# Contents


**1 Basic** 1
- 1.1 .vimrc . . . 1
- 1.2 Increase Stack Size . . . 1
- 1.3 digitDP . . . 1
- 1.4 DP(convex hull optimization) . . . 2
- 1.5 simulated annealing . . . 2

**2 Graph** 3
- 2.1 HLD . . . 3
- 2.2 Hungarian . . . 4
- 2.3 KM . . . 5
- 2.4 Bi-vertex-connected Subgraph . . . 5
- 2.5 Bi-edge-connected Subgraph . . . 6
- 2.6 SCC . . . 7
- 2.7 Steiner Tree( PECaveros ) . . . 7
- 2.8 Edmond's Matching Algorithm . . . 7
- 2.9 Tree Decomposition . . . 7
- 2.10 Tree Longest Path . . . 9

**3 Flow** 9
- 3.1 Dinic Maxflow . . . 9

**4 Data Structure** 10
- 4.1 Disjoint Set . . . 10
- 4.2 Djs + Seg . . . 11
- 4.3 Sparse Table . . . 11
- 4.4 Link Cut Tree . . . 12
- 4.5 Treap . . . 12

**5 Math** 13
- 5.1 Prime Table . . . 13
- 5.2 Miller Rabin Prime Test . . . 13
- 5.3 Extended Euclidean Algorithm . . . 13
- 5.4 Gauss Elimination . . . 14
- 5.5 FFT . . . 14
- 5.6 NNT . . . 15
- 5.7 Big Number . . . 16

**6 string** 18
- 6.1 Palindromic Tree . . . 18
- 6.2 Suffix Array . . . 19
- 6.3 Longest Palindromic Substring . . . 19

**7 geometry** 19
- 7.1 Point Class . . . 19
- 7.2 Intersection of Circles/Lines/Segments . . . 19
- 7.3 Convex Hull . . . 20
- 7.4 Half Plane Intersection . . . 20


# 1 Basic

## 1.1 .vimrc

```
imap jj <Esc>

sy on
se sw=4 ts=4 sts=4 et nu sc hls cc=69
filet plugin indent on
nm <F5> :!"./%<"<CR>
nm <F6> :!"./%<" < input.txt<CR>
au FileType cpp no <F9> :!g++ % -o
 \ %< -std=c++14 -O3 -Wall -Wextra
 \ -Wshadow -Wno-unused-result<CR>
no <expr> <silent> <Home> col('.') ==
 \ match(getline('.'),'\S') + 1
 \ ? '0' : '^'
im <silent> <Home> <C-O><Home>
```

## 1.2 Increase Stack Size

```
//stack resize
asm( "mov %0,%%esp\n" ::"g"(mem+10000000) );
//change esp to rsp if 64-bit system

//stack resize (linux)
#include <sys/resource.h>
void increase_stack_size() {
  const rlim_t ks = 64*1024*1024;
  struct rlimit rl;
  int res=getrlimit(RLIMIT_STACK, &rl);
  if(res==0){
    if(rl.rlim_cur<ks){
      rl.rlim_cur=ks;
      res=setrlimit(RLIMIT_STACK, &rl);
    }
  }
}
```

## 1.3 digitDP

```
/**
简介
顾名思义，所谓的数位DP就是按照数字的个，十，百，千……
    位数进行的DP。
数位DP的题目有着非常明显的性质：
    询问[l,r]的区间内，有多少的数字满足某个性质
做法根据前缀和的思想，求出[0,l-1]和[0,r]中满足性质的数
    的个数，然后相减即可。

算法核心

LL dfs(int x,int pre,int bo,int limit);
一般需要以上参数（当然具体情况具体分析）。

    x表示当前的数位（一般都是从高位到低位）
    pre表示前一位的数字
    bo可以表示一些附加条件：是否有前项0，是否当前已经符
        合条件……
    limit这个很重要！它表示当前数位是否受到上一位的限
        制，比较抽象，举例说明
    如果上限是135，前两位已经是1和3了，现在到了个位，个
        位只能是5以下的数字

注：如果当前受限，不能够记忆化，也不能返回记忆化的结果
为了避免多次调用时 每次上限不同 而导致的错
**/
//http://acm.csie.org/ntujudge/view_code.php?id=106844
// Multiples
LL x;
int digit[100];
LL ten_pow[ 15 ];
bool ava[15];
LL dp[15][2][1000000];
LL dfs(int len, LL mod, bool bo, bool limit) {
```

```
    if( len < 0 ) return mod == 0;
    if( !limit && dp[len][bo][mod] != -1 ) return dp[
        len][bo][mod];
    int up = limit? digit[len] : 9;
    LL ret = 0;
    for(int i = 0; i <= up; i++) if( ava[i] || (i==0&&
        bo) ) {
        ret += dfs( len-1, (mod+ten_pow[len]*i)%x, bo
            &&(!i), limit&&(i==up) );
    }
    if( !limit ) dp[len][bo][mod] = ret;
    return ret;
}
LL solve(LL num) {
    int len = 0; digit[0] = 0;
    while( num ) {
        digit[len++] = num%10;
        num /= 10;
    }
    return dfs(len-1, 0,1, 1);
}
bool check(LL num) {
    while( num ) {
        if ( !ava[ num%10 ] ) return false;
        num /= 10;
    }
    return true;
}
int main() {
    LL A, B;
    cin>>x>>A>>B;
    ten_pow[0] = 1;
    mem( dp, -1);
    for(int i = 1; i < 15; i++)
        ten_pow[i] = (ten_pow[i-1]*10)%x;
    string dig; cin>>dig;
    mem(ava, false);
    for(char c : dig) ava[ c-'0' ] = 1;

    if( x <= 1000000 ) {
        cout<< solve(B) - solve(A-1) <<endl;
    }else {
        LL ans = 0;
        LL cur = 0;
        while( cur < A ) cur += x;
        while( cur <= B ) {
            if( check(cur) ) ans++;
            cur += x;
        }
        cout<<ans<<endl;
    }
}
```

## 1.4 DP(convex hull optimization)

```
//http://codeforces.com/contest/311/problem/B
struct line{
    LL slope, inter;
    LL value(LL x) { return x*slope + inter; }
};
bool check(line x, line y, line z) {
    return (z.slope - y.slope) * (z.inter - x.inter )
        >=
            ( z.slope - x.slope) * (z.inter - y.inter) ;
}

#define maxn 100005
int n, m, p;
LL a[maxn], d[maxn], dp[101][maxn], s[maxn];
int main() {
    cin>> n >> m >> p;
    for(int i = 2; i<=n; ++i) {
        d[i] = getint();
        d[i] += d[i-1];
    }
    for(int i = 1; i<=m; ++i) {
        int h; scanf("%d %lld", &h, a+i);
        a[i] -= d[h];
    }
    sort( a+1, a+1+m );
    for(int i=1;i<=m;i++) s[i] = a[i]+s[i-1];
    //start dp
    for(int i=1; i<=p;i++) {
        if( i == 1 ) {
            for(int j=1;j<=m;j++) dp[i][j] = j*a[j] - s
                [j];
        }else {
            deque<line> dq;
            dq.pb( {0, 0} );
            for(int j=1;j<=m;j++) {
                while( dq.size() >= 2 && dq[0].value(-a
                    [j]) > dq[1].value(-a[j]) ) dq.
                    pop_front();
                dp[i][j] = dq[0].value(-a[j]);

                line newline{ j, dp[i-1][j]+s[j] };
                while( dq.size() >= 2 && check(dq[dq.
                    size()-2], dq.back(), newline) ) dq
                    .pop_back();
                dq.pb( newline );
            /*
                if( i==1 ) {
                    dp[i][j] = j*a[j] - s[j];
                }else {
                    LL mn = 0;
                    for(int k = 1; k < j; k++) {
                        mn = min(mn, dp[i-1][k] + s[k]
                            - a[j]*k );
                    }
                    dp[i][j] = mn + a[j]*j-s[j];
                    // apply convex hull optimization
                }
            */
                dp[i][j] += a[j]*j - s[j];
            }
        }
    }
    cout << dp[p][m] << endl;
}
```

## 1.5 simulated annealing

```
//http://mikucode.blogspot.tw/2015/03/algorithm.html
//尋找和所有點距離和最小的點
#include <cstdio>
#include <cstdlib>
#include <cmath>
#define F(n) Fi(i,n)
#define Fi(i,n) for(int i=0;i<n;i++)
#define N 1010
using namespace std;
int X[N],Y[N],n;
inline double pow2(double x){
    return x*x;
}
double check(double x,double y){
    double ans=0;
    F(n)ans+=sqrt(pow2(x-X[i])+pow2(y-Y[i]));
    return ans;
}
int main(){

    while(~scanf("%d",&n) ) {
        F(n) scanf("%d%d",X+i,Y+i);
        double x=0,y=0,tx,ty,tans,l=10000,ans;
        ans=check(x,y);
        while(l>1e-4) {
            int tmp=rand();
            tx=x+l*cos(tmp);ty=y+l*sin(tmp);
            tans=check(tx,ty);
            if(tans<ans) ans=tans,x=tx,y=ty;
            else l*=0.9;
        }
        printf("%.9f\n",2*ans);
    }
}
```

```
//尋找兩個點使他們跟給定的四個點最小生成樹最小
#include <cstdio>
#include <cstdlib>
```

```
#include <cmath>
#include <algorithm>
#define F(n) Fi(i,n)
#define Fi(i,n) Fl(i,0,n)
#define Fl(i,l,n) for(int i=l;i<n;i++)
#define N 10
using namespace std;
int X[N],Y[N],n,F[N],e;
struct E{
    int a,b;
    double c;
}G[N*2];
struct V{
    double x,y;
    V operator+(double l){
        int tmp=rand();
        return (V){x+l*cos(tmp),y+l*sin(tmp)};
    }
}v[N];
int find(int x){
    return x==F[x]?x:F[x]=find(F[x]);
}
inline double pow2(double x){
    return x*x;
}
double check(V s1,V s2){
    double ans=0;
    e=0;v[4]=s1,v[5]=s2;
    F(5)Fl(j,i+1,6)
        G[e++]=(E){i,j,sqrt(pow2(v[i].x-v[j].x)+pow2(v[
            i].y-v[j].y))};
    F(6)F[i]=i;
    sort(G,G+e,[](E a,E b){return a.c<b.c;});
    F(e){
        if(find(G[i].a)!=find(G[i].b)){
            ans+=G[i].c;
            F[find(G[i].a)]=find(G[i].b);
        }
    }
    return ans;
}
int main(){
    scanf("%d",&n);
    while(n--){
        F(4) scanf("%lf%lf",&v[i].x,&v[i].y);
        double ttans,tans,ans,l1=10000,l2;
        V s1=(V){0,0},s2=(V){0,0},ts1,ts2,tmp;
        ans=check(s1,s2);
        while(l1>1e-3){
            l2=10000;
            ts1=s1+l1;
            tans=check(ts1,s2);
            tmp=s2;
            while(l2>1e-3){
                ts2=s2+l2;
                ttans=check(ts1,ts2);
                if(ttans<tans)tans=ttans,s2=ts2;
                else l2*=0.9;
            }
            if(tans<ans)ans=tans,s1=ts1;
            else l1*=0.9,s2=tmp;
        }
        printf("%f\n",2*ans);
    }
}
```

