arr[sink]) return false;
30     Flow += arr[sink];
31     for (int now = sink,i;now != source;now = toit[i^1])
32     {
33         i = pre[now]; Cost += cost[pre[now]]*arr[sink];
34         cap[i] -= arr[sink]; cap[i^1] += arr[sink];
35     }
36     return true;

```

```
37 }
```

4.4 Difference Constraints

```
1  // bzoj2330
2  #include<iostream>
3  #include<stack>
4  #include<queue>
5  #include<cstdio>
6  #include<cstdlib>
7  using namespace std;
8
9  #define maxn 100010
10 #define maxm 200010
11 int cnt,side[maxn],next[maxm],toit[maxm],cost[maxm],d[maxn];
12 int nside[maxn],nnext[maxm],ntoit[maxm],ncost[maxm],m,tot;
13 int dfn[maxn],low[maxn],sum[maxn],id[maxn],arr[maxn],n;
14 bool vis[maxn]; stack <int> S;
15
16 inline void add(int a,int b,int c)
17 {
18     next[++cnt] = side[a]; side[a] = cnt;
19     toit[cnt] = b; cost[cnt] = c;
20 }
21
22 inline void ins(int a,int b,int c)
23 {
24     nnext[++cnt] = nside[a]; nside[a] = cnt;
25     ntoit[cnt] = b; ncost[cnt] = c; ++d[b];
26 }
27
28 inline void dfs(int now)
29 {
30     S.push(now); dfn[now] = low[now] = ++cnt;
31     for (int i = side[now];i;i = next[i])
32         if (!vis[toit[i]])
33         {
34             if (!dfn[toit[i]]) dfs(toit[i]);
35             low[now] = min(low[toit[i]],low[now]);
36         }
37     if (low[now] == dfn[now])
38     {
39         ++tot;
40         while (S.top() != now) id[S.top()] = tot,vis[S.top()] = true,S.pop();
41         id[S.top()] = tot,vis[S.top()] = true,S.pop();
42     }
43 }
44
45 inline bool rebuild()
```



```

46 {
47     cnt = 0;
48     for (int i = 1; i <= n; ++i)
49         for (int j = side[i]; j; j = next[j])
50             {
51                 if (id[toit[j]] == id[i]) sum[id[i]] += cost[j];
52                 else ins(id[i], id[toit[j]], cost[j]);
53             }
54     for (int i = 1; i <= tot; ++i) if (sum[i]) return false;
55     return true;
56 }
57
58 inline void topsort()
59 {
60     queue<int> team;
61     for (int i = 1; i <= tot; ++i) if (!d[i]) team.push(i), arr[i] = 1;
62     while (!team.empty())
63     {
64         int now = team.front(); team.pop();
65         for (int i = nside[now]; i; i = nnext[i])
66             {
67                 arr[ntoit[i]] = max(arr[now] + ncost[i], arr[ntoit[i]]);
68                 if (!--d[ntoit[i]]) team.push(ntoit[i]);
69             }
70     }
71 }
72
73 int main()
74 {
75     freopen("2330.in", "r", stdin);
76     freopen("2330.out", "w", stdout);
77     scanf("%d %d", &n, &m);
78     for (int i = 1; i <= m; ++i)
79     {
80         int x, a, b; scanf("%d %d %d", &x, &a, &b);
81         if (x == 1) add(a, b, 0), add(b, a, 0);
82         else if (x == 2) add(a, b, 1);
83         else if (x == 3) add(b, a, 0);
84         else if (x == 4) add(b, a, 1);
85         else add(a, b, 0);
86     }
87     cnt = 0; for (int i = n; i; --i) if (!dfn[i]) dfs(i);
88     if (!rebuild()) printf("-1"), exit(0);
89     topsort();
90     long long ans = 0;
91     for (int i = 1; i <= n; ++i) ans += (long long)arr[id[i]];
92     printf("%lld", ans);
93     fclose(stdin); fclose(stdout);
94     return 0;

```

```
95 }
```

4.5 Dinic Algorithm

```
1  // dinic
2  int source,sink,cnt = 1;
3  int d[maxv],side[maxv],cur[maxv],side[maxe],nxt[maxe],toit[maxe],cap[maxe]; bool in[maxv];
4
5  inline void add(int a,int b,int c) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b;
   ↪  cap[cnt] = c; }
6  inline void ins(int a,int b,int c) { add(a,b,c); add(b,a,0); }
7
8  inline bool bfs()
9  {
10     queue <int> team; team.push(source); d[source] = 0;
11     memset(in,false,tot+10); in[source] = true; team.push(source);
12     while (!team.empty())
13     {
14         int now = team.front(); team.pop(); cur[now] = side[now];
15         for (int i = side[now];i;i = nxt[i])
16         {
17             if (!cap[i]) continue;
18             if (!in[toit[i]])
19                 in[toit[i]] = true,d[toit[i]] = d[now]+1,team.push(toit[i]);
20         }
21     }
22     return in[sink];
23 }
24
25 inline int dfs(int now,int f)
26 {
27     if (now == sink||!f) return f;
28     int used = 0,w;
29     for (int &i = cur[now];i;i = nxt[i])
30         if (cap[i]&& d[toit[i]] == d[now]+1)
31         {
32             w = dfs(toit[i],min(cap[i],f-used));
33             used += w; cap[i] -= w; cap[i^1] += w;
34             if (used == f) break;
35         }
36     return used;
37 }
38
39 inline int dinic(int S,int T)
40 {
41     source = S; sink = T; int ret = 0;
42     while (bfs()) ret += dfs(source,inf);
43     return ret;
44 }
```

4.6 Dominator Tree

```

1  //建出来的树点的编号 i 在原图中是 redfn[i]
2  int
   ↳ N,M,Ts,cnt,side[maxn],nxt[maxn],toit[maxn],dfn[maxn],redfn[maxn],idom[maxn],best[maxn],semi[maxn];
3  int ans[maxn],anc[maxn],fa[maxn],child[maxn],size[maxn]; vector <int>
   ↳ prod[maxn],bucket[maxn],son[maxn];
4
5  inline void init()
6  {
7      cnt = 1; memset(side,0,sizeof side); memset(ans,0,sizeof ans);
8      for (int i = 0;i <= N;++i) prod[i].clear(),bucket[i].clear(),son[i].clear();
9  }
10
11 inline void add(int a,int b) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; }
12
13 inline int gi()
14 {
15     char ch; int ret = 0,f = 1;
16     do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
17     if (ch == '-') f = -1,ch = getchar();
18     do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
19     return ret*f;
20 }
21
22 inline void dfs(int now)
23 {
24     dfn[now] = ++Ts; redfn[Ts] = now;
25     anc[Ts] = idom[Ts] = child[Ts] = size[Ts] = 0;
26     semi[Ts] = best[Ts] = Ts;
27     for (int i = side[now];i;i = nxt[i])
28     {
29         if (!dfn[toit[i]])
30             dfs(toit[i]),fa[dfn[toit[i]]] = dfn[now];
31         prod[dfn[toit[i]]].push_back(dfn[now]);
32     }
33 }
34
35 inline void compress(int now)
36 {
37     if (anc[anc[now]] != 0)
38     {
39         compress(anc[now]);
40         if (semi[best[now]] > semi[best[anc[now]]])
41             best[now] = best[anc[now]];
42         anc[now] = anc[anc[now]];
43     }
44 }
45
46 inline int eval(int now)

```

```

47 {
48     if (!anc[now]) return now;
49     else
50     {
51         compress(now);
52         return semi[best[anc[now]]] >= semi[best[now]] ? best[now] : best[anc[now]];
53     }
54 }
55
56 inline void link(int v, int w)
57 {
58     int s = w;
59     while (semi[best[w]] < semi[best[child[w]]])
60     {
61         if (size[s] + size[child[child[s]]] >= 2 * size[child[s]])
62             anc[child[s]] = s, child[s] = child[child[s]];
63         else size[child[s]] = size[s], s = anc[s] = child[s];
64     }
65     best[s] = best[w]; size[v] += size[w];
66     if (size[v] < 2 * size[w]) swap(s, child[v]);
67     while (s) anc[s] = v, s = child[s];
68 }
69
70 inline void lengauer_tarjan()
71 {
72     memset(dfn, 0, sizeof dfn); memset(fa, -1, sizeof fa); Ts = 0;
73     dfs(N); fa[1] = 0;
74     for (int w = Ts; w > 1; --w)
75     {
76         for (auto x: prod[w])
77         {
78             int u = eval(x);
79             if (semi[w] > semi[u]) semi[w] = semi[u];
80         }
81         bucket[semi[w]].push_back(w);
82         link(fa[w], w); if (!fa[w]) continue;
83         for (auto x: bucket[fa[w]])
84         {
85             int u = eval(x);
86             if (semi[u] < fa[w]) idom[x] = u;
87             else idom[x] = fa[w];
88         }
89         bucket[fa[w]].clear();
90     }
91     for (int w = 2; w <= Ts; ++w)
92         if (idom[w] != semi[w])
93             idom[w] = idom[idom[w]];
94     idom[1] = 0;
95     for (int i = Ts; i > 1; --i)

```

```

96     {
97         if (fa[i] == -1) continue;
98         son[idom[i]].push_back(i);
99     }
100 }
101
102 // 例题: 询问 i 号点到 N 号点所有必经点编号和
103 #include<algorithm>
104 #include<cstring>
105 #include<iostream>
106 #include<cstdio>
107 #include<cstdlib>
108 using namespace std;
109
110 const int maxn = 100010;
111 int
112     ↪ N,M,Ts,cnt,side[maxn],nxt[maxn],toit[maxn],dfn[maxn],redfn[maxn],idom[maxn],best[maxn],semi[maxn];
113 int ans[maxn],anc[maxn],fa[maxn],child[maxn],size[maxn]; vector<int>
114     ↪ prod[maxn],bucket[maxn],son[maxn];
115
116 inline void init()
117 {
118     cnt = 1; memset(side,0,sizeof side); memset(ans,0,sizeof ans);
119     for (int i = 0;i <= N;++i) prod[i].clear(),bucket[i].clear(),son[i].clear();
120 }
121
122 inline void add(int a,int b) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; }
123
124 inline int gi()
125 {
126     char ch; int ret = 0,f = 1;
127     do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
128     if (ch == '-') f = -1,ch = getchar();
129     do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
130     return ret*f;
131 }
132
133 inline void dfs(int now)
134 {
135     dfn[now] = ++Ts; redfn[Ts] = now;
136     anc[Ts] = idom[Ts] = child[Ts] = size[Ts] = 0;
137     semi[Ts] = best[Ts] = Ts;
138     for (int i = side[now];i;i = nxt[i])
139     {
140         if (!dfn[toit[i]])
141             dfs(toit[i]),fa[dfn[toit[i]]] = dfn[now];
142         prod[dfn[toit[i]]].push_back(dfn[now]);
143     }
144 }

```

```

143
144 inline void compress(int now)
145 {
146     if (anc[anc[now]] != 0)
147     {
148         compress(anc[now]);
149         if (semi[best[now]] > semi[best[anc[now]]])
150             best[now] = best[anc[now]];
151         anc[now] = anc[anc[now]];
152     }
153 }
154
155 inline int eval(int now)
156 {
157     if (!anc[now]) return now;
158     else
159     {
160         compress(now);
161         return semi[best[anc[now]]] >= semi[best[now]] ? best[now] : best[anc[now]];
162     }
163 }
164
165 inline void link(int v, int w)
166 {
167     int s = w;
168     while (semi[best[w]] < semi[best[child[w]]])
169     {
170         if (size[s] + size[child[child[s]]] >= 2 * size[child[s]])
171             anc[child[s]] = s, child[s] = child[child[s]];
172         else size[child[s]] = size[s], s = anc[s] = child[s];
173     }
174     best[s] = best[w]; size[v] += size[w];
175     if (size[v] < 2 * size[w]) swap(s, child[v]);
176     while (s) anc[s] = v, s = child[s];
177 }
178
179 inline void lengauer_tarjan()
180 {
181     memset(dfn, 0, sizeof dfn); memset(fa, -1, sizeof fa); Ts = 0;
182     dfs(N); fa[1] = 0;
183     for (int w = Ts; w > 1; --w)
184     {
185         for (auto x : prod[w])
186         {
187             int u = eval(x);
188             if (semi[w] > semi[u]) semi[w] = semi[u];
189         }
190         bucket[semi[w]].push_back(w);
191         link(fa[w], w); if (!fa[w]) continue;

```

```

192     for (auto x:bucket[fa[w]])
193     {
194         int u = eval(x);
195         if (semi[u] < fa[w]) idom[x] = u;
196         else idom[x] = fa[w];
197     }
198     bucket[fa[w]].clear();
199 }
200 for (int w = 2; w <= Ts; ++w)
201     if (idom[w] != semi[w])
202         idom[w] = idom[idom[w]];
203 idom[1] = 0;
204 for (int i = Ts; i > 1; --i)
205 {
206     if (fa[i] == -1) continue;
207     son[idom[i]].push_back(i);
208 }
209 }
210
211 inline void get_ans(int now)
212 {
213     ans[redfn[now]] += redfn[now];
214     for (auto x:son[now])
215         ans[redfn[x]] += ans[redfn[now]], get_ans(x);
216 }
217
218 int main()
219 {
220     //freopen("I.in", "r", stdin);
221     while (scanf("%d %d", &N, &M) != EOF)
222     {
223         init();
224         for (int i = 1; i <= M; ++i)
225             a = gi(), b = gi(), add(a, b);
226         lengauer_tarjan(); get_ans(1);
227         for (int i = 1; i <= N; ++i)
228             printf("%d%c", ans[i], " \n"[i == N]);
229     }
230     return 0;
231 }

```

4.7 Hungary

```

1 //匈牙利算法
2 //Version1
3 inline bool find(int x)
4 {
5     if (cor[x]) return false;
6     for (int i = side[x]; i; i = next[i]) if (!used[toit[i]])

```

