

NCTU_TaNoShiI

Contents

1 String	1	9.3 SCC	17
1.1 AC	1	9.4 GeneralGraphMaximumValueMatch	17
1.2 SuffixAutomata	2	9.5 Stable Marriage	19
1.3 MinLexicographicalRotate	2	9.6 BCCvertex	19
1.4 ZvaluePalindromes	3	9.7 MaxClique	20
1.5 SuffixArray	3	9.8 BCCedge	20
1.6 Zvalue	3		
2 Math	3	10 無權邊的生成樹個數 Kirchhoff's Theorem	21
2.1 MillerRabin	3	11 monge	21
2.2 FFT	3	12 四心	21
2.3 Extgcd	4	13 Runge-Kutta	21
2.4 Pollard'sRho	4	14 Householder Matrix	21
3 Other	4		
3.1 Annealing	4		
3.2 DLX	4		
3.3 MahattanMST	5		
3.4 MoOnTree	6		
3.5 Det	6		
4 DataStructure	6		
4.1 PersistentTreap	6		
4.2 Pbds Kth	7		
4.3 PbdsHeap	7		
4.4 Heavy-LightDecomposition	7		
4.5 KDtree	8		
5 Default	9		
5.1 Default	9		
6 Flow	9		
6.1 Minmumwiegthmatchclique	9		
6.2 CostFlow	9		
6.3 MincutTree	10		
6.4 Dinic	10		
6.5 GeneralGraphmatch	10		
6.6 KM	11		
6.7 SWmincut	11		
7 Geometry	12		
7.1 Circleintersection	12		
7.2 Fermat's Point	12		
7.3 Pointoperators	12		
7.4 3DConvexHull	13		
7.5 Halfplaneintersection	13		
7.6 ConvexHull	14		
7.7 Triangulation	14		
7.8 K-closet Pair	15		
7.9 MCC	15		
7.10 LineIntersection	16		
7.11 PointToLine	16		
8 JAVA	16		
8.1 Big Integer	16		
8.2 Prime	16		
9 Graph	16		
9.1 MMC	16		
9.2 DMST	17		

1 String

1.1 AC

```

1 struct Node{
2     Node *index[30];
3     Node *fail;
4     int word;
5     int num;
6     Node(){
7         for(int i=0;i<30;i++)
8             index[i]=NULL;
9         fail=NULL;
10        word=0;
11        num=-1;
12    }
13 }*root=new Node();
14 void add(char c[]){
15     Node *n=root;
16     for(int i=0;c[i]!='\0';i++){
17
18         if(!n->index[c[i]-'a'])
19             n->index[c[i]-'a']=new Node();
20         n=n->index[c[i]-'a'];
21     }
22     n->word=1;
23     n->num++;
24 }
25 void ac(){
26     queue<Node*> q;
27     q.push(root);
28     root->fail=NULL;
29     while(!q.empty()){
30         Node *n=q.front();
31         q.pop();
32         for(int i=0;i<30;i++){
33             if(n->index[i]){
34                 q.push(n->index[i]);
35                 Node* p=n->fail;
36                 while(p!=NULL&&!p->index[i])
37                     p=p->fail;
38                 if(p)
39                     n->index[i]->fail=p->index[i];
40             }
41             else
42                 n->index[i]->fail=root;
43         }
44     }
45 }
46 void search(char c[]){
47     Node *n=root;
48     for(int i=0;c[i]!='\0';i++){
49
50         while(!n->index[c[i]-'a']&&n!=root){
51             n=n->fail;
52         }
53         if(n->index[c[i]-'a'])
54             n=n->index[c[i]-'a'];

```

```

55 Node *p=n;
56 while(p){
57     if(p->num!=-1)
58     {
59         ans[p->num]++;
60     }
61     p=p->fail;
62 }
63 }
64 }
65 void del(Node *n=root){
66     for(int i=0;i<30;i++)
67     if(n->index[i])
68     del(n->index[i]);
69     free(n);
70 }

```

1.2 SuffixAutomata

```

1 // BZOJ 3998
2 const int MAX_N = 500000 + 10;
3 struct Node {
4     static Node mem[MAX_N<<1] , *pmem;
5     Node *ch[26] , *fail;
6     int mx , val;
7     ll dp;
8     int tag , deg;
9     Node():mx(0),fail(0),dp(0),val(0),tag(0),deg(0){
10         MS(ch , 0);
11     }
12 }
13 Node::mem[MAX_N<<1] , *Node::pmem = Node::mem , *
14     root
15     , *last;
16 int T , N;
17 char s[MAX_N];
18 inline void init() {
19     last = root = new (Node::pmem++)Node();
20 }
21 inline int idx(char c) {
22     return c - 'a';
23 }
24 inline void insert(char c) {
25     c = idx(c);
26     Node *p = last;
27     Node *np = new (Node::pmem++)Node();
28     np->mx = p->mx + 1;
29     np->val = 1;
30     while(p && !p->ch[c]) {
31         p->ch[c] = np;
32         np->deg++;
33         p = p->fail;
34     }
35     if(!p) np->fail = root;
36     else
37     {
38         Node *q = p->ch[c];
39         if(q->mx == p->mx + 1) np->fail = q;
40         else
41         {
42             Node *nq = new (Node::pmem++)Node();
43             nq->mx = p->mx + 1;
44             nq->val = 0;
45             memcpy(nq->ch , q->ch , sizeof(q->ch));
46             REP(i , 26) {
47                 if(nq->ch[i]) nq->ch[i]->deg++;
48             }
49             nq->fail = q->fail;
50             q->fail = np->fail = nq;
51             while(p && p->ch[c] == q) {
52                 p->ch[c] = nq;
53                 q->deg--;
54                 nq->deg++;
55                 p = p->fail;
56             }
57         }
58     }
59     last = np;

```

```

60 inline void bfs() {
61     static Node* que[MAX_N<<1];
62     int l = 0 , r = 0;
63     que[r++] = root;
64     root->tag = 2;
65     vector<Node*> vec;
66     while(l < r) {
67         Node *u = que[l++];
68         REP(i , 26) {
69             if(u->ch[i]) {
70                 if(--u->ch[i]->deg == 0 && u->ch[i]
71 ]->
72                 tag != 1) {
73                     u->ch[i]->tag = 1;
74                     que[r++] = u->ch[i];
75                     vec.PB(u->ch[i]);
76                 }
77             }
78         }
79     }
80     for(int i = SZ(vec) - 1; i >= 0; i--) {
81         Node *u = vec[i];
82         if(T) {
83             if(u->fail) u->fail->val += u->val;
84             else u->val = 1;
85         }
86         root->val = 0;
87         for(int i = SZ(vec) - 1; i >= 0; i--) {
88             Node *u = vec[i];
89             u->dp = u->val;
90             REP(j , 26) {
91                 if(u->ch[j]) u->dp += u->ch[j]->dp;
92             }
93         }
94         REP(i , 26) {
95             if(root->ch[i]) root->dp += root->ch[i]->dp;
96         }
97     }
98     inline void solve(int k) {
99         Node *p = root;
100         if(k > p->dp || k <= 0) {
101             puts("-1");
102             return;
103         }
104         while(k > 0) {
105             int flag = 0;
106             REP(i , 26) {
107                 if(!p->ch[i]) continue;
108                 if(k <= p->ch[i]->dp) {
109                     putchar('a' + i);
110                     k -= p->ch[i]->val;
111                     p = p->ch[i];
112                     flag = 1;
113                     break;
114                 }
115             }
116             else k -= p->ch[i]->dp;
117         }
118         if(!flag) break;
119     }
120 }
121 int main() {
122     scanf("%s",s);
123     int n = strlen(s);
124     N = n;
125     init();
126     REP(i , n) insert(s[i]);
127     int K;
128     scanf("%d%d",&T,&K);
129     bfs();
130     solve(K);
131     return 0;
132 }

```

1.3 MinLexicographicalRotate

```

1 string mcp(string s){
2     int n = s.length();

```

```

3   s += s;
4   int i=0, j=1;
5   while (i<n && j<n){
6       int k = 0;
7       while (k < n && s[i+k] == s[j+k]) k++;
8       if (s[i+k] <= s[j+k]) j += k+1;
9       else i += k+1;
10      if (i == j) j++;
11  }
12  int ans = i < n ? i : j;
13  return s.substr(ans, n);
14 }

```

1.4 ZvaluePalindromes

```

1 inline void manacher(char *s,int len,int *z){
2     int l=0,r=0;
3     for(int i=1;i<len;++i){
4         z[i]=r>i?min(z[2*l-i],r-i):1;
5         while(s[i+z[i]]==s[i-z[i]])++z[i];
6         if(z[i]+i>r)r=z[i]+i,l=i;
7     }
8 }

```

1.5 SuffixArray

```

1 int ss[N];
2 int heigh[N];
3 int sa[N];
4 int rank[N];
5 int length;
6 int val[30];
7 int c[N]; // counting sort array
8 int temp[2][N];
9 void suffix_array()
10 {
11     int A = 250;
12     int* rank = temp[0];
13     int* new_rank = temp[1];
14     for (int i=0; i<A; ++i) c[i] = 0;
15     for (int i=0; i<length; ++i) c[rank[i] = ss[i]]++;
16     for (int i=1; i<A; ++i) c[i] += c[i-1];
17     for (int i=length-1; i>=0; --i) sa[--c[ss[i]]] = i;
18     for (int n=1; n<length; n*=2)
19     {
20         for (int i=0; i<A; ++i) c[i] = 0;
21         for (int i=0; i<length; ++i) c[rank[i]]++;
22         for (int i=1; i<A; ++i) c[i] += c[i-1];
23         int* sa2 = new_rank;
24         int r = 0;
25         for (int i=length-n; i<length; ++i)
26             sa2[r++] = i;
27         for (int i=0; i<length; ++i)
28             if (sa[i] >= n)
29                 sa2[r++] = sa[i] - n;
30         for (int i=length-1; i>=0; --i)
31             sa[--c[rank[sa2[i]]]] = sa2[i];
32         new_rank[sa[0]] = r = 0;
33         for (int i=1; i<length; ++i)
34         {
35             if (!(rank[sa[i-1]] == rank[sa[i]] &&
36                 sa[i-1]+n < length && // stable
37                 rank[sa[i-1]+n] == rank[sa[i]+n]))
38                 r++;
39             new_rank[sa[i]] = r;
40         }
41         swap(rank, new_rank);
42         if (r == length-1) break;
43         A = r + 1;
44     }
45 }
46 void lcp_array()
47 {

```

```

48     for (int i=0; i<length; ++i)
49         rank[sa[i]] = i;
50
51     for (int i=0, lcp=0, h=0; i<length; i++)
52         if (rank[i] == 0)
53             heigh[0] = 0;
54         else
55         {
56             int j = sa[rank[i]-1];
57             if (lcp > 0) lcp-=val[ss[i-1]-'a'],h--;
58             while (ss[i+h] == ss[j+h]) lcp+=val[ss[i
59                 +h]-'a'],h++;
60             heigh[rank[i]] = lcp;
61         }

```

1.6 Zvalue

```

1 inline void z_alg1(char *s,int len,int *z){
2     int l=0,r=0;
3     z[0]=len;
4     for(int i=1;i<len;++i){
5         z[i]=r>i?min(r-i+1,z[z[l]-(r-i+1)]):0;
6         while(i+z[i]<len&&s[z[i]]==s[i+z[i]])++z[i];
7         if(i+z[i]-1>r)r=i+z[i]-1,l=i;
8     }
9 }

```

2 Math

2.1 MillerRabin

```

1 // 4759123141 2, 7, 61
2 //2^64 2, 325, 9375, 28178, 450775, 9780504,
3 //1795265022
4 bool Isprime(LL n)
5 {
6     if (n == 2) return true;
7     if (n < 2 || n % 2 == 0) return false;
8     LL u = n - 1, t = 0;
9     while (u % 2 == 0) {u >>= 1; t++;}
10    LL sprp[7] = {2, 325, 9375, 28178, 450775,
11        9780504, 1795265022};
12    for (int k=0; k<7; ++k)
13    {
14        LL a = sprp[k] % n;
15        if (a == 0 || a == 1 || a == n-1) continue;
16        long long x = f_pow(a, u, n);
17        if (x == 1 || x == n-1) continue;
18        for (int i = 0; i < t-1; i++)
19        {
20            x = f_pow(x, 2, n);
21            if (x == 1) return false;
22            if (x == n-1) break;
23        }
24        if (x == n-1) continue;
25        return false;
26    }
27    return true;
28 }

```

2.2 FFT

```

1 #define N 524288
2 #define pi acos(-1)
3 typedef complex<double> C;
4 int n,m,i,t,g[N];
5 C a[N],b[N];
6 void FFTinit(){
7     for (i=1;i<N;i++) g[i]=g[i>>1]>>1|((i&1)<<18);
8 }