# 2 Graph

## 2.1 HLD

```
//we can reference the problem Greatest graph
///http://acm.csie.org/ntujudge/problemdata/2582.pdf
//this template operate on edges
#define maxn 100005
struct segment_tree{
    #define right(x) x << 1 | 1
    #define left(x) x << 1
    int* arr;
    int m[4*maxn];
    int tag[4*maxn];
    const int inf = 1e9;

    void init() {
        //memset(tag, -1, sizeof(tag));
        fill(tag, tag+4*maxn, inf);
    }
    void pull(int ind) {
        m[ind] = min(m[right(ind)] ,m[left(ind)]);
    };
    void push(int ind) {
        if(tag[ind] != inf) {
            tag[left(ind)] = min(tag[left(ind)], tag[
                ind]);
            tag[right(ind)] = min(tag[right(ind)], tag[
                ind]);
            m[left(ind)] = min( m[left(ind)], tag[left(
                ind)]);
            m[right(ind)] = min( m[right(ind)], tag[
                right(ind)]);
            tag[ind] = inf;
        }
    }
    /// root => 1
    void build(int ind, int l, int r) {

        if( r - l == 1) {
            m[ind] =  arr[l];
            return;
        }
        int mid = (l+r)>>1;
        build( left(ind), l, mid );
        build( right(ind), mid, r );
        pull(ind);
    }
    int query_min(int ind, int L, int R, int ql, int qr
        ) {
        if( L >= qr || R <= ql ) return 1e9;
        if( R <= qr && L >= ql ) {
            return m[ind];
        }
        push(ind);
        int mid = (L+R)>>1;
        return min( query_min(left(ind), L, mid, ql, qr
            ), query_min(right(ind), mid, R, ql, qr));
    }
    void modify(int ind, int L, int R, int ql, int qr,
        int x) {
        if( L >= qr || R <= ql ) return;
        if( R <= qr && L >= ql ) {

            m[ind] = min(m[ind], x);
            tag[ind] = min(tag[ind], x);

            return;
        }
        push(ind);
        int mid = (L+R)>>1;
        modify(left(ind), L, mid, ql, qr, x);
        modify(right(ind), mid, R, ql, qr, x);
        pull(ind);
    }
};

int seg_arr[maxn];
struct Tree{
    segment_tree seg;
    int n;
    struct Edge { int u, v, c; };
    vector<Edge> e;
    void addEdge(int x, int y, int c) {
        G[x].pb( SZ(e) );
        G[y].pb( SZ(e) );
        e.pb( Edge{x, y, c}  );
    }
    int siz[maxn],max_son[maxn],pa[maxn],dep[maxn];
    /*size of subtree、index of max_son, parent index、
        depth*/
    int link_top[maxn],link[maxn],timer;
    /*chain top、index in segtree、time stamp*/
    std::vector<int >G[maxn];
    void init(int N) {
```

```
        n = N;
        e.clear();
        for(int i = 1; i <= n; i++) G[i].clear();
        timer=0;
        pa[1] = 1;
        dep[1] = 0;
    }
    void find_max_son(int x){
        siz[x]=1;
        max_son[x]=-1;
        for(int e_ind : G[x]) {
            int v = e[e_ind].u == x ? e[e_ind].v : e[
                e_ind].u ;
            if( v == pa[x] )continue;
            pa[v] = x; dep[v] = dep[x] + 1;
            find_max_son(v);
            if(max_son[x] == -1 || siz[v] > siz[max_son
                [x]])
                max_son[x] = v;
            siz[x] += siz[v];
        }
    }
    void build_link(int x,int top){
        link[x] = timer++;/*記錄x點的時間戳*/
        link_top[x] = top;
        if(max_son[x] != -1)
            build_link( max_son[x], top);/*優先走訪最大
                孩子*/

        for(int e_ind : G[x]) {
            int v = e[e_ind].u == x ? e[e_ind].v : e[
                e_ind].u ;

            if( v == pa[x] ) {
                seg_arr[ link[x] ] = e[e_ind].c;
            }
            if( v == max_son[x] || v == pa[x] )continue
                ;
            // edge from x => v
            build_link(v, v);
        }
    }
    inline int lca(int a,int b){
        /*求LCA，可以在過程中對區間進行處理*/
        int ta=link_top[a],tb=link_top[b];
        while(ta != tb){
            if(dep[ta]<dep[tb]){
                std::swap(ta,tb);
                std::swap(a,b);
            }
            //interval [ link[ta], link[a] ]
            a = pa[ta];
            ta = link_top[a];
        }
        return dep[a] < dep[b] ? a:b;
    }

    int modify(int a,int b, int c){

        int ta=link_top[a], tb=link_top[b];
        while(ta != tb){
            if(dep[ta]<dep[tb]){
                std::swap(ta,tb);
                std::swap(a,b);
            }
            //interval [ link[ta],link[a] ]
            //same interval if operate on edges
            seg.modify(1, 1, n, link[ta], link[a]+1, c)
                ;
            a = pa[ta];
            ta = link_top[a];
        }
        //a, b are on the same chain
        if( a == b ) ; // interval [ link[a], link[a]
            ], if operate on edges => no edge
        else {
            if(dep[a]>dep[b])
                swap(a,b);
            //interval [ link[a],link[b] ]
            // if operate on edges ==> [ link[ max_son[
                a] ], link[b] ]
                seg.modify(1, 1, n, link[ max_son[a] ],
                    link[b]+1, c);
            }
        }
        /*
        void modify(int a, int b, int c) {
            if( a==b ) return;
            if( link_top[a] == link_top[b]) {
                if( dep[a] > dep[b] ) swap(a, b);
                seg.modify(1, 1, n, link[a]+1, link[b]+1, c
                    );
                assert( link[a]+1 == link[ max_son[a]] );
                return;
            }
            if(dep[link_top[a]] < dep[link_top[b]])
                swap(a, b);
            // a is the node with deeper link_top
            seg.modify( 1, 1, n, link[link_top[a]], link[a]
                + 1, c);
            modify( pa[link_top[a]], b, c);
        }
        */

        /// Heavy Light Decomposition
        void HLD() {
            // root is indexed 1 here !
            find_max_son(1);
            build_link(1, 1);
        }
}tree;


int main() {
    int T;cin>>T;
    while(T--) {
        int n,m ;
        scanf("%d %d",&n, &m);
        int ans = 0;
        tree.init(n);
        for(int i=0;i<n-1;i++) {
            int a, b, c;
            scanf("%d%d%d",&a,&b,&c);
            //a--, b--; be careful here
            tree.addEdge(a, b, c);
            ans += c;
        }
        tree.HLD();

        tree.seg.arr = seg_arr;
        tree.seg.build(1, 1, n);

    }
    return 0;
}
```

## 2.2 Hungarian

```
// edge and node index starting from 0
// dfs  version below
//complexity O ( V*E )
/* to do
#define __maxNodes
num_left = ?
*/
struct Edge {
    int from;
    int to;
    int weight;
    Edge(int f, int t, int w):from(f), to(t), weight(w)
        {}
};
vector<int> G[__maxNodes]; /* G[i] 存储顶点 i 出发的边
    的编号 */
vector<Edge> edges;
int num_nodes;
int num_left;
int num_right;
int num_edges;
int matching[__maxNodes]; /* matching result */
int check[__maxNodes];
```

```
bool dfs(int u) {
    for (auto i = G[u].begin(); i != G[u].end(); ++i) {
         // 对 u 的每个邻接点
        int v = edges[*i].to;
        if (!check[v]) {     // 要求不在交替路中
            check[v] = true; // 放入交替路
            if (matching[v] == -1 || dfs(matching[v]))
                {
                // 如果是未盖点，说明交替路为增广路，则
                    交换路径，并返回成功
                matching[v] = u;
                matching[u] = v;
                return true;
            }
        }
    }
    return false; // 不存在增广路，返回失败
}
int hungarian() {
    int ans = 0;
    memset(matching, -1, sizeof(matching));
    for (int u=0; u < num_left; ++u) {
        if (matching[u] == -1) {
            memset(check, 0, sizeof(check));
            if (dfs(u)) ++ans;
        }
    }
    return ans;
}
```

## 2.3 KM

```
// 最小帶權匹配~ km算法
//http://acm.csie.org/ntujudge/contest_view.php?id=836&
    contest_id=449
#include <bits/stdc++.h>
using namespace std;

struct bipartite {
    #define maxn 602
    #define INF 0xffffff
    int sx[maxn], sy[maxn], mat[maxn][maxn];
    int x[maxn], y[maxn], link[maxn];
    int N, M, slack;

    int DFS(int t) {
        int tmp;
        sx[t] = 1;
        for (int i = 0; i < M; i++) {
            if (!sy[i]) {
                tmp = x[t] + y[i] - mat[t][i];
                if (tmp == 0) {
                    sy[i] = 1;
                    if (link[i] == -1 || DFS(link[i]))
                    {
                    link[i] = t;
                    return 1;
                    }
                }
                else if (tmp < slack) slack = tmp;
            }
        }
        return 0;
    }
    int KM() {
        for (int i = 0; i < N; i++) {
            x[i] = 0;
            for (int j = 0; j < M; j++) {
                if (mat[i][j] > x[i]) x[i] = mat[i][j];
            }
        }
        for (int j = 0; j < M; j++) { y[j] = 0; }
        memset(link, -1, sizeof(link));
        for (int i = 0; i < N; i++) {
            while (1) {
                memset(sx, 0, sizeof(sx));
                memset(sy, 0, sizeof(sy));
                slack = INF;
                if (DFS(i)) break;
                for (int j = 0; j < N; j++) {
                    if (sx[j]) x[j] -= slack;
                }
                for (int j = 0; j < M; j++) {
                    if (sy[j]) y[j] += slack;
                }
            }
        }

        int ans = 0;
        int cnt = 0;
        int t;
        for (int i = 0; i < M; i++)
        {
            t = link[i];
            if (t >= 0 && mat[t][i] != -INF)
            {
                cnt ++;
                ans += mat[t][i];
            }
        }
        // 最大權 : 沒有負號
        return -ans;
    }
    void init(int n,int m) {
        N = n, M = m;
        for (int i = 0; i < N; i++)
            for (int j = 0; j < M; j++)
                mat[i][j] = -INF;
    }
    void input() {
        for(int i = 0; i < N; i++)
            for(int j =0;j<M;j++) {
                // fill in mat[i][j]
                // stands for the weighting , but
                    negative sign !
                // if 最大權 : 沒有負號
            }

    }
}km;

int main(){
    int n,E;
    while (scanf("%d", &n) != EOF)
    {
        km.init(n, n);
        km.input();
        cout<< km.KM() <<endl;
    }
    return 0;
}
```

## 2.4 Bi-vertex-connected Subgraph

```
#include <bits/stdc++.h>
using namespace std;
#ifdef DEBUG
    #define debug(...) printf(__VA_ARGS__)
#else
    #define debug(...) (void)0
#endif
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define PII pair<long long, long long>
#define fi first
#define se second
#define all(x) (x).begin(),(x).end()
#define SZ(x) ((int)(x).size())
const int inf = 0x7fffffff; //beware overflow
const LL INF = 0x7fffffffffffffff; //beware overflow
#define mem(x, y) memset(x, (y), sizeof(x) );
#define IOS ios_base::sync_with_stdio(0); cin.tie(0)
template<typename A, typename B>
ostream& operator <<(ostream &s, const pair<A,B> &p) {
     return s<<"("<<p.first<<","<<p.second<<")";
}
```

```
template<typename T>
ostream& operator <<(ostream &s, const vector<T> &c) {
    s << "[ ";
    for (auto it : c) s << it << " ";
    s << "]";
    return s;
}
template<typename T>
ostream& operator << (ostream &o, const set<T> &st) {
    o << "{";
    for (auto it=st.begin(); it!=st.end(); it++) o << (
        it==st.begin() ? "" : ", ") << *it;
    return o << "}";
}
template<typename T1, typename T2>
ostream& operator << (ostream &o, const map<T1, T2> &mp
    ) {
    o << "{";
    for (auto it=mp.begin(); it!=mp.end(); it++) {
        o << (it==mp.begin()?"":", ") << it->fi << ":"
            << it->se;
    }
    o << "}";
    return o;
}

//    regard every vbcc as a set of edges
/** needed for tarjan **/
#define maxn 100005
#define maxm 100005
int n, m;
struct Edge{int s, t;};
vector<Edge> edge;
int dfn[maxn], low[maxn];
stack<int> st;
bool vis[maxn];
int Time;
bool vis_e[maxm];
int bcnt, vbb[maxm];
vector<int> vb[maxm];
vector<int> G[maxn];
/** **/

void tarjan(int s){
    dfn[s] = low[s] = ++Time;
    vis[s] = true;
    for(int e_ind : G[s]){
        if(!vis_e[e_ind]){
            vis_e[e_ind] = true; st.push(e_ind);
            int to = edge[e_ind].s + edge[e_ind].t - s;
            if(!vis[to]){
                tarjan(to);
                low[s] = min(low[s], low[to]);
                if(low[to] >= dfn[s]){
                    vb[bcnt].clear();
                    while(1){
                        int t = st.top();st.pop();
                        vbb[t] = bcnt;
                        vb[bcnt].push_back(t);
                        if(t == e_ind) break;
                    }
                    bcnt++;
                }
            }else
                low[s] = min(low[s], dfn[to]);
        }
    }
}
void init_tarjan() {

    mem(vis, false); mem(vis_e, false);
    Time = bcnt = 0; edge.clear();
    for(int i = 1; i <= n; i++) G[i].clear();
}

int main() {

    cin >> n >> m;
    init_tarjan();
    for(int i = 0; i < m; i++) {
        int a, b; scanf("%d %d", &a, &b);
        edge.push_back(Edge{a,b});
        G[a].push_back((int)edge.size()-1);
        G[b].push_back((int)edge.size()-1);
    }
    tarjan(1);

}
```