```

7      {
8          used[toit[i]] = true;
9          if (!cho[toit[i]] || find(cho[toit[i]]))
10             {
11                 cho[toit[i]] = x; map[x] = toit[i];
12                 return true;
13             }
14     }
15     return false;
16 }
17
18 inline void hungry()
19 {
20     for (int i = 1; i <= p; ++i)
21         memset(used, false, sizeof(used)), find(i);
22     for (int i = 1; i <= m; ++i)
23     {
24         memset(used, false, sizeof(used)), cho[map[i]] = 0;
25         find(i), cor[i] = true;
26     }
27 }
28 //Version2
29 inline int find(int x)
30 {
31     for (int i = 1; i <= n; ++i)
32         if (f[x][i] && !used[i])
33         {
34             used[i] = true;
35             if (!cho[i] || find(cho[i])) { cho[i] = x; return true; }
36         }
37     return false;
38 }
39
40 inline int hungry()
41 {
42     int ret = 0;
43     for (int i = 1; i <= n; ++i)
44     {
45         memset(used, false, sizeof(used));
46         if (find(i)) ret++;
47     }
48     return ret;
49 }

```

4.8 Isap Algorithm

```

1 // isap: 有毒
2 inline void bfs()
3 {

```



```

4     queue <int> team; memcpy(cur,side,4*(N+1));
5     team.push(sink); d[sink] = 1; in[sink] = true;
6     while (!team.empty())
7     {
8         int now = team.front(); team.pop(); nd[d[now]]++;
9         for (int i = side[now];i;i = nxt[i])
10            if (cap[i^1] && !in[toit[i]])
11                in[toit[i]] = true,d[toit[i]] = d[now]+1,team.push(toit[i]);
12    }
13    for (int i = 1;i <= N;++i) if (!in[i]) nd[d[i] = N+1]++;
14 }
15 inline int isap()
16 {
17     int res = 0,now = source,ca = inf;
18     bfs();
19     while (d[source] <= N)
20     {
21         if (now == sink)
22         {
23             while (now != source)
24             {
25                 cap[pre[now]] -= ca; cap[pre[now]^1] += ca;
26                 now = toit[pre[now]^1];
27             }
28             res += ca; ca = inf;
29         }
30         bool flag = false; arr[now] = ca;
31         for (int i = cur[now];i;i = nxt[i])
32             if (cap[i]&&d[toit[i]]==d[now]-1)
33             {
34                 cur[now] = pre[toit[i]] = i; ca = min(ca,cap[i]);
35                 now = toit[i]; flag = true; break;
36             }
37         if (flag) continue; if (!--nd[d[now]]) break; int arg = N;
38         for (int i = side[now];i;i = nxt[i])
39             if (cap[i]&&d[toit[i]] < arg) arg = d[toit[i]];
40         ++nd[d[now]=arg+1]; cur[now] = side[now];
41         if (now != source) ca = arr[now = toit[pre[now]^1]];
42     }
43     return res;
44 }
45
46 int source;          // 源点
47 int sink;            // 汇点
48 int p[max_nodes];    // 可增广路上的上一条弧的编号
49 int num[max_nodes];  // 和 t 的最短距离等于 i 的节点数量
50 int cur[max_nodes];  // 当前弧下标
51 int d[max_nodes];    // 残量网络中节点 i 到汇点 t 的最短距离
52 bool visited[max_nodes];

```

```

53
54 // 预处理, 反向 BFS 构造 d 数组
55 bool bfs()
56 {
57     memset(visited, 0, sizeof(visited));
58     queue<int> Q;
59     Q.push(sink);
60     visited[sink] = 1;
61     d[sink] = 0;
62     while (!Q.empty()) {
63         int u = Q.front();
64         Q.pop();
65         for (iterator_t ix = G[u].begin(); ix != G[u].end(); ++ix) {
66             Edge &e = edges[*ix]^1;
67             if (!visited[e.from] && e.capacity > e.flow) {
68                 visited[e.from] = true;
69                 d[e.from] = d[u] + 1;
70                 Q.push(e.from);
71             }
72         }
73     }
74     return visited[source];
75 }
76
77 // 增广
78 int augment()
79 {
80     int u = sink, df = __inf;
81     // 从汇点到源点通过 p 追踪增广路径, df 为一路上最小的残量
82     while (u != source) {
83         Edge &e = edges[p[u]];
84         df = min(df, e.capacity - e.flow);
85         u = edges[p[u]].from;
86     }
87     u = sink;
88     // 从汇点到源点更新流量
89     while (u != source) {
90         edges[p[u]].flow += df;
91         edges[p[u]^1].flow -= df;
92         u = edges[p[u]].from;
93     }
94     return df;
95 }
96
97 int max_flow()
98 {
99     int flow = 0;
100     bfs();
101     memset(num, 0, sizeof(num));

```

```

102     for (int i = 0; i < num_nodes; i++) num[d[i]]++;
103     int u = source;
104     memset(cur, 0, sizeof(cur));
105     while (d[source] < num_nodes) {
106         if (u == sink) {
107             flow += augment();
108             u = source;
109         }
110         bool advanced = false;
111         for (int i = cur[u]; i < G[u].size(); i++) {
112             Edge& e = edges[G[u][i]];
113             if (e.capacity > e.flow && d[u] == d[e.to] + 1) {
114                 advanced = true;
115                 p[e.to] = G[u][i];
116                 cur[u] = i;
117                 u = e.to;
118                 break;
119             }
120         }
121         if (!advanced) { // retreat
122             int m = num_nodes - 1;
123             for (iterator_t ix = G[u].begin(); ix != G[u].end(); ++ix)
124                 if (edges[*ix].capacity > edges[*ix].flow)
125                     m = min(m, d[edges[*ix].to]);
126             if (--num[d[u]] == 0) break; // gap 优化
127             num[d[u] = m+1]++;
128             cur[u] = 0;
129             if (u != source)
130                 u = edges[p[u]].from;
131         }
132     }
133     return flow;
134 }
135 //By mxh
136 #define maxn 1010
137 const int INF=1<<30;
138 int n,m;
139 int S,T;
140 struct Edge
141 {
142     int v,flow,next;
143 } e[510010];
144 int g[maxn],tot=1;//tot 初值必须赋为 1
145 void addedge(int x,int y,int flow)
146 {
147     e[++tot].v=y;e[tot].flow=flow;e[tot].next=g[x];g[x]=tot;
148     e[++tot].v=x;e[tot].flow=0;e[tot].next=g[y];g[y]=tot;
149 }
150 int w[maxn],hash[maxn],d[maxn];

```



```

200         flow=INF;
201     }
202     break;
203 }
204 if (!bo) continue;
205 int minx=n,pos=0;
206 for (int i=g[u];i;i=e[i].next)
207     if (e[i].flow && d[e[i].v]<minx) minx=d[e[i].v],pos=i;
208 p[u]=pos;
209 hash[d[u]]--;
210 if (hash[d[u]]==0) break;
211 d[u]=minx+1;
212 hash[d[u]]++;
213 if (u!=S) u=prel[u],flow=w[u];
214 }
215 return ans;
216 }

```

4.9 Kuhn-Munkres Algorithm

```

1 // Truly  $O(n^3)$ , 最大权匹配
2 // 邻接矩阵, 不能连的边设为  $-INF$ , 求最小权匹配时边权取负, 但不能连的还是  $-INF$ , 使用时先对  $1 \rightarrow n$  调用 hungary(), 再
3 struct KM
4 {
5     int w[maxn][maxn],lx[maxn],ly[maxn],match[maxn],way[maxn],slack[maxn];
6     bool used[maxn];
7
8     inline void init()
9     {
10         for (int i = 1;i <= N;++i)
11             match[i] = lx[i] = ly[i] = way[i] = 0;
12     }
13
14     inline void hungary(int x)
15     {
16         match[0] = x; int j0 = 0;
17         for (int j = 0;j <= N;++j)
18             slack[j] = inf,used[j] = false;
19         do
20         {
21             used[j0] = true;
22             int i0 = match[j0],delta = inf,j1 = 0;
23             for (int j = 1;j <= N;++j)
24                 if (!used[j])
25                 {
26                     int cur = -w[i0][j]-lx[i0]-ly[j];
27                     if (cur < slack[j])
28                         slack[j] = cur,way[j] = j0;
29                     if (slack[j] < delta)

```

```

30         delta = slack[j], j1 = j;
31     }
32     for (int j = 0; j <= N; ++j)
33     {
34         if (used[j]) lx[match[j]] += delta, ly[j] -= delta;
35         else slack[j] -= delta;
36     }
37     j0 = j1;
38 }
39 while (match[j0]);
40 do
41 {
42     int j1 = way[j0];
43     match[j0] = match[j1];
44     j0 = j1;
45 }
46 while (j0);
47 }
48
49 inline void work() { for (int i = 1; i <= N; ++i) hungary(i); }
50
51 inline int get_ans()
52 {
53     int sum = 0;
54     for (int i = 1; i <= N; ++i)
55     {
56         // if (w[match[i]][i] == -inf) ; //无解
57         if (match[i] > 0) sum += w[match[i]][i];
58     }
59     return sum;
60 }
61 }km;
62 //最小权匹配
63 struct KM
64 {
65     int w[maxn][maxn], lx[maxn], ly[maxn], match[maxn], way[maxn], slack[maxn]; bool used[maxn];
66
67     inline void init()
68     {
69         for (int i = 1; i <= N; ++i)
70             match[i] = lx[i] = ly[i] = way[i] = 0;
71     }
72
73     inline void hungary(int x)
74     {
75         match[0] = x; int j0 = 0;
76         for (int j = 0; j <= N; ++j)
77             slack[j] = -inf, used[j] = false;
78         do

```

```

79     {
80         used[j0] = true;
81         int i0 = match[j0], delta = -inf, j1 = 0;
82         for (int j = 1; j <= N; ++j)
83             if (!used[j])
84                 {
85                     int cur = -w[i0][j] - lx[i0] - ly[j];
86                     if (cur > slack[j]) slack[j] = cur, way[j] = j0;
87                     if (slack[j] > delta) delta = slack[j], j1 = j;
88                 }
89         for (int j = 0; j <= N; ++j)
90             {
91                 if (used[j]) lx[match[j]] += delta, ly[j] -= delta;
92                 else slack[j] -= delta;
93             }
94         j0 = j1;
95     }
96     while (match[j0]);
97     do
98     {
99         int j1 = way[j0];
100         match[j0] = match[j1];
101         j0 = j1;
102     }
103     while (j0);
104 }
105
106 inline void work() { for (int i = 1; i <= N; ++i) hungary(i); }
107
108 inline int get_ans()
109 {
110     int sum = 0;
111     for (int i = 1; i <= N; ++i)
112     {
113         // if (w[match[i]][i] == inf) ; // 无解
114         if (match[i] > 0) sum += w[match[i]][i];
115     }
116     return sum;
117 }
118 }km;

```

4.10 Maximal Matching in General Graphs

```

1 // 接口 int matching(), 返回最大匹配数, G 为邻接矩阵
2 inline void push(int x)
3 {
4     team.push(x); check[x] = true;
5     if (!treec[x]) tra[++cnt] = x, treec[x] = true;
6 }

```

```

7  inline int root(int x) { return f[x]?f[x] = root(f[x]):x; }
8
9  inline void clear()
10 {
11     for (int i = 1,j;i <= cnt;++i)
12     {
13         j = tra[i]; father[j] = 0,f[j] = 0;
14         check[j] = treec[j] = false;
15     }
16 }
17
18 inline int lca(int u,int v)
19 {
20     int len = 0;
21     for (;u;u = father[match[u]])
22         pathc[path[++len] = u = root(u)] = true;
23     for (;v = father[match[v]])
24         if (pathc[v = root(v)]) break;
25     for (int i = 1;i <= len;++i)
26         pathc[path[i]] = false;
27     return v;
28 }
29
30 inline void reset(int u,int p)
31 {
32     for (int v;root(u) != p;)
33     {
34         if (!check[v = match[u]]) push(v);
35         if (!f[u]) f[u] = p; if (!f[v]) f[v] = p;
36         u = father[v]; if (root(u) != p) father[u] = v;
37     }
38 }
39
40 inline void flower(int u,int v)
41 {
42     int p = lca(u,v);
43     if (root(u) != p) father[u] = v;
44     if (root(v) != p) father[v] = u;
45     reset(u,p); reset(v,p);
46 }
47
48 inline bool find(int x)
49 {
50     while (!team.empty()) team.pop();
51     cnt = 0; push(x);
52     while (!team.empty())
53     {
54         int i = team.front(); team.pop();
55         for (int j = 1;j <= N;++j)

```



```

56         if (G[i][j]&&root(i) != root(j)&&match[j] != i)
57         {
58             if (match[j]&&father[match[j]]) flower(i,j);
59             else if (!father[j])
60             {
61                 father[tra[++cnt] = j] = i; treec[j] = true;
62                 if (match[j]) push(match[j]);
63                 else
64                 {
65                     for (int k = i,l = j,p;k;l = p,k = father[l])
66                         p = match[k],match[k] = l,match[l] = k;
67                     return true;
68                 }
69             }
70         }
71     }
72     return false;
73 }
74
75 inline int matching()
76 {
77     memset(father,0,sizeof father); memset(f,0,sizeof f); memset(path,0,sizeof path);
78     memset(tra,0,sizeof tra); memset(match,0,sizeof match); memset(check,false,sizeof check);
79     memset(treec,false,sizeof treec); memset(pathc,false,sizeof pathc);
80     int ret = cnt = 0;
81     for (int i = 1;i <= N;++i)
82     {
83         if (match[i]) continue;
84         if (find(i)) ++ret; clear();
85     }
86     return ret;
87 }

```

4.11 Maximal Weighted Matching in General Graphs

```

1  // 接口 int matching(), 返回最大匹配数,G 为邻接矩阵
2  inline void push(int x)
3  {
4      team.push(x); check[x] = true;
5      if (!treec[x]) tra[++cnt] = x,treec[x] = true;
6  }
7  inline int root(int x) { return f[x]?f[x] = root(f[x]):x; }
8
9  inline void clear()
10 {
11     for (int i = 1,j;i <= cnt;++i)
12     {
13         j = tra[i]; father[j] = 0,f[j] = 0;
14         check[j] = treec[j] = false;

```