```

```

9 void FFT(C *a,int f)
10 {
11     int i,j,k,p;
12     for (i=0;i<N;i++)
13         if (g[i]>i) swap(a[i],a[g[i]]);
14     for (i=1;i<N;i<=1)
15     {
16         C e(cos(pi/i),f*sin(pi/i));
17         for (j=0;j<N;j+=i<=1)
18         {
19             C w(1,0);for (k=0;k<i;k++,w*=e)
20             {
21                 C x=a[j+k],y=w*a[j+k+i];
22                 a[j+k]=x+y;a[j+k+i]=x-y;
23             }
24         }
25     }
26 }
27 int res[400005];
28 int main()
29 {
30     FFTinit();
31     FFT(a,1);
32     FFT(b,1);
33     for(i=0;i<N;i++) a[i]=a[i]*b[i];
34     FFT(a,-1);
35     for (i=0;i<n+m;i++)
36         (int)a[i].real()/N+0.5)
37 }

```

2.3 Extgcd

```

1 typedef pair<int, int> pii;
2 pii gcd(int a, int b){
3     if(b == 0) return mp(1, 0);
4     else{
5         int p = a / b;
6         pii q = gcd(b, a % b);
7         return make_pair(q.y, q.x - q.y * p);
8     }
9 }

```

2.4 Pollard'sRho

```

1 // does not work when n is prime
2 inline LL f(LL x , LL mod) {
3     return (x * x % mod + 1) % mod;
4 }
5 inline LL pollard_rho(LL n) {
6     if(!(n&1)) return 2;
7     while(true) {
8         LL y = 2 , x = rand() % (n - 1) + 1 , res = 1;
9         for(int sz = 2; res == 1; sz *= 2) {
10             for(int i = 0; i < sz && res <= 1; i++) {
11                 x = f(x , n);
12                 res = __gcd(abs(x - y) , n);
13             }
14             y = x;
15         }
16         if (res != 0 && res != n) return res;
17     }
18 }

```

3 Other

3.1 Annealing

```

1 double distForAllPoints(double x, double y,
2     vector< pair<int, int> > &D) {
3     double sum = 0;
4     for(int i = D.size()-1; i >= 0; i--) {

```

```

5         sum += hypot(D[i].first - x, D[i].second - y);
6     }
7     return sum;
8 }
9 double randDouble() {
10     return (rand() % 32767) / 32767.0;
11 }
12 double annealing(vector< pair<int, int> > &D) {
13     #define S_MUL 0.6f
14     #define S_LEN 1000
15     #define T_CNT 10
16     #define E_CNT 10
17     double step = S_LEN;
18     double x[E_CNT], y[E_CNT], val[E_CNT];
19     double Lx, Ly, Rx, Ry, tx, ty, tcost;
20     Lx = Rx = D[0].first;
21     Ly = Ry = D[0].second;
22     for(int i = 0; i < D.size(); i++) {
23         Lx = min(Lx, (double)D[i].first);
24         Rx = max(Rx, (double)D[i].first);
25         Ly = min(Ly, (double)D[i].second);
26         Ry = max(Ry, (double)D[i].second);
27     }
28     for(int i = 0; i < E_CNT; i++) {
29         x[i] = randDouble() * (Rx - Lx) + Lx;
30         y[i] = randDouble() * (Ry - Ly) + Ly;
31         val[i] = distForAllPoints(x[i], y[i], D);
32     }
33     while(step > 0.1) {
34         for(int i = 0; i < E_CNT; i++) {
35             for(int j = 0; j < T_CNT; j++) {
36                 tx = x[i] + randDouble() * 2 * step - step;
37                 ty = y[i] + randDouble() * 2 * step - step;
38                 tcost = distForAllPoints(tx, ty, D);
39                 if(tcost < val[i]) {
40                     val[i] = tcost, x[i] = tx, y[i] = ty;
41                 }
42             }
43         }
44         step *= S_MUL;
45     }
46     double ret = val[0];
47     for(int i = 0; i < E_CNT; i++) {
48         ret = min(ret, val[i]);
49     }
50     printf("%.0lf\n", ret);
51 }
52 int main() {
53     int testcase, N;
54     scanf("%d", &testcase);
55     while(testcase--) {
56         scanf("%d", &N);
57         vector< pair<int, int> > D;
58         int x, y;
59         for(int i = 0; i < N; i++) {
60             scanf("%d %d", &x, &y);
61             D.push_back(make_pair(x, y));
62         }
63         annealing(D);
64         if(testcase)
65             puts("");
66     }
67     return 0;
68 }

```

3.2 DLX

```

1 struct DLX{
2     int n,m,len;
3     int U[maxnode],D[maxnode],R[maxnode],L[maxnode],
4         Row[maxnode],Col[maxnode];
5     int H[maxn];
6     int S[maxm];
7     int ansd,ans[maxn];
8
9     void init(int _n,int _m){
10         n = _n;m = _m;
11         for(int i = 0; i <= m; i++){

```

```

12         U[i] = D[i] = i;
13         L[i] = i-1;
14         R[i] = i+1;
15     }
16     R[m] = 0, L[0] = m;
17     len = m;
18     for(int i = 1; i <= n; i++)
19         H[i] = -1;
20 }
21
22 void link(int r, int c){
23     ++S[Col[++len]=c];
24     Row[len] = r;
25     D[len] = D[c];
26     U[D[c]] = len;
27     U[len] = c;
28     D[c] = len;
29     if(H[r] < 0)
30         H[r] = L[len] = R[len] = len;
31     else{
32         R[len] = R[H[r]];
33         L[R[H[r]]] = len;
34         L[len] = H[r];
35         R[H[r]] = len;
36     }
37 }
38
39 void del(int c){
40     L[R[c]] = L[c];
41     R[L[c]] = R[c];
42     for(int i = D[c]; i != c; i = D[i]){
43         for(int j = R[i]; j != i; j = R[j]){
44             U[D[j]] = U[j];
45             D[U[j]] = D[j];
46             --S[Col[j]];
47         }
48     }
49 }
50
51 void resume(int c){
52     for(int i = U[c]; i != c; i = U[i]){
53         for(int j = L[i]; j != i; j = L[j]){
54             ++S[Col[U[D[j]]]=D[U[j]]=j];
55         }
56     }
57     L[R[c]] = R[L[c]] = c;
58 }
59
60 void dance(int d){
61     //剪枝
62     if(ansd != -1 && ansd <= d)
63         return;
64     if(R[0] == 0){
65         if(ansd == -1)
66             ansd = d;
67         else if(d < ansd)
68             ansd = d;
69         return;
70     }
71     int c = R[0];
72     for(int i = R[0]; i != 0; i = R[i]){
73         if(S[i] < S[c])
74             c = i;
75     }
76     del(c);
77     for(int i = D[c]; i != c; i = D[i]){
78         ans[d] = Row[i];
79         for(int j = R[i]; j != i; j = R[j])
80             del(Col[j]);
81         dance(d+1);
82         for(int j = L[i]; j != i; j = L[j])
83             resume(Col[j]);
84     }
85     resume(c);
86 }
87 };

```

```

1 #include<bits/stdc++.h>
2 #define REP(i,n) for(int i=0;i<n;i++)
3 using namespace std;
4 typedef long long LL;
5 const int N=200100;
6 int n,m;
7 struct PT {int x,y,z,w,id;}p[N];
8 inline int dis(const PT &a,const PT &b){return abs(a
    .xb.x)+abs(a.y-b.y);}
9 inline bool cpx(const PT &a,const PT &b){return a.x
    !=b.
10 x? a.x>b.x:a.y>b.y;}
11 inline bool cpz(const PT &a,const PT &b){return a.z<
    b.z
12 ;}
13 struct E{int a,b,c;}e[8*N];
14 bool operator<(const E&a,const E&b){return a.c<b.c;}
15 struct Node{
16     int L,R,key;
17 }node[4*N];
18 int s[N];
19 int F(int x){return s[x]==x?s[x]:F(s[x]);}
20 void U(int a,int b){s[F(b)]=F(a);}
21 void init(int id,int L,int R) {
22     node[id]=(Node){L,R,-1};
23     if(L==R)return
24     ;
25     init(id*2,L,(L+R)/2);
26     init(id*2+1,(L+R)/2+1,R);
27 }
28 void ins(int id,int x) {
29     if(node[id].key==-1 || p[node[id].key].w>p[x].w)
30         node[
31         id].key=x;
32     if(node[id].L==node[id].R)return
33     ;
34     if(p[x].z<=(node[id].L+node[id].R)/2)ins(id*2,x)
35     ;
36     else ins(id*2+1,x);
37 }
38 int Q(int id,int L,int R){
39     if(R<node[id].L || L>node[id].R)return -1;
40     if(L<=node[id].L && node[id].R<=R)return node[id
    ].key ;
41     int a=Q(id*2,L,R),b=Q(id*2+1,L,R);
42     if(b==-1 || (a!=-1 && p[a].w<p[b].w)) return a;
43     else return b;
44 }
45 void calc() {
46     REP(i,n) {
47         p[i].z=p[i].y-p[i].x;
48         p[i].w=p[i].x+p[i].y;
49     }
50     sort(p,p+n,cpz);
51     int cnt=0,j,k;
52     for
53     (int i=0;i<n;i=j){
54         for(j=i+1;p[j].z==p[i].z && j<n;j++);
55         for(k=i,cnt++;k<j;k++)p[k].z=cnt;
56     }
57     init(1,1,cnt);
58     sort(p,p+n,cpx);
59     REP(i,n) {
60         j=Q(1,p[i].z,cnt);
61         if(j!=-1)e[m++]=(E){p[i].id,p[j].id,dis(p[i
    ],p[j])
62         };
63         ins(1,i);
64     }
65 }
66 LL MST() {
67     LL r=0;
68     sort(e,e+m);
69     REP(i,m) {
70         if(F(e[i].a)==F(e[i].b))continue;
71         U(e[i].a,e[i].b);
72         r+=e[i].c;
73     }
74     return r;
75 }
76 int main(){
77     int ts;

```

3.3 MahattanMST

```

76 scanf("%d", &ts);
77 while (ts-->0) {
78     m = 0;
79     scanf("%d",&n);
80     REP(i,n) {scanf("%d%d",&p[i].x,&p[i].y);p[i].id=s[i]=i;}
81     calc();
82     REP(i,n)p[i].y = -p[i].y;
83     calc();
84     REP(i,n)swap(p[i].x,p[i].y);
85     calc();
86     REP(i,n)p[i].x=-p[i].x;
87     calc();
88     printf("%lld\n",MST()*2);
89 }
90 return 0;
91 }

```

3.4 MoOnTree

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define IOS ios_base::sync_with_stdio(0); cin.tie(0);
4 #define SZ(x) ((int)((x).size()))
5 const int MX = 500005;
6 const int SQ = 1400;
7 const int LOG = 17;
8 struct BIT {
9     int bit[MX];
10     int lb(int x) { return x & -x; }
11     void add(int p, int v) {
12         p++;
13         for (int i=p; i<MX; i+=lb(i)) bit[i] += v;
14     }
15     int qry() {
16         int v = 0;
17         for (int i=1<<LOG; i>0; i>>=1) {
18             if ((v|i) < MX and bit[v|i]==i) v |= i;
19         }
20         return v;
21     }
22 }bit;
23 struct Query {
24     int l,r,qid;
25 }qry[MX];
26 struct Edge {
27     int v,x;
28 };
29 int N,Q,timestamp[MX],ans[MX];
30 int in[MX],cnt[MX];
31 vector<Edge> E[MX];
32 vector<Edge> seq;
33 void DFS(int u, int f) {
34     timestamp[u] = SZ(seq);
35     for (auto it:E[u]) {
36         if (it.v == f) continue;
37         seq.push_back(it);
38         DFS(it.v,u);
39         seq.push_back(it);
40     }
41 }
42 void poke(int id) {
43     int v = seq[id].v;
44     int x = seq[id].x;
45     in[v] ^= 1;
46     cnt[x] += in[v] ? 1 : -1;
47     if (in[v] and cnt[x] == 1) bit.add(x, 1);
48     if (!in[v] and cnt[x] == 0) bit.add(x, -1);
49 }
50 int main() {
51     IOS;
52     cin >> N >> Q;
53     for (int i=0; i<N-1; i++) {
54         int u,v,x;
55         cin >> u >> v >> x;
56         x = min(x,N);
57         E[u].push_back({v,x});
58         E[v].push_back({u,x});

```