## 2.5 Bi-edge-connected Subgraph

```
/** needed for tarjan **/
#define maxn 100005
#define maxm 100005
int n, m;
int dfn[maxn], low[maxn];
stack<int> st;
int Time;
int bcnt;
vector<int> G[maxn];
bool in_cyc[maxn];
/** **/

void tarjan(int s, int p){
    dfn[s] = low[s] = ++Time;
    st.push(s);
    for(int to : G[s]) if( to != p ){
        if(!dfn[to]) {
            tarjan(to, s);
            low[s] = min(low[s], low[to]);
            if( low[to] > dfn[s]) {
                // is cut_edge
                // pop stack 的過程也可以寫在這
                // 但最後(after tarjan)還要多判stack
                    not empty的情況
                /*
                if( low[to] > dfn[s]) {

                in_cyc[bcnt] = st.top()!=to;
                while(1){
                    int t = st.top();st.pop();
                    id[t] = bcnt;
                    if(t == to) break;
                }
                bcnt++;
                */
            }

            }
        }else
            low[s] = min(low[s], dfn[to]);
    }

    if(low[s] == dfn[s]){
        in_cyc[bcnt] = st.top()!=s;
        while(1){
            int t = st.top();st.pop();
            id[t] = bcnt;
            if(t == s) break;
        }
        bcnt++;
    }
}
void init_tarjan() {
    Time = bcnt = 0;
}
int main() {
  cin >> n >> m;
  init_tarjan();
  for(int i = 0; i < m; i++) {
      int a, b; scanf("%d %d", &a, &b);
      G[a].pb(b), G[b].pb(a);
  }
  mem( in_cyc , false);
  tarjan(1, 1);

}
```

## 2.6 SCC

```
#include <bits/stdc++.h>

using namespace std;
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define PII pair<long long, long long>
#define fi first
#define se second

const int inf = 1e9;
const LL INF = 1e18;
const int mod = 1e9+7;
#define maxn 100050

int n, m;
vector<int> g[maxn];
stack<int> Stack;
int scnt, Time;
int belong[maxn], dfn[maxn], low[maxn], indegree[maxn];
bool instack[maxn];
void input(){
  cin >> n >> m;
  for(int i = 0; i < m; i++){
    int a, b; scanf("%d%d", &a, &b);
    g[a].pb(b);
  }
}
void init() {
  scnt = Time = 0;
  for(int i = 1; i <= n; i++)
    g[i].clear();
  while(!Stack.empty()) Stack.pop();
  memset(indegree, 0, sizeof(indegree));
  memset(dfn, 0, sizeof(dfn));
  memset(instack, false, sizeof(instack));
}
void dfs(int u) {
  dfn[u] = low[u] = ++Time;
  Stack.push(u); instack[u] = true;
  for(int v : g[u]) {
    if ( !dfn[v] ) {
      dfs(v);
      low[u] = min(low[u], low[v]);
    }
    else if(instack[v])
      low[u] = min(low[u], dfn[v]);
  }
  if(low[u] == dfn[u]) {
    scnt++;
    int tp;
    do{
      tp = Stack.top(); Stack.pop();
      instack[tp] = false;
      belong[tp] = scnt;
    } while(tp != u);
  }
}
void tarjan() {
  for(int i = 1; i <= n; i++)
    if(!dfn[i])
      dfs(i);
}
int main(){
  int T;cin >> T;
  while(T--){
    init();
    input();
    tarjan();
    for(int i = 1; i <= n; i++) {
      for(int v : g[i]) {
        if(belong[v] != belong[i])
          indegree[belong[v]]++;
      }
    }
    LL ans = 0;
    for(int i = 1; i <= scnt; i++)
      if(!indegree[i]) ans++;
    cout << ans << endl;
  }
  return 0;
}
```

## 2.7 Steiner Tree( PECaveros )

```
// Minimum Steiner Tree
// O(V 3^T + V^2 2^T)
struct SteinerTree{
#define V 33
#define T 8
#define INF 1023456789
  int n , dst[ V ][ V ] , dp[ 1 << T ][ V ] , tdst[ V
      ];
  void init( int _n ){
    n = _n;
    for( int i = 0 ; i < n ; i ++ ){
      for( int j = 0 ; j < n ; j ++ )
        dst[ i ][ j ] = INF;
      dst[ i ][ i ] = 0;
    }
  }
  void add_edge( int ui , int vi , int wi ){
    dst[ ui ][ vi ] = min( dst[ ui ][ vi ] , wi );
    dst[ vi ][ ui ] = min( dst[ vi ][ ui ] , wi );
  }
  void shortest_path(){
    for( int k = 0 ; k < n ; k ++ )
      for( int i = 0 ; i < n ; i ++ )
        for( int j = 0 ; j < n ; j ++ )
          dst[ i ][ j ] = min( dst[ i ][ j ],
                               dst[ i ][ k ] + dst[ k
                                 ][ j ] );
  }
  int solve( const vector<int>& ter ){
    int t = (int)ter.size();
    for( int i = 0 ; i < ( 1 << t ) ; i ++ )
      for( int j = 0 ; j < n ; j ++ )
        dp[ i ][ j ] = INF;
    for( int i = 0 ; i < n ; i ++ )
      dp[ 0 ][ i ] = 0;
    for( int msk = 1 ; msk < ( 1 << t ) ; msk ++ ){
      if( msk == ( msk & (-msk) ) ){
        int who = __lg( msk );
        for( int i = 0 ; i < n ; i ++ )
          dp[ msk ][ i ] = dst[ ter[ who ] ][ i ];
        continue;
      }
      for( int i = 0 ; i < n ; i ++ )
        for( int submsk = ( msk - 1 ) & msk ; submsk ;
                 submsk = ( submsk - 1 ) & msk )
            dp[ msk ][ i ] = min( dp[ msk ][ i ],
                                  dp[ submsk ][ i ] +
                                    dp[ msk ^ submsk
                                    ][ i ] );
      for( int i = 0 ; i < n ; i ++ ){
        tdst[ i ] = INF;
        for( int j = 0 ; j < n ; j ++ )
          tdst[ i ] = min( tdst[ i ],
                           dp[ msk ][ j ] + dst[ j ][ i
                             ] );
      }
      for( int i = 0 ; i < n ; i ++ )
        dp[ msk ][ i ] = tdst[ i ];
    }
    int ans = INF;
    for( int i = 0 ; i < n ; i ++ )
      ans = min( ans , dp[ ( 1 << t ) - 1 ][ i ] );
    return ans;
  }
} solver;
```

## 2.8 Edmond's Matching Algorithm

```
//http://acm.csie.org/ntujudge/contest_view.php?id=370&
    contest_id=466
#include <bits/stdc++.h>
```

```cpp
using namespace std;
//带花树,Edmonds's matching algorithm,一般图最大匹配
// have to be a undirected graph
#define MAXN 505
vector<int>G[MAXN];//用vector存圖
int pa[MAXN],match[MAXN],st[MAXN],S[MAXN],vis[MAXN];
int t,n;
inline int lca(int u,int v){//找花的花托
    for(++t;;swap(u,v)){
        if(u==0)continue;
        if(vis[u]==t)return u;
        vis[u]=t;//這種方法可以不用清空vis陣列
        u=st[pa[match[u]]];
    }
}
#define qpush(u) q.push(u),S[u]=0
inline void flower(int u,int v,int l,queue<int> &q){
    while(st[u]!=l){
        pa[u]=v;//所有未匹配邊的pa都是雙向的
        if(S[v=match[u]]==1)qpush(v);//所有奇點變偶點
        st[u]=st[v]=l,u=pa[v];
    }
}
inline bool bfs(int u){
    for(int i=1;i<=n;++i)st[i]=i;//st[i]表示第i個點的集合
    memset(S+1,-1,sizeof(int)*n);//-1:沒走過 0:偶點 1:奇點
    queue<int>q;qpush(u);
    while(q.size()){
        u=q.front(),q.pop();
        for(size_t i=0;i<G[u].size();++i){
            int v=G[u][i];
            if(S[v]==-1){
                pa[v]=u,S[v]=1;
                if(!match[v]){//有增廣路直接擴充
                    for(int lst;u;v=lst,u=pa[v])
                        lst=match[u],match[u]=v,match[v]=u;
                    return 1;
                }
                qpush(match[v]);
            }else if(!S[v]&&st[v]!=st[u]){
                int l=lca(st[v],st[u]);//遇到花, 做花的處理
                flower(v,u,l,q),flower(u,v,l,q);
            }
        }
    }
    return 0;
}
inline int blossom(){
    memset(pa+1,0,sizeof(int)*n);
    memset(match+1,0,sizeof(int)*n);
    int ans=0;
    for(int i=1;i<=n;++i)
        if(!match[i]&&bfs(i))++ans;
    return ans;
}
void solve() {
    cin>>n;
    int m;cin>>m;
    while(m--) {
        int a,b;
        scanf("%d %d", &a, &b);
        a++, b++;
        // since node indexed [ 1 .. n ] in this template
        #define pb push_back
        //Multiedge and self-cycles are not forbidden
        G[a].pb( b );
        G[b].pb( a );
    }
    cout<< blossom() <<endl;
    for(int i = 1; i <= n; i++) G[i].clear();
}
int main() {
    int t;cin>>t;
    while(t--) solve();
}
```