```

15     }
16 }
17
18 inline int lca(int u,int v)
19 {
20     int len = 0;
21     for (;u;u = father[match[u]])
22         pathc[path[++len] = u = root(u)] = true;
23     for (;v = father[match[v]])
24         if (pathc[v = root(v)]) break;
25     for (int i = 1;i <= len;++i)
26         pathc[path[i]] = false;
27     return v;
28 }
29
30 inline void reset(int u,int p)
31 {
32     for (int v;root(u) != p;)
33     {
34         if (!check[v = match[u]]) push(v);
35         if (!f[u]) f[u] = p; if (!f[v]) f[v] = p;
36         u = father[v]; if (root(u) != p) father[u] = v;
37     }
38 }
39
40 inline void flower(int u,int v)
41 {
42     int p = lca(u,v);
43     if (root(u) != p) father[u] = v;
44     if (root(v) != p) father[v] = u;
45     reset(u,p); reset(v,p);
46 }
47
48 inline bool find(int x)
49 {
50     while (!team.empty()) team.pop();
51     cnt = 0; push(x);
52     while (!team.empty())
53     {
54         int i = team.front(); team.pop();
55         for (int j = 1;j <= N;++j)
56             if (G[i][j]&&root(i) != root(j)&&match[j] != i)
57             {
58                 if (match[j]&&father[match[j]]) flower(i,j);
59                 else if (!father[j])
60                 {
61                     father[tra[++cnt] = j] = i; treec[j] = true;
62                     if (match[j]) push(match[j]);
63                     else

```

```

64         {
65             for (int k = i, l = j, p; k; l = p, k = father[l])
66                 p = match[k], match[k] = l, match[l] = k;
67             return true;
68         }
69     }
70 }
71 }
72 return false;
73 }
74
75 inline int matching()
76 {
77     memset(father, 0, sizeof father); memset(f, 0, sizeof f); memset(path, 0, sizeof path);
78     memset(tra, 0, sizeof tra); memset(match, 0, sizeof match); memset(check, false, sizeof check);
79     memset(treec, false, sizeof treec); memset(pathc, false, sizeof pathc);
80     int ret = cnt = 0;
81     for (int i = 1; i <= N; ++i)
82     {
83         if (match[i]) continue;
84         if (find(i)) ++ret; clear();
85     }
86     return ret;
87 }

```

4.12 Maximum Cardinality Search

```

1  // BZOJ 1006
2  #include <algorithm>
3  #include <queue>
4  #include <cstdio>
5  #include <cstdlib>
6  #include <set>
7  using namespace std;
8
9  #define maxn 10010
10 #define maxc 510
11 #define maxm 1000010
12 int tot, n, m, cnt, color[maxn][maxc], label[maxn], all;
13 int side[maxn], next[maxm*2], toit[maxm*2], per[maxn];
14 bool in[maxn];
15 struct node
16 {
17     int key, ord;
18     friend bool operator < (node a, node b) {return a.key > b.key; }
19 };
20 multiset <node> S;
21
22 inline void add(int a, int b)

```

```

23 {
24     next[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b;
25 }
26
27 inline void ins(int a,int b){add(a,b); add(b,a);}
28
29 inline void mcs()
30 {
31     int i,u;
32     for (i = 1;i <= n;++i) S.insert((node){0,i});
33     while (all < n)
34     {
35         u = (*S.begin()).ord; S.erase(S.begin()); if (in[u]) continue;
36         in[u] = true; per[++all] = u;
37         for (i = side[u];i;i = next[i])
38             if (!in[toit[i]])
39             {
40                 label[toit[i]]++;
41                 S.insert((node){label[toit[i]],toit[i]});
42             }
43     }
44 }
45
46 inline void paint()
47 {
48     int p,i,j,t;
49     for (p = 1;p <= n;++p)
50     {
51         i = per[p];
52         for (j = 1;j <= tot;++j)
53             if (!color[i][j]) {t = j; break; }
54         if (j == tot + 1) t = ++tot;
55         for (j = side[i];j;j = next[j])
56             color[toit[j]][t] = true;
57     }
58 }
59
60 int main()
61 {
62     freopen("1006.in","r",stdin);
63     freopen("1006.out","w",stdout);
64     scanf("%d %d",&n,&m);
65     for (int i = 1;i <= m;++i)
66     { int a,b; scanf("%d %d",&a,&b); ins(a,b); }
67     mcs();
68     paint();
69     printf("%d",tot);
70     fclose(stdin); fclose(stdout);
71     return 0;

```

72 }

4.13 Network Flow with Lower Bound

1. 无源汇有上下界可行流

设原来源点为 *Source*，汇点是 *Sink*。新建一个超级源 *SuperSource* 和超级汇 *SuperSink*。对于原网络中的每一条边 $u \rightarrow v$ ，上界 U ，下界 L ，将它拆分为三条边：

- (1) $u \rightarrow \text{SuperSink}$ ，容量为 L 。
- (2) $\text{SuperSource} \rightarrow v$ ，容量为 L 。
- (3) $u \rightarrow v$ ，容量为 $U - L$ 。

最后添加边 $\text{Sink} \rightarrow \text{Source}$ ，容量为 $+\infty$ 。在新建的网络上，计算从 *SuperSource* 到 *SuperSink* 的最大流。若每条从 *SuperSource* 发出的边都满流，说明存在可行流，否则不。每条边实际流量为容量下界 + 附加流中它的流量。

2. 有源汇有上下界可行流

在“无源汇有上下界可行流”建图上，新增一条 $\text{Sink} \rightarrow \text{Source}$ 的边，容量为 $+\infty$ 即可。

3. 有源汇有上下界最大流

在“有源汇有上下界可行流”建图上，先判断是否存在可行流，若存在可行流，拆掉 $\text{Sink} \rightarrow \text{Source}$ 的边后，接着在图中 $\text{Source} \rightarrow \text{Sink}$ 最大流增广加上原可行流即为最大流答案。（若存在可行流，去掉下界后最大流即为原图有源汇有上下界最大流）

4. 有源汇有上下界最小流

在“有源汇有上下界可行流”建图上，先判断是否存在可行流，若存在可行流，拆掉 $\text{Sink} \rightarrow \text{Source}$ 的边后，用可行流减去在图中 $\text{Sink} \rightarrow \text{Source}$ 增广的最大流即为最小流答案。

在实现时，可以吧 *SuperSource* 连向同一节点的多条边合成一条（容量合并。从同一节点指向 *SuperSink* 的多条边也应合并）。

对于费用流，只需要改变将网络流算法改成费用流算法。对于原网络中的每一条边 $u \rightarrow v$ ，上界 U ，下界 L ，费用 c ，将它拆分为三条边：

- (1) $u \rightarrow \text{SuperSink}$ ，容量为 L ，费用 c 。
- (2) $\text{SuperSource} \rightarrow v$ ，容量为 L ，费用 0 。
- (3) $u \rightarrow v$ ，容量为 $U - L$ ，费用 c 。

4.14 Point Biconnected Component

```

1 // Source: HackerRank - bonnie-and-clyde
2 #include<algorithm>
3 #include<vector>
4 #include<stack>
5 #include<iostream>
6 #include<cstdio>

```

```

7  #include<cstdlib>
8  using namespace std;
9
10 const int maxn = 400010;
11 int N,M,Q,cnt = 1,side[maxn],toit[maxn],nxt[maxn],f[maxn][25],father[maxn],low[maxn];
12 int tot,dep[maxn],dfn[maxn],nside[maxn],ntoit[maxn],nnxt[maxn]; bool cut[maxn];
13 stack <int> S; vector <int> bel[maxn],bcc[maxn]; bool vis[maxn];
14
15 inline int find(int a) { if (father[a] != a) father[a] = find(father[a]); return father[a]; }
16
17 inline void add(int a,int b) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; }
18 inline void ins(int a,int b) { add(a,b); add(b,a); }
19 inline void nadd(int a,int b) { nnxt[++cnt] = nside[a]; nside[a] = cnt; ntoit[cnt] = b; }
20 inline void nins(int a,int b) { nadd(a,b); nadd(b,a); }
21
22 inline int gi()
23 {
24     char ch; int ret = 0,f = 1;
25     do ch = getchar(); while (!(ch >= '0' && ch <= '9') && ch != '-');
26     if (ch == '-') f = -1, ch = getchar();
27     do ret = ret*10+ch-'0', ch = getchar(); while (ch >= '0' && ch <= '9');
28     return ret*f;
29 }
30
31 inline void tj(int now,int fa)
32 {
33     dfn[now] = low[now] = ++cnt; int child = 0;
34     for (int i = side[now]; i; i = nxt[i])
35     {
36         if (toit[i] == fa) continue;
37         if (!dfn[toit[i]])
38         {
39             S.push(i>>1); tj(toit[i],now); ++child;
40             low[now] = min(low[now],low[toit[i]]);
41             if (low[toit[i]] >= dfn[now])
42             {
43                 cut[now] = true; ++tot;
44                 while (true)
45                 {
46                     int t = S.top(); S.pop();
47                     bel[toit[t<<1]].push_back(tot); bel[toit[t<<1|1]].push_back(tot);
48                     bcc[tot].push_back(toit[t<<1]); bcc[tot].push_back(toit[t<<1|1]);
49                     if (t == (i>>1)) break;
50                 }
51             }
52         }
53         else low[now] = min(low[now],dfn[toit[i]]);
54     }
55     if (!fa && child == 1) cut[now] = false;

```

```

56 }
57
58 inline void build()
59 {
60     vector<int> cuts; cnt = 1;
61     for (int i = 1; i <= tot; ++i)
62     {
63         sort(bcc[i].begin(), bcc[i].end());
64         bcc[i].erase(unique(bcc[i].begin(), bcc[i].end()), bcc[i].end());
65     }
66     for (int i = 1; i <= N; ++i) if (cut[i]) cuts.push_back(i);
67     for (auto x: cuts)
68     {
69         sort(bel[x].begin(), bel[x].end());
70         bel[x].erase(unique(bel[x].begin(), bel[x].end()), bel[x].end());
71         ++tot; for (auto y: bel[x]) nins(tot, y);
72         bel[x].clear(); bel[x].push_back(tot); bcc[tot].push_back(x);
73     }
74 }
75
76 inline void dfs(int now)
77 {
78     vis[now] = true;
79     for (int i = 1; (1<<i) <= dep[now]; ++i) f[now][i] = f[f[now][i-1]][i-1];
80     for (int i = nside[now]; i; i = nnxt[i])
81     {
82         if (vis[ntoit[i]]) continue; f[ntoit[i]][0] = now;
83         dep[ntoit[i]] = dep[now] + 1; dfs(ntoit[i]);
84     }
85 }
86
87 inline int jump(int a, int b) { for (int i = 0; b; ++i, b >>= 1) if (b&1) a = f[a][i]; return a; }
88 inline int lca(int a, int b)
89 {
90     if (dep[a] < dep[b]) swap(a, b);
91     a = jump(a, dep[a] - dep[b]); if (a == b) return a;
92     for (int i = 0; i >= 0; )
93     {
94         if (f[a][i] != f[b][i]) a = f[a][i], b = f[b][i], ++i;
95         else --i;
96     }
97     return f[a][0];
98 }
99
100 inline bool check(int u, int v, int w)
101 {
102     if (find(u) != find(v) || find(v) != find(w)) return false;
103     if (u == w || v == w) return true; if (u == v) return false;
104     int uu = bel[u][0], vv = bel[v][0], ww = bel[w][0], su, sv;

```

```

105     if (uu == ww || vv == ww) return true;
106     if (lca(uu, ww) == ww) su = jump(uu, dep[uu] - dep[ww] - 1); else su = f[ww][0];
107     if (lca(vv, ww) == ww) sv = jump(vv, dep[vv] - dep[ww] - 1); else sv = f[ww][0];
108     if (su == sv)
109     {
110         if (!cut[w]) return false;
111         else
112         {
113             if (su == uu || sv == vv) return true; int ssu, ssv;
114             if (lca(su, uu) == su) ssu = jump(uu, dep[uu] - dep[su] - 1); else ssu = f[su][0];
115             if (lca(sv, vv) == sv) ssv = jump(vv, dep[vv] - dep[sv] - 1); else ssv = f[sv][0];
116             if (ssu == ssv) return false; else return true;
117         }
118     }
119     else return true;
120 }
121
122 int main()
123 {
124     freopen("J.in", "r", stdin);
125     freopen("J.out", "w", stdout);
126     N = gi(); M = gi(); Q = gi();
127     for (int i = 1; i <= N; ++i) father[i] = i;
128     for (int i = 1, a, b; i <= M; ++i)
129     {
130         ins(a = gi(), b = gi());
131         a = find(a), b = find(b);
132         if (a != b) father[a] = b;
133     }
134     cnt = 0; for (int i = 1; i <= N; ++i) if (!dfn[i]) tj(i, 0);
135     build(); for (int i = 1; i <= N; ++i) if (!vis[i]) dfs(i);
136     while (Q--)
137     {
138         int u = gi(), v = gi(), w = gi();
139         if (check(u, v, w)) puts("YES"); else puts("NO");
140     }
141     return 0;
142 }

```

4.15 Steiner Tree

```

1  /*
2  *  Steiner Tree: 求, 使得指定  $K$  个点连通的生成树的最小总权值
3  *   $st[i]$  表示顶点  $i$  的标记值, 如果  $i$  是指定集合内第  $m(0 \leq m < K)$  个点, 则  $st[i] = 1 \leq m$ 
4  *   $endSt = 1 \leq K$ 
5  *   $dptree[i][state]$  表示以  $i$  为根, 连通状态为  $state$  的生成树值
6  */
7  inline void update(int &x, int y) { if (x == -1) x = y; else if (x > y) x = y; }
8  inline void spfa(int state)

```



```

9  {
10     while (!team.empty())
11     {
12         int now = team.front(); team.pop();
13         for (int i = side[now]; i; i = nxt[i])
14         {
15             int v = toit[i];
16             if (f[v][st[v]|state] == -1 || f[v][st[v]|state] > f[now][state]+len[i])
17             {
18                 f[v][st[v]|state] = f[now][state]+len[i];
19                 if ((st[v]|state) != state || vis[v][state]) continue;
20                 vis[v][state] = true; team.push(v);
21             }
22         }
23         vis[now][state] = false;
24     }
25 }
26 inline int work()
27 {
28     endSt = 1<<(K<<1);
29     memset(f,-1,sizeof(f)); memset(st,0,sizeof(st)); memset(dp,-1,sizeof(dp));
30     memset(vis,false,sizeof(vis)); memset(side,0,sizeof(side));
31     for (int i = 1; i <= K; ++i) st[i] = 1<<(i-1);
32     for (int i = 1; i <= K; ++i) st[N-K+i] = 1<<(i+K-1);
33     for (int i = 1; i <= N; ++i) f[i][st[i]] = 0;
34     for (int j = 1; j < endSt; ++j)
35     {
36         for (int i = 1; i <= N; ++i)
37         {
38             if (!st[i] || (st[i]&j))
39                 for (int sub = (j-1)&j; sub; sub = (sub-1)&j)
40                 {
41                     int x = sub|st[i], y = (j-sub)|st[i];
42                     if (f[i][x] != -1 && f[i][y] != -1)
43                         update(f[i][j], f[i][x]+f[i][y]);
44                 }
45             if (f[i][j] != -1) team.push(i), vis[i][j] = true;
46         }
47         spfa(j);
48     }
49 }

```

4.16 Stoer Wagner Algorithm