```

59     }
60     DFS(1,1);
61     for (int i=1; i<=Q; i++) {
62         int u,v;
63         cin >> u >> v;
64         int l = timestamp[u], r = timestamp[v];
65         if (l > r) swap(l,r);
66         r--;
67         qry[i] = {l,r,i};
68     }
69     sort(qry+1,qry+1+Q, [](Query a, Query b) {
70         return make_pair(a.l/SQ,a.r) < make_pair(b.l/SQ,b.r);
71     });
72     int curL = 1, curR = 0;
73     for (int i=1; i<=Q; i++) {
74         int ql=qry[i].l,qr=qry[i].r;
75         while (curL > ql) poke(--curL);
76         while (curR < qr) poke(++curR);
77         while (curL < ql) poke(curL++);
78         while (curR > qr) poke(curR--);
79         ans[qry[i].qid] = bit.qry();
80     }
81     for (int i=1; i<=Q; i++) cout << ans[i] << "\n";
82     return 0;
83 }
84 }

```

3.5 Det

```

1 LL det(LL a[][20],int n)
2 {
3     LL ret=1;
4     for(int i=1;i<n;i++)
5     {
6         for(int j=i+1;j<n;j++)
7             while(a[j][i])
8             {
9                 LL t=a[i][i]/a[j][i];
10                for(int k=i;k<n;k++)
11                    a[i][k]=a[i][k]-a[j][k]*t;
12                for(int k=i;k<n;k++)
13                    swap(a[i][k],a[j][k]);
14                ret=-ret;
15            }
16        if(a[i][i]==0)return 0;
17        ret=ret*a[i][i];
18    }
19    return ret;
20 }
21 }

```

4 DataStructure

4.1 PersistentTreap

```

1 const int MEM = 16000004;
2 struct Treap {
3     static Treap nil, mem[MEM], *pmem;
4     Treap *l, *r;
5     char val;
6     int size;
7     Treap () : l(&nil), r(&nil), size(0) {}
8     Treap (char _val) : l(&nil), r(&nil), val(_val), size(1) {}
9 } Treap::nil, Treap::mem[MEM], *Treap::pmem = Treap::nil;
10 mem;
11 int size(const Treap *t) { return t->size; }
12 void pull(Treap *t) {
13     if (!size(t)) return;
14     t->size = size(t->l) + size(t->r) + 1;
15 }
16 Treap* merge(Treap *a, Treap *b) {

```



```

18 if (!size(a)) return b;
19 if (!size(b)) return a;
20 Treap *t;
21 if (rand() % (size(a) + size(b)) < size(a)) {
22     t = new (Treap::pmem++) Treap(*a);
23     t->r = merge(a->r, b);
24 } else {
25     t = new (Treap::pmem++) Treap(*b);
26     t->l = merge(a, b->l);
27 }
28 pull(t);
29 return t;
30 }
31 void split(Treap *t, int k, Treap *&a, Treap *&b) {
32     if (!size(t)) a = b = &Treap::nil;
33     else if (size(t->l) + 1 <= k) {
34         a = new (Treap::pmem++) Treap(*t);
35         split(t->r, k - size(t->l) - 1, a->r, b);
36         pull(a);
37     } else {
38         b = new (Treap::pmem++) Treap(*t);
39         split(t->l, k, a, b->l);
40         pull(b);
41     }
42 }
43 int nv;
44 Treap *rt[50005];
45 void print(const Treap *t) {
46     if (!size(t)) return;
47     print(t->l);
48     cout << t->val;
49     print(t->r);
50 }
51 int main(int argc, char** argv) {
52     IOS;
53     rt[nv=0] = &Treap::nil;
54     Treap::pmem = Treap::mem;
55     int Q, cmd, p, c, v;
56     string s;
57     cin >> Q;
58     while (Q--) {
59         cin >> cmd;
60         if (cmd == 1) {
61             // insert string s after position p
62             cin >> p >> s;
63             Treap *tl, *tr;
64             split(rt[nv], p, tl, tr);
65             for (int i=0; i<s.size(); i++)
66                 tl = merge(tl, new (Treap::pmem++) Treap
67                     (s[i]));
68             rt[++nv] = merge(tl, tr);
69         } else if (cmd == 2) {
70             // remove c characters starting at
71             position
72             Treap *tl, *tm, *tr;
73             cin >> p >> c;
74             split(rt[nv], p-1, tl, tm);
75             split(tm, c, tm, tr);
76             rt[++nv] = merge(tl, tr);
77         } else if (cmd == 3) {
78             // print c characters starting at
79             position p, in version v
80             Treap *tl, *tm, *tr;
81             cin >> v >> p >> c;
82             split(rt[v], p-1, tl, tm);
83             split(tm, c, tm, tr);
84             print(tm);
85             cout << "n";
86         }
87     }
88     return 0;
89 }

```

4.2 Pbds Kth

```

1 #include <bits/extc++.h>
2 using namespace __gnu_pbds;
3 typedef tree<int, null_type, less<int>, rb_tree_tag,

```

```

4 tree_order_statistics_node_update> set_t;
5 int main()
6 {
7     // Insert some entries into s.
8     set_t s;
9     s.insert(12);s.insert(505);
10    // The order of the keys should be: 12, 505.
11    assert(*s.find_by_order(0) == 12);
12    assert(*s.find_by_order(3) == 505);
13    // The order of the keys should be: 12, 505.
14    assert(s.order_of_key(12) == 0);
15    assert(s.order_of_key(505) == 1);
16    // Erase an entry.
17    s.erase(12);
18    // The order of the keys should be: 505.
19    assert(*s.find_by_order(0) == 505);
20    // The order of the keys should be: 505.
21    assert(s.order_of_key(505) == 0);
22 }

```

4.3 PbdsHeap

```

1 #include <bits/extc++.h>
2 typedef __gnu_pbds::priority_queue<int> heap_t;
3 heap_t a,b;
4 int main() {
5     a.clear();b.clear();
6     a.push(1);a.push(3);
7     b.push(2);b.push(4);
8     assert(a.top() == 3);
9     assert(b.top() == 4);
10    // merge two heap
11    a.join(b);
12    assert(a.top() == 4);
13    assert(b.empty());
14    return 0;
15 }

```

4.4 Heavy-LightDecomposition

```

1 #define N
2 void init();//implement
3 int n,fa[N],belong[N],dep[N],sz[N],que[N];
4 int step,line[N],stPt[N],edPt[N];
5 vector<int> v[N], chain[N];
6 void DFS(int u){
7     vector<int> &c = chain[belong[u]];
8     for (int i=c.size()-1; i>=0; i--){
9         int v = c[i];
10        stPt[v] = step;
11        line[step++] = v;
12    }
13    for (int i=0; i<(int)c.size(); i++){
14        u = c[i];
15        for (vector<int>::iterator it=v[u].begin();
16            it!=v[u].end();it++){
17            if (fa[u] == *it || (i && *it == c[i-1]))
18                continue;
19            DFS(*it);
20        }
21        edPt[u] = step-1;
22    }
23 }
24 void build_chain(int st){
25     int fr,bk;
26     fr=bk=0; que[bk++] = 1; fa[st]=st; dep[st]=0;
27     while (fr < bk){
28         int u=que[fr++];
29         for (vector<int>::iterator it=v[u].begin();
30             it!=v[u].end();it++){
31             if (*it == fa[u]) continue;
32             que[bk++] = *it;
33             dep[*it] = dep[u]+1;
34             fa[*it] = u;
35         }
36     }
37 }

```

```

34 for (int i=bk-1,u,pos; i>=0; i--){
35     u = que[i]; sz[u] = 1; pos = -1;
36     for (vector<int>::iterator it=v[u].begin();
37         it!=v[u].end();it++){
38         if (*it == fa[u]) continue;
39         sz[u] += sz[*it];
40         if (pos==-1 || sz[*it]>sz[pos]) pos=*it;
41     }
42     if (pos == -1) belong[u] = u;
43     else belong[u] = belong[pos];
44     chain[belong[u]].pb(u);
45 }
46 step = 0;
47 DFS(st);
48 int getLCA(int u, int v){
49     while (belong[u] != belong[v]){
50         int a = chain[belong[u]].back();
51         int b = chain[belong[v]].back();
52         if (dep[a] > dep[b]) u = fa[a];
53         else v = fa[b];
54     }
55     return sz[u] >= sz[v] ? u : v;
56 }
57 vector<pii> getPathSeg(int u, int v){
58     vector<pii> ret1,ret2;
59     while (belong[u] != belong[v]){
60         int a = chain[belong[u]].back();
61         int b = chain[belong[v]].back();
62         if (dep[a] > dep[b]){
63             ret1.pb(mp(stPt[a],stPt[u]));
64             u = fa[a];
65         } else {
66             ret2.pb(mp(stPt[b],stPt[v]));
67             v = fa[b];
68         }
69     }
70     if (dep[u] > dep[v]) swap(u,v);
71     ret1.pb(mp(stPt[u],stPt[v]));
72     reverse(ret2.begin(), ret2.end());
73     ret1.insert(ret1.end(),ret2.begin(),ret2.end());
74     return ret1;
75 }
76 // Usage
77 void build(){
78     build_chain(1); //change root
79     init();
80 }
81 int get_answer(int u, int v){
82     int ret = -2147483647;
83     vector<pii> vec = getPathSeg(u,v);
84     for (vector<pii>::iterator it =vec.begin();it!=
85         vec.end();it++){
86         // check answer with segment [it.F, it.S]
87         return ret;
88     }

```

4.5 KDtree

```

1 struct KDTree {
2     struct Node {
3         int x,y,x1,y1,x2,y2;
4         int id,f;
5         Node *L, *R;
6     }tree[MXN];
7     int n;
8     Node *root;
9     long long dis2(int x1, int y1, int x2, int y2) {
10         long long dx = x1-x2;
11         long long dy = y1-y2;
12         return dx*dx+dy*dy;
13     }
14     static bool cmpx(Node& a, Node& b){ return a.x<b
15         .x; }
16     static bool cmpy(Node& a, Node& b){ return a.y<b
17         .y; }
18     void init(vector<pair<int,int>> ip) {
19         n = ip.size();
20         for (int i=0; i<n; i++) {

```

```

19         tree[i].id = i;
20         tree[i].x = ip[i].first;
21         tree[i].y = ip[i].second;
22     }
23     root = build_tree(0, n-1, 0);
24 }
25 Node* build_tree(int L, int R, int dep) {
26     if (L>R) return nullptr;
27     int M = (L+R)/2;
28     tree[M].f = dep%2;
29     nth_element(tree+L, tree+M, tree+R+1, tree[M
30     ].f ?
31     cmpx : cmpy);
32     tree[M].x1 = tree[M].x2 = tree[M].x;
33     tree[M].y1 = tree[M].y2 = tree[M].y;
34     tree[M].L = build_tree(L, M-1, dep+1);
35     if (tree[M].L) {
36         tree[M].x1 = min(tree[M].x1, tree[M].L->
37         x1);
38         tree[M].x2 = max(tree[M].x2, tree[M].L->
39         x2);
40         tree[M].y1 = min(tree[M].y1, tree[M].L->
41         y1);
42         tree[M].y2 = max(tree[M].y2, tree[M].L->
43         y2);
44     }
45     tree[M].R = build_tree(M+1, R, dep+1);
46     if (tree[M].R) {
47         tree[M].x1 = min(tree[M].x1, tree[M].R->
48         x1);
49         tree[M].x2 = max(tree[M].x2, tree[M].R->
50         x2);
51         tree[M].y1 = min(tree[M].y1, tree[M].R->
52         y1);
53         tree[M].y2 = max(tree[M].y2, tree[M].R->
54         y2);
55     }
56     return tree+M;
57 }
58 int touch(Node* r, int x, int y, long long d2){
59     long long dis = sqrt(d2)+1;
60     if (x<r->x1-dis || x>r->x2+dis || y<r->y1-
61     dis || y>
62     r->y2+dis)
63         return 0;
64     return 1;
65 }
66 void nearest(Node* r, int x, int y, int &mID,
67     long
68     &md2) {
69     if (!r || !touch(r, x, y, md2)) return;
70     long long d2 = dis2(r->x, r->y, x, y);
71     if (d2 < md2 || (d2 == md2 && mID < r->id))
72     {
73         mID = r->id;
74         md2 = d2;
75     }
76     // search order depends on split dim
77     if ((r->f == 0 && x < r->x) ||
78         (r->f == 1 && y < r->y)) {
79         nearest(r->L, x, y, mID, md2);
80         nearest(r->R, x, y, mID, md2);
81     } else {
82         nearest(r->R, x, y, mID, md2);
83         nearest(r->L, x, y, mID, md2);
84     }
85 }
86 int query(int x, int y) {
87     int id = 1029384756;
88     long long d2 = 102938475612345678LL;
89     nearest(root, x, y, id, d2);
90     return id;
91 }
92 }tree;

```


5 Default

5.1 Default