## 2.9 Tree Decomposition

```cpp
//codeforces Digit Tree
//http://codeforces.com/problemset/problem/715/C
#include <bits/stdc++.h>
using namespace std;
#ifdef DEBUG
    #define debug(...) printf(__VA_ARGS__)
#else
    #define debug(...) (void)0
#endif
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define PII pair<long long, long long>
#define fi first
#define se second
#define all(x) (x).begin(),(x).end()
#define SZ(x) ((int)(x).size())
const int inf = 0x7fffffff; //beware overflow
const LL INF = 0x7fffffffffffffff; //beware overflow
#define mem(x, y) memset(x, (y), sizeof(x) );
#define IOS ios_base::sync_with_stdio(0); cin.tie(0)
template<typename A, typename B>
ostream& operator <<(ostream &s, const pair<A,B> &p) {
     return s<<"("<<p.first<<","<<p.second<<")";
}
template<typename T>
ostream& operator <<(ostream &s, const vector<T> &c) {
    s << "[ ";
    for (auto it : c) s << it << " ";
    s << "]";
    return s;
}
template<typename T>
ostream& operator << (ostream &o, const set<T> &st) {
    o << "{";
    for (auto it=st.begin(); it!=st.end(); it++) o << (
        it==st.begin() ? "" : ", ") << *it;
    return o << "}";
}
template<typename T1, typename T2>
ostream& operator << (ostream &o, const map<T1, T2> &mp
    ) {
    o << "{";
    for (auto it=mp.begin(); it!=mp.end(); it++) {
        o << (it==mp.begin()?"":", ") << it->fi << ":"
            << it->se;
    }
    o << "}";
    return o;
}

typedef long long ll;

bool isprime[100005];
vector<LL> primes;
LL M, PHI;
#define MOD M
ll modpow(ll a, ll b) {
  ll r = 1;
  while(b) {
    if(b&1) r=(r*a)%MOD;
    a=(a*a)%MOD;
    b >>= 1;
  }
  return r;
}
void Sieve(int n) {
  memset(isprime, 1, sizeof(isprime));
  isprime[1] = false;
  for(int i = 2; i <= n; i++) {
    if(isprime[i]) {
      primes.pb(i);
      for(int j = 2*i; j <= n; j += i)
        isprime[j] = false;
    }
  }
}
```

```
LL phi(LL n) {
  ll num = 1; ll num2 = n;
  for(ll i = 0; primes[i]*primes[i] <= n; i++) {
    if(n%primes[i]==0) {
      num2/=primes[i];
      num*=(primes[i]-1);
    }
    while(n%primes[i]==0) {
      n/=primes[i];
    }
  }
  if(n>1) {
    num2/=n; num*=(n-1);
  }
  n = 1;
  num *= num2;
  return num;
}
ll inv(ll a) {
  return modpow(a, PHI-1);
}
#define maxn 100005
struct edge{
    int u, v, dig;
    int no(int x) {
        return x == u ? v : u;
    }
};
vector<edge> e;
vector<int> G[maxn];
LL n,  ans;
bool vis[maxn];
int sz[maxn], dep[maxn];
LL tenPow[maxn];
int dfs(int u, int p, int d) {
    sz[u] = 1;
    dep[u] = d;
    for(int eind : G[u] ) {
        int v = e[eind].no(u);
        if(  v == p || vis[v] ) continue;
        sz[u] += dfs(v, u, d+1);
    }
    return sz[u];
}
int findCenter(int u, int p, int treesize) {
    for(int eind : G[u] ) {
        int v = e[eind].no(u);
        if( v == p || vis[v] ) continue;
        if( sz[v]*2 > treesize)
            return findCenter( v, u, treesize);
    }
    return u;
}

LL up[maxn], down[maxn];
int belong[maxn];
map<LL, LL> tot;
vector< map<LL, LL> > vec;
vector<int> pt;

void calc(int u, int p, int b, int d) {
    pt.pb( u );
    belong[u] = b;
    dep[u] = d;

    int id =  find_if( all(G[u])  ,[u,p](int x) { return
        e[x].no(u) == p; }) - G[u].begin();
    down[u] = ( down[p]*10 + e[ G[u][id] ].dig )%M;
    up[u] =  (tenPow[ d-1 ]*e[ G[u][id] ].dig + up[p] )
        %M;

    for(int eind : G[u]) {
        int v = e[eind].no(u);
        if( vis[v] || v == p ) continue;
        calc( v, u, b, d+1);
    }

    vec[b][ up[u] ]++;
    tot[ up[u] ]++;
}

LL solve(int cent) {
    //cent is the root now
    vector<int> L;
    for(int eind : G[cent]) {
        int v = e[eind].no(cent);
        if( !vis[v] ) {
            L.pb( v );
        }
    }
    vec.clear();
    vec.resize( SZ(L), {} );
    tot.clear();
    up[cent] = down[cent] = 0;
    dep[cent] = 0;
    pt.clear();
    for(int i = 0; i < SZ(L); i++)
        calc( L[i], cent, i, 1);

    LL ret = 0;
    for(int u : pt) {
        LL tmp = (-down[u]+M)%M;
        tmp = ( tmp*inv( tenPow[ dep[u] ] ))%M;
        ret += tot[ tmp ] - vec[ belong[u] ][ tmp ];
    }
    assert( (LL)count_if(all(pt), [] (int x) { return
        up[x] == 0; } ) == tot[0]);
    LL tmp = tot[0] + (LL)count_if(all(pt), [] (int x)
        { return down[x] == 0; } );
    debug("%lld\n", tmp);
    return ret+tmp;
}
void solveAll(int node) {
    dfs(node, -1, 0);
    int cent = findCenter(node, -1, sz[node]);
    ans += solve( cent );
    debug("%d %lld\n", cent, ans);
    vis[cent] = true;
    for(int eind : G[cent] ) {
        int v = e[eind].no(cent);
        if( vis[v] ) continue;
        solveAll(v);
    }
}
int main() {
    cin>>n>>M;
  Sieve( 100000 );
    PHI = phi(M);
    for(int i = 0; i < n-1; i++) {
        int a, b, c; scanf("%d %d %d", &a, &b, &c);
        G[a].pb( SZ(e) ); G[b].pb( SZ(e) );
        e.pb( edge{a, b, c} );
    }
    //init
    tenPow[0] = 1;
    for(int i = 1; i < maxn; i++) tenPow[i] = (tenPow[i
        -1]*10)%M;
    ans = 0;
    mem( vis, false);
    solveAll(0);
    cout<<ans<<endl;
}
```

## 2.10 Tree Longest Path

```
/** codeforces 592D - Super M **/
#include <bits/stdc++.h>

using namespace std;

#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define PII pair<long long, long long>
#define fi first
#define se second

const int inf = 1e9;
const LL INF = 1e18;
const int mod = 1e9+7;
#define maxn 123460
```

```
int n, m;
vector<int> g[maxn];
bool is[maxn];
int dep[maxn], R, max_depth, A;
int cnt[maxn], parent[maxn];

bool dfs(int u, int par = 0){
  parent[u] = par;
  dep[u] = dep[par] + 1;
  if(dep[u] > max_depth && is[u])
    max_depth = dep[u], R = u;
  bool ret = is[u];
  for(int v : g[u])
    if(v != par)
      ret |= dfs(v, u);
  if(ret) A++;
  return ret;
}

int find_center(int start) {
  R = start; dep[0] = -1; max_depth = 0;
  dfs(start);
  max_depth = 0; dep[R] = -1;
  dfs(R, R);
  int ret = R, d = max_depth/2;
  while( d>0 ) {
    d--;
    ret = parent[ret];
  }
  return ret;
}
int S, dis, max_length;
bool dfs1(int u, int par = 0) {
  dep[u] = dep[par] + 1;
  if(is[u])
    if(dep[u] > max_length)
      max_length = dep[u], S = u;
    else if(dep[u] == max_length && u < S)
      S = u;

  bool c = false;
  for(int v : g[u])
    if( v != par )
      dfs1(v, u);
}
int main(){
  cin >> n >> m;
  for(int i = 0; i < n-1; i++){
    int a, b; scanf("%d%d",&a, &b);
    g[a].pb(b), g[b].pb(a);
  }
  memset(is, false, sizeof(is));
  int tmp;
  for(int i = 0; i < m; i++){
    cin>>tmp;is[tmp] = true;
  }
  int C = find_center(tmp);
  dep[0] = -1;S = inf;dis = (max_depth+1)/2;
  // distance(center, any other node) <= (longestpath +
      1) / 2
  dfs1(C);
  if( max_depth & 1)
    dfs1(parent[C]);
  cout << S << endl << A-2-max_depth << endl;
  return 0;
}
```

# 3 Flow

## 3.1 Dinic Maxflow

```
//http://acm.csie.org/ntujudge/problem.php?id=2581
//French Fries Festival
//dinic runs in O( V^2*E )
#include <bits/stdc++.h>
using namespace std;
#ifdef DEBUG
    #define debug(...) printf(__VA_ARGS__)
#else
    #define debug(...) (void)0
#endif
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define PII pair<long long, long long>
#define fi first
#define se second
#define all(x) (x).begin(),(x).end()
#define SZ(x) ((int)(x).size())
const int inf = 0x7ffffff; //beware overflow
const LL INF = 0x7fffffffffffffff; //beware overflow
#define mem(x, y) memset(x, (y), sizeof(x) );
#define IOS ios_base::sync_with_stdio(0); cin.tie(0)
template<typename A, typename B>
ostream& operator <<(ostream &s, const pair<A,B> &p) {
     return s<<"("<<p.first<<","<<p.second<<")";
}
template<typename T>
ostream& operator <<(ostream &s, const vector<T> &c) {
    s << "[ ";
    for (auto it : c) s << it << " ";
    s << "]";
    return s;
}
template<typename T>
ostream& operator << (ostream &o, const set<T> &st) {
    o << "{";
    for (auto it=st.begin(); it!=st.end(); it++) o << (
        it==st.begin() ? "" : ", ") << *it;
    return o << "}";
}
template<typename T1, typename T2>
ostream& operator << (ostream &o, const map<T1, T2> &mp
    ) {
    o << "{";
    for (auto it=mp.begin(); it!=mp.end(); it++) {
        o << (it==mp.begin()?"":", ") << it->fi << ":"
            << it->se;
    }
    o << "}";
    return o;
}
#define maxn 500
struct Edge{ int to, cap, rev; };
struct Dinic{
    vector<Edge> G[maxn];
    int dis[maxn], iter[maxn];
    void init(int n) {
        //zero based
      for(int i = 0; i < n; i++) G[i].clear();
    }
    void addEdge(int from, int to, int cap) {
        vector<Edge>::iterator it;
        if( ( it=find_if( all(G[from]), [to](Edge& e) {
             return e.to == to; } )) != G[from].end() )
             {
            it->cap += cap;
            return;
        }
      G[from].pb(Edge{to, cap, (int)G[to].size()});
      G[to].pb(Edge{from, 0, (int)G[from].size()-1});
        //if undirected 0 will be cap
    }
    bool bfs(int s, int t) {
      memset(dis, -1, sizeof(dis));
      queue<int> que;
      que.push(s); dis[s] = 0;
      while(!que.empty()) {
        int tp = que.front(); que.pop();
        for(Edge &e : G[tp]) {
          if(e.cap > 0 && dis[e.to] == -1)
            dis[e.to] = dis[tp] + 1, que.push(e.to);
        }
      }
      return dis[t] != -1;
    }
    int dfs(int v, int t, int f) {
      if( v == t ) return f;
      for(int &i = iter[v]; i < G[v].size(); i++) {
```

```
        Edge &e = G[v][i];
        if(e.cap > 0 && dis[v] < dis[e.to]) {
          int d = dfs(e.to, t, min(f, e.cap));
          if(d > 0) {
            e.cap -= d;
            G[e.to][e.rev].cap += d;
            f += d;
            return d;
          }
        }
      }
      return 0;
    }
    int maxFlow(int s, int t) {
      int ret = 0;
      while( bfs(s, t) ) {
        memset(iter, 0, sizeof(iter));
        int f;
        while(( f = dfs(s, t, inf) )> 0 )
          ret += f;
      }
      return ret;
    }
}dinic, dinic2;
void solve() {
    int n,m,k; cin>>n>>m>>k;
    // flow problem with lower bounds;
    int s = 0, t = n+2, ss = n+3, tt = n+4;
    dinic.init( n+5 );
    dinic.addEdge(s, 1, k);
    dinic.addEdge(n+1, t, k);
    //
    int slb = 0;
    while(m--) {
        int l, r, a, b; scanf("%d %d %d %d", &l, &r, &a
            , &b);
        slb += a;
        r++;

        dinic.addEdge(l, r, b-a);
        dinic.addEdge(ss, r, a);
        dinic.addEdge(l, tt, a);
    }
    dinic2  = dinic;

    dinic.addEdge(t, s, k);
    int f1 = dinic.maxFlow(ss, tt);
    if( !all_of( all(dinic.G[ss]), [](Edge x) { return
        x.cap == 0; } ) ) {
        puts("-1"); return;
    }

    dinic2.addEdge(ss, s, 1e9);
    dinic2.addEdge(t, tt, 1e9);

    int f2 = dinic2.maxFlow(ss, tt);
    // maxflow in current graph is f2 - slb
    printf("%d\n", (f2 - slb)*n );
}
int main() {
    int t;cin>>t;
    while(t--)
        solve();
}
```

# 4 Data Structure

## 4.1 Disjoint Set

```
struct Disjoint_set {
    #define MAX_N 500005
    // define MAX_N
    int pa[MAX_N], Rank[MAX_N];
    int sz[MAX_N];
    void init_union_find(int V) {
        for(int i=0; i<V; i++) {
            pa[i] = i;
            Rank[i] = 0;
            sz[i] = 1;
        }
    }
    int find(int x) {
        return x == pa[x] ? x : pa[x] = find(pa[x]);
    }

    int unite(int x, int y) {
        x = find(x), y = find(y);
        int S = sz[x]+sz[y];
        if(x != y){
            if(Rank[x] < Rank[y]) {
                pa[x] = y;
                sz[y]=S;
                return y;
            }
            else{
                pa[y] = x;
                sz[x] = S;
                if(Rank[x] == Rank[y]) Rank[x] ++;
                return x;
            }
        }
    }
    bool same(int x, int y) {
        return find(x) == find(y);
    }
}
```