```

1  int G[maxn][maxn], node[maxn], dis[maxn]; bool visit[maxn];
2
3  inline int solve(int n)
4  {
5      if (n == 1) return inf;

```

```

6     int answer = inf;
7     for (int i = 0; i < n; ++i) node[i] = i;
8     while (n > 1)
9     {
10        int mx = 1;
11        for (int i = 0; i < n; ++i)
12        {
13            dis[node[i]] = G[node[0]][node[i]];
14            if (dis[node[i]] > dis[node[mx]]) mx = i;
15        }
16        int prev = 0;
17        memset(visit, false, sizeof visit);
18        visit[node[0]] = true;
19        for (int i = 1; i < n; ++i)
20        {
21            if (i == n-1)
22            {
23                answer = min(answer, dis[node[mx]]);
24                for (int k = 0; k < n; ++k)
25                    G[node[k]][node[prev]] = (G[node[prev]][node[k]] += G[node[k]][node[mx]]);
26                node[mx] = node[--n];
27            }
28            visit[node[mx]] = true; prev = mx; mx = -1;
29            for (int j = 1; j < n; ++j)
30                if (!visit[node[j]])
31                {
32                    dis[node[j]] += G[node[prev]][node[j]];
33                    if (mx == -1 || dis[node[mx]] < dis[node[j]]) mx = j;
34                }
35        }
36    }
37    return answer;
38 }

```

4.17 Strongly Connected Component

```

1  int dfn[maxn], low[maxn], timestamp;
2  stack <int> stk; vector <int> scc[maxn];
3  void tarjan(int now)
4  {
5      dfn[now] = low[now] = ++timestamp;
6      stk.push(now);
7      for (int i = side[now]; i; i = nxt[i])
8      {
9          if (!dfn[toit[i]])
10             tarjan(toit[i]), low[now] = min(low[now], low[toit[i]]);
11          else if (!bel[toit[i]]) low[now] = min(low[now], dfn[toit[i]]);
12      }
13      if (dfn[now] == low[now])

```

```

14     {
15         ++tot;
16         while (stk.top() != now)
17         {
18             scc[tot].push_back(stk.top());
19             bel[stk.top()] = tot; stk.pop();
20         }
21         scc[tot].push_back(stk.top());
22         bel[stk.top()] = tot; stk.pop();
23     }
24 }

```

4.18 Virtual Tree

```

1  int N,cnt,timestamp,dfn[maxn],f[maxn][25],side[maxn],H[maxn];
2  int dep[maxn],toit[maxn],nxt[maxn],last[maxn],cost[maxn],stk[maxn];
3  ll best[maxn],g[maxn];
4
5  inline void add(int a,int b,int c) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b;
   ⇨ cost[cnt] = c; }
6  inline void ins(int a,int b,int c) { add(a,b,c); add(b,a,c); }
7
8  inline void nadd(int a,int b,int idc)
9  {
10     if (a == b) return;
11     if (last[a] != idc) side[a] = 0,last[a] = idc;
12     if (last[b] != idc) side[b] = 0,last[b] = idc;
13     nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b;
14 }
15
16 inline bool cmp(int a,int b) { return dfn[a] < dfn[b]; }
17
18 inline void dfs(int now)
19 {
20     dfn[now] = ++timestamp;
21     for (int i = 1;(1<<i) <= dep[now];++i)
22         f[now][i] = f[f[now][i-1]][i-1];
23     for (int i = side[now];i;i = nxt[i])
24         if (toit[i] != f[now][0])
25         {
26             best[toit[i]] = min(best[now],(ll)cost[i]);
27             dep[toit[i]] = dep[now]+1;
28             f[toit[i]][0] = now; dfs(toit[i]);
29         }
30 }
31
32 inline int jump(int a,int step) { for (int i = 0;step;step >>= 1,++i) if (step&1) a = f[a][i];
   ⇨ return a; }
33 inline int lca(int a,int b)

```

```

34 {
35     if (dep[a] < dep[b]) swap(a,b);
36     a = jump(a,dep[a]-dep[b]);
37     if (a == b) return a;
38     for (int i = 0; i >= 0;)
39     {
40         if (f[a][i] != f[b][i])
41             a = f[a][i], b = f[b][i], ++i;
42         else --i;
43     }
44     return f[a][0];
45 }
46
47 inline void work(int idc)
48 {
49     cnt = 0; int K = gi(), tot, top;
50     for (int i = 1; i <= K; ++i) H[i] = gi();
51     sort(H+1, H+K+1, cmp); H[tot = 1] = H[1];
52     for (int i = 2; i <= K; ++i) if (lca(H[tot], H[i]) != H[tot]) H[++tot] = H[i];
53     stk[top = 1] = 1;
54     for (int i = 1; i <= tot; ++i)
55     {
56         int ans = lca(H[i], stk[top]);
57         while (true)
58         {
59             if (dep[ans] >= dep[stk[top-1]]) { nadd(ans, stk[top--], idc); break; }
60             nadd(stk[top-1], stk[top], idc); --top;
61         }
62         if (stk[top] != ans) stk[++top] = ans;
63         if (stk[top] != H[i]) stk[++top] = H[i];
64     }
65     while (--top) nadd(stk[top], stk[top+1], idc);
66     // dp(1); printf("%lld\n", g[1]);
67 }

```

4.19 Zhu-Liu Algorithm

```

1 struct Directed_MT
2 {
3     struct Edge
4     {
5         int u, v, w;
6         inline Edge() = default;
7         inline Edge(int _u, int _v, int _w) : u(_u), v(_v), w(_w) {}
8     };
9     int n, m, vis[maxn], pre[maxn], id[maxn], in[maxn]; Edge edges[maxn];
10
11     inline void init(int _n) { n = _n; m = 0; }
12     inline void AddEdge(int u, int v, int w) { edges[m++] = Edge(u, v, w); }

```

```

13     inline int work(int root)
14     {
15         int ret = 0;
16         while (true)
17         {
18             // 初始化
19             for (int i = 0; i < n; ++i) in[i] = inf+1;
20             for (int i = 0; i < m; ++i)
21             {
22                 int u = edges[i].u, v = edges[i].v;
23                 // 找寻最小入边, 删除自环
24                 if (edges[i].w < in[v] && u != v)
25                     in[v] = edges[i].w, pre[v] = u;
26             }
27             // 如果没有最小入边, 表示该点不连通, 则最小树形图形成失败
28             for (int i = 0; i < n; ++i)
29             {
30                 if (i == root) continue;
31                 if (in[i] == inf+1) return inf;
32             }
33             int cnt = 0; // 记录缩点
34             memset(id, -1, sizeof id); memset(vis, -1, sizeof vis);
35             in[root] = 0;
36             for (int i = 0; i < n; ++i)
37             {
38                 ret += in[i]; int v = i;
39                 // 找寻自环
40                 while (vis[v] != i && id[v] == -1 && v != root)
41                     vis[v] = i, v = pre[v];
42                 if (v != root && id[v] == -1)
43                 {
44                     // 这里不能从 i 开始找, 因为 i 有可能不在自环内
45                     for (int u = pre[v]; u != v; u = pre[u]) id[u] = cnt;
46                     id[v] = cnt++;
47                 }
48             }
49             // 如果没有自环了, 表示最小树形图成功了
50             if (!cnt) break;
51             // 找到那些不是自环的, 重新给那些点进行标记
52             for (int i = 0; i < n; ++i)
53                 if (id[i] == -1) id[i] = cnt++;
54             for (int i = 0; i < m; ++i)
55             {
56                 int u = edges[i].u, v = edges[i].v;
57                 edges[i].v = id[v]; edges[i].u = id[u];
58                 if (id[u] != id[v]) edges[i].w -= in[v];
59             }
60             // 缩点完后, 点的数量就变了
61             n = cnt; root = id[root];

```

```

62     }
63     return ret;
64 }
65 }MT;

```

4.20 ZKW Cost Flow

```

1  // To be written
2  bool spfa()
3  {
4      memset(mark,0,sizeof(mark));
5      memset(d,0x7,sizeof(d));
6      d[T] = 0; mark[T] = 1;
7      queue<int> team;
8      team.push(T);
9      while (!team.empty())
10     {
11         int now = team.front();
12         team.pop();
13         for (int i = head[now];i;i = e[i].next)
14             if (e[i^1].v&& d[e[i].to] > d[now]-e[i].c)
15                 {
16                     d[e[i].to] = d[now]-e[i].c;
17                     if (!mark[e[i].to])
18                         {
19                             mark[e[i].to] = true;
20                             team.push(e[i].to);
21                         }
22                 }
23         mark[now] = false;
24     }
25     if (d[0] > 10000000) return false;
26     return true;
27 }
28
29 int dfs(int x,int f)
30 {
31     if (x == T)
32     {
33         mark[T] = 1;
34         return f;
35     }
36     int used = 0,w;
37     mark[x] = true;
38     for (int i = head[x];i;i = e[i].next)
39         if (!mark[e[i].to]&&e[i].v&& d[x]-e[i].c==d[e[i].to])
40             {
41                 w = f - used;
42                 w = dfs(e[i].to,small(e[i].v,w));

```

```
43         ans += w*e[i].c;
44         e[i].v -= w;
45         e[i^1].v += w;
46         used += w;
47         if (used == f) return f;
48     }
49     return used;
50 }
51
52 void zkw()
53 {
54     while (spfa())
55     {
56         mark[T] = 1;
57         while (mark[T])
58         {
59             memset(mark,0,sizeof(mark));
60             dfs(0,inf);
61         }
62     }
63 }
```

Chapter 5

Number Theory

5.1 Baby Step Giant Step

```
1 // To Be Verified
2 // 求出最小的 t 使得  $X^t = Y \bmod mod$ 
3 inline int bsgs(int X,int Y,int mod)
4 {
5     int m = ceil(sqrt(mod+0.5)),mul = 1,res = 1;
6     if (Y == 1) return 0;
7     hash.clear(); hash[Y] = 0;
8     for (int i = 1;i <= m;++i)
9     {
10         mul = ((ll)mul*(ll)X)%mod;
11         if (mul == Y) return i;
12         hash[(ll)Y*(ll)mul%mod] = i;
13     }
14     res = mul;
15     for (int i = 2;(i-1)*m <= mod;++i)
16     {
17         res = (ll)res*(ll)mul%mod;
18         if (hash.find(res) != hash.end()) return i*m-hash[res];
19     }
20     return -1;
21 }
```

5.2 Chinese Remainder Theorem

```
1 //快速乘
2 inline ll qsc(ll a,ll b,ll mod)
3 {
4     ll ret = 0; a %= mod,b %= mod;
5     for (;b;b >>= 1)
6     {
7         if (b&1)
8         {
```



```

9         ret += a;
10        if (ret >= mod) ret -= mod;
11    }
12    a += a; if (a >= mod) a -= mod;
13 }
14 return ret;
15 }
16
17 inline ll msm(ll a,ll b,ll mod)
18 {
19     ll ret = 1;
20     for (;b;b >>= 1,a = qsc(a,a,mod)) if (b&1) ret = qsc(ret,a,mod);
21     return ret;
22 }
23
24 inline ll crt()
25 {
26     ll lcm = 1,ret = 0;
27     for (int i = 1;i <= K;++i) lcm *= (ll)P[i];
28     for (int i = 1;i <= K;++i)
29     {
30         ll tm = lcm/P[i];
31         ll inv = msm(tm,P[i]-2,P[i]);
32         ret = (ret+qsj(qsj(tm,inv,lcm),res[i],lcm))%lcm;
33     }
34     return ret;
35 }

```

5.3 Extended Euclidean Algorithm

```

1  //By yxj
2  inline ll exgcd(ll a,ll b,ll c) //ax mod b = c
3  {
4      if (a == 0) return -1;
5      else if (c % a == 0) return c/a;
6      ll t = exgcd(b % a,a,((-c % a)+a)%a);
7      if (t == -1) return -1;
8      return (t*b+c)/a;
9  }
10
11 //Input:a,b,&x,&y,ax+by = gcd(a,b)
12 //Output:gcd(a,b)
13 inline int exgcd(int a,int b,int &x,int &y)
14 {
15     if (!b) { x = 1,y = 0; return a; }
16     else
17     {
18         int r = exgcd(b,a%b,y,x);
19         y -= x*(a/b); return r;

```

```

20     }
21 }

```

5.4 Linearly Sieve

```

1  //欧拉函数
2  inline void ready()
3  {
4      phi[1] = 1;
5      for (int i = 2; i < maxn; ++i)
6      {
7          if (!exist[i]) phi[i] = i-1, prime[++tot] = i;
8          for (int j = 1; j <= tot; ++j)
9          {
10             if (i*prime[j] >= maxn) break;
11             exist[i*prime[j]] = true;
12             if (i % prime[j] == 0)
13                 { phi[i*prime[j]] = phi[i]*prime[j]; break; }
14             else phi[i*prime[j]] = phi[i]*phi[prime[j]];
15         }
16     }
17 }
18 //莫比乌斯函数
19 inline void ready()
20 {
21     mu[1] = 1;
22     for (int i = 2; i <= 50000; ++i)
23     {
24         if (!exist[i]) { prime[++tot] = i; mu[i] = -1; }
25         for (int j = 1; j <= tot && prime[j]*i <= 50000; ++j)
26         {
27             exist[i*prime[j]] = true;
28             if (i % prime[j] == 0) { mu[i*prime[j]] = 0; break; }
29             mu[i*prime[j]] = -mu[i];
30         }
31     }
32 }

```

5.5 N-Power Residue

```

1  //Input: p, N, a p is a prime
2  //Output: the solutions to equation  $x^N \equiv a \pmod{p}$  in  $[0, p-1]$ 
3  inline vector<int> residue(int p, int N, int a)
4  {
5      int g = PrimitiveRoot(p); ll m = bsgs(g, a, p);
6      vector<int> ret;
7      if (!a) { ret.push_back(0); return ret; }
8      if (m == -1) return ret;
9      ll A = N, B = p-1, C = m, x, y, d = exgcd(A, B, x, y);

```