```

1 #include<bits/stdc++.h>
2 #define mp(a,b) make_pair((a),(b))
3 #define pii pair<int,int>
4 #define pdd pair<double,double>
5 #define pll pair<LL,LL>
6 #define pb(x) push_back(x)
7 #define x first
8 #define y second
9 #define sqr(x) ((x)*(x))
10 #define EPS 1e-6
11 #define mii map<int,int>
12 #define MEM(x) memset(x,0,sizeof(x))
13 #define MEMS(x) memset(x,-1,sizeof(x))
14 #define pi 3.14159265359
15 // #define INF 0x7fffffff
16 #define IOS ios_base::sync_with_stdio(0); cin.tie(0)
17 #define N 300005
18 using namespace std;
19 typedef long long LL;

```

```

45         stk.clear();
46         if (!onstk[i] && SPFA(i)){
47             found = 1;
48             while (stk.size()>=2){
49                 int u = stk.back(); stk.
pop_back();
50                 int v = stk.back(); stk.
pop_back();
51                 match[u] = v;
52                 match[v] = u;
53             }
54         }
55         if (!found) break;
56     }
57     int ret = 0;
58     for (int i=0; i<n; i++){
59         ret += edge[i][match[i]];
60         ret /= 2;
61     }
62     return ret;
63 }
64 }graph;

```

6 Flow

6.1 Minunwieghtmatchclique

```

1 struct Graph {
2     // Minimum General Weighted Matching (Perfect
3     // Match) clique
4     static const int MXN = 105;
5     int n, edge[MXN][MXN];
6     int match[MXN], dis[MXN], onstk[MXN];
7     vector<int> stk;
8     void init(int _n) {
9         n = _n;
10        MEM(edge);
11    }
12    void add_edge(int u, int v, int w) {
13        edge[u][v] = edge[v][u] = w;
14    }
15    bool SPFA(int u){
16        if (onstk[u]) return true;
17        stk.pb(u);
18        onstk[u] = 1;
19        for (int v=0; v<n; v++){
20            if (u != v && match[u] != v && !onstk[v]
21                && (dis[u] - edge[u][v] > dis[v])){
22                int m = match[v];
23                if (dis[m] > dis[u] - edge[v][m] +
24                    edge[u][v]){
25                    dis[m] = dis[u] - edge[v][m] +
26                        edge[u][v];
27                    onstk[v] = 1;
28                    stk.pb(v);
29                    if (SPFA(m)) return true;
30                    stk.pop_back();
31                    onstk[v] = 0;
32                }
33            }
34        }
35        onstk[u] = 0;
36        stk.pop_back();
37        return false;
38    }
39    int solve() {
40        // find a match
41        for (int i=0; i<n; i+=2){
42            match[i] = i+1;
43            match[i+1] = i;
44        }
45        while (true){
46            int found = 0;
47            MEM(dis); MEM(onstk);
48            for (int i=0; i<n; i++){

```

6.2 CostFlow

```

1 struct CostFlow {
2     static const int MXN = 205;
3     static const long long INF = 102938475610293847
4     LL;
5     struct Edge {
6         int v, r;
7         long long f, c;
8         Edge(int a, int b, int _c, int d):v(a),r(b),f(
9             _c),c(d){}
10    };
11    int n, s, t, prv[MXN], prvl[MXN], inq[MXN];
12    long long dis[MXN], fl, cost;
13    vector<Edge> E[MXN];
14    void init(int _n, int _s, int _t) {
15        n = _n; s = _s; t = _t;
16        for (int i=0; i<n; i++) E[i].clear();
17        fl = cost = 0;
18    }
19    void add_edge(int u, int v, long long f, long
20        long c) {
21        E[u].pb(Edge(v, E[v].size(), f, c));
22        E[v].pb(Edge(u, E[u].size()-1, 0, -c));
23    }
24    pll flow() {
25        while (true) {
26            for (int i=0; i<n; i++) {
27                dis[i] = INF;
28                inq[i] = 0;
29            }
30            dis[s] = 0;
31            queue<int> que;
32            que.push(s);
33            while (!que.empty()) {
34                int u = que.front(); que.pop();
35                inq[u] = 0;
36                for (int i=0; i<E[u].size(); i++) {
37                    int v = E[u][i].v;
38                    long long w = E[u][i].c;
39                    if (E[u][i].f > 0 && dis[v] >
40                        dis[u] + w) {
41                        prv[v] = u; prvl[v] = i;
42                        dis[v] = dis[u] + w;
43                        if (!inq[v]) {
44                            inq[v] = 1;
45                            que.push(v);
46                        }
47                    }
48                }
49            }
50            if (dis[t] == INF) break;
51            long long tf = INF;
52            for (int v=t, u, l; v!=s; v=u) {

```

```

51         u=prv[v]; l=prvL[v];
52         tf = min(tf, E[u][l].f);
53     }
54     for (int v=t, u, l; v!=s; v=u) {
55         u=prv[v]; l=prvL[v];
56         E[u][l].f -= tf;
57         E[v][E[u][l].r].f += tf;
58     }
59     cost += tf * dis[t];
60     fl += tf;
61 }
62 return {fl, cost};
63 }
64 }flow;

```

6.3 MincutTree

```

1 set<int> temp;
2 int Vis[3005];
3 int cvis[3005];
4 void dfs(int n){
5     Vis[n]=1;
6     for(auto it=v[n].begin();it!=v[n].end();it++){
7         if(val[n][*it]>flow[n][*it]&&!Vis[*it]){
8             dfs(*it);
9             if(cvis[*it])
10                 temp.insert(*it);
11         }
12     }
13 }
14 int n;
15 int dc(set<int> s,int flag){
16     if(s.size()==1)
17         return *s.begin();
18     for(int i=0;i<n;i++){
19         for(auto it=v[i].begin();it!=v[i].end();it++){
20             flow[i][*it]=0;
21         }
22         for(auto it=s.begin();it!=s.end();it++){
23             cvis[*it]=1;
24         }
25         int res=Flow(*s.begin(),*s.rbegin());
26         MEM(Vis);
27         dfs(*s.begin());
28         temp.insert(*s.begin());
29         for(auto it=s.begin();it!=s.end();it++){
30             cvis[*it]=0;
31         }
32         set<int> s1,s2;
33         swap(s1,temp);
34         temp.clear();
35         for(auto it=s1.begin();it!=s1.end();it++){
36             s.erase(*it);
37             swap(s2,s);
38             int x=dc(s1,0);
39             int y=dc(s2,1);
40             vt[x].pb(mp(y,res));
41             vt[y].pb(mp(x,res));
42             if(flag==0)
43                 return x;
44             else
45                 return y;
46         }
47     }
48 }

```

6.4 Dinic

```

1 struct Dinic{
2     static const int MXN = 10000;
3     struct Edge{ int v,f,re; Edge(int a,int b,int c)
4         :v(a),f(b),re(c){}};
5     int n,s,t,level[MXN];
6     vector<Edge> E[MXN];
7     void init(int _n, int _s, int _t){
8         n = _n; s = _s; t = _t;
9         for (int i=0; i<=n; i++) E[i].clear();
10    }
11    void add_edge(int u, int v, int f){

```

```

11        E[u].pb(Edge(v,f,E[v].size()));
12        E[v].pb(Edge(u,0,E[u].size()-1));//direct
13    }
14    bool BFS(){
15        MEMS(level);
16        queue<int> que;
17        que.push(s);
18        level[s] = 0;
19        while (!que.empty()){
20            int u = que.front(); que.pop();
21            for (auto it : E[u]){
22                if (it.f > 0 && level[it.v] == -1){
23                    level[it.v] = level[u]+1;
24                    que.push(it.v);
25                }
26            }
27        }
28        return level[t] != -1;
29    }
30    int DFS(int u, int nf){
31        if (u == t) return nf;
32        int res = 0;
33        for (auto &it : E[u]){
34            if (it.f > 0 && level[it.v] == level[u]
35                ]+1){
36                int tf = DFS(it.v, min(nf,it.f));
37                res += tf; nf -= tf; it.f -= tf;
38                E[it.v][it.re].f += tf;
39                if (nf == 0) return res;
40            }
41        }
42        if (!res) level[u] = -1;
43        return res;
44    }
45    int flow(int res=0){
46        while ( BFS() )
47            res += DFS(s,2147483647);
48        return res;
49    }flow;

```

6.5 GeneralGraphmatch

```

1 struct GenMatch { // 1-base
2     static const int MAXN = 505;
3     int V;
4     bool el[MAXN][MAXN];
5     int pr[MAXN];
6     bool inq[MAXN],inp[MAXN],inb[MAXN];
7     queue<int> qe;
8     int st,ed;
9     int nb;
10    int bk[MAXN],djs[MAXN];
11    int ans;
12    void init(int _V) {
13        V = _V;
14        MEM(el); MEM(pr);
15        MEM(inq); MEM(inp); MEM(inb);
16        MEM(bk); MEM(djs);
17        ans = 0;
18    }
19    void add_edge(int u, int v) {
20        el[u][v] = el[v][u] = 1;
21    }
22    int lca(int u,int v) {
23        memset(inp,0,sizeof(inp));
24        while(1) {
25            u = djs[u];
26            inp[u] = true;
27            if(u == st) break;
28            u = bk[pr[u]];
29        }
30        while(1) {
31            v = djs[v];
32            if(inp[v]) return v;
33            v = bk[pr[v]];
34        }
35        return v;
36    }

```

```

37 void upd(int u) {
38     int v;
39     while(djs[u] != nb) {
40         v = pr[u];
41         inb[djs[u]] = inb[djs[v]] = true;
42         u = bk[v];
43         if(djs[u] != nb) bk[u] = v;
44     }
45 }
46 void blo(int u,int v) {
47     nb = lca(u,v);
48     memset(inb,0,sizeof(inb));
49     upd(u); upd(v);
50     if(djs[u] != nb) bk[u] = v;
51     if(djs[v] != nb) bk[v] = u;
52     for(int tu = 1; tu <= V; tu++)
53         if(inb[djs[tu]]) {
54             djs[tu] = nb;
55             if(!inq[tu]){
56                 qe.push(tu);
57                 inq[tu] = 1;
58             }
59         }
60 }
61 void flow() {
62     memset(inq,false,sizeof(inq));
63     memset(bk,0,sizeof(bk));
64     for(int i = 1; i <= V; i++)
65         djs[i] = i;
66     while(qe.size()) qe.pop();
67     qe.push(st);
68     inq[st] = 1;
69     ed = 0;
70     while(qe.size()) {
71         int u = qe.front(); qe.pop();
72         for(int v = 1; v <= V; v++)
73             if(el[u][v] && (djs[u] != djs[v]) && (pr
74 [u] !=
75 v)) {
76                 if((v == st) || ((pr[v] > 0) && bk[
77 pr[v]] >
78 0))
79                     blo(u,v);
80                 else if(bk[v] == 0) {
81                     bk[v] = u;
82                     if(pr[v] > 0) {
83                         if(!inq[pr[v]]) qe.push(pr[v]
84 ));
85                     } else {
86                         ed = v;
87                         return;
88                     }
89                 }
90             }
91         }
92     }
93 void aug() {
94     int u,v,w;
95     u = ed;
96     while(u > 0) {
97         v = bk[u];
98         w = pr[v];
99         pr[v] = u;
100         pr[u] = v;
101         u = w;
102     }
103 }
104 int solve() {
105     memset(pr,0,sizeof(pr));
106     for(int u = 1; u <= V; u++)
107         if(pr[u] == 0) {
108             st = u;
109             flow();
110             if(ed > 0) {
111                 aug();
112                 ans ++;
113             }
114         }
115     }
116     return ans;
117 }
118 }gp;

```

6.6 KM

```

1 typedef pair<long long, long long> pll;
2 struct KM{
3     // Maximum Bipartite Weighted Matching (Perfect
4     Match)
5     static const int MXN = 650;
6     static const int INF = 2147483647; // long long
7     int n,match[MXN],vx[MXN],vy[MXN];
8     int edge[MXN][MXN],lx[MXN],ly[MXN],slack[MXN];
9     // ^^^^ long long
10    void init(int _n){
11        n = _n;
12        for (int i=0; i<n; i++){
13            for (int j=0; j<n; j++){
14                edge[i][j] = 0;
15            }
16        }
17    void add_edge(int x, int y, int w){ // long long
18        edge[x][y] = w;
19    }
20    bool DFS(int x){
21        vx[x] = 1;
22        for (int y=0; y<n; y++){
23            if (vy[y]) continue;
24            if (lx[x]+ly[y] > edge[x][y]){
25                slack[y] = min(slack[y], lx[x]+ly[y]
26 ]-edge[x][y
27 ]);
28            } else {
29                vy[y] = 1;
30                if (match[y] == -1 || DFS(match[y]))
31                {
32                    match[y] = x;
33                    return true;
34                }
35            }
36        }
37        return false;
38    }
39    int solve(){
40        fill(match,match+n,-1);
41        fill(lx,lx+n,-INF);
42        fill(ly,ly+n,0);
43        for (int i=0; i<n; i++){
44            for (int j=0; j<n; j++){
45                lx[i] = max(lx[i], edge[i][j]);
46            }
47            for (int i=0; i<n; i++){
48                fill(slack,slack+n,INF);
49                while (true){
50                    fill(vx,vx+n,0);
51                    fill(vy,vy+n,0);
52                    if ( DFS(i) ) break;
53                    int d = INF; // long long
54                    for (int j=0; j<n; j++){
55                        if (!vy[j]) d = min(d, slack[j]);
56                    }
57                    for (int j=0; j<n; j++){
58                        if (vx[j]) lx[j] -= d;
59                        if (vy[j]) ly[j] += d;
60                        else slack[j] -= d;
61                    }
62                }
63            }
64        }
65        int res=0;
66        for (int i=0; i<n; i++){
67            res += edge[match[i]][i];
68        }
69        return res;
70    }
71 }graph;

```

6.7 SWmincut