## 4.2 Djs + Seg

```
// demo ==> undo djs + segtree with offline
// this program doesn't consider the problem of
    overflowing varaible ans
// http://acm.csie.org/ntujudge/view_code.php?id
    =108190&contest_id=472

#define maxn 100005
#define maxm 500005
//can be used to solve dynamic connectivity problem
//can be used with segment tree ==> offline
struct DisjointSet {
  // save() is like recursive
  // undo() is like return
  int n, fa[maxn], sz[maxn];
  vector<pair<int*,int>> h;
  vector<int> sp;
  int ans;
  void init(int tn) {
    ans = 0;
    n=tn;
    for (int i=0; i<n; i++) {
      fa[i]=i;
      sz[i]=1;
    }
    sp.clear(); h.clear();
  }
  void assign(int *k, int v) {
    h.pb({k, *k});
    *k=v;
  }
  void save() { sp.pb(SZ(h)); }
  void undo() {
    assert(!sp.empty());
    int last=sp.back(); sp.pop_back();
    while (SZ(h)!=last) {
      auto x=h.back(); h.pop_back();
      *x.fi=x.se;
    }
  }
  int f(int x) {
    while (fa[x]!=x) x=fa[x];
    return x;
  }
  void uni(int x, int y) {
    x=f(x); y=f(y);
    if (x==y) return ;
    if (sz[x]<sz[y]) swap(x, y);
    //nans stands for new answer
    int t = sz[x]+sz[y];
```

```
    int nans = ans - (sz[x]*sz[x]-sz[x]) - (sz[y]*sz[y]-sz[y]) + t*t-t;
    assign(&sz[x], sz[x]+sz[y]);
    assign(&fa[y], x);
    assign(&ans, nans);
  }
}djs;

int n, m;
map<int, int> ma[maxn];
vector<pii> seg[4*maxm];
LL ans[maxm];
void add(int ql, int qr, int a, int b, int id=1, int l=0, int r=m) {
    if( qr <= l || ql >= r ) return;
    if( l >= ql && r <= qr ) {
        seg[id].pb( mp(a, b) );
        return ;
    }
    int mid = (l+r)>>1;
    add( ql, qr, a, b, id*2, l, mid);
    add( ql, qr, a, b, id*2+1, mid, r);
}
void dfs(int u=1, int l=0, int r=m) {

    djs.save();
    for(pii v : seg[u] )  djs.uni( v.fi, v.se );

    if( r-l > 1 ) {
        int mid = (l+r)>>1;
        dfs(u*2, l, mid);
        dfs(u*2+1, mid, r);
    }else {
        // do sth here
        ans[l] = djs.ans;
    }

    djs.undo();
}
int main() {
    scanf("%d %d", &n, &m);
    for(int i = 0; i < m; i++) {
        int a, b; scanf("%d %d",&a, &b);
        a--, b--; if( b < a ) swap(a, b);

        if( ma[a].count(b) ) {
            add(ma[a][b], i, a, b);
            ma[a].erase(b);
        }else ma[a][b] = i;
    }
    for(int i = 0; i < n; i++) if( !ma[i].empty() ) {
        for(auto p : ma[i])
            add( p.se, m, i, p.fi);
    }
    djs.init(n);
    dfs();
    for(int i =0; i < m;i++) printf("%lld\n", ans[i]);
}
```

## 4.3 Sparse Table

```
//codeforces 689D
#define maxn 200005

template< typename T, typename Cmp = less<T> >
struct RMQ {
    T d[maxn][20];
    Cmp cmp;
    int w[maxn], sz;

    void init(const T *a, int n) {
        int i, j;

        for (w[0] = -1, i = 1; i <= n; ++i) w[i] = (i & (i - 1)) ? w[i - 1] : w[i - 1] + 1;
        for (sz = n, i = 0; i < n; ++i) d[i][0] = a[i];
        for (j = 1; (1 << j) <= n; ++j) {
            for (i = 0; i + (1 << j) <= n; ++i) {
                d[i][j] = cmp(d[i][j - 1], d[i + (1 << (j - 1))][j - 1]) ? d[i][j - 1] : d
                    [i + (1 << (j - 1))][j - 1];
            }
        }
    }
    // index of a [l .. r]
    const T &query(int l, int r) const {
        int x = w[r - l + 1];
        return cmp(d[l][x], d[r - (1 << x) + 1][x]) ? d
            [l][x] : d[r - (1 << x) + 1][x];
    }
};
int a[maxn], b[maxn];
int n;
RMQ<int> s;
RMQ<int, greater<int> > t;

int main() {
    cin>>n;
    for(int i = 0;i < n; i++) scanf("%d", &a[i]);
    for(int i = 0; i < n; i++) scanf("%d", &b[i]);

    s.init(b, n);
    t.init(a, n);
    int c, d;
    LL ans = 0;
    for(int i=0;i<n;i++) {

        if( a[i] > b[i]) continue;

        int ub = n+1, lb = i;
        while( ub-lb>1) {
            int mid = (ub+lb)>>1;
            if( t.query(i, mid-1) - s.query(i, mid-1) >
                0) ub = mid;
            else lb = mid;
        }
        int up = ub;

        ub = n+1, lb = i;
         while( ub-lb>1) {
            int mid = (ub+lb)>>1;
            if( t.query(i, mid-1) - s.query(i, mid-1)
                >= 0) ub = mid;
            else lb = mid;
        }
        int down = ub;
        ans += up-down;
    }
    cout << ans << endl;

    return 0;
}
```

## 4.4 Link Cut Tree

```
//https://github.com/yzgysjr/ACM-ICPC-Templates/blob/
    master/Data%20Structure/Link%20Cut%20Tree.cpp
struct node { int rev; node *pre, *ch[2]; } base[MAXN],
     nil, *null;
typedef node *tree;
#define isRoot(x) (x->pre->ch[0] != x && x->pre->ch[1]
    != x)
#define isRight(x) (x->pre->ch[1] == x)
inline void MakeRev(tree t) { if (t != null) { t->rev
    ^= 1; swap(t->ch[0], t->ch[1]); } }
inline void PushDown(tree t) { if (t->rev) { MakeRev(t
    ->ch[0]); MakeRev(t->ch[1]); t->rev = 0; } }
inline void Rotate(tree x) {
  tree y = x->pre; PushDown(y); PushDown(x);
  int d = isRight(x);
  if (!isRoot(y)) y->pre->ch[isRight(y)] = x; x->pre =
      y->pre;
  if ((y->ch[d] = x->ch[!d]) != null) y->ch[d]->pre = y
      ;
  x->ch[!d] = y; y->pre = x; Update(y);
}
inline void Splay(tree x) {
  PushDown(x); for (tree y; !isRoot(x); Rotate(x)) {
    y = x->pre; if (!isRoot(y)) Rotate(isRight(x) !=
        isRight(y) ? x : y);
```

```
  } Update(x);
}
inline void Splay(tree x, tree to) {
  PushDown(x); for (tree y; (y = x->pre) != to; Rotate(
      x)) if (y->pre != to)
    Rotate(isRight(x) != isRight(y) ? x : y);
  Update(x);
}
inline tree Access(tree t) {
  tree last = null; for (; t != null; last = t, t = t->
      pre) Splay(t),t->ch[1] = last, Update(t);
  return last;
}
inline void MakeRoot(tree t) { Access(t); Splay(t);
    MakeRev(t); }
inline tree FindRoot(tree t) { Access(t); Splay(t);
    tree last = null;
  for ( ; t != null; last = t, t = t->ch[0]) PushDown(t
      ); Splay(last); return last;
}
inline void Join(tree x, tree y) { MakeRoot(y); y->pre
    = x; }
inline void Cut(tree t) {Access(t); Splay(t); t->ch
    [0]->pre = null; t->ch[0] = null; Update(t);}
inline void Cut(tree x, tree y) {
  tree upper = (Access(x), Access(y));
  if (upper == x) { Splay(x); y->pre = null; x->ch[1] =
       null; Update(x); }
  else if (upper == y) { Access(x); Splay(y); x->pre =
      null; y->ch[1] = null; Update(y); }
  else assert(0); // `impossible to happen`
}
inline int Query(tree a, tree b) { // `query the cost
    in path a <-> b, lca inclusive`
  Access(a); tree c = Access(b); // c is lca
  int v1 = c->ch[1]->maxCost; Access(a);
  int v2 = c->ch[1]->maxCost;
  return max(max(v1, v2), c->cost);
}
void Init() {
  null = &nil; null->ch[0] = null->ch[1] = null->pre =
       null; null->rev = 0;
  Rep(i, 1, N) { node &n = base[i]; n.rev = 0; n.pre =
      n.ch[0] = n.ch[1] = null; }
}
//compressed version
//http://trinklee.blog.163.com/blog/static
    /238158060201521101957375/
const int N=30010;
int n,fa[N],son[N][2],val[N],siz[N],stmp,rev[N];
#define swap(a,b) (stmp=a,a=b,b=stmp)
void pu(int t){siz[t]=siz[son[t][0]]+siz[son[t][1]]+1;}
void pd(int t){rev[t]?rev[t]=0,rev[son[t][0]]^=1,rev[
    son[t][1]]^=1,swap(son[t][0],son[t][1]),1:1;}
bool nr(int t){return son[fa[t]][0]==t||son[fa[t]][1]==
    t;}
void rtt(int t,int f=0,bool p=0){
    p=son[f=fa[t]][1]==t,
    fa[t]=fa[f],nr(f)?son[fa[f]][son[fa[f]][1]==f]=t:1,
    (son[f][p]=son[t][!p])?fa[son[f][p]]=f:1,
    pu(son[fa[f]=t][!p]=f);
}
void pv(int t){if(nr(t))pv(fa[t]);pd(t);}
void splay(int t,int f=0){
    for(pv(t);nr(t);rtt(t))nr(f=fa[t])?
    rtt(son[f][1]==t^son[fa[f]][1]==f?t:f),1:1;pu(t);
}
void access(int t,int la=0){for(;t;splay(t),son[t][1]=
    la,la=t,t=fa[t]);}
void makeroot(int t){access(t),splay(t),rev[t]^=1;}
void link(int u,int v){makeroot(u),fa[u]=v;}
void cut(int u,int v){makeroot(u),access(v),splay(v),
    son[v][0]=fa[u]=0;}
```

## 4.5 Treap

```
#include <bits/stdc++.h>
using namespace std;

struct Treap{
    Treap *l, *r;
    int pri, key, val;
    Treap(int _val, int _key):
        val(_val), key(_key), l(NULL), r(NULL), pri(
            rand()){}
};

/// We assure that key value in A treap is greater than
     that in treap B
Treap *merge( Treap *a, Treap *b){
    if(a==NULL || b==NULL) return (!a) ? b : a;
    if(a->pri > b->pri){
        a->r = merge(a->r, b);
        return a;
    }else{
        b->l = merge(a, b->l);
        return b;
    }
}
void split(Treap *t, int k, Treap *&a, Treap *&b){
    if( !t ) a = b = NULL;
    else if( t->key <= k){
        a = t;
        split(t->r, k, a->r, b);
    }else{
        b = t;
        split(t->l, k, a, b->l);
    }
}
Treap* insert( Treap *t, int k, int _val){
    Treap *tl, *tr;
    split(t, k, tl, tr);
    return merge(tl, merge(new Treap(_val, k) , tr));
}
Treap* remove( Treap* t, int k){
    Treap *tl, *tr;
    split(t, k-1, tl, t);
    split(t, k, t, tr);
    return merge(tl, tr);
}
int main(){

    return 0;
}
```

# 5 Math

## 5.1 Prime Table

```
#include <bits/stdc++.h>
using namespace std;
struct Prime_table {

    int prime[1000000]={2,3,5,7};
    int sz=4;
    // biggest prime < ub
    int ub=(1<<20);

    int check(int num){
        int k = 0;
        for(k = 0; k < sz && prime[k]*prime[k] <= num;
            k++){
            if( num % prime[k]==0)   return 0;
        }
        return 1;
    }
    void buildprime(){
        int currentPrime=7;
        int j=4;
        for(sz=4,j=4; currentPrime<ub; sz++, j=6-j){
            currentPrime=currentPrime+j;
            if (check(currentPrime)) {
                prime[sz] = currentPrime;
            }
            else{
                sz--;
            }
        }
     }
```

```
}ptable;
```