```

10     if (C % d) return ret;
11     x *= (C / d)%B;
12     ll delta = B / d;
13     for (int i = 0; i < d; ++i)
14     {
15         x += delta; if (x >= B) x -= B;
16         ret.push_back((int)qsm(g,x,p));
17     }
18     sort(ret.begin(),ret.end());
19     ret.erase(unique(ret.begin(),ret.end()),ret.end());
20     return ret;
21 }

```

5.6 Number Theoretic Transformation

```

1  // The First Version
2  struct node
3  {
4      int a[maxn*2],len;
5      inline void NTT(int loglen,int len,int on)
6      {
7          for (int i = 0,j,t,p;i < len;++i)
8          {
9              for (j = 0,t = i,p = 0;j < loglen;++j,t >>= 1)
10                 p <<= 1,p |= t&1;
11                 if (p > i) swap(a[p],a[i]);
12             }
13             for (int s = 1,k = 2;s <= loglen;++s,k <<= 1)
14             {
15                 int wn; if (on) wn = e[s]; else wn = ine[s];
16                 for (int i = 0;i < len;i += k)
17                 {
18                     int w = 1;
19                     for (int j = 0;j < (k >> 1);++j,w = (ll)wn*w%rhl)
20                     {
21                         int u = a[i+j],v = (ll)w*a[i+j+(k>>1)]%rhl;
22                         a[i+j] = u+v; if (a[i+j] >= rhl) a[i+j] -= rhl;
23                         a[i+j+(k>>1)] = u-v;
24                         if (a[i+j+(k>>1)] < 0) a[i+j+(k>>1)] += rhl;
25                     }
26                 }
27             }
28             if (!on)
29             {
30                 int inv = qsm(len,rhl-2,rhl);
31                 for (int i = 0;i < len;++i) a[i] = (ll)a[i]*inv%rhl;
32             }
33         }
34         friend inline bool operator *(node x,node y)

```

```

35     {
36         int loglen = 0, len;
37         for (; (1<<loglen)<x.len+y.len; ++loglen); len = 1<<loglen;
38         x.NTT(loglen, len, 1); y.NTT(loglen, len, 1);
39         for (int i = 0; i < (1<<loglen); ++i) x.a[i] = (ll)x.a[i]*y.a[i]%rhl;
40         x.NTT(loglen, len, 0);
41     }
42 };
43
44 int main()
45 {
46     for (int i = 1; i < 20; ++i)
47         e[i] = qsm(gg, (rhl-1)>>i, rhl), ine[i] = qsm(e[i], rhl-2, rhl);
48 }
49
50 // The Second Version
51 typedef long long ll;
52 ll e[20], ine[20];
53
54 inline ll qsm(ll a, int b, int c)
55 {
56     ll ret = 1;
57     for (; b >>= 1, (a *= a) %= c; if (b&1) (ret *= a) %= c;
58     return ret;
59 }
60
61 inline void NTT(ll *a, int loglen, int len, int on)
62 {
63     for (int i = 0, j, t, p; i < len; ++i)
64     {
65         for (j = 0, t = i, p = 0; j < loglen; ++j, t >>= 1)
66             p <= 1, p |= t&1;
67         if (p > i) swap(a[p], a[i]);
68     }
69     for (int s = 1, k = 2; s <= loglen; ++s, k <= 1)
70     {
71         int wn; if (on) wn = e[s]; else wn = ine[s];
72         for (int i = 0; i < len; i += k)
73         {
74             int w = 1;
75             for (int j = 0; j < (k >> 1); ++j, w = (ll)wn*w%lhh)
76             {
77                 int u = a[i+j], v = (ll)w*a[i+j+(k>>1)]%lhh;
78                 a[i+j] = u+v; if (a[i+j] >= lhh) a[i+j] -= lhh;
79                 a[i+j+(k>>1)] = u-v;
80                 if (a[i+j+(k>>1)] < 0) a[i+j+(k>>1)] += lhh;
81             }
82         }
83     }

```

```

84     if (!on)
85     {
86         int inv = qsm(len, lhh-2, lhh);
87         for (int i = 0; i < len; ++i) a[i] = a[i]*inv%lhh;
88     }
89 }
90
91 struct Polynomial
92 {
93     int len; ll array[maxn<<2];
94     inline Polynomial(int _len = 0):len(_len) {}
95     inline Polynomial(ll a[], int n):len(n) { for (int i = 0; i < n; ++i) array[i] = a[i]; }
96     inline ll operator [] (int n) const { return array[n]; }
97     inline ll &operator [] (int n) { return array[n]; }
98     inline void set(int n) { len = n; }
99     inline void set(int n, ll a[]) { len = n; for (int i = 0; i < n; ++i) array[i] = a[i]; }
100    inline void extend(int key)
101    {
102        for (int i = len; i < (1<<key); ++i)
103            array[i] = 0;
104    }
105    inline void cut(int key) { len = key; }
106    inline void transform(int loglen, int on) { NTT(array, loglen, 1<<loglen, on); }
107 }; //变量只能定义在全局，不然会 re
108
109 inline Polynomial multiply(Polynomial &pa, Polynomial &ret) // self-multiply
110 {
111     int loglen = 0;
112     while ((1<<loglen) < (pa.len<<1)-1) ++loglen;
113     pa.extend(1<<loglen); pa.transform(loglen, 1);
114     for (int i = 0; i < (1<<loglen); ++i) ret[i] = pa[i]*pa[i]%lhh;
115     ret.transform(loglen, 0); ret.cut((pa.len<<1)-1);
116     return ret;
117 }
118 inline Polynomial multiply(Polynomial &pa, Polynomial &pb, Polynomial &ret)
119 {
120     int loglen = 0;
121     while ((1<<loglen) < (pa.len+pb.len-1)) ++loglen;
122     pa.extend(1<<loglen); pa.transform(loglen, 1);
123     pb.extend(1<<loglen); pb.transform(loglen, 1);
124     for (int i = 0; i < (1<<loglen); ++i) ret[i] = pa[i]*pb[i]%lhh;
125     ret.transform(loglen, 0); ret.cut(pa.len+pb.len-1);
126     return ret;
127 }
128
129 int main()
130 {
131     for (int i = 1; i < 20; ++i)
132         e[i] = qsm(g, (lhh-1)>>i, lhh), ine[i] = qsm(e[i], lhh-2, lhh);

```

```
133 }
```

5.7 Pollard Rho Algorithm

```
1  const int prime[] = {0,2,3,5,7,11,13,17,19,23,29,31};
2
3  inline ll mul(ll a,ll b,ll p) { return (a*b-((ll)((ld)a/p*b+1e-3)*p)+p)%p; }
4
5  inline bool check(ll m)
6  {
7      if (m <= 2) return m == 2;
8      ll tmp = m-1; int t = 0;
9      while (!(tmp&1)) ++t,tmp >>= 1;
10     for (int i = 1;i <= 10;++i)
11     {
12         int a = prime[i];
13         if (a == m) return true;
14         ll w = qsm(a,tmp,m);
15         for (int it = 1;it <= t;++it)
16         {
17             ll pf = mul(w,w,m);
18             if (pf == 1&&(w != 1&&w != m-1)) return false;
19             w = pf;
20         }
21         if (w != 1) return false;
22     }
23     return true;
24 }
25 inline void rho(ll m)
26 {
27     if (check(m)) { fac[++nn] = m; return; }
28     while (true)
29     {
30         ll X = (ll)rand()*rand()%(m-1)+1,Y = X;
31         ll c = (ll)rand()*rand()%(m-1)+1; int i,j;
32         for (i = j = 2;++i)
33         {
34             X = (mul(X,X,m)+c) % m;
35             ll d = __gcd(abs(X-Y),m);
36             if (1 < d&&d < m) { rho(d),rho(m/d); return; }
37             if (X == Y) break; if (i == j) Y = X,j <= 1;
38         }
39     }
40 }
41 inline void factor(ll m) { nn = 0; if (m > 1) rho(m); sort(fac+1,fac+nn+1); }
42
43
44 //__int128 Version
45 typedef __int128 int128;
```

```

46 inline int128 mul(int128 a,int128 b,int128 mod)
47 {
48     int128 ret = 0; a %= mod,b %= mod;
49     for (;b;b >>= 1)
50     {
51         if (b&1)
52         {
53             ret += a;
54             if (ret >= mod) ret -= mod;
55         }
56         a += a; if (a >= mod) a -= mod;
57     }
58     return ret;
59 }
60
61 inline int128 qsm(int128 a,int128 b,int128 mod)
62 {
63     int128 ret = 1;
64     for (;b;b >>= 1,a = mul(a,a,mod)) if (b&1) ret = mul(ret,a,mod);
65     return ret;
66 }
67
68 inline void ready()
69 {
70     for (int i = 2;i <= 100;++i)
71     {
72         if (prime[i]) continue; prime[++tot] = i;
73         for (int j = i*i;j <= 100;j += i) prime[j] = 1;
74     }
75 }
76
77 inline int128 gi()
78 {
79     int128 ret = 0; char ch;
80     do ch = getchar(); while (!(ch >= '0'&&ch <= '9'));
81     do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
82     return ret;
83 }
84
85 inline int128 gcd(int128 a,int128 b) { if (b == 0) return a; return gcd(b,a%b); }
86
87 inline int128 Abs(int128 a) { if (a < 0) return -a; return a; }
88
89 inline bool check(int128 m)
90 {
91     if (m <= 2) return m == 2;
92     int128 tmp = m-1; int t = 0;
93     while (!(tmp&1)) ++t,tmp >>= 1;
94     for (int i = 1;i <= tot;++i)

```

```

95     {
96         int a = prime[i];
97         if (a == m) return true;
98         int128 w = qsm(a,tmp,m);
99         for (int it = 1; it <= t; ++it)
100         {
101             int128 pf = mul(w,w,m);
102             if (pf == 1 && (w != 1 && w != m-1)) return false;
103             w = pf;
104         }
105         if (w != 1) return false;
106     }
107     return true;
108 }
109 inline void rho(int128 m)
110 {
111     if (check(m)) { fac[++nn] = m; return; }
112     while (true)
113     {
114         int128 X = (int128)rand()*(int128)rand()%(m-1)+1, Y = X;
115         int128 c = (int128)rand()*(int128)rand()%(m-1)+1; int i,j;
116         for (i = j = 2; ++i)
117         {
118             X = (mul(X,X,m)+c)%m;
119             int128 d = gcd(Abs(X-Y),m);
120             if (1 < d && d < m) { rho(d), rho(m/d); return; }
121             if (X == Y) break; if (i == j) Y = X, j <= 1;
122         }
123     }
124 }
125
126 inline void factor(int128 m) { nn = 0; if (m > 1) rho(m); sort(fac+1,fac+nn+1); }

```

5.8 Primitive Root

```

1  //Input: A prime p
2  //Output: p's primitive root
3  vector <ll> a;
4
5  inline g_test(ll g, ll p)
6  {
7      for (ll i = a.size()-1; i >= 0; --i)
8          if (qsm(g,(p-1)/a[i],p) == 1) return 0;
9      return 1;
10 }
11
12 inline ll PrimitiveRoot(ll p)
13 {
14     ll tmp = p - 1;

```



```

15     for (ll i = 2; i <= tmp/i; ++i)
16     {
17         if (!(tmp % i))
18         {
19             a.push_back(i);
20             while (!(tmp%i)) tmp /= i;
21         }
22         if (tmp != 1) a.push_back(tmp);
23     }
24     for (ll g = 1; ; ++g) if (g_test(g,p)) return g;
25 }

```

5.9 Quadratic Residue

```

1  //判断是否存在 x, 使得  $x^2 \equiv a \pmod n$ , 存在返回最小 x, 否则返回 -1
2  inline int modsqr(int a, int n)
3  {
4      int b, k, i, x;
5      if (n == 2) return a & 1;
6      if (qsm(a, (n-1)>>1, n) == 1)
7      {
8          if (n % 4 == 3) x = qsm(a, (n+1)>>2, n);
9          else
10         {
11             for (b = 1; qsm(b, (n-1)>>1, n) == 1; ++b);
12             i = (n-1)>>1; k = 0;
13             do
14             {
15                 i >>= 1, k >>= 1;
16                 if (!((qsm(a, i, n) * (ll)qsm(b, k, n) + 1) % n)) k += ((n-1)>>1);
17             }
18             while (!(i&1));
19             x = (qsm(a, (i+1)>>1, n) * (ll)qsm(b, k>>1, n)) % n;
20         }
21         if ((x << 1) > n) x = n-x;
22         return x;
23     }
24     return -1;
25 }

```

5.10 Single Variable Modulus Linear Equation

```

1  //Input: a, b, n
2  //Output: All the solutions in  $[0, n)$  to the equation  $ax \equiv b \pmod n$ 
3  inline vector <ll> LineModEquation(ll a, ll b, ll n)
4  {
5      ll x, y, d = exgcd(a, n, x, y); vector <ll> ans;
6      if (!(b % d))
7      {

```

```
8         x %= n; x += n; x % n;
9         ans.push_back(x*(b/d)%n);
10        for (ll i= 1;i < d;++i) ans.push_back((ans[0]+i*n/d)%n);
11        //若找最小的，直接就是 (ans[0]%(n/d))
12    }
13    return ans;
14 }
```

Chapter 6

Numerical Algorithms

6.1 Counting Integral Points under Straight Line

```
1 // \sum_{i=0}^{n-1} (a+bi)/m
2 inline ll count(ll n,ll a,ll b,ll m)
3 {
4     if (!b) return n*(a/m);
5     else if (a >= m) return n*(a/m)+count(n,a%m,b,m);
6     else if (b >= m) return (n-1)*n/2*(b/m)+count(n,a,b%m,m);
7     else return count((a+b*n)/m,(a+b*n)%m,m,b);
8 }
```

6.2 Evaluation of Expression

```
1 #include<bitset>
2 #include<stack>
3 #include<iostream>
4 #include<cstdio>
5 #include<cstdlib>
6 using namespace std;
7
8 const int maxn = 200010;
9 int T,N,M,pri[256],match[maxn]; bitset <maxn> A,B; char s[maxn];
10
11 inline int gi()
12 {
13     char ch; int ret = 0,f = 1;
14     do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
15     if (ch == '-') f = -1,ch = getchar();
16     do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
17     return ret*f;
18 }
19
20 inline bitset <maxn> calc(int l,int r)
21 {
```