```

1 struct SW{ // O(V^3)
2     static const int MXN = 514;
3     int n,vst[MXN],del[MXN];
4     int edge[MXN][MXN],wei[MXN];
5     void init(int _n){
6         n = _n;
7         MEM(edge);

```

```

8     MEM(del);
9 }
10 void add_edge(int u, int v, int w){
11     edge[u][v] += w;
12     edge[v][u] += w;
13 }
14 void search(int &s, int &t){
15     MEM(vst); MEM(wei);
16     s = t = -1;
17     while (true){
18         int mx=-1, cur=0;
19         for (int i=0; i<n; i++){
20             if (!del[i] && !vst[i] && mx<wei[i])
21                 cur = i, mx = wei[i];
22             if (mx == -1) break;
23             vst[cur] = 1;
24             s = t;
25             t = cur;
26             for (int i=0; i<n; i++){
27                 if (!vst[i] && !del[i]) wei[i] += edge[
28                     cur][i];
29             }
30         }
31         int solve(){
32             int res = 2147483647;
33             for (int i=0,x,y; i<n-1; i++){
34                 search(x,y);
35                 res = min(res,wei[y]);
36                 del[y] = 1;
37                 for (int j=0; j<n; j++){
38                     edge[x][j] = (edge[j][x] += edge[y][j]);
39                 }
40             }
41             return res;
42         }
43     }
44 }graph;

```

7 Geometry

7.1 Circleintersection

```

1 using ld = double;
2 vector<pdd> interCircle(pdd o1, double r1, pdd o2,
3 double r2) {
4     ld d2 = (o1 - o2) * (o1 - o2);
5     ld d = sqrt(d2);
6     if (d > r1+r2) return {};
7     pdd u = 0.5*(o1+o2) + ((r2*r2-r1*r1)/(2*d2))*(o1
8 -o2);
9     double A = sqrt((r1+r2+d) * (r1-r2+d) * (r1+r2-d)
10 *(-r1+r2+d));
11     pdd v = A / (2*d2) * pdd(o1.S-o2.S, -o1.F+o2.F);
12     return {u+v, u-v};
13 }

```

7.2 Fermat's Point

```

1 #define F(n) Fi(i,n)
2 #define Fi(i,n) Fl(i,0,n)
3 #define Fl(i,l,n) for(int i=(l);i<(int)(n);++i)
4 #include <bits/stdc++.h>
5 // #include <ext/pb_ds/assoc_container.hpp>
6 // #include <ext/pb_ds/priority_queue.hpp>
7 using namespace std;
8 // using namespace __gnu_pbds;
9 const double pi = acos(-1), eps = 1e-9;
10 const double st = sin(pi/3), ct = cos(pi/3);
11 struct point {
12     point(double x_ = 0, double y_ = 0): x(x_), y(y_)
13     {}
14     double x, y;
15     inline friend istream& operator>>(istream &is,
16     point &p) {
17         is >> p.x >> p.y;
18     }
19 };

```

```

16     return is;
17 }
18 inline friend ostream& operator<<(ostream &os,
19 const point &p) {
20     os << p.x << ' ' << p.y;
21     return os;
22 }
23 struct line {
24     line(double a_ = 0, double b_ = 0, double c_ = 0):
25         a(a_), b(b_), c(c_) {}
26     double a, b, c;
27     inline double calc(point p) {
28         return a*p.x+b*p.y;
29     }
30     inline double calc(double a, double b, point p) {
31         return a*p.x+b*p.y;
32     }
33     inline double dist2(point a, point b) {
34         return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
35     }
36     inline point rot(point O, point p) {
37         p.x -= O.x, p.y -= O.y;
38         return point(O.x+p.x*ct-p.y*st, O.y+p.x*st+p.y*ct)
39         ;
40     }
41     inline line cln(point a, point b) {
42         return line(a.y-b.y, b.x-a.x, calc(a.y-b.y, b.x-a.
43             x, a));
44     }
45     inline point ntse(line f, line g) {
46         double det = f.a*g.b-g.a*f.b, dx = f.c*g.b-g.c*f.b
47             , dy = f.a*g.c-g.a*f.c;
48         return point(dx/det, dy/det);
49     }
50     inline point fema(point a, point b, point c) {
51         double la = dist2(b, c), lb = dist2(a, c), lc =
52             dist2(a, b);
53         double sa = sqrt(la), sb = sqrt(lb), sc = sqrt(lc)
54             ;
55         if ((lb+lc-la)/(2.0*sb*sc) < -0.5 + eps)
56             return a;
57         if ((la+lc-lb)/(2.0*sa*sc) < -0.5 + eps)
58             return b;
59         if ((la+lb-lc)/(2.0*sa*sb) < -0.5 + eps)
60             return c;
61         point t1 = rot(a, b), t2 = rot(b, a);
62         if (dist2(c, t1) < dist2(c, t2)) swap(t1, t2);
63         point s1 = rot(b, c), s2 = rot(c, b);
64         if (dist2(a, s1) < dist2(a, s2)) swap(s1, s2);
65         return ntse(cln(c, t1), cln(a, s1));
66     }
67 int main() {
68     ios_base::sync_with_stdio(false);
69     cin.tie(NULL);
70     point a, b, c;
71     cin >> a >> b >> c;
72     cout << setprecision(10) << fixed << fema(a, b, c)
73         << '\n';
74 }

```

7.3 Pointoperators

```

1 #define x first
2 #define y second
3 #define cpdd const pdd
4 struct pdd : pair<double, double> {
5     using pair<double, double>::pair;
6     pdd operator + (cpdd &p) const {
7         return {x+p.x, y+p.y};
8     }
9     pdd operator - () const {
10         return {-x, -y};
11     }
12     pdd operator - (cpdd &p) const {
13         return (*this) + (-p);
14     }
15     pdd operator * (double f) const {

```

```

16     return {f*x, f*y};
17 }
18 double operator * (cpdd &p) const {
19     return x*p.x + y*p.y;
20 }
21 };
22 double abs(cpdd &p) { return hypot(p.x, p.y); }
23 double arg(cpdd &p) { return atan2(p.y, p.x); }
24 double cross(cpdd &p, cpdd &q) { return p.x*q.y - p.
    y*q
25 .x; }
26 double cross(cpdd &p, cpdd &q, cpdd &o) { return
    cross(
27 p-o, q-o); }
28 pdd operator * (double f, cpdd &p) { return p*f; }
    //!! Not f*p !!

```

7.4 3DConvexHull

```

1 int flag[MXN][MXN];
2 struct Point{
3     ld x,y,z;
4     Point operator - (const Point &b) const {
5         return (Point){x-b.x,y-b.y,z-b.z};
6     }
7     Point operator * (const ld &b) const {
8         return (Point){x*b,y*b,z*b};
9     }
10    ld len() const { return sqrtl(x*x+y*y+z*z); }
11    ld dot(const Point &a) const {
12        return x*a.x+y*a.y+z*a.z;
13    }
14    Point operator * (const Point &b) const {
15        return (Point){y*b.z-b.y*z,z*b.x-b.z*x,x*b.y
        -b.x*y
16    };
17 }
18 };
19 Point ver(Point a, Point b, Point c) {
20     return (b - a) * (c - a);
21 }
22 vector<Face> convex_hull_3D(const vector<Point> pt)
23 {
24     int n = SZ(pt);
25     REP(i,n) REP(j,n)
26     flag[i][j] = 0;
27     vector<Face> now;
28     now.push_back((Face){0,1,2});
29     now.push_back((Face){2,1,0});
30     int ftop = 0;
31     for (int i=3; i<n; i++){
32         ftop++;
33         vector<Face> next;
34         REP(j, SZ(now)) {
35             Face& f=now[j];
36             ld d=(pt[i]-pt[f.a]).dot(ver(pt[f.a], pt
37 [f.b], pt
38 [f.c]));
39             if (d <= 0) next.push_back(f);
40             int ff = 0;
41             if (d > 0) ff=ftop;
42             else if (d < 0) ff=-ftop;
43             flag[f.a][f.b] = flag[f.b][f.c] = flag[f
44 .c][f.a]
45 = ff;
46         }
47         REP(j, SZ(now)) {
48             Face& f=now[j];
49             if (flag[f.a][f.b] > 0 and flag[f.a][f.b
50 ] != flag
51 [f.b][f.a])
52             next.push_back((Face){f.a,f.b,i});
53             if (flag[f.b][f.c] > 0 and flag[f.b][f.c
54 ] != flag
55 [f.c][f.b])
56             next.push_back((Face){f.b,f.c,i});
57             if (flag[f.c][f.a] > 0 and flag[f.c][f.a
58 ] != flag
59 [f.a][f.c])
60             next.push_back((Face){f.c,f.a,i});
61         }
62         now=next;
63     }
64     return now;
65 }

```

```

54         next.push_back((Face){f.c,f.a,i});
55     }
56     now=next;
57 }
58 return now;
59 }

```

7.5 Halfplaneintersection

```

1 typedef pdd Point;
2 typedef vector<Point> Polygon;
3 typedef pair<Point,Point> Line;
4 #define N 10
5 #define p1 first
6 #define p2 second
7 pdd operator-(const pdd &a,const pdd &b){
8     return mp(a.x-b.x,a.y-b.y);
9 }
10 pdd operator+(const pdd &a,const pdd &b){
11     return mp(a.x+b.x,a.y+b.y);
12 }
13 pdd operator*(const pdd &a,const double &b){
14     return mp(b*a.x,b*a.y);
15 }
16 double cross(Point a, Point b){
17     return a.x * b.y - a.y * b.x;
18 }
19 double cross(Point o, Point a, Point b){
20     return cross(a-o,b-o);
21 }
22 double cross(Line l, Point p){
23     return cross(l.p1, l.p2, p);
24 }
25 double arg(const pdd &a){
26     return atan2(a.y,a.x);
27 }
28 bool parallel(Line l1, Line l2){
29     return cross(l1.p2 - l1.p1, l2.p2 - l2.p1) < 1e
        -8&&cross(l1.p2 - l1.p1, l2.p2 - l2.p1) > -1e
        -8;
30 }
31 Point intersection(Line l1, Line l2){
32     Point& a1 = l1.p1, &a2 = l1.p2;
33     Point& b1 = l2.p1, &b2 = l2.p2;
34     Point a = a2 - a1, b = b2 - b1, s = b1 - a1;
35     return a1 + a * (cross(b, s) / cross(b, a));
36 }
37 bool cmp(Line l1, Line l2){
38     return arg(l1.p2 - l1.p1) < arg(l2.p2 - l2.p1);
39 }
40 Polygon halfplane_intersection(vector<Line> hp){
41     sort(hp.begin(), hp.end(), cmp);
42     int L = 0, R = 0;
43     vector<Line> l(N);
44     vector<Point> p(N);
45     l[R] = hp[0];
46     for (int i=1; i<hp.size(); i++)
47     {
48         while (L < R && cross(hp[i], p[R-1]) < 0) R
49         --;
50         while (L < R && cross(hp[i], p[L]) < 0) L
51         ++;
52         l[++R] = hp[i];
53         if (parallel(l[R-1], hp[i]) &&
54             cross(l[R-1], hp[i].p1) > 0) l[R] = hp[i
55 ];
56         if (L < R) p[R-1] = intersection(l[R], l[R
57 -1]);
58     }
59     while (L < R && cross(l[L], p[R-1]) < 0) R--;
60     if (R-L <= 1) return Polygon();
61     if (L < R) p[R] = intersection(l[L], l[R]);
62     Polygon ch;
63     for (int i=L; i<=R; i++) ch.push_back(p[i]);
64     ch.resize(unique(ch.begin(), ch.end()) - ch.
65 begin());
66     if (ch.size() > 1 && ch.front() == ch.back())
67     ch.pop_back();
68     return ch;
69 }

```