## 5.2 Miller Rabin Prime Test

```
#include <bits/stdc++.h>

using namespace std;

typedef long long LL;
LL mul(LL a, LL b, const LL mod) {
    LL x = 0,y = a % mod;
    while (b > 0) {
        if ( b&1 )
            x = (x + y) % mod;
        y = (y * 2) % mod;
        b >>= 1;
    }
    return x % mod;
}
/*
LL mul(LL lhs, LL rhs, const LL mod ) {
    return ( lhs * rhs ) % mod;
}
*/
LL mypow(LL b, LL e,const LL mod) {
    LL x = 1;
    LL y = b;
    while ( e ) {
        if ( e&1 ) x = mul(x, y, mod);
        y = mul(y, y, mod);
        e >>= 1;
    }
    return x;
}
const int testbase[] = {2, 3, 5, 7, 11, 13, 17, 19, 23,
     29, 31, 37};
bool isprime(const LL p) {
    if (p < 2) return false;
    if (p != 2 && !(p&1) ) return false;
    LL d = p - 1;
    while ( !(d&1) ) d >>= 1;
    for( int a : testbase ) {

        LL td = d;
        if( a >= p-1 ) return true;
        LL st = mypow(a, td, p);
        while ( td != p - 1 && st != 1 && st != p - 1)
            {
            st = mul(st, st, p);
            td <<= 1;
        }
        if ( st != p - 1 && !(td&1) ) return false;
    }
    return true;
}
int main() {
    int T;
    scanf("%d",&T);
    while(T--) {
        LL q;
        scanf("%lld",&q);
        puts(isprime(q)?"YES":"NO");
    }
    return 0;
}
```

## 5.3 Extended Euclidean Algorithm

```
/** normal gcd function using recursion **/
int gcd(int a, int b){
    if(b == 0) return a;
    return gcd(b, a%b);
}
// Find solution of ax + by = gcd(a, b)
// ps : x, y may be negative
int extgcd(int a, int b, int& x, int& y){
    int d = a;
    if(b != 0) {
        d = extgcd(b, a%b, y, x);
        y -= (a/b) * x;
    }else {
        x = 1, y = 0;
    }
    return d;
}
```

## 5.4 Gauss Elimination

```
// solving linear equations with gauss elimination
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

void print(vector< vector<double> > A) {
    int n = A.size();
    for (int i=0; i<n; i++) {
        for (int j=0; j<n+1; j++) {
            cout << A[i][j] << "\t";
            if (j == n-1) {
                cout << "| ";
            }
        }
        cout << "\n";
    }
    cout << endl;
}

vector<double> gauss(vector< vector<double> > A) {
    int n = A.size();

    for (int i=0; i<n; i++) {
        // Search for maximum in this column
        double maxEl = abs(A[i][i]);
        int maxRow = i;
        for (int k=i+1; k<n; k++) {
            if (abs(A[k][i]) > maxEl) {
                maxEl = abs(A[k][i]);
                maxRow = k;
            }
        }

        // Swap maximum row with current row (column by
             column)
        for (int k=i; k<n+1;k++) {
            double tmp = A[maxRow][k];
            A[maxRow][k] = A[i][k];
            A[i][k] = tmp;
        }

        // Make all rows below this one 0 in current
            column
        for (int k=i+1; k<n; k++) {
            double c = -A[k][i]/A[i][i];
            for (int j=i; j<n+1; j++) {
                if (i==j) {
                    A[k][j] = 0;
                } else {
                    A[k][j] += c * A[i][j];
                }
            }
        }
    }

    // Solve equation Ax=b for an upper triangular
        matrix A
    vector<double> x(n);
    for (int i=n-1; i>=0; i--) {
        x[i] = A[i][n]/A[i][i];
        for (int k=i-1;k>=0; k--) {
            A[k][n] -= A[k][i] * x[i];
        }
    }
    return x;
}

int main() {
```

```
    int n;
    cin >> n;

    vector<double> line(n+1,0);
    vector< vector<double> > A(n,line);

    // Read input data
    for (int i=0; i<n; i++) {
        for (int j=0; j<n; j++) {
            cin >> A[i][j];
        }
    }

    for (int i=0; i<n; i++) {
        cin >> A[i][n];
    }

    // Print input
    print(A);

    // Calculate solution
    vector<double> x(n);
    x = gauss(A);

    // Print result
    cout << "Result:\t";
    for (int i=0; i<n; i++) {
        cout << x[i] << " ";
    }
    cout << endl;
}
```

## 5.5 FFT

```
typedef long double ld;
/* N must be 2^k and greater than array.size()
 * FFT( a );
 * FFT( b );
 * for(int i = 0; i<N; ++i) c[i] = conj(a[i] * b[i]);
 * FFT( c );
 * for(int i = 0; i<N; ++i) c[i] = conj(c[i]);
 * for(int i = 0; i<N; ++i) c[i] /= N;
 */
void FFT(vector< complex<ld> >& v) {
    int N = v.size();
    for(int i = 1, j = 0; i<N; ++i) {
        for(int k = N>>1; !((j^=k)&k); k>>=1);
        if(i>j) swap(v[i],v[j]);
    }
    for(int k = 2; k<=N; k<<=1) {
        ld w = -2.0*pi/k;
        complex<ld> deg(cos(w),sin(w));
        for(int j = 0; j<N; j+=k) {
            complex<ld> theta(1,0);
            for(int i = j; i<j+k/2; ++i) {
                complex<ld> a = v[i];
                complex<ld> b = v[i+k/2]*theta;
                v[i] = a+b;
                v[i+k/2] = (a-b);
                theta *= deg;
            }
        }
    }
}
```

```
//http://sd-invol.github.io/2016/02/13/FFT-mod-prime/
struct Complex {
    double x , y;
    Complex (double _x = 0 , double _y = 0) {
        x = _x , y = _y;
    }
    Complex operator + (const Complex &r) const {
        return Complex(x + r.x , y + r.y);
    }
    Complex operator - (const Complex &r) const {
        return Complex(x - r.x , y - r.y);
    }
    Complex operator * (const Complex &r) const {
        return Complex(x * r.x - y * r.y , x * r.y + y
            * r.x);
    }
    Complex conj () const {
        return Complex(x , -y);
    }
    double operator = (const double a) {
        *this = Complex(a , 0);
        return a;
    }
};
const double pi = acos(-1.0);
//fft with modulo, code referenced from the internet
/*
    fftMod::fftPrepare(len);
    fftMod::convolution(res, le, ri, len, r-1);
*/
namespace fftMod{
    const int N = 1 << 18;
    const int Mod = 1e9 + 7;
    // to do, M should be about sqrt(Mod)
    const int M = 32768;
    int p[N] , I[N] ;
    int t1[N] , t2[N] , t3[N];

    Complex w[N];
    int rev[N];

    void fftPrepare(int n) {
        int LN = __builtin_ctz(n);
        for (int i = 0 ; i < n ; ++ i) {
            double ang = 2 * pi * i / n;
            w[i] = Complex(cos(ang) , sin(ang));
            rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (
                LN - 1));
        }
    }
    void FFT(Complex P[], int n, int oper) {
        for (int i = 0 ; i < n ; i ++) {
            if (i < rev[i]) {
                swap(P[i], P[rev[i]]);
            }
        }
        for (int d = 0; (1 << d) < n; d++) {
            int m = 1 << d, m2 = m * 2 , rm = n / m2;
            for (int i = 0; i < n; i += m2) {
                for (int j = 0; j < m; j++) {
                    Complex &P1 = P[i + j + m], &P2 = P
                        [i + j];
                    Complex t = w[rm * j] * P1;
                    P1 = P2 - t;
                    P2 = P2 + t;
                }
            }
        }
    }

    Complex A[N] , B[N] , C1[N] , C2[N];
    void convolution(vector<int> &res , vector<int> &a,
        vector<int> &b, int len, int K) {
        // a[ 0 .. len ) and b[ 0 .. len )'s
            convolution % Mod
        // stored in res[ 0 .. K+1 )
        for (int i = 0 ; i < len ; ++ i) {
            A[i] = Complex(a[i] / M , a[i] % M);
            B[i] = Complex(b[i] / M , b[i] % M);
        }
       FFT(A , len , 1); FFT(B , len , 1);

        for (int i = 0 ; i < len ; ++ i) {
            int j = i ? len - i : i;
            Complex a1 = (A[i] + A[j].conj()) * Complex
                (0.5 , 0);
            Complex a2 = (A[i] - A[j].conj()) * Complex
                (0 , -0.5);
            Complex b1 = (B[i] + B[j].conj()) * Complex
                (0.5 , 0);
            Complex b2 = (B[i] - B[j].conj()) * Complex
                (0 , -0.5);
            Complex c11 = a1 * b1 , c12 = a1 * b2;
            Complex c21 = a2 * b1 , c22 = a2 * b2;
            C1[j] = c11 + c12 * Complex(0 , 1);
            C2[j] = c21 + c22 * Complex(0 , 1);
        }
```

```
        FFT(C1 , len , -1); FFT(C2 , len , -1);

        for (int i = 0 ; i <= K ; ++ i) {
            int x = (LL)(C1[i].x / len + 0.5) % Mod;
            int y1 = (LL)(C1[i].y / len + 0.5) % Mod;
            int y2 = (LL)(C2[i].x / len + 0.5) % Mod;
            int z = (LL)(C2[i].y / len + 0.5) % Mod;
            res[i] = ((LL)x * M * M + (LL)(y1 + y2) * M
                + z) % Mod;
        }
    }
};
```

## 5.6 NNT

```
/*
NTT( a );
NTT( b );
for(int i = 0; i<N: ++i)
    c[i] = (long long) a[i] * b[i] % mod;
NTT( c, true );
for(int i = 0; i<N; ++i)
    c[i] = (786433LL-12) * c[i] % mod;
*/

constexpr int mod = 786433;
constexpr int N = 65536;

void NTT(vector< int >& v, bool flag = false)
{
    for(int i = 1, j = 0; i<N; ++i)
    {
        for(int k = N>>1; !((j^=k)&k); k>>=1);
        if(i>j) swap(v[i],v[j]);
    }
    for(int k = 2; k<=N; k<<=1)
    {
        int deg = mypow(flag ? 524289 : 3, N / k);
        for(int j = 0; j<N; j+=k)
        {
            int theta = 1;
            for(int i = j; i<j+k/2; ++i)
            {
                int a = v[i];
                int b = (long long) v[i+k/2]*theta%mod;
                v[i] = (a+b) % mod;
                v[i+k/2] = (a-b+mod)%mod;
                theta = (long long) theta * deg % mod;
            }
        }
    }
}
```

```
constexpr int mod = 1e9+7;
typedef vector<int> VEC;
// ntt + Crt, code referenced from the internet
namespace nttCrt {
    constexpr int magic[3] = {1004535809, 998244353,
        104857601};
    constexpr int MOD = 1000000007;
    constexpr int G = 3;
    int P;
    inline int quick_mod(int x, int k, int MOD) {
        int ans = 1;
        while (k) {
            if (k&1) ans = 1LL * ans * x % MOD;
            x = 1LL * x * x % MOD;
            k >>= 1;
        }
        return ans;
    }
    inline void change(int *y, int len) {
        for(int i = 1, j = len / 2; i < len - 1; i++) {
            if(i < j) swap(y[i], y[j]);
            //交换互为小标反转的元素，i<j保证交换一次
            //i做正常的+1，j左反转类型的+1,始终保持i和j
                是反转的
            int k = len / 2;
            while(j >= k) {
                j -= k;
                k /= 2;
            }
            if(j < k) j += k;
        }
    }
    inline void ntt(int *y, int len, int on) {
        change(y, len);
        for(int h = 2; h <= len; h <<= 1) {
            int wn = quick_mod(G, (P - 1) / h, P);
            for(int j = 0; j < len; j += h) {
                int w = 1;
                for(int k = j; k < j + h / 2; k++) {
                    int u = y[k] % P;
                    int t = 1LL * w * y[k + h / 2] % P;
                    y[k] = (u + t) % P;
                    y[k + h / 2] = ((u - t) % P + P) %
                        P;
                    w = 1LL * w * wn % P;
                }
            }
        }
        if(on == -1) {
            for (int i = 1; i < len / 2; i++)
                swap(y[i], y[len - i]);
            int inv = quick_mod(len, P - 2, P);
            for(int i = 0; i < len; i++)
                y[i] = 1LL * y[i] * inv % P;
        }
    }

    int n;
    int r[3][3];

    inline int CRT(int *a) {
        int sb[3] = {a[0], a[1], a[2]};
        for(int i = 0; i < 3; ++i) {
            for(int j = 0; j < i; ++j) {
                int t = (sb[i] - sb[j]) % magic[i];
                if(t < 0) t += magic[i];
                sb[i] = 1LL * t * r[j][i] % magic[i];
            }
        }
        int mul = 1, ans = sb[0] % MOD;
        for(int i = 1; i < 3; ++i) {
            mul = 1LL * mul * magic[i - 1] % MOD;
            ans = (ans + 1LL * sb[i] * mul) % MOD;
        }
        return ans;
    }
    int tmp[maxn][3];
    int x1[maxn*2], x2[maxn*2];

    inline void gao(vector<int>& res, vector<int> &a,
        vector<int> &b ,int len, int kk) {

        for (int ti = 0; ti < 3; ti++) {
            P = magic[ti];
            int k;
            for ( k = 0; k < SZ(a) && k < len; k++) x1[
                k] = a[k];
            for ( ;k < len; k++) x1[k] = 0;
            for ( k = 0; k < SZ(b) && k < len; k++) x2[
                k] = b[k];
            for ( ;k < len; k++) x2[k] = 0;

            ntt(x1, len, 1); ntt(x2, len, 1);
            for (int i = 0; i < len; i++) x1[i] = 1LL *
                x1[i] * x2[i] % P;
            ntt(x1, len, -1);

            for (int i = 0; i <= kk; i++) tmp[i][ti] =
                x1[i];
        }
        for (int i = 0; i <= kk; i++) res[i] = CRT(tmp
            [i]) ;
    }
    inline void init() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                r[i][j] = quick_mod(magic[i], magic[j]
                    - 2, magic[j]);
```

```
            }
        }
    }
};
```