```

22     if (l > r) return bitset <maxn>();
23     while (match[l] == r) ++l,r--;
24     if (l == r) { if (s[l] == 'A') return A; else return B; }
25     int cur = 0; pair <int,int> mn(1<<30,0);
26     for (int i = l;i <= r;++i)
27     {
28         if (s[i] == '(') cur += 10;
29         else if (s[i] == ')') cur -= 10;
30         else if (pri[s[i]])
31             if (make_pair(cur+pri[s[i]],i) < mn)
32                 mn = make_pair(cur+pri[s[i]],i);
33     }
34     int pos = mn.second; auto L = calc(l,pos-1),R = calc(pos+1,r);
35     if (s[pos] == '+') return L|R;
36     else if (s[pos] == '*') return L&R;
37     else return ~R;
38 }
39
40 int main()
41 {
42     freopen("H.in","r",stdin);
43     freopen("H.out","w",stdout);
44     pri['+'] = 1; pri['*'] = 2; pri['-'] = 3;
45     while (++T)
46     {
47         N = gi(),M = gi(); if (!N) break;
48         A.reset(); B.reset(); printf("Case %d: ",T);
49         for (int K = gi();K--;) A[gi()] = 1;
50         for (int K = gi();K--;) B[gi()] = 1;
51         scanf("%s",s+1); stack <int> S;
52         for (int i = 1;i <= M;++i) match[i] = -1;
53         for (int i = 1;i <= M;++i)
54         {
55             if (s[i] == '(') S.push(i);
56             else if (s[i] == ')')
57             {
58                 int t = S.top(); S.pop();
59                 match[match[t] = i] = t;
60             }
61         }
62         int tot = 0; auto ans = calc(1,M);
63         for (int i = 1;i <= N;++i) tot += ans[i]; printf("%d",tot);
64         for (int i = 1;i <= N;++i) if (ans[i]) printf(" %d",i); putchar('\n');
65     }
66     return 0;
67 }

```

6.3 Fast Fourier Transformation

```

1  // The First Version
2  struct Vir
3  {
4      double re,im;
5      inline Vir(double _re = 0,double _im = 0):re(_re),im(_im) {}
6      friend inline Vir operator*(const Vir &a,const Vir &b) { return
↪   Vir(a.re*b.re-a.im*b.im,a.re*b.im+a.im*b.re); }
7      friend inline Vir operator+(const Vir &a,const Vir &b) { return Vir(a.re+b.re,a.im+b.im); }
8      friend inline Vir operator-(const Vir &a,const Vir &b) { return Vir(a.re-b.re,a.im-b.im); }
9      friend inline Vir operator/(const Vir &a,double r) { return Vir(a.re/r,a.im/r); }
10 }pa[maxn],pb[maxn];
11
12 inline void fft(Vir *a,int loglen,int len,int on)
13 {
14     for (register int i = 0,j,t,p;i < len;++i)
15     {
16         for (p = j = 0,t = i;j < loglen;++j,t >>= 1)
17             p <= 1,p |= (t&1);
18         if (p > i) swap(a[p],a[i]);
19     }
20     for (register int m = 2,s = 1;s <= loglen;++s,m <= 1)
21     {
22         register Vir w(cos(2*pi*on/m),sin(2*pi*on/m));
23         for (int i = 0;i < len;i += m)
24         {
25             register Vir wn(1,0);
26             for (register int j = 0;j < (m>>1);++j,wn = wn*w)
27             {
28                 register Vir u = a[i+j],v = wn*a[i+j+(m>>1)];
29                 a[i+j] = u+v; a[i+j+(m>>1)] = u-v;
30             }
31         }
32     }
33     if (on == -1) for (int i = 0;i < len;++i) a[i] = a[i]/len;
34 }
35
36 inline void work()
37 {
38     int loglen = 0,len;
39     while ((1<<loglen) < len) ++loglen; len = 1 << loglen;
40     fft(pa,loglen,len,1); fft(pb,loglen,len,1);
41     for (int i = 0;i < len;++i) pa[i] = pa[i]*pb[i];
42     fft(pa,loglen,len,-1);
43 }
44
45 //The Second Version
46 const double pi = acos(-1.0);
47 struct Complex

```

```

48 {
49     double re,im;
50     inline Complex() = default;
51     inline Complex(double _re,double _im):re(_re),im(_im) {}
52     friend inline Complex operator*(const Complex &a,const Complex &b) { return
↪ Complex(a.re*b.re-a.im*b.im,a.re*b.im+a.im*b.re); }
53     friend inline Complex operator+(const Complex &a,const Complex &b) { return
↪ Complex(a.re+b.re,a.im+b.im); }
54     friend inline Complex operator-(const Complex &a,const Complex &b) { return
↪ Complex(a.re-b.re,a.im-b.im); }
55     friend inline Complex operator/(const Complex &a,double r) { return Complex(a.re/r,a.im/r);
↪ }
56 };
57
58 inline void FFT(Complex *a,int loglen,int len,int on)
59 {
60     for (register int i = 0,j,t,p;i < len;++i)
61     {
62         for (p = j = 0,t = i;j < loglen;++j,t >= 1)
63             p <= 1,p |= (t&1);
64         if (p > i) swap(a[p],a[i]);
65     }
66     for (register int m = 2,s = 1;s <= loglen;++s,m <= 1)
67     {
68         register Complex w(cos(2*pi*on/m),sin(2*pi*on/m));
69         for (int i = 0;i < len;i += m)
70         {
71             register Complex wn(1,0);
72             for (register int j = 0;j < (m>>1);++j,wn = wn*w)
73             {
74                 register Complex u = a[i+j],v = wn*a[i+j+(m>>1)];
75                 a[i+j] = u+v; a[i+j+(m>>1)] = u-v;
76             }
77         }
78     }
79     if (on == -1) for (int i = 0;i < len;++i) a[i] = a[i]/len;
80 }
81
82 struct Polynomial
83 {
84     int len; Complex array[maxn<<2];
85     inline Polynomial(int _len = 0):len(_len) {}
86     inline Polynomial(Complex a[],int n):len(n) { for (int i = 0;i < n;++i) array[i] = a[i]; }
87     inline Complex operator [] (int n) const { return array[n]; }
88     inline Complex &operator [] (int n) { return array[n]; }
89     inline void set(int n) { len = n; }
90     inline void set(int n,Complex a[]) { len = n; for (int i = 0;i < n;++i) array[i] = a[i]; }
91     inline void extend(int key)
92     {

```

```

93         for (int i = len; i < (1<<key); ++i)
94             array[i] = Complex(0,0);
95     }
96     inline void cut(int key) { len = key; }
97     inline void transform(int loglen, int on) { FFT(array, loglen, 1<<loglen, on); }
98 }; //变量只能定义在全局, 不然会 re
99
100 inline Polynomial multiply(Polynomial &pa, Polynomial &ret) // self-multiply
101 {
102     int loglen = 0;
103     while ((1<<loglen) < (pa.len<<1)-1) ++loglen;
104     pa.extend(1<<loglen); pa.transform(loglen, 1);
105     for (int i = 0; i < (1<<loglen); ++i) ret[i] = pa[i]*pa[i];
106     ret.transform(loglen, -1); ret.cut((pa.len<<1)-1);
107     return ret;
108 }
109 inline Polynomial multiply(Polynomial &pa, Polynomial &pb, Polynomial &ret)
110 {
111     int loglen = 0;
112     while ((1<<loglen) < (pa.len+pb.len-1)) ++loglen;
113     pa.extend(1<<loglen); pa.transform(loglen, 1);
114     pb.extend(1<<loglen); pb.transform(loglen, 1);
115     for (int i = 0; i < (1<<loglen); ++i) ret[i] = pa[i]*pb[i];
116     ret.transform(loglen, -1); ret.cut(pa.len+pb.len-1);
117     return ret;
118 }

```

6.4 Fast Input and Output

```

1 // Input and Output of Int
2 // Be careful of Max_Int and Min_Int
3 inline int gi()
4 {
5     char ch; int ret = 0, f = 1;
6     do ch = getchar(); while (!(ch >= '0' && ch <= '9') && ch != '-');
7     if (ch == '-') f = -1, ch = getchar();
8     do ret = ret*10+ch-'0', ch = getchar(); while (ch >= '0' && ch <= '9');
9     return ret*f;
10 }
11
12 inline void pi(int a)
13 {
14     if (!a) putchar('0');
15     if (a < 0) a = -a, putchar('-');
16     int num[10], n = 0;
17     while (a) num[n++] = a%10, a /= 10;
18     for (int i = n-1; i >= 0; --i) putchar('0'+num[i]);
19 }

```

6.5 Fraction Class

```

1  typedef long long ll;
2  struct Fraction
3  {
4      ll num,den;
5      inline Fraction(ll a = 0,ll b = 1)
6      {
7          if (den < 0) a = -a,b = -b;
8          assert(b != 0); ll g = gcd(abs(a),b);
9          num = a/g; den = b/g;
10     }
11     friend inline Fraction operator +(const Fraction &a,const Fraction &b) const { return
↪ Fraction(a.num*b.den+b.num*a.den,a.den*b.den); }
12     friend inline Fraction operator -(const Fraction &a,const Fraction &b) const { return
↪ Fraction(a.num*b.den-b.num*a.den,a.den*b.den); }
13     friend inline Fraction operator *(const Fraction &a,const Fraction &b) const { return
↪ Fraction(a.num*b.num,a*den*b.den); }
14     friend inline Fraction operator /(const Fraction &a,const Fraction &b) const { return
↪ Fraction(a.num*b.den,a*den*b.num); }
15     friend inline bool operator <(const Fraction &a,const Fraction &b) const { return
↪ a.num*b.den < a.den*b.num; }
16     friend inline bool operator <=(const Fraction &a,const Fraction &b) const { return
↪ a.num==b.num&& a.den==b.den; }
17 };

```

6.6 Gray Code

```

1  //0-2n-1 的格雷码
2  inline vector <int> GrayCreat(int n)
3  {
4      vector <int> res;
5      for (int i = 0;i < (1<<n);++i) res.push_back(i^(i>>1));
6      return res;
7  }

```

6.7 Numerical Integration

```

1  //self-adapt simpson
2  inline long double simpson(long double l,long double r,long double mid,long double Cl,long
↪ double Cr,long double Cm)
3  {
4      long double tCl = calc((l+mid)/2),tCr = calc((mid+r)/2);
5      long double
↪ ans=(r-l)*(Cl+Cr+4*Cm)/6,lans=(mid-l)*(Cl+Cm+4*tCl)/6,rans=(r-mid)*(Cr+Cm+4*tCr)/6;
6      if (r-l <= 1e-3&&fabs(lans+rans-ans)<eps) return ans;
7      // if (dep > lim&&fabs(lans+rans-ans)<eps) return ans;
8      else return simpson(l,mid,(l+mid)/2,Cl,Cm,tCl)+simpson(mid,r,(mid+r)/2,Cm,Cr,tCr);
9  }

```



```

10
11 //romberg---To Be Verified
12 template <class T>
13 inline double romberg(const T &f, double a, double b, double eps = 1e-8)
14 {
15     vector <double> t; double h = b-a, last, cur;
16     int k = 1, i = 1;
17     t.push_back(h*(f(a)+f(b))/2);
18     do
19     {
20         last = t.back(); cur = 0; double x = a+h/2;
21         for (int j = 0; j < k; ++j) cur += f(x), x += h;
22         cur = (t[0]+h*cur)/2;
23         double k1 = 4.0/3, k2 = 1.0/3;
24         for (int j = 0; j < i; ++j)
25         {
26             double temp = k1*cur-k2*t[j];
27             t[j] = cur; cur = temp; k2 /= 4*k1-k2; k1 = k2+1;
28         }
29         t.push_back(cur); k *= 2; h /= 2; ++i;
30     }
31     while (fabs(last - cur) > eps);
32     return t.back();
33 }

```

6.8 Simplex

6.8.1 Description

有 n 个实数变量 x_1, x_2, \dots, x_n 和 m 条约束，其中第 i 条约束形如 $\sum_{j=1}^n a_{i,j}x_j \leq b_i$ 。

此外这 n 个变量需要满足非负性限制， $x_j \geq 0$ 。

在满足上述所有条件的情况下，你需要指定每个变量 x_j 的取值，使得目标函数 $F = \sum_{j=1}^n c_j x_j$ 的值最大。

6.8.2 Input

第一行三个正整数 n, m, t 。其中 $t \in \{0, 1\}$ 。

第二行有 n 个整数 c_1, c_2, \dots, c_n ，整数间均用一个空格分隔。

接下来 m 行，每行代表一条约束，其中第 i 行有 $n+1$ 个整数 $a_{i1}, a_{i2}, \dots, a_{in}, b_i$ ，整数间均用一个空格分隔。

6.8.3 Output

如果不存在满足所有约束的解，仅输出一行 “Infeasible”。

如果对于任意的 M ，都存在一组解使得目标函数的值大于 M ，仅输出一行 “Unbounded”。

否则，第一行输出一个实数，表示目标函数的最大值 F 。

如果 $t = 1$, 那么你还需要输出第二行, 用空格隔开的 n 个非负实数, 表示此时 x_1, x_2, \dots, x_n 的取值, 如有多组方案请任意输出其中一个。

6.8.4 Code

```

1 // uoj 179
2 #include<iostream>
3 #include<cstdio>
4 #include<cstdlib>
5 using namespace std;
6
7 #define maxn (30)
8 #define eps (1e-8)
9
10 int N,M,op,tot,q[maxn],idx[maxn],idy[maxn]; double a[maxn][maxn],A[maxn];
11
12 inline void pivot(int x,int y)
13 {
14     swap(idy[x],idx[y]);
15     double tmp = a[x][y]; a[x][y] = 1/a[x][y];
16     for (int i = 0;i <= N;++i) if (y != i) a[x][i] /= tmp;
17     tot = 0; for (int i = 0;i <= N;++i) if (i != y && (a[x][i] > eps || a[x][i] < -eps)) q[++tot] =
    ↪ i;
18     for (int i = 0;i <= M;++i)
19     {
20         if ((x == i) || (a[i][y] < eps && a[i][y] > -eps)) continue;
21         for (int j = 1;j <= tot;++j) a[i][q[j]] -= a[x][q[j]]*a[i][y];
22         a[i][y] = -a[i][y]/tmp;
23     }
24 }
25
26 int main()
27 {
28     freopen("179.in","r",stdin);
29     freopen("179.out","w",stdout);
30     scanf("%d %d %d",&N,&M,&op); srand(233);
31     for (int i = 1;i <= N;++i) scanf("%lf",a[0]+i);
32     for (int i = 1;i <= M;++i)
33     {
34         for (int j = 1;j <= N;++j) scanf("%lf",a[i]+j);
35         scanf("%lf",a[i]);
36     }
37     for (int i = 1;i <= N;++i) idx[i] = i;
38     for (int i = 1;i <= M;++i) idy[i] = i+N;
39     while (true)
40     {
41         int x = 0,y = 0;
42         for (int i = 1;i <= M;++i) if (a[i][0] < -eps && ((!x) || (rand()&1))) x = i; if (!x) break;
43         for (int i = 1;i <= N;++i) if (a[x][i] < -eps && ((!y) || (rand()&1))) y = i; if (!y) return
    ↪ puts("Infeasible"),0;

```