```

64 }
65 double cal(Polygon p){
66     if(p.empty())
67         return 0;
68     p.pb(*p.begin());
69     double ans=0;
70     for(int i=0;i<p.size()-1;i++){
71         ans+=p[i].x*p[i+1].y;
72         ans-=p[i].y*p[i+1].x;
73     }
74     ans/=2;
75     ans=abs(ans);
76     return ans;
77 }

```

7.6 ConvexHull

```

1 sort(p,p+n);
2 pii ans[N];
3 ans[0]=p[0];
4 int k=0;
5 int now=0;
6 for(int yy=0;yy<2;yy++){
7     for(int i=1;i<n;i++){
8         while(now!=k&&(p[i].y-ans[now-1].y)*(ans[now].x-
9             ans[now-1].x)<=(p[i].x-ans[now-1].x)*(ans[now].
10                y-ans[now-1].y)){
11             now--;
12         }
13         ans[++now]=p[i];
14     }
15     k=now;
16     reverse(p,p+n);
17 }

```

7.7 Triangulation

```

1 bool inCircle(pdd a, pdd b, pdd c, pdd d) {
2     b = b - a;
3     c = c - a;
4     d = d - a;
5     if (cross(b, c) < 0) swap(b, c);
6     double m[3][3] = {
7         {b.x, b.y, b*b},
8         {c.x, c.y, c*c},
9         {d.x, d.y, d*d}
10    };
11    double det = m[0][0] * (m[1][1]*m[2][2] - m
12        [1][2]*m
13        [2][1])
14    + m[0][1] * (m[1][2]*m[2][0] - m[1][0]*m
15        [2][2])
16    + m[0][2] * (m[1][0]*m[2][1] - m[1][1]*m
17        [2][0]);
18    return det < 0;
19 }
20 bool intersect(pdd a, pdd b, pdd c, pdd d) {
21     return cross(b, c, a) * cross(b, d, a) < 0 and
22         cross(d, a, c) * cross(d, b, c) < 0;
23 }
24 const double EPS = 1e-12;
25 struct Triangulation {
26     static const int MXN = 1e5+5;
27     int N;
28     vector<int> ord;
29     vector<pdd> pts;
30     set<int> E[MXN];
31     vector<vector<int>> solve(vector<pdd> p) {
32         N = SZ(p);
33         ord.resize(N);
34         for (int i=0; i<N; i++) {
35             E[i].clear();
36             ord[i] = i;
37         }
38         sort(ALL(ord), [&p](int i, int j) {
39             return p[i] < p[j];

```

```

39     });
40     pts.resize(N);
41     for (int i=0; i<N; i++) pts[i] = p[ord[i]];
42     go(0, N);
43     vector<vector<int>> res(N);
44     for (int i=0; i<N; i++) {
45         int o = ord[i];
46         for (auto x: E[i]) {
47             res[o].PB(ord[x]);
48         }
49     }
50     return res;
51 }
52 void add_edge(int u, int v) {
53     E[u].insert(v);
54     E[v].insert(u);
55 }
56 void remove_edge(int u, int v) {
57     E[u].erase(v);
58     E[v].erase(u);
59 }
60 void go(int l, int r) {
61     int n = r - l;
62     if (n <= 3) {
63         for (int i=l; i<r; i++)
64             for (int j=i+1; j<r; j++) add_edge(i, j);
65         return;
66     }
67     int md = (l+r)/2;
68     go(l, md);
69     go(md, r);
70     int il = l, ir = r-1;
71     while (1) {
72         int nx = -1;
73         for (auto i: E[il]) {
74             double cs = cross(pts[il], pts[i],
75 pts[
76         ir]);
77         if (cs > EPS ||
78             (abs(cs) < EPS and abs(pts[i]-pts[
79             ir]) < abs(pts[il]-pts[ir]))) {
80             nx = i;
81             break;
82         }
83     }
84     if (nx != -1) {
85         il = nx;
86         continue;
87     }
88     for (auto i: E[ir]) {
89         double cs = cross(pts[ir], pts[i],
90 pts[
91         il]);
92         if (cs < -EPS ||
93             (abs(cs) < EPS and abs(pts[i]-pts[
94             il]) < abs(pts[ir]-pts[il]))) {
95             nx = i;
96             break;
97         }
98     }
99     if (nx != -1) {
100         ir = nx;
101     } else break;
102 }
103 add_edge(il, ir);
104 while (1) {
105     int nx = -1;
106     bool is2 = false;
107     AcThPaUNpPuAmCmBkCfEsFmMdNoLr 19
108     for (int i: E[il]) {
109         if (cross(pts[il], pts[i], pts[ir])
110 < -
111             EPS and
112             (nx == -1 or inCircle(pts[il], pts[
113             ir], pts[nx], pts[i]))) nx = i;
114     }
115     for (int i: E[ir]) {
116         if (cross(pts[ir], pts[i], pts[il])
117 >
118             EPS and

```



```

116         (nx == -1 or inCircle(pts[il], pts[
117         ir], pts[nx], pts[i]))) nx = i,
118         is2 = 1;
119     }
120     if (nx == -1) break;
121     int a = il, b = ir;
122     if (is2) swap(a, b);
123     for (auto i: E[a]) {
124         if (intersect(pts[a], pts[i], pts[b
125         ],
126         pts[nx])) {
127             remove_edge(a, i);
128         }
129     }
130     if (is2) {
131         add_edge(il, nx);
132         ir = nx;
133     } else {
134         add_edge(ir, nx);
135         il = nx;
136     }
137 }
138 } tri;

```

7.8 K-closet Pair

```

1 #define F(n) Fi(i,n)
2 #define Fi(i,n) Fl(i,0,n)
3 #define Fl(i,l,n) for(int i=(l);i<(int)(n);++i)
4 #include <bits/stdc++.h>
5 // #include <ext/pb_ds/assoc_container.hpp>
6 // #include <ext/pb_ds/priority_queue.hpp>
7 using namespace std;
8 // using namespace __gnu_pbds;
9 typedef long long ll;
10 struct point {
11     point(ll x_ = 0, ll y_ = 0): x(x_), y(y_) {} ll x
12     , y;
13     inline bool operator<(const point &e_) const {
14         return (x != e_.x ? x < e_.x : y < e_.y);
15     }
16     inline friend istream& operator>>(istream &is_,
17     point& e_) {
18         is_ >> e_.x >> e_.y;
19         return is_;
20     }
21 };
22 int k;
23 priority_queue<ll> PQ;
24 inline ll dist2(const point &e1, const point &e2) {
25     ll res = (e1.x-e2.x)*(e1.x-e2.x)+(e1.y-e2.y)*(e1.y
26     -e2.y);
27     PQ.push(res);
28     if (PQ.size() > k) {
29         PQ.pop();
30     }
31     return res;
32 }
33 #define N 500005
34 point p[N];
35 queue<point> Q;
36 ll closet_point(int l, int m, int r, ll delta2) {
37     ll xmid = p[m-1].x;
38     while (!Q.empty()) {
39         Q.pop();
40     }
41     for (int i = l, j = m; i < m; ++i) {
42         if ((p[i].x-xmid)*(p[i].x-xmid) >= delta2) {
43             continue;
44         }
45         while (j < r && p[j].y < p[i].y && (p[j].y-p[i].
46         y)*(p[j].y-p[i].y) < delta2) {
47             if ((p[j].x-xmid)*(p[j].x-xmid) < delta2) {
48                 Q.push(p[j]);
49             }
50             ++j;
51         }
52     }
53 }

```

```

48     while (!Q.empty() && Q.front().y < p[i].y && (Q.
49     front().y-p[i].y)*(Q.front().y-p[i].y) > delta2
50     ) {
51         Q.pop();
52     }
53     while (!Q.empty()) {
54         delta2 = min(delta2, dist2(p[i], Q.front()));
55         Q.pop();
56     }
57     return delta2;
58 }
59 ll find_distance(int l, int r) {
60     if (r - l <= 3000) {
61         ll ans = 0x3f3f3f3f3f3f3f3f;
62         for (int i = l; i < r; ++i)
63             for (int j = i+1; j < r; ++j)
64                 ans = min(ans, dist2(p[i], p[j]));
65         return ans;
66     }
67     int m = (l+r)/2;
68     ll delta2 = min(find_distance(l, m), find_distance
69     (m, r));
70     return min(delta2, closet_point(l, m, r, delta2));
71 }
72 int main() {
73     ios_base::sync_with_stdio(false);
74     cin.tie(NULL);
75     int n;
76     cin >> n >> k;
77     F(n) cin >> p[i];
78     sort(p, p+n);
79     find_distance(0, n);
80     cout << PQ.top() << '\n';
81 }

```

7.9 MCC

```

1 struct Mcc{
2     // return pair of center and r^2
3     static const int MAXN = 1000100;
4     int n;
5     pdd p[MAXN], cen;
6     double r2;
7     void init(int _n, pdd _p[]){
8         n = _n;
9         memcpy(p, _p, sizeof(pdd)*n);
10    }
11    double sqr(double a){ return a*a; }
12    double abs2(pdd a){ return a.a*a.a; }
13    pdd center(pdd p0, pdd p1, pdd p2) {
14        pdd a = p1-p0;
15        pdd b = p2-p0;
16        double c1=abs2(a)*0.5;
17        double c2=abs2(b)*0.5;
18        double d = a.x*b.y-b.x*a.y;
19        double x = p0.x + (c1 * b.y - c2 * a.y) / d;
20        double y = p0.y + (a.x * c2 - b.x * c1) / d;
21        return pdd(x,y);
22    }
23    pair<pdd,double> solve(){
24        random_shuffle(p,p+n);
25        r2=0;
26        for (int i=0; i<n; i++){
27            if (abs2(cen-p[i]) <= r2) continue;
28            cen = p[i];
29            r2 = 0;
30            for (int j=0; j<i; j++){
31                if (abs2(cen-p[j]) <= r2) continue;
32                cen = 0.5 * (p[i]+p[j]);
33                r2 = abs2(cen-p[j]);
34                for (int k=0; k<j; k++){
35                    if (abs2(cen-p[k]) <= r2)
36                        continue;
37                    cen = center(p[i],p[j],p[k]);
38                    r2 = abs2(cen-p[k]);
39                }
40            }
41        }
42    }
43 }

```

```

41         return {cen,r2};
42     }
43 }mcc;

```

```

24         if(n%2 == 0)ans=BigInteger.ZERO.subtract
        (ans);
25         System.out.println(ans);
26     }
27 }
28 }

```

7.10 LineIntersection

```

1 pdd interPnt(pdd p1, pdd p2, pdd q1, pdd q2, bool &
    res)
2 {
3     double f1 = cross(p2, q1, p1);
4     double f2 = -cross(p2, q2, p1);
5     double f = (f1 + f2);
6     if(fabs(f) < EPS) {
7         res = false;
8         return {};
9     }
10    res = true;
11    return (f2 / f) * q1 + (f1 / f) * q2;
12 }

```

7.11 PointToLine

```

1 double cal(const pii &a,const pii &b,const pii &c){
2     int hi=dot(mp(a.x-b.x,a.y-b.y),mp(c.x-b.x,c.y-b.y)
    );
3     if(hi<=0){
4         return dis(a,b);
5     }
6     hi=dot(mp(a.x-c.x,a.y-c.y),mp(b.x-c.x,b.y-c.y));
7     if(hi<=0){
8         return dis(c,a);
9     }
10    if(b.x==c.x)
11    return abs(a.x-b.x);
12    if(b.y==c.y)
13    return abs(a.y-b.y);
14    double B=(double)(b.x-c.x)/(b.y-c.y);
15    double C=(double)(b.y*c.x-b.x*c.y)/(b.y-c.y);
16    return abs(-a.x+B*a.y+C)/sqrt(1+sqr(B));
17 }

```

8 JAVA

8.1 Big Integer

```

1 import java.math.*;
2 import java.io.*;
3 import java.util.*;
4 public class Main{
5     public static void main(String []argv){
6         c[0][0]=BigInteger.ONE;
7         for(int i=1;i<3001;i++){
8             c[i][0]=BigInteger.ONE;
9             c[i][i]=BigInteger.ONE;
10            for(int j=1;j<i;j++)c[i][j]=c[i-1][j].
add(c[i-1][j-1]);
11        }
12        Scanner scanner = new Scanner(System.in);
13        int T = scanner.nextInt();
14        BigInteger x;
15        BigInteger ans;
16        while(T-- > 0){
17            ans = BigInteger.ZERO;
18            int n = scanner.nextInt();
19            for(int i=0;i<n;i++){
20                x = new BigInteger(scanner.next());
21                if(i%2 == 1)ans=ans.subtract(c[n-1][
i].multiply(x));
22            else ans=ans.add(c[n-1][i].multiply(
x));
23        }

```

8.2 Prime