## 5.7 Big Number

```
//http://blog.csdn.net/hackbuteer1/article/details
    /6595881
#include<iostream>
#include<string>
#include<iomanip>
#include<algorithm>
using namespace std;

#define MAXN 9999
#define MAXSIZE 10
#define DLEN 4

class BigNum
{
private:
  int a[500];      //可以控制大数的位数
  int len;         //大数长度
public:
  BigNum(){ len = 1;memset(a,0,sizeof(a)); }   //构造函
      数
  BigNum(const int);          //将一个int类型的变量转化为
      大数
  BigNum(const char*);        //将一个字符串类型的变量转化
      为大数
  BigNum(const BigNum &);   //拷贝构造函数
  BigNum &operator=(const BigNum &);     //重载赋值运算
      符，大数之间进行赋值运算

  friend istream& operator>>(istream&,  BigNum&);    //
      重载输入运算符
  friend ostream& operator<<(ostream&,  BigNum&);    //
      重载输出运算符

  BigNum operator+(const BigNum &) const;    //重载加法
      运算符，两个大数之间的相加运算
  BigNum operator-(const BigNum &) const;    //重载减法
      运算符，两个大数之间的相减运算
  BigNum operator*(const BigNum &) const;    //重载乘法
      运算符，两个大数之间的相乘运算
  BigNum operator/(const int   &) const;     //重载除法
      运算符，大数对一个整数进行相除运算

  BigNum operator^(const int   &) const;     //大数的n次
      方运算
  int    operator%(const int   &) const;     //大数对一个
      int类型的变量进行取模运算
  bool   operator>(const BigNum & T)const;   //大数和另
      一个大数的大小比较
  bool   operator>(const int & t)const;      //大数和一
      个int类型的变量的大小比较

  void print();        //输出大数
};
BigNum::BigNum(const int b)      //将一个int类型的变量转
    化为大数
{
  int c,d = b;
  len = 0;
  memset(a,0,sizeof(a));
  while(d > MAXN)
  {
    c = d - (d / (MAXN + 1)) * (MAXN + 1);
    d = d / (MAXN + 1);
    a[len++] = c;
  }
  a[len++] = d;
}
BigNum::BigNum(const char*s)      //将一个字符串类型的变
    量转化为大数
{
  int t,k,index,l,i;
  memset(a,0,sizeof(a));
  l=strlen(s);
  len=l/DLEN;
  if(l%DLEN)
    len++;
  index=0;
  for(i=l-1;i>=0;i-=DLEN)
  {
    t=0;
    k=i-DLEN+1;
    if(k<0)
      k=0;
    for(int j=k;j<=i;j++)
      t=t*10+s[j]-'0';
    a[index++]=t;
  }
}
BigNum::BigNum(const BigNum & T) : len(T.len)  //拷贝构
    造函数
{
  int i;
  memset(a,0,sizeof(a));
  for(i = 0 ; i < len ; i++)
    a[i] = T.a[i];
}
BigNum & BigNum::operator=(const BigNum & n)    //重载赋
    值运算符，大数之间进行赋值运算
{
  int i;
  len = n.len;
  memset(a,0,sizeof(a));
  for(i = 0 ; i < len ; i++)
    a[i] = n.a[i];
  return *this;
}
istream& operator>>(istream & in,  BigNum & b)    //重载
    输入运算符
{
  char ch[MAXSIZE*4];
  int i = -1;
  in>>ch;
  int l=strlen(ch);
  int count=0,sum=0;
  for(i=l-1;i>=0;)
  {
    sum = 0;
    int t=1;
    for(int j=0;j<4&&i>=0;j++,i--,t*=10)
    {
      sum+=(ch[i]-'0')*t;
    }
    b.a[count]=sum;
    count++;
  }
  b.len =count++;
  return in;

}
ostream& operator<<(ostream& out,  BigNum& b)    //重载
    输出运算符
{
  int i;
  cout << b.a[b.len - 1];
  for(i = b.len - 2 ; i >= 0 ; i--)
  {
    cout.width(DLEN);
    cout.fill('0');
    cout << b.a[i];
  }
  return out;
}

BigNum BigNum::operator+(const BigNum & T) const   //两
    个大数之间的相加运算
{
  BigNum t(*this);
  int i,big;      //位数
  big = T.len > len ? T.len : len;
```

```cpp
    for(i = 0 ; i < big ; i++)
    {
        t.a[i] +=T.a[i];
        if(t.a[i] > MAXN)
        {
            t.a[i + 1]++;
            t.a[i] -=MAXN+1;
        }
    }
    if(t.a[big] != 0)
        t.len = big + 1;
    else
        t.len = big;
    return t;
}
BigNum BigNum::operator-(const BigNum & T) const    //两个大数之间的相减运算
{
    int i,j,big;
    bool flag;
    BigNum t1,t2;
    if(*this>T)
    {
        t1=*this;
        t2=T;
        flag=0;
    }
    else
    {
        t1=T;
        t2=*this;
        flag=1;
    }
    big=t1.len;
    for(i = 0 ; i < big ; i++)
    {
        if(t1.a[i] < t2.a[i])
        {
            j = i + 1;
            while(t1.a[j] == 0)
                j++;
            t1.a[j--]--;
            while(j > i)
                t1.a[j--] += MAXN;
            t1.a[i] += MAXN + 1 - t2.a[i];
        }
        else
            t1.a[i] -= t2.a[i];
    }
    t1.len = big;
    while(t1.a[len - 1] == 0 && t1.len > 1)
    {
        t1.len--;
        big--;
    }
    if(flag)
        t1.a[big-1]=0-t1.a[big-1];
    return t1;
}

BigNum BigNum::operator*(const BigNum & T) const    //两个大数之间的相乘运算
{
    BigNum ret;
    int i,j,up;
    int temp,temp1;
    for(i = 0 ; i < len ; i++)
    {
        up = 0;
        for(j = 0 ; j < T.len ; j++)
        {
            temp = a[i] * T.a[j] + ret.a[i + j] + up;
            if(temp > MAXN)
            {
                temp1 = temp - temp / (MAXN + 1) * (MAXN + 1);
                up = temp / (MAXN + 1);
                ret.a[i + j] = temp1;
            }
            else
            {
                up = 0;
                ret.a[i + j] = temp;
            }
        }
        if(up != 0)
            ret.a[i + j] = up;
    }
    ret.len = i + j;
    while(ret.a[ret.len - 1] == 0 && ret.len > 1)
        ret.len--;
    return ret;
}
BigNum BigNum::operator/(const int & b) const    //大数对一个整数进行相除运算
{
    BigNum ret;
    int i,down = 0;
    for(i = len - 1 ; i >= 0 ; i--)
    {
        ret.a[i] = (a[i] + down * (MAXN + 1)) / b;
        down = a[i] + down * (MAXN + 1) - ret.a[i] * b;
    }
    ret.len = len;
    while(ret.a[ret.len - 1] == 0 && ret.len > 1)
        ret.len--;
    return ret;
}
int BigNum::operator %(const int & b) const    //大数对一个int类型的变量进行取模运算
{
    int i,d=0;
    for (i = len-1; i>=0; i--)
    {
        d = ((d * (MAXN+1))% b + a[i])% b;
    }
    return d;
}
BigNum BigNum::operator^(const int & n) const    //大数的n次方运算
{
    BigNum t,ret(1);
    int i;
    if(n<0)
        exit(-1);
    if(n==0)
        return 1;
    if(n==1)
        return *this;
    int m=n;
    while(m>1)
    {
        t=*this;
        for( i=1;i<<1<=m;i<<=1)
        {
            t=t*t;
        }
        m-=i;
        ret=ret*t;
        if(m==1)
            ret=ret*(*this);
    }
    return ret;
}
bool BigNum::operator>(const BigNum & T) const    //大数和另一个大数的大小比较
{
    int ln;
    if(len > T.len)
        return true;
    else if(len == T.len)
    {
        ln = len - 1;
        while(a[ln] == T.a[ln] && ln >= 0)
            ln--;
        if(ln >= 0 && a[ln] > T.a[ln])
            return true;
        else
            return false;
    }
    else
        return false;
}
```

```
bool BigNum::operator >(const int & t) const    //大数和一个int类型的变量的大小比较
{
  BigNum b(t);
  return *this>b;
}

void BigNum::print()    //输出大数
{
  int i;
  cout << a[len - 1];
  for(i = len - 2 ; i >= 0 ; i--)
  {
    cout.width(DLEN);
    cout.fill('0');
    cout << a[i];
  }
  cout << endl;
}
int main(void)
{
  int i,n;
  BigNum x[101];      //定义大数的对象数组
  x[0]=1;
  for(i=1;i<101;i++)
    x[i]=x[i-1]*(4*i-2)/(i+1);
  while(scanf("%d",&n)==1 && n!=-1) {
    x[n].print();
  }
}
```

# 6 string

## 6.1 Palindromic Tree

```
/**
回文自動機包含以下元素:

    狀態St, 所有節點的集合, 一開始兩個節點, 0表示偶數長度串的根和1表示奇數長度串的根
    last 新增加一個字符後所形成的最長回文串的節點編號
    s 當前的字符串(一開始設s[0]=-1(可以是任意一個在串S中不會出現的字符))
    n 表示添加的字符個數


每個節點代表一個不同的回文子字串, 我們在每個節點會儲存一些數值:

    len 表示所代表的回文子字串長度
    next[c] 表示所代表的回文子字串在頭尾各增加一個字符c後的回文字串其節點編號
    sufflink 表示所代表的回文子字串不包括本身的最長後綴回文子串的節點編號
    cnt(非必要) 表示所代表的回文子字串在整體字串出現的次數(在建構完成後呼叫count()才能計算)
    //num(非必要) 表示所代表的回文子字串其後綴為回文字串的個數 <== not included
**/
struct palindromic_tree{
    struct node{
        int next[26],sufflink,len; /*這些是必要的元素*/
        int l, r; // this node is s[ l .. r ]
        int cnt, num;           /*這些是額外維護的元素*/
        node(int l=0):sufflink(0),len(l),cnt(0),num(0){
            for(int i=0;i<26;++i)next[i]=0;
        }
    };
    std::vector<node> St;
    std::string s; //current string [ 1 .. n ]
    int last,n;
    palindromic_tree():St(2),last(1),n(0){
        St[0].sufflink=1;
        St[1].len=-1;
        s.push_back(-1);
    }
    inline void clear(){
        St.clear();
        s.clear();
        last=1;
        n=0;
        St.push_back(0);
        St.push_back(-1);
        St[0].sufflink=1;
        s.push_back(-1);
    }
    inline int get_sufflink(int x){
        while( s[n-St[x].len-1] != s[n] ) x=St[x].
            sufflink;
        return x;
    }
    inline void add(int c){
        s.push_back(c-='a');
        ++n;
        int cur=get_sufflink(last);
        if(!St[cur].next[c]){
            int now=St.size();
            St.push_back(St[cur].len+2);
            St[now].sufflink=St[get_sufflink(St[cur].
                sufflink)].next[c];
            /*不用擔心會找到空節點, 由證明的過程可知*/
            St[cur].next[c]=now;
            St[now].num=St[St[now].sufflink].num+1;
            St[now].l = n - St[now].len + 1, St[now].r
                = n;
        }
        last=St[cur].next[c];
        ++St[last].cnt;
    }
    inline void count(){/*cnt必須要在構造完後呼叫count
        ()去計算*/
        std::vector<node>::reverse_iterator i=St.rbegin
            ();
        for(;i!=St.rend();++i) {
            St[i->sufflink].cnt+=i->cnt;
        }
    }
    inline int size(){/*傳回其不同的回文子串個數*/
        return St.size()-2;
    }
}ptree;
```