```

44     pivot(x,y);
45 }
46 while (true)
47 {
48     int x = 0,y = 0; double mn = 1e15;
49     for (int i = 1;i <= N;++i) if (a[0][i] > eps) { y = i; break; } if (!y) break;
50     for (int i = 1;i <= M;++i) if (a[i][y] > eps && a[i][0]/a[i][y] < mn) mn =
↪ a[i][0]/a[i][y],x = i; if (!x) return puts("Unbounded"),0;
51     pivot(x,y);
52 }
53 printf("%.8lf\n",-a[0][0]); if (!op) return 0;
54 for (int i = 1;i <= M;++i) if (idy[i] <= N) A[idy[i]] = a[i][0];
55 for (int i = 1;i <= N;++i) printf("%.8lf ",A[i]);
56 fclose(stdin); fclose(stdout);
57 return 0;
58 }

```

6.9 Solutions of Equation of Higher Order

```

1  // vector <double> solve(vector <double> coef,int n)
2  // coef 方程的系数; n 方程的系数
3  // 输出所有实数解
4  const double EPS = 1e-15,inf = 1e12;
5
6  inline int sign(double x) { return x < -EPS?-1:x > EPS; }
7
8  inline double get(const vector <double> &coef,double x)
9  {
10     double e = 1,s = 0;
11     for (int i = 0;i < coef.size();++i) s += coef[i]*e,e *= x;
12     return s;
13 }
14
15 inline double find(const vector <double> &coef,int n,double lo,double hi)
16 {
17     double sign_lo,sign_hi;
18     if ((sign_lo = sign(get(coef,lo)))== 0) return lo;
19     if ((sign_hi = sign(get(coef,hi)))== 0) return hi;
20     if (sign_lo*sign_hi > 0) return inf;
21     for (int step = 0;step < 100&&hi-lo > EPS;++step)
22     {
23         double m = (lo+hi)/2; int sign_mid = sign(get(coef,m));
24         if (sign_mid == 0) return m;
25         else if (sign_lo*sign_mid < 0) hi = m;
26         else lo = m;
27     }
28     return (lo+hi)/2;
29 }
30

```

```
31 inline vector <double> solve(const vector <double> &coef, int n)
32 {
33     vector <double> ret;
34     if (n == 1)
35     {
36         if (sign(coef[1])) ret.push_back(-coef[0]/coef[1]);
37         return ret;
38     }
39     vector <double> dcoef(n);
40     for (int i = 0; i < n; ++i) dcoef[i] = coef[i+1]*(i+1);
41     vector <double> droot = solve(dcoef, n-1);
42     droot.insert(droot.begin(), -inf);
43     droot.push_back(inf);
44     for (int i = 0; i+1 < droot.size(); ++i)
45     {
46         double tmp = find(coef, n, droot[i], droot[i+1]);
47         if (tmp < inf) ret.push_back(tmp);
48     }
49     return ret;
50 }
```

Chapter 7

String Algorithms

7.1 Aho-Corasick Automaton

```
1  // ac 自动机
2  inline int newnode()
3  {
4      memset(nxt[L],-1,sizeof(nxt[L]));
5      return ++L-1;
6  }
7  inline void init() { L = 0; root = newnode(); }
8  inline void insert()
9  {
10     int len = strlen(buf),now = root;
11     for (int i = 0;i < len;++i)
12     {
13         if (nxt[now][buf[i]-'0'] == -1)
14             nxt[now][buf[i]-'0'] = newnode();
15         now = nxt[now][buf[i]-'0'];
16     }
17     end[now] = true;
18 }
19 inline void build()
20 {
21     int now = root; queue <int> team;
22     fail[root] = root;
23     for (int i = 0;i < 10;++i)
24     {
25         if (nxt[now][i] == -1) nxt[now][i] = root;
26         else fail[nxt[now][i]] = root,team.push(nxt[now][i]);
27     }
28     while (!team.empty())
29     {
30         now = team.front(); team.pop();
31         for (int i = 0;i < 10;++i)
32         {
33             if (nxt[now][i] == -1)
```

```

34         nxt[now][i] = nxt[fail[now]][i];
35     else
36     {
37         fail[nxt[now][i]] = nxt[fail[now]][i];
38         team.push(nxt[now][i]);
39     }
40 }
41 }
42 }

```

7.2 Knuth-Morris-Pratt Algorithm

```

1  // To Be Verified
2  void cal_next(char *str, int *next, int len)
3  {
4      int i,j;
5      next[0] = -1;
6      for (int i = 1; i < len; i++)
7      {
8          j = next[i - 1];
9          while (str[j+1] != str[i]&&(j >= 0)) j = next[j];
10         if (str[i] == str[j+1]) next[i] = j + 1;
11         else next[i] = -1;
12     }
13 }
14
15 int KMP(char *str,int slen, char *ptr,int plen,int *next)
16 {
17     int s_i = 0,p_i = 0;
18     while (s_i < slen&&p_i < plen)
19     {
20         if (str[s_i] == ptr[p_i]) s_i++,p_i++;
21         else
22         {
23             if (!p_i) s_i++;
24             else p_i = next[p_i-1] + 1;
25         }
26     }
27     return (p_i == plen)?(s_i - plen):-1;
28 }

```

7.3 Manacher Algorithm

```

1  // Correct but to Be Rewritten
2  inline void ready()
3  {
4      for (int i = 1;i <= 2*l1+1;++i)
5          { if (i & 1) bac[i] = '#'; else bac[i] = s[i>>1]; }
6      bin[0] = 1;

```

```

7     for (int i = 1; i <= l1; ++i)
8         hash[i] = hash[i-1]*37+s[i]-'A'+1, bin[i] = 37*bin[i-1];
9 }
10
11 inline void manacher()
12 {
13     rad[1] = 1; int best = 1;
14     for (int i = 2; i <= 2*l1+1; ++i)
15     {
16         int j;
17         if (best+rad[best]-1 < i) j = 1;
18         else j = min(rad[2*best-i], best+rad[best]-i)+1;
19         while (i-j+1 && i+j-1 <= 2*l1+1 && bac[i-j+1] == bac[i+j-1])
20         {
21             if (bac[i+j-1] != '#')
22             {
23                 ull h = (hash[(i+j-1)>>1] - hash[((i-j+1)>>1)-1]*bin[j]);
24                 if (!exist1[h%rh11] || !exist2[h%rh12] || !exist3[h%rh13])
25                 {
26                     exist1[h%rh11] = exist2[h%rh12] = exist3[h%rh13] = true;
27                     ++tot, have[tot][0] = (i-j+1)>>1;
28                     have[tot][1] = (i+j-1)>>1;
29                 }
30             }
31             ++j;
32         }
33         rad[i] = j-1;
34         if (i+rad[i] > best+rad[best]) best = i;
35     }
36 }

```

7.4 Palindrome Automaton

```

1  // Correct but to Be Rewritten
2  struct pat
3  {
4      int next[maxn][26], fail[maxn], cnt[maxn], len[maxn], s[maxn], last, n, p;
5      inline int newnode(int l) { cnt[p] = 0; len[p] = 1; return p++; }
6      inline void init() { last = n = p = 0; newnode(0); newnode(-1); s[0] = -1; fail[0] = 1; }
7      inline int getfail(int x) { while (s[n-len[x]-1] != s[n]) x = fail[x]; return x; }
8      inline void add(int c)
9      {
10         c -= 'a'; s[++n] = c; int cur = getfail(last);
11         if (!next[cur][c])
12         {
13             int now = newnode(len[cur]+2);
14             fail[now] = next[getfail(fail[cur])][c];
15             next[cur][c] = now;
16         }

```

```

17     last = next[cur][c]; cnt[last]++;
18 }
19 }

```

7.5 Suffix Array

```

1  // 记得最后填一个字符集中没有的字符
2  inline void build(char *buf, int *Sa, int *Rank, int *Height, int n, int now, int m)
3  {
4      int i, j, k, *x = t1, *y = t2;
5      memset(c, 0, 4*m);
6      for (i = 0; i < n; ++i) c[x[i] = buf[i] - 'A']++;
7      for (i = 1; i < m; ++i) c[i] += c[i-1];
8      for (i = n-1; i >= 0; --i) Sa[--c[x[i]]] = i;
9      for (k = 1; k < n; k <= 1)
10     {
11         int p = 0;
12         for (i = n-k; i < n; ++i) y[p++] = i;
13         for (i = 0; i < n; ++i) if (Sa[i] >= k) y[p++] = Sa[i] - k;
14         memset(c, 0, 4*m);
15         for (i = 0; i < n; ++i) c[x[y[i]]]++;
16         for (i = 1; i < m; ++i) c[i] += c[i-1];
17         for (i = n-1; i >= 0; --i) Sa[--c[x[y[i]]]] = y[i];
18         swap(x, y); p = 1; x[Sa[0]] = 0;
19         for (i = 1; i < n; ++i)
20             x[Sa[i]] = y[Sa[i-1]] == y[Sa[i]] && y[Sa[i-1]+k] == y[Sa[i]+k] ? p-1 : p++;
21         if (p >= n) break; m = p;
22     }
23     for (i = 0; i < n; ++i) Rank[Sa[i]] = i;
24     for (i = k = 0; i < n; ++i)
25     {
26         if (k) --k; if (!Rank[i]) continue;
27         j = Sa[Rank[i]-1];
28         while (i+k < n && j+k < n && buf[i+k] == buf[j+k]) ++k;
29         Height[Rank[i]] = k;
30     }
31 }

```

7.6 Suffix Automaton

```

1  // Correct but to Be Rewritten
2  struct SAM
3  {
4      int tot, tail, cnt, p, np, q, nq, sz[maxn], arr[maxn], step[maxn], tran[maxn][26], parent[maxn];
5      inline SAM() { tail = tot = 1; }
6      inline void insert(int c)
7      {
8          p = tail; np = tail = ++tot; step[np] = step[p]+1;
9          for (; !tran[p][c]; p = parent[p]) tran[p][c] = np;

```



```

10     if (!p) parent[np] = 1;
11     else
12     {
13         q = tran[p][c];
14         if (step[p]+1 == step[q]) parent[np] = q;
15         else
16         {
17             nq = ++tot; step[nq] = step[p]+1;
18             memcpy(tran[nq],tran[q],104);
19             parent[nq] = parent[q]; parent[np] = parent[q] = nq;
20             for (;tran[p][c] == q;p = parent[p]) tran[p][c] = nq;
21         }
22     }
23     sz[np] = 1;
24 }
25
26 inline void dfs(int now)
27 {
28     if (vis[now]) return; vis[now] = true;
29     for (int i = 0;i < 26;++i)
30         if (tran[now][i]) dfs(tran[now][i]),arr[now] += arr[tran[now][i]];
31     arr[now] += sz[now];
32 }
33
34 inline void build()
35 {
36     if (!mode) for (int i = 1;i <= tot;++i) sz[i] = 1;
37     else
38     {
39         for (int i = 2;i <= tot;++i) ++d[parent[i]];
40         queue <int> team; for (int i = 1;i <= tot;++i) if (!d[i]) team.push(i);
41         while (!team.empty())
42         {
43             int now = team.front(); team.pop();
44             sz[parent[now]] += sz[now];
45             if (!--d[parent[now]]) team.push(parent[now]);
46         }
47     }
48     sz[1] = 0; dfs(1);
49 }
50 inline void work()
51 {
52     int now = 1,l = 0,rank = 0; memset(s,0,N+1);
53     if (K > arr[1]) puts("-1");
54     else
55     {
56         while (true)
57         {
58             rank += sz[now]; if (rank >= K) break;

```

```
59         for (int i = 0; i < 26; ++i)
60         {
61             if (rank+arr[tran[now][i]] < K) rank += arr[tran[now][i]];
62             else { s[++l] = 'a'+i; now = tran[now][i]; break; }
63         }
64     }
65     printf("%s", s+1);
66 }
67 }
68 }sam;
```

Chapter 8

Others

8.1 Calculation of Date

```
1 //ya 年 ma 月 da 日与 yb 年 mb 月 db 日相差几天
2 const int days = 365, s[] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
3 inline bool isleap(int y)
4 {
5     if ((!(y%400)||y%100)&&!(y%4)) return true;
6     return false;
7 }
8
9 inline int leap(int y)
10 {
11     if (!y) return 0;
12     return y/4-y/100+y/400;
13 }
14
15 inline int calc(int day,int mon,int year)
16 {
17     int res = (year-1)*days+leap(year-1);
18     for (int i = 1;i < mon;++i) res += s[i];
19     if (isleap(year)&&mon > 2) res++;
20     res += day; return res;
21 }
22
23 inline int count_day(int da,int ma,int ya,int db,int mb,int yb)
24 {
25     int resa = calc(da,ma,ya);
26     int resb = calc(db,mb,yb);
27     return abs(resa-resb);
28 }
```

8.2 Java Hints

```

1  // Code 1
2  import java.io.*;
3  import java.math.*;
4  import java.util.*;
5
6  public class Main
7  {
8      final static int lhh = 998244353,maxn = 1655;
9      static long jc[] = new long [maxn];
10
11     static int calc(BigInteger N)
12     {
13         if (N.compareTo(BigInteger.ONE) <= 0) return 0;
14         // System.out.println(N);
15         int l = 2,r = 1650,mid;
16         while (l <= r)
17         {
18             mid = (l+r)>>1;
19             if (((BigInteger.valueOf(mid)).pow(mid)).compareTo(N) <= 0) l = mid+1;
20             else r = mid-1;
21         }
22         // System.out.println(l+" "+r);
23         int ret = (int)jc[r]-1,d = 1; if (ret < 0) ret += lhh;
24         int digit[] = new int[d]; BigInteger _d = BigInteger.valueOf(l);
25         for (int i = 0;i < d;++i)
26         {
27             digit[i] = N.mod(_d).intValue();
28             N = N.divide(_d);
29         }
30         // for (int i = d-1;i >= 0;--i) System.out.print(digit[i]);
31         // System.out.println();
32         boolean cho[] = new boolean[d],safe = false,exist = true;
33         Arrays.fill(cho,false);
34         int per[] = new int [d];
35         for (int i = d-1;i >= 0;--i)
36         {
37             int cur = -1;
38             if (safe)
39             {
40                 int down = i == d-1?1:0;
41                 for (int j = d-1;j >= down;--j)
42                 {
43                     if (cho[j] == true) continue;
44                     cur = j; break;
45                 }
46                 if (cur == -1) { exist = false; break; }
47                 cho[cur] = true; per[i] = cur;
48             }

```