```

1 import java.math.*;
2 import java.io.*;
3 import java.util.*;
4 public class Main{
5     public static void main(String []argv){
6         Scanner scanner = new Scanner(System.in);
7         int T = scanner.nextInt();
8         for (int cs = 0 ; cs < T ; cs++){
9             if (cs != 0) { System.out.println(""); }
10            int a = scanner.nextInt();
11            int b = scanner.nextInt();
12            for (int i = a ; i <= b ; i++) {
13                BigInteger x = BigInteger.valueOf(i)
;
14                if (x.isProbablePrime(5) == true) {
15                    System.out.println(x);
16                }
17            }
18        }
19    }
20 }

```

9 Graph

9.1 MMC

```

1 /* minimum mean cycle 最小平均值環*/
2 const int MXN = 16004;
3 const int MAXE = 1805;
4 const int MAXN = 35;
5 const double inf = 1029384756;
6 const double eps = 1e-6;
7 struct Edge {
8     int v,u;
9     double c;
10 };
11 int n,m,prv[MAXN][MAXN], prve[MAXN][MAXN], vst[MAXN
];
12 Edge e[MAXE];
13 vector<int> edgeID, cycle, rho;
14 double d[MAXN][MAXN];
15 inline void bellman_ford() {
16     for(int i=0; i<n; i++) d[0][i]=0;
17     for(int i=0; i<n; i++) {
18         fill(d[i+1], d[i+1]+n, inf);
19         for(int j=0; j<m; j++) {
20             int v = e[j].v, u = e[j].u;
21             if(d[i][v]<inf && d[i+1][u]>d[i][v]+e[j
].c) {
22                 d[i+1][u] = d[i][v]+e[j].c;
23                 prv[i+1][u] = v;
24                 prve[i+1][u] = j;
25             }
26         }
27     }
28 }
29 double karp_mmc() {
30     // returns inf if no cycle, mmc otherwise
31     double mmc=inf;
32     int st = -1;
33     bellman_ford();
34     for(int i=0; i<n; i++) {
35         double avg=-inf;
36         for(int k=0; k<n; k++) {

```

```

37         if(d[n][i]<inf-eps) avg=max(avg,(d[n][i]
]-d[k][i])
38         /(n-k));
39         else avg=max(avg,inf);
40     }
41     if (avg < mmc) tie(mmc, st) = tie(avg, i);
42 }
43 MEM(vst); edgeID.clear(); cycle.clear(); rho.
clear();
44 for (int i=n; !vst[st]; st=prv[i--][st]) {
45     vst[st]++;
46     edgeID.pb(prve[i][st]);
47     rho.pb(st);
48 }
49 while (vst[st] != 2) {
50     int v = rho.back(); rho.pop_back();
51     cycle.pb(v);
52     vst[v]++;
53 }
54 reverse(edgeID.begin(),edgeID.end());
55 edgeID.resize(cycle.size());
56 return mmc;
57 }

```

9.2 DMST

```

1 struct zhu_liu{
2     static const int MAXN=1100,MAXM=1005005;
3     struct node{
4         int u,v;
5         LL w,tag;
6         node *l,*r;
7         node(int u=0,int v=0,LL w=0):u(u),v(v),w(w),tag
(0),l(0),r(0){}
8         void down(){
9             w+=tag;
10            if(l)l->tag+=tag;
11            if(r)r->tag+=tag;
12            tag=0;
13        }
14    }mem[MAXN];
15    node *pq[MAXN*2],*E[MAXN*2];
16    int st[MAXN*2],id[MAXN*2],m,from[MAXN*2];
17    void init(int n){
18        for(int i=1;i<=n;++i){
19            pq[i]=E[i]=0;
20            st[i]=id[i]=i;
21            from[i]=0;
22        }m=0;
23    }
24    node *merge(node *a,node *b){//skew heap
25        if(!a||!b)return a?a:b;
26        a->down(),b->down();
27        if(b->w<a->w)return merge(b,a);
28        if(b->w==a->w&&b->v<a->v)return merge(b,a);//
29        swap(a->l,a->r);
30        a->l=merge(b,a->l);
31        return a;
32    }
33    void add_edge(int u,int v,LL w){
34        if(u!=v)pq[v]=merge(pq[v],&(mem[m++]=node(u,v,w)
));
35    }
36    int find(int x,int *st){
37        return st[x]==x?x:st[x]=find(st[x],st);
38    }
39    LL build(int root,int n){
40        LL ans=0;int N=n,all=n;
41        for(int i=1;i<=N;++i){
42            if(i==root||!pq[i])continue;
43            while(pq[i]){
44                pq[i]->down(),E[i]=pq[i];
45                pq[i]=merge(pq[i]->l,pq[i]->r);
46                if(find(E[i]->u,id)!=find(i,id))break;
47            }
48            if(find(E[i]->u,id)==find(i,id))continue;
49            from[E[i]->v]=E[i]->u;
50            ans+=E[i]->w;
51            if(find(E[i]->u,st)==find(i,st)){

```

```

52                if(pq[i])pq[i]->tag-=E[i]->w;
53                pq[++N]=pq[i],id[N]=N;
54                for(int u=find(E[i]->u,id);u!=i;u=find(E[u]
]->u,id)){
55                    if(pq[u])pq[u]->tag-=E[u]->w;
56                    id[find(u,id)]=N;
57                    pq[N]=merge(pq[N],pq[u]);
58                }
59                st[N]=find(i,st);
60                id[find(i,id)]=N;
61            }else st[find(i,st)]=find(E[i]->u,st),--all;
62        }
63        return all==1?ans:-1;//圖不連通就無解
64    }
65 }MST;

```

9.3 SCC

```

1 struct Scc{
2     int n, nScc, vst[MAXN], bln[MAXN];
3     vector<int> E[MAXN], rE[MAXN], vec;
4     void init(int _n){
5         n = _n;
6         for (int i=0; i<MAXN; i++){
7             E[i].clear();
8             rE[i].clear();
9         }
10    }
11    void add_edge(int u, int v){
12        E[u].pb(v);
13        rE[v].pb(u);
14    }
15    void DFS(int u){
16        vst[u]=1;
17        for (auto v : E[u])
18            if (!vst[v]) DFS(v);
19        vec.pb(u);
20    }
21    void rDFS(int u){
22        vst[u] = 1;
23        bln[u] = nScc;
24        for (auto v : rE[u])
25            if (!vst[v]) rDFS(v);
26    }
27    void solve(){
28        nScc = 0;
29        vec.clear();
30        MEM(vst);
31        for (int i=0; i<n; i++)
32            if (!vst[i]) DFS(i);
33        reverse(vec.begin(),vec.end());
34        FZ(vst);
35        for (auto v : vec){
36            if (!vst[v]){
37                rDFS(v);
38                nScc++;
39            }
40        }
41    }
42 };

```

9.4 GeneralGraphMaximunValueMatch

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 //from vfleaking
4 //自己進行一些進行一些小修改
5 #define INF INT_MAX
6 #define MAXN 400
7 struct edge{
8     int u,v,w;
9     edge(){}
10    edge(int u,int v,int w):u(u),v(v),w(w){}
11 };
12 int n,n_x;
13 edge g[MAXN*2+1][MAXN*2+1];

```

```

14 int lab[MAXN*2+1];
15 int match[MAXN*2+1], slack[MAXN*2+1], st[MAXN*2+1], pa[
    MAXN*2+1];
16 int flower_from[MAXN*2+1][MAXN+1], S[MAXN*2+1], vis[
    MAXN*2+1];
17 vector<int> flower[MAXN*2+1];
18 queue<int> q;
19 inline int e_delta(const edge &e){ // does not work
    inside blossoms
20     return lab[e.u]+lab[e.v]-g[e.u][e.v].w*2;
21 }
22 inline void update_slack(int u, int x){
23     if(!slack[x] || e_delta(g[u][x])<e_delta(g[slack[x]
        ][x]))slack[x]=u;
24 }
25 inline void set_slack(int x){
26     slack[x]=0;
27     for(int u=1; u<=n; ++u)
28         if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)
29             update_slack(u, x);
30 }
31 void q_push(int x){
32     if(x<=n)q.push(x);
33     else for(size_t i=0; i<flower[x].size(); i++)q.push(
        flower[x][i]);
34 }
35 inline void set_st(int x, int b){
36     st[x]=b;
37     if(x>n)for(size_t i=0; i<flower[x].size(); ++i)
38         set_st(flower[x][i], b);
39 }
40 inline int get_pr(int b, int xr){
41     int pr=find(flower[b].begin(), flower[b].end(), xr)-
        flower[b].begin();
42     if(pr%2==1){//檢查他在前一層圖是奇點還是偶點
43         reverse(flower[b].begin()+1, flower[b].end());
44         return (int)flower[b].size()-pr;
45     }else return pr;
46 }
47 inline void set_match(int u, int v){
48     match[u]=g[u][v].v;
49     if(u>n){
50         edge e=g[u][v];
51         int xr=flower_from[u][e.u], pr=get_pr(u, xr);
52         for(int i=0; i<pr; ++i)set_match(flower[u][i],
            flower[u][i+1]);
53         set_match(xr, v);
54         rotate(flower[u].begin(), flower[u].begin()+pr,
            flower[u].end());
55     }
56 }
57 inline void augment(int u, int v){
58     for(;;){
59         int xnv=st[match[u]];
60         set_match(u, v);
61         if(!xnv)return;
62         set_match(xnv, st[pa[xnv]]);
63         u=st[pa[xnv]], v=xnv;
64     }
65 }
66 inline int get_lca(int u, int v){
67     static int t=0;
68     for(++t; u!=v; swap(u, v)){
69         if(u==0)continue;
70         if(vis[u]==t)return u;
71         vis[u]=t; //這種方法可以不用清空v陣列
72         u=st[match[u]];
73         if(u)u=st[pa[u]];
74     }
75     return 0;
76 }
77 inline void add_blossom(int u, int lca, int v){
78     int b=n+1;
79     while(b<=n&&st[b])++b;
80     if(b>n_x)++n_x;
81     lab[b]=0, S[b]=0;
82     match[b]=match[lca];
83     flower[b].clear();
84     flower[b].push_back(lca);
85     for(int x=u, y; x!=lca; x=st[pa[y]])
86         flower[b].push_back(x), flower[b].push_back(y=st[
            match[x]]), q_push(y);
87     reverse(flower[b].begin()+1, flower[b].end());
88     for(int x=v, y; x!=lca; x=st[pa[y]])
89         flower[b].push_back(x), flower[b].push_back(y=st[
            match[x]]), q_push(y);
89     set_st(b, b);
90     for(int x=1; x<=n_x; ++x)g[b][x].w=g[x][b].w=0;
91     for(int x=1; x<=n; ++x)flower_from[b][x]=0;
92     for(size_t i=0; i<flower[b].size(); ++i){
93         int xs=flower[b][i];
94         for(int x=1; x<=n_x; ++x)
95             if(g[b][x].w==0 || e_delta(g[xs][x])<e_delta(g[b]
                [x]))
96                 g[b][x]=g[xs][x], g[x][b]=g[x][xs];
97         for(int x=1; x<=n; ++x)
98             if(flower_from[xs][x])flower_from[b][x]=xs;
99     }
100     set_slack(b);
101 }
102 inline void expand_blossom(int b){ // S[b] == 1
103     for(size_t i=0; i<flower[b].size(); ++i)
104         set_st(flower[b][i], flower[b][i]);
105     int xr=flower_from[b][g[b][pa[b]].u], pr=get_pr(b,
        xr);
106     for(int i=0; i<pr; i+=2){
107         int xs=flower[b][i], xns=flower[b][i+1];
108         pa[xs]=g[xns][xs].u;
109         S[xs]=1, S[xns]=0;
110         slack[xs]=0, set_slack(xns);
111         q_push(xns);
112     }
113     S[xr]=1, pa[xr]=pa[b];
114     for(size_t i=pr+1; i<flower[b].size(); ++i){
115         int xs=flower[b][i];
116         S[xs]=-1, set_slack(xs);
117     }
118     st[b]=0;
119 }
120 inline bool on_found_edge(const edge &e){
121     int u=st[e.u], v=st[e.v];
122     if(S[v]==-1){
123         pa[v]=e.u, S[v]=1;
124         int nu=st[match[v]];
125         slack[v]=slack[nu]=0;
126         S[nu]=0, q_push(nu);
127     }else if(S[v]==0){
128         int lca=get_lca(u, v);
129         if(!lca)return augment(u, v), augment(v, u), true;
130         else add_blossom(u, lca, v);
131     }
132     return false;
133 }
134 inline bool matching(){
135     memset(S+1, -1, sizeof(int)*n_x);
136     memset(slack+1, 0, sizeof(int)*n_x);
137     q=queue<int>();
138     for(int x=1; x<=n_x; ++x)
139         if(st[x]==x&&!match[x])pa[x]=0, S[x]=0, q_push(x);
140     if(q.empty())return false;
141     for(;;){
142         while(q.size()){
143             int u=q.front(); q.pop();
144             if(S[st[u]]==1)continue;
145             for(int v=1; v<=n; ++v)
146                 if(g[u][v].w>0&&st[u]!=st[v]){
147                     if(e_delta(g[u][v])==0){
148                         if(on_found_edge(g[u][v]))return true;
149                     }else update_slack(u, st[v]);
150                 }
151         }
152         int d=INF;
153         for(int b=n+1; b<=n_x; ++b)
154             if(st[b]==b&&S[b]==1)d=min(d, lab[b]/2);
155         for(int x=1; x<=n_x; ++x)
156             if(st[x]==x&&slack[x]){
157                 if(S[x]==-1)d=min(d, e_delta(g[slack[x]][x]));
158             }else if(S[x]==0)d=min(d, e_delta(g[slack[x]][
                x])/2);
159         for(int u=1; u<=n; ++u){
160             if(S[st[u]]==0){
161                 if(lab[u]<=d)return 0;
162             }