## 6.2 Suffix Array

## 6.3 Longest Palindromic Substring

```
//ntu judge Earse
#define maxn 200001
char t[maxn];
char s[maxn * 2];
int z[maxn * 2];
int N ;
int longest_palindromic_substring() {
    // t穿插特殊字元, 存放到s。
    int n = strlen(t);
    N = n * 2 + 1;
    memset(s, '.', N);
    for (int i=0; i<n; ++i) s[i*2+1] = t[i];
    s[N] = '\0';
    z[0] = 1;    // if無須使用, then無須計算。

    int L = 0, R = 0;
    for (int i=1; i<N; ++i) // 從z[1]開始
    {
        z[i] = (R > i) ? min(z[2*L-i], R-i) : 1;
        while (i-z[i] >= 0 && i+z[i] < N &&
               s[i-z[i]] == s[i+z[i]]) z[i]++;
        if (i+z[i] > R) L = i, R = i+z[i];
    }

    /*
    // 尋找最長迴文子字串的長度
    n = 0;
    int p = 0;
    for (int i=1; i<N; ++i) // 從z[1]開始
```

```
        if (z[i] > n)
            n = z[p = i];
    */
    // longest 從中心到外端的長度 => (n-2)/2
    //cout << "最長迴文子字串的長度是" << (2*n-1) / 2;

    /*
    // 印出最長迴文子字串，記得別印特殊字元。
        for (int i=p-z[p]+1; i<=p+z[p]-1; ++i)
            if (i & 1) {
                cout << s[i];
            }
    */
    return (2*n-1)/2;
}
int nxt[maxn * 2];
int main() {
    int T;cin>>T;
    while(T--) {
        scanf("%s", t);
        #ifdef DEBUG
            cout << longest_palindromic_substring() <<
                endl;
        #else
            longest_palindromic_substring();
        #endif
        memset(nxt, -1, sizeof(nxt));
        for(int i = 0; i < N; i++) {
            nxt[ i-z[i]+1 ] = i+1;
        }
        int leftmost = 0;
        for(int i = 0; i < N; i++) {
            leftmost = max(leftmost, nxt[i]);
            nxt[i] = max(leftmost, nxt[i]);
        }
        int ans = 0;
        for(int cur = 0; cur<N-1;) {
            cur = nxt[cur];
            ans++;
        }
        cout<< ans <<endl;
    }
    return 0;
}
```

# 7 geometry

## 7.1 Point Class

```
const double eps = 1e-10;
#define N 100
struct P {
    double x, y;
    P(double _x=0, double _y=0) :x(_x), y(_y) {};
    void read() {
        scanf("%lf%lf",&x,&y);
    }
    void print() {
        printf("%f %f\n", x, y);
    }
} p[N];
bool operator <( P a, P b ) { return tie(a.x,a.y)<tie(b
    .x,b.y); }
P operator +( P a, P b ) { return P{a.x+b.x,a.y+b.y}; }
P operator -( P a, P b ) { return P{a.x-b.x,a.y-b.y}; }
P operator *(  P b, double a ) { return P{a*b.x,a*b.y};
    }
P operator /( P a, double b ) { return P{a.x/b,a.y/b};
    }
P& operator /=( P &a, double b ) { return a=a/b; }
double operator *( P a, P b ) { return a.x*b.x+a.y*b.y;
    }
double operator ^( P a, P b ) { return a.x*b.y-a.y*b.x;
    }
double x( P o, P a, P b ) { return (a-o)^(b-o); }
double dot( P o, P a, P b ) { return (a-o)*(b-o); }
```

## 7.2 Intersection of Circles/Lines/Segments

//PECaveros

```
vector<P> interCircle( P o1 , double r1 , P o2 , double
    r2 ){
  double d2 = ( o1 - o2 ) * ( o1 - o2 );
  double d = sqrt(d2);
  if( d > r1 + r2 ) return {};
  P u = (o1+o2)*0.5 + (o1-o2)*((r2*r2-r1*r1)/(2*d2));
  double A = sqrt((r1+r2+d)*(r1-r2+d)*(r1+r2-d)*(-r1+r2
      +d));
  P v = P( o1.y-o2.y , -o1.x + o2.x ) * A / (2*d2);
  return {u+v, u-v};
}
```

```
P interPnt( P p1, P p2, P q1, P q2){
  double f1 = ( p2 - p1 ) ^ ( q1 - p1 );
  double f2 = ( p2 - p1 ) ^ ( p1 - q2 );
  double f = ( f1 + f2 );
  if( fabs( f ) < eps ) return Pt( nan(""), nan("") );
  return q1 * ( f2 / f ) + q2 * ( f1 / f );
}
```

```
int ori( const PLL& o , const PLL& a , const PLL& b ){
  LL ret = ( a - o ) ^ ( b - o );
  return ret / max( 1ll , abs( ret ) );
}
// p1 == p2 || q1 == q2 need to be handled
bool banana( const PLL& p1 , const PLL& p2 ,
             const PLL& q1 , const PLL& q2 ){
  if( ( ( p2 - p1 ) ^ ( q2 - q1 ) ) == 0 ){ // parallel
    if( ori( p1 , p2 , q1 ) ) return false;
    return ( ( p1 - q1 ) * ( p2 - q1 ) ) <= 0 ||
           ( ( p1 - q2 ) * ( p2 - q2 ) ) <= 0 ||
           ( ( q1 - p1 ) * ( q2 - p1 ) ) <= 0 ||
           ( ( q1 - p2 ) * ( q2 - p2 ) ) <= 0;
  }
  return (ori( p1, p2, q1 ) * ori( p1, p2, q2 )<=0) &&
         (ori( q1, q2, p1 ) * ori( q1, q2, p2 )<=0);
}
```

## 7.3 Convex Hull

```
#define REP(i,n) for ( int i=0; i<int(n); i++ )
int n;
void input() {
    scanf("%d",&n);
    REP(i,n) p[i].read();
}


P findCenter() {
    p[n]=p[0];
    P center=P{0,0};
    REP(i,n) {
        double v=p[i]*p[i+1];
        center.x += (p[i].x+p[i+1].x)*v;
        center.y += (p[i].y+p[i+1].y)*v;
    }
    double area=0;
    REP(i,n) area+=p[i]*p[i+1];
    area /= 2;
    center /= 6*area;
    return center;
}

P q1[N],q2[N],q[N];
void convex() {
    sort(p,p+n);
    int m1=0,m2=0;
    REP(i,n) {
        while ( m1>=2 && X(q1[m1-2],q1[m1-1],p[i]) >= 0
            ) m1--;
        while ( m2>=2 && X(q2[m2-2],q2[m2-1],p[i]) <= 0
            ) m2--;
        q1[m1++]=q2[m2++]=p[i];
    }
    int m=0;
    REP(i,m1) q[m++]=q1[i];
```

```
    for ( int i=m2-2; i>=1; i-- ) q[m++]=q2[i];
    q[m]=q[0];
}
void solve() {
    convex();
    // continue ...
}
```

## 7.4 Half Plane Intersection

```
//http://acm.csie.org/ntujudge/problemdata/2575.pdf
//http://www.csie.ntnu.edu.tw/~u91029/Half-
    planeIntersection.html

/**
預先使用四個半平面，設定一個極大的正方形邊界，讓半平面
    交集擁有邊界。
二、逐一加入每個半平面，求出當下的半平面交集（凸多邊
    形）。
online 演算法，隨時維護一個半平面交集。每次更新需時 O(N
    )，總時間複雜度為 O(N^2)，  N 是半平面數目。
**/

#include <bits/stdc++.h>
using namespace std;
#define mp make_pair

typedef complex<double> Point;
typedef vector<Point> Polygon;
typedef pair<Point,Point> Line;
#define x real()
#define y imag()

// 兩向量叉積
double cross(Point& a, Point& b) {
    return a.x * b.y - a.y * b.x;
}

// 向量oa與向量ob進行叉積
double cross(Point& o, Point& a, Point& b) {
    return (a.x-o.x) * (b.y-o.y) - (a.y-o.y) * (b.x-o.x
        );
}

// 多邊形面積
double area(Polygon& p) {
    double a = 0;
    int n = p.size();
    for (int i=0; i<n; ++i)
        a += cross(p[i], p[(i+1)%n]);
    return fabs(a) / 2;
}

// 兩線交點
Point intersection(Point& a1, Point& a2, Point& b1,
    Point& b2) {
    Point a = a2 - a1, b = b2 - b1, s = b1 - a1;
    return a1 + a * cross(b, s) / cross(b, a);
}

// 一個凸多邊形與一個半平面的交集
Polygon halfplane_intersection(Polygon& p, Line& line)
    {
    Polygon q;
    Point p1 = line.first, p2 = line.second;

    // 依序窮舉凸多邊形所有點，判斷是否在半平面上。
    // 如果凸多邊形與半平面分界線有相交，就求交點。
    int n = p.size();
    for (int i=0; i<n; ++i)
    {
        double c = cross(p1, p2, p[i]);
        double d = cross(p1, p2, p[(i+1)%n]);
        if (c >= 0) q.push_back(p[i]);
        if (c * d < 0) q.push_back(intersection(p1, p2,
            p[i], p[(i+1)%n]));
    }
    return q;
}
#define maxn 550
//Line line[maxn];
Point v[maxn];
double ans[maxn];
int main() {
    int T;cin>>T;
    while(T--) {
        int n;
        double  w, h;
        scanf("%d %lf %lf", &n, &w, &h);
        // 預先設定一個極大的正方形邊界
        Polygon p, org;
        /** initialize
         p.push_back(Point(-1e9,-1e9));
        p.push_back(Point(-1e9,+1e9));
        p.push_back(Point(+1e9,-1e9));
        p.push_back(Point(+1e9,+1e9));
        **/
        p.push_back(Point(0,0));
        p.push_back(Point(0,h));
        p.push_back(Point(w,h));
        p.push_back(Point(w,0));
        org = p;
        for(int i =0;i < n;i ++) {
            double a, b;
            scanf("%lf %lf", &a, &b);
            v[i] = Point(a, b);
        }
        // 每一個半平面都與目前的半平面交集求交集
        for (int i=0; i<n; ++i)
        {
            p = org;
            for(int j = 0; j < n; j++) {
                if(i==j) continue;
                Line line;
                // find perpendicular line to line i_j
                Point a( (v[i].x+v[j].x)/2, (v[i].y+v[j
                    ].y)/2 );
                Point b(a.x+(v[i].y-v[j].y), a.y-(v[i].
                    x-v[j].x));

                line = cross(a, b, v[i]) >=0 ? mp(a, b)
                    : mp(b,a);
                p = halfplane_intersection(p, line);
                if (area(p) == 0) break;     // 退化或者
                    空集合
            }
            ans[i] = area(p);
        }
        for(int i = 0;i < n;i ++) printf("%.9f\n", ans[
            i]);
    }
}
/*
10
3 4 4
1 1 2 2 3 3
*/
```