```

49         else
50         {
51             if (cho[digit[i]] == true)
52             {
53                 // System.out.println(i+": "+digit[i]);
54                 while (i < d)
55                 {
56                     cur = -1;
57                     int down = i == d-1?1:0;
58                     for (int j = digit[i]-1; j >= down; --j)
59                     {
60                         if (cho[j] == true) continue;
61                         cur = j; break;
62                     }
63                     // System.out.println(cur+": "+i);
64                     if (cur == -1) { ++i; if (i < d) cho[per[i]] = false; }
65                     else { cho[cur] = true; per[i] = cur; break; }
66                 }
67
68                 if (cur == -1) { exist = false; break; }
69                 safe = true;
70             }
71             else
72             {
73                 if (digit[i] == 0 && i == d-1) { exist = false; break; }
74                 per[i] = digit[i];
75                 cho[digit[i]] = true;
76             }
77         }
78     }
79     // for (int i = d-1; i >= 0; --i) System.out.print(per[i]);
80     // System.out.println();
81     if (!exist) return ret;
82     for (int i = d-1; i >= 0; --i)
83     {
84         int tmp = per[i];
85         for (int j = d-1; j > i; --j)
86             if (per[j] < per[i]) --tmp;
87         ret += jc[i]*tmp%lhh;
88         if (ret >= lhh) ret -= lhh;
89     }
90     ret++; if (ret >= lhh) ret -= lhh;
91     ret -= jc[d-1]; if (ret < 0) ret += lhh;
92     // System.out.println(ret);
93     return ret;
94 }
95
96 public static void main(String args[])
97 {

```

```

98     jc[0] = 1;
99     for (int i = 1; i <= 1650; ++i)
100         jc[i] = jc[i-1]*(long)i%lhh;
101     Scanner cin = new Scanner(System.in);
102     int T = cin.nextInt();
103
104     while (T-- > 0)
105     {
106         BigInteger l = cin.nextBigInteger(), r = cin.nextBigInteger();
107         int ans = calc(r)-calc(l.subtract(BigInteger.ONE));
108         if (ans < 0) ans += lhh; System.out.println(ans);
109     }
110     // calc(BigInteger.valueOf(123455));
111 }
112 }
113
114 //Code 2
115 import java.io.*;
116 import java.util.*;
117 import java.math.*;
118 public class Main
119 {
120     static BigDecimal ratio[] = new BigDecimal[110];
121     public static void main(String[] args)
122     {
123         Scanner cin = new Scanner(System.in);
124         int T = cin.nextInt();
125         for (int Case = 1; Case <= T; ++Case)
126         {
127             int N = cin.nextInt();
128             for (int i = 1; i <= N; ++i)
129             {
130                 String S = cin.next();
131                 String[] str = S.split(":");
132                 BigDecimal a = new BigDecimal(str[0]), b = new BigDecimal(str[1]);
133                 ratio[i] = a.divide(a.add(b), 30, BigDecimal.ROUND_HALF_EVEN);
134             }
135             Arrays.sort(ratio, 1, N+1);
136             BigDecimal res = new BigDecimal(0), _1 = new BigDecimal(1); int ans = 0;
137             for (int i = 1; i <= N; ++i)
138             {
139                 res = res.add(ratio[i]);
140                 if (res.compareTo(_1) < 0) ans = i;
141                 else break;
142             }
143             System.out.println("Case #" + Case + ": " + ans);
144         }
145     }
146 }

```

```

147
148 // Code 3
149 import java.math.*;
150 import java.util.*;
151 public class Main
152 {
153     static BigInteger d,ret,temp,yy;
154     static int n,dd;
155     static boolean mark = true;
156     static BigInteger[] a = new BigInteger[20];
157     public static void main(String[] args)
158     {
159         Scanner in = new Scanner (System.in);
160         n = in.nextInt();
161         temp = BigInteger.ONE;
162         ret = BigInteger.ZERO;
163         for (int i = 0;i < n;++i)
164         {
165             int k = in.nextInt();
166             a[i] = BigInteger.valueOf(k);
167             d = temp.gcd(a[i]);
168             temp = temp.multiply(a[i]).divide(d);
169         }
170         for (int i = 1;i < (1<n);++i)
171         {
172             mark = false; yy = BigInteger.ONE;
173             for (int j = 0;j < n;++j) if (((1 << j) & i) > 0) { mark = !mark; d = a[j].gcd(yy);
↪ yy = yy.multiply(a[j]).divide(d); }
174             if (mark) ret = ret.add(temp.divide(yy));
175             else ret = ret.subtract(temp.divide(yy));
176         }
177         d = ret.gcd(temp);
178         System.out.println(ret.divide(d));
179         System.out.println(temp.divide(d));
180     }
181 }
182
183 // Code 4
184 import java.io.*;
185 import java.math.*;
186 import java.util.*;
187
188 public class Main
189 {
190     public static String reverse(String str) { return new
↪ StringBuffer(str).reverse().toString(); }
191
192     public static void main(String args[])
193     {

```

```

194 Scanner cin = new Scanner(System.in);
195 int T = cin.nextInt(); BigInteger zero = BigInteger.valueOf(0);
196 while (T-- > 0)
197 {
198     int base1 = cin.nextInt(), base2 = cin.nextInt();
199     String S = cin.next(); int len = S.length();
200     System.out.println(base1+" "+S);
201     BigInteger res = BigInteger.valueOf(0), b1 = BigInteger.valueOf(base1), b2 =
↪ BigInteger.valueOf(base2);
202     for (int i = 0; i < len; ++i)
203     {
204         res = res.multiply(b1);
205         int rep = 0;
206         if (S.charAt(i) >= '0' && S.charAt(i) <= '9') rep = S.charAt(i) - '0';
207         else if (S.charAt(i) >= 'A' && S.charAt(i) <= 'Z') rep = 10 + S.charAt(i) - 'A';
208         else rep = 36 + S.charAt(i) - 'a';
209         res = res.add(BigInteger.valueOf(rep));
210     }
211     String ret = new String();
212     // System.out.println(res);
213     if (res.compareTo(zero) == 0) ret += '0';
214     else
215         while (res.compareTo(zero) > 0)
216         {
217             long val = res.remainder(b2).longValue();
218             // System.out.println(val);
219             if (val < 10) ret += (char)(val + '0');
220             else if (val < 36) ret += (char)(val + 'A' - 10);
221             else ret += (char)(val + 'a' - 36);
222             res = res.divide(b2);
223         }
224     System.out.println(base2+" "+reverse(ret)+"\n");
225 }
226 }
227 }

```

8.3 Emacs Configuration-Full

```

1  ;;-----General-----
2  (global-set-key (kbd "M-n") 'forward-paragraph)
3  (global-set-key (kbd "M-p") 'backward-paragraph)
4  (global-set-key (kbd "<C-f12>") 'print-buffer)
5  (global-set-key (kbd "C-`") 'set-mark-command)
6  (global-set-key (kbd "<f5>") 'eshell)
7  ;; 修改透明度
8  (set-frame-parameter (selected-frame) 'alpha (list 78 78))
9  (add-to-list 'default-frame-alist (cons 'alpha (list 78 78)))
10 ;; set the cursor
11 (setq-default cursor-type 'bar)

```



```

12 ;; comment-dwim
13 (defun strong-comment-dwim-line (&optional arg)
14
15   ↪ "Replacement for the comment-dwim command. If no region is selected and current line is not blank and we a
16   (interactive "*P")
17   (comment-normalize-vars)
18   (if (and (not (region-active-p)) (not (looking-at "[\t]*$")))
19       (comment-or-uncomment-region (line-beginning-position) (line-end-position))
20       (comment-dwim arg)))
21 (global-set-key (kbd "M-;") 'strong-comment-dwim-line)
22 ;; Remove tool bar
23 (tool-bar-mode -1)
24 ;; 不要滚动栏，现在都用滚轴鼠标了，可以不用滚动栏了
25 (scroll-bar-mode -1)
26 ;; 设置行号
27 (global-linum-mode t)
28 ;; Change the Title
29 (setq frame-title-format "%b@lmxyy'-emacs")
30 ;; 滚动页面时比较舒服，不要整页的滚动
31 (setq scroll-step 1
32       scroll-margin 3
33       scroll-conservatively 10000)
34 ;; 设置界面 start
35 (if window-system
36     (setq default-frame-alist
37           (append
38             '( (top . 0)
39               (left . 0)
40               (width . 300)
41               (height . 300))
42             default-frame-alist)))
43 )
44 ;; 允许 emacs 和外部其他程序的粘贴
45 (setq x-select-enable-clipboard t)
46 ;; 去掉菜单栏，我将 F10 绑定为显示菜单栏，万一什么东西忘了，需要菜单栏了可以摁 F10 调出，再摁 F10 就去掉菜单
47 (menu-bar-mode -1)
48 ;; 显示列号
49 (setq column-number-mode t)
50 ;; 开启语法高亮。
51 (global-font-lock-mode 1)
52 ;; Auto Indent
53 (global-set-key (kbd "RET") 'newline-and-indent)
54
55 ;; -----Tex Mode-----
56 ;; Emacs 加载 Auctex
57 (load "auctex.el" nil t t)
58 (load "preview-latex.el" nil t t)
59 (setq Tex-auto-save t)

```

```

60 (setq Tex-parse-self t)
61 (setq-default Tex-master nil)
62 ;; 设置编译信息
63 (defun compile-xelatex ()
64   (interactive)
65   (compile (format "xelatex -shell-escape %s" (buffer-name))))
66 (global-set-key (kbd "<f6>") 'compile-xelatex)
67 (defun compile-latex ()
68   (interactive)
69   (compile (format "latex -shell-escape %s" (buffer-name))))
70 (global-set-key (kbd "C-<f6>") 'compile-latex)
71
72 ;; -----C++ Mode-----
73 ;; 设置编译信息
74 (defun compile-cpp ()
75   (interactive)
76   (compile (format "g++ -o %s %s -g -lm -Wall -std=c++11" (file-name-sans-extension
77     ↪ (buffer-name))(buffer-name))))
78 (global-set-key (kbd "<f9>") 'compile-cpp)
79 (defun compile-cpp-02 ()
80   (interactive)
81   (compile (format "g++ -o %s %s -g -lm -Wall -std=c++11 -02" (file-name-sans-extension
82     ↪ (buffer-name))(buffer-name))))
83 (global-set-key (kbd "C-<f9>") 'compile-cpp-02)
84 ;; 设置一键调试
85 (global-set-key (kbd "<f8>") 'gud-gdb)
86 ;; 设置 tab 为 4 个空格的宽度
87 (setq c-basic-offset 4)
88 (setq default-tab-width 4)
89
90 ;; -----Java Mode-----
91 ;; 设置编译信息
92 (defun compile-java ()
93   (interactive)
94   (compile (format "javac %s" (buffer-name))))
95 (global-set-key (kbd "<f7>") 'compile-java)
96
97 ;; -----Org Mode-----
98 (setq org-startup-indented t)
99 ;; The following lines are always needed. Choose your own keys.
100 (add-to-list 'auto-mode-alist '("\\.org\\'" . org-mode))
101 (add-hook 'org-mode-hook 'turn-on-font-lock) ; not needed when global-font-lock-mode is on
102 (global-set-key "\C-cl" 'org-store-link)
103 (global-set-key "\C-ca" 'org-agenda)
104 (global-set-key "\C-cb" 'org-iswitchb)
105 ;; Auto-line-breaking
106 ;; 这一句定义一个函数
107 (defun my-org-mode() (setq truncate-lines nil))
108 ;; 把这个函数加入到 `org-mode-hook' 中，这样每次打开 org 文件时都会先执行这个函数

```

```

107 (add-hook 'org-mode-hook 'my-org-mode)
108
109 ;; -----Ido Mode-----
110 ;; 启用 ido 模式
111 (ido-mode t)
112
113 ;; -----Verilog Mode-----
114 ;; 设置编译信息
115 (defun compile-verilog ()
116   (interactive)
117   (compile (format "iverilog -o %s %s" (file-name-sans-extension (buffer-name))(buffer-name))))
118 (global-set-key (kbd "<f12>") 'compile-verilog)
119
120
121 ;; -----Custom Sets-----
122 (custom-set-variables
123   ;; custom-set-variables was added by Custom.
124   ;; If you edit it by hand, you could mess it up, so be careful.
125   ;; Your init file should contain only one such instance.
126   ;; If there is more than one, they won't work right.
127   '(column-number-mode t)
128   '(cua-mode t nil (cua-base))
129   '(inhibit-startup-screen t)
130   '(menu-bar-mode nil)
131   '(tool-bar-mode nil))
132 (custom-set-faces
133   ;; custom-set-faces was added by Custom.
134   ;; If you edit it by hand, you could mess it up, so be careful.
135   ;; Your init file should contain only one such instance.
136   ;; If there is more than one, they won't work right.
137   '(default ((t (:family "Courier 10 Pitch" :foundry "bitstream" :slant normal :weight bold
138   ↪ :height 151 :width normal)))))

```

8.4 Emacs Configuration-Competition

```

1  ;; Default Font: Courier 10 Pitch Bold   Size: 15
2  ;; Remember to set CUA-mode and save your options.
3  (global-set-key (kbd "M-n") 'forward-paragraph)
4  (global-set-key (kbd "M-p") 'backward-paragraph)
5  (global-linum-mode t)
6  (defun compile-cpp ()
7    (interactive)
8    (compile (format "g++ -o %s %s -g -lm -Wall -std=c++11" (file-name-sans-extension
9    ↪ (buffer-name))(buffer-name))))
9  (global-set-key (kbd "<f9>") 'compile-cpp)
10 (global-set-key (kbd "<f8>") 'gud-gdb)
11 (setq default-tab-width 4)
12 (setq c-basic-offset 4)
13 (global-set-key (kbd "RET") 'newline-and-indent)

```