```

```

163     lab[u] -= d;
164 } else if (S[st[u]] == 1) lab[u] += d;
165 }
166 for (int b = n + 1; b <= n_x; ++b)
167     if (st[b] == b) {
168         if (S[st[b]] == 0) lab[b] += d * 2;
169         else if (S[st[b]] == 1) lab[b] -= d * 2;
170     }
171 q = queue<int>();
172 for (int x = 1; x <= n_x; ++x)
173     if (st[x] == x && slack[x] && st[slack[x]] != x &&
174         e_delta(g[slack[x]][x]) == 0)
175         if (on_found_edge(g[slack[x]][x])) return true;
176 for (int b = n + 1; b <= n_x; ++b)
177     if (st[b] == b && S[b] == 1 && lab[b] == 0) expand_blossom(b);
178 }
179 return false;
180 }
181 inline pair<long long, int> weight_blossom() {
182     memset(match + 1, 0, sizeof(int) * n);
183     n_x = n;
184     int n_matches = 0;
185     long long tot_weight = 0;
186     for (int u = 0; u <= n; ++u) st[u] = u, flower[u].clear();
187     int w_max = 0;
188     for (int u = 1; u <= n; ++u)
189         for (int v = 1; v <= n; ++v) {
190             flower_from[u][v] = (u == v ? u : 0);
191             w_max = max(w_max, g[u][v].w);
192         }
193     for (int u = 1; u <= n; ++u) lab[u] = w_max;
194     while (matching()) ++n_matches;
195     for (int u = 1; u <= n; ++u)
196         if (match[u] && match[u] < u)
197             tot_weight += g[u][match[u]].w;
198     return make_pair(tot_weight, n_matches);
199 }
200 inline void init_weight_graph() {
201     for (int u = 1; u <= n; ++u)
202         for (int v = 1; v <= n; ++v)
203             g[u][v] = edge(u, v, 0);
204 }
205 int main() {
206     int m;
207     scanf("%d", &m);
208     init_weight_graph();
209     for (int i = 0; i < m; ++i) {
210         int u, v, w;
211         scanf("%d%d%d", &u, &v, &w);
212         g[u][v].w = g[v][u].w = w;
213     }
214     printf("%lld\n", weight_blossom().first);
215     for (int u = 1; u <= n; ++u) printf("%d ", match[u]); puts("");
216     return 0;
217 }

```

9.5 Stable Marriage

```

1 #define F(n) Fi(i, n)
2 #define Fi(i, n) Fl(i, 0, n)
3 #define Fl(i, l, n) for(int i = l ; i < n ; ++i)
4 #include <bits/stdc++.h>
5 using namespace std;
6 int D, quota[205], weight[205][5];
7 int S, scoretoDep[12005][205], score[5];
8 int P, prefer[12005][85], iter[12005];
9 int ans[12005];
10 typedef pair<int, int> PII;
11 map<int, int> samescore[205];
12 typedef priority_queue<PII, vector<PII>, greater<PII>> QQQ;
13 QQQ pri[205];
14 void check(int d) {
15     PII t = pri[d].top();
16     int v;

```

```

17     if (pri[d].size() - samescore[d][t.first] + 1 <=
18         quota[d]) return;
19     while (pri[d].top().first == t.first) {
20         v = pri[d].top().second;
21         ans[v] = -1;
22         --samescore[d][t.first];
23         pri[d].pop();
24     }
25 void push(int s, int d) {
26     if (pri[d].size() < quota[d]) {
27         pri[d].push(PII(scoretoDep[s][d], s));
28         ans[s] = d;
29         ++samescore[s][scoretoDep[s][d]];
30     } else if (scoretoDep[s][d] >= pri[d].top().first) {
31         pri[d].push(PII(scoretoDep[s][d], s));
32         ans[s] = d;
33         ++samescore[s][scoretoDep[s][d]];
34         check(d);
35     }
36 }
37 void f() {
38     int over;
39     while (true) {
40         over = 1;
41         Fi (q, S) {
42             if (ans[q] != -1 || iter[q] >= P) continue;
43             push(q, prefer[q][iter[q]++]);
44             over = 0;
45         }
46         if (over) break;
47     }
48 }
49 main() {
50     ios::sync_with_stdio(false);
51     cin.tie(NULL);
52     int sadmit, stof, dexceed, dfew;
53     while (cin >> D, D) { // Beware of the input
54         format or judge may troll us.
55         sadmit = stof = dexceed = dfew = 0;
56         memset(iter, 0, sizeof(iter));
57         memset(ans, 0, sizeof(ans));
58         Fi (q, 205) {
59             pri[q] = QQQ();
60             samescore[q].clear();
61         }
62         cin >> S >> P;
63         Fi (q, D) {
64             cin >> quota[q];
65             Fi (w, S) cin >> weight[q][w];
66         }
67         Fi (q, S) {
68             Fi (w, S) cin >> score[w];
69             Fi (q, D) {
70                 scoretoDep[q][w] = 0;
71                 F (S) scoretoDep[q][w] += weight[w][i] *
72                     score[i];
73             }
74         }
75         Fi (q, S) Fi (w, P) {
76             cin >> prefer[q][w];
77             --prefer[q][w];
78         }
79         f();
80         Fi (q, D) sadmit += pri[q].size();
81         Fi (q, S) if (ans[q] == prefer[q][0]) ++stof;
82         Fi (q, D) if (pri[q].size() > quota[q]) ++
83             dexceed;
84         Fi (q, D) if (pri[q].size() < quota[q]) ++dfew;
85         cout << sadmit << ' ' << stof << ' ' << dexceed
86             << ' ' << dfew << '\n';
87     }
88 }

```

9.6 BCCVertex

```

1 const int MXN = 16004;
2 struct BccVertex {

```



```

3   int n,nScc,step,dfn[MXN],low[MXN];
4   vector<int> E[MXN],sccv[MXN];
5   int top,stk[MXN];
6   void init(int _n) {
7       n = _n;
8       nScc = step = 0;
9       for (int i=0; i<n; i++) E[i].clear();
10  }
11  void add_edge(int u, int v) {
12      E[u].pb(v);
13      E[v].pb(u);
14  }
15  void DFS(int u, int f) {
16      dfn[u] = low[u] = step++;
17      stk[top++] = u;
18      for (auto v:E[u]) {
19          if (v == f) continue;
20          if (dfn[v] == -1) {
21              DFS(v,u);
22              low[u] = min(low[u], low[v]);
23              if (low[v] >= dfn[u]) {
24                  int z;
25                  sccv[nScc].clear();
26                  do {
27                      z = stk[--top];
28                      sccv[nScc].pb(z);
29                  } while (z != v);
30                  sccv[nScc].pb(u);
31                  nScc++;
32              }
33              else {
34                  low[u] = min(low[u],dfn[v]);
35              }
36          }
37      }
38      vector<vector<int>> solve() {
39          vector<vector<int>> res;
40          for (int i=0; i<n; i++) {
41              dfn[i] = low[i] = -1;
42          }
43          for (int i=0; i<n; i++) {
44              if (dfn[i] == -1) {
45                  top = 0;
46                  DFS(i,i);
47              }
48          }
49          for(int i=0;i<nScc;i++) res.pb(sccv[i]);
50          return res;
51      }
52  }graph;

```

9.7 MaxClique

```

1  class MaxClique {
2  public:
3      static const int MV = 210;
4      int V;
5      int el[MV][MV/30+1];
6      int dp[MV];
7      int ans;
8      int s[MV][MV/30+1];
9      vector<int> sol;
10     void init(int v) {
11         V = v; ans = 0;
12         MEMS(el); MEMS(dp);
13     }
14     /* Zero Base */
15     void addEdge(int u, int v) {
16         if(u > v) swap(u, v);
17         if(u == v) return;
18         el[u][v/32] |= (1<<(v%32));
19     }
20     bool dfs(int v, int k) {
21         int c = 0, d = 0;
22         for(int i=0; i<(V+31)/32; i++) {
23             s[k][i] = el[v][i];
24             if(k != 1) s[k][i] &= s[k-1][i];
25             c += __builtin_popcount(s[k][i]);
26         }

```

```

27         if(c == 0) {
28             if(k > ans) {
29                 ans = k;
30                 sol.clear();
31                 sol.push_back(v);
32                 return 1;
33             }
34             return 0;
35         }
36         for(int i=0; i<(V+31)/32; i++) {
37             for(int a = s[k][i]; a ; d++) {
38                 if(k + (c-d) <= ans) return 0;
39                 int lb = a&(-a), lg = 0;
40                 a ^= lb;
41                 while(lb!=1) {
42                     lb = (unsigned int)(lb) >> 1;
43                     lg ++;
44                 }
45                 int u = i*32 + lg;
46                 if(k + dp[u] <= ans) return 0;
47                 if(dfs(u, k+1)) {
48                     sol.push_back(v);
49                     return 1;
50                 }
51             }
52         }
53         return 0;
54     }
55     int solve() {
56         for(int i=V-1; i>=0; i--) {
57             dfs(i, 1);
58             dp[i] = ans;
59         }
60         return ans;
61     }
62 };

```

9.8 BCCedge

```

1  vector<vector<int>> v;
2  int vis[100005],lwn[100005];
3  vector<int> stk;
4  int f[100005];
5  int bln[100005];
6  int Find(int a){
7      if(bln[a]==a)return a;
8      return bln[a]=Find(bln[a]);
9  }
10 int t;
11 void dfs(int a,int p){
12     stk.pb(a);
13     bln[a]=a;
14     vis[a]=lwn[a]=++t;
15     int cnt=0;
16     for(int i=0;i<v[a].size();i++){
17         int x=v[a][i];
18         if(x!=p||cnt==1){
19             if(vis[x]==0){
20                 dfs(x,a);
21                 if(lwn[x]>vis[a]){
22                     int fa=Find(x);
23                     f[x]=Find(a);
24                     while(stk.back()!=x){
25                         bln[stk.back()]=fa;
26                         stk.pop_back();
27                     }
28                     bln[stk.back()]=fa;
29                     stk.pop_back();
30                 }
31                 lwn[a]=min(lwn[a],lwn[x]);
32             }
33             else{
34                 lwn[a]=min(lwn[a],vis[x]);
35             }
36         }
37         else{
38             cnt++;
39         }
40     }

```


41 }

10 無權邊的生成樹個數 Kirchhoff's Theorem

1. 定義 $n \times m$ 矩陣 $E = (a_{i,j})$, n 為點數, m 為邊數, 若 i 點在 j 邊上, i 為小點 $a_{i,j} = 1$, i 為大點 $a_{i,j} = -1$, 否則 $a_{i,j} = 0$ 。
(證明省略)

4. 令 $E(E^T) = Q$, 他是一種有負號的 kirchhoff 的矩陣, 取 Q 的子矩陣即為 $F(F^T)$
結論: 做 Q 取子矩陣算 \det 即為所求。(除去第一行第一列 by mz)

11 monge

$$i \leq i' < j \leq j'$$

$$m(i, j) + m(i', j') \leq m(i', j) + m(i, j')$$

$$k(i, j-1) \leq k(i, j) \leq k(i+1, j)$$

12 四心

$$\frac{sa \cdot A + sb \cdot B + sc \cdot C}{sa + sb + sc}$$

外心 $\sin 2A : \sin 2B : \sin 2C$

內心 $\sin A : \sin B : \sin C$

垂心 $\tan A : \tan B : \tan C$

重心 $1 : 1 : 1$

13 Runge-Kutta

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = f(t_n, y_n)$$

$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$$

$$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$$

$$k_4 = f(t_n + h, y_n + hk_3)$$

14 Householder Matrix

$$I - 2 \frac{vv^T}{v^T v}$$