# Contents

# 1 Setting

## 1.1 Header

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

using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int, int> pii;
typedef pair<ll,ll> pll;

#define Fi first
#define Se second
#define pb(x) push_back(x)
#define sz(x) (int)x.size()
#define rep(i, n) for(int i=0;i<n;i++)
#define repp(i, n) for(int i=1;i<=n;i++)
#define all(x) x.begin(), x.end()

#define VA_NUM_ARGS(...) VA_NUM_ARGS_IMPL_((0,__VA_ARGS__, 6,5,4,3,2,1))
#define VA_NUM_ARGS_IMPL_(tuple) VA_NUM_ARGS_IMPL tuple
#define VA_NUM_ARGS_IMPL(_0,_1,_2,_3,_4,_5,_6,N,...) N
#define macro_dispatcher(macro, ...) macro_dispatcher_(macro, VA_NUM_ARGS(
  __VA_ARGS__))
#define macro_dispatcher_(macro, nargs) macro_dispatcher__(macro, nargs)
#define macro_dispatcher__(macro, nargs) macro_dispatcher___(macro, nargs)
#define macro_dispatcher___(macro, nargs) macro ## nargs
#define Debug1(a)           cout<<#a<<"="<<(a)<<"\n"
#define Debug2(a,b)         cout<<#a<<"="<<(a)<<", "<<#b<<"="<<(b)<<"\n"
#define Debug3(a,b,c)       cout<<#a<<"="<<(a)<<", "<<#b<<"="<<(b)<<", "<<#c
  <<"="<<(c)<<"\n"
#define Debug4(a,b,c,d)     cout<<#a<<"="<<(a)<<", "<<#b<<"="<<(b)<<", "<<#c
  <<"="<<(c)<<", "<<#d<<"="<<(d)<<"\n"
#define Debug5(a,b,c,d,e)   cout<<#a<<"="<<(a)<<", "<<#b<<"="<<(b)<<", "<<#c
  <<"="<<(c)<<", "<<#d<<"="<<(d)<<", "<<#e<<"="<<(e)<<"\n"
#define Debug6(a,b,c,d,e,f) cout<<#a<<"="<<(a)<<", "<<#b<<"="<<(b)<<", "<<#c
  <<"="<<(c)<<", "<<#d<<"="<<(d)<<", "<<#e<<"="<<(e)<<", "<<#f<<"="<<(f)<<"\n"
#define Debug(...) macro_dispatcher(Debug, __VA_ARGS__)(__VA_ARGS__)
#define DA(a,n) cout<<#a<<"=["; printarray(a,n); cout<<"]\n"
#define DAR(a,n,s) cout<<#a<<"["<<s<<"-"<<n-1<<"]=["; printarray(a,n,s); cout
  <<"]\n"

#define TT1 template<class T>
#define TT1T2 template<class T1, class T2>
```

```
#define TT1T2T3 template<class T1, class T2, class T3>
template<class T, size_t N> ostream& operator << (ostream& os, const array<T, N
  >& v);
TT1T2 ostream& operator << (ostream& os, const pair<T1, T2>& p){ return os
  <<"("<<p.first<<", "<< p.second<<")"; }
TT1 ostream& operator << (ostream& os, const vector<T>& v){       bool f=1;os
  <<"[";for(auto& i : v) { if (!f)os << ", ";os<<i;f=0;}return os << "]"; }
template<class T, size_t N> ostream& operator << (ostream& os, const array<T, N
  >& v) {     bool f=1;os<<"[";for(auto& i : v) { if (!f)os << ", ";os<<i;f=0;}
  return os << "]"; }
TT1T2 ostream& operator << (ostream& os, const set<T1, T2>&v){    bool f=1;os
  <<"[";for(auto& i : v) { if (!f)os << ", ";os<<i;f=0;}return os << "]"; }
TT1T2 ostream& operator << (ostream& os, const multiset<T1,T2>&v){bool f=1;os
  <<"[";for(auto& i : v) { if (!f)os << ", ";os<<i;f=0;}return os << "]"; }
TT1T2T3 ostream& operator << (ostream& os, const map<T1,T2,T3>& v){ bool f = 1;
  os << "["; for (auto& ii : v) { if (!f)os << ", "; os << "(" << ii.first << "
  -> " << ii.second << ") "; f = 0; }return os << "]"; }
TT1T2 ostream& operator << (ostream& os, const multimap<T1, T2>& v){ bool f = 1;
   os << "["; for (auto& ii : v) { if (!f)os << ", "; os << "(" << ii.first << "
   -> " << ii.second << ") "; f = 0; }return os << "]"; }
TT1T2 ostream& operator << (ostream& os, priority_queue<T1, T2> v) { bool f = 1;
   os << "["; while (!v.empty()) { auto x = v.top(); v.pop(); if (!f) os << ",
  "; f = 0; os << x; } return os << "]"; }
TT1T2 void printarray(const T1& a, T2 sz, T2 beg = 0){ for (T2 i = beg; i<sz; i
  ++) cout << a[i] << " "; cout << endl; }
```

### 1.2 vimrc

```
syntax on
set nu ai ci si nobk et ar ru nocp hls
set bs=2 ts=4 sw=4 sts=4
set cb=unnamed
set mouse=an
command PS vsp %:r.in|sp %:r.out|vert res 30|wa
command RIO wall|!g++ -O2 -std=c++14 -Wall -lm %:r.cpp && ./a.out < %:r.in > %:r
  .out
command RI  wall|!g++ -O2 -std=c++14 -Wall -lm %:r.cpp && ./a.out < %:r.in
```

### 1.3 Sublime text

```
{
    "shell_cmd": "g++ -O2 -std=c++11 \"${file}\" -o \"${file_path}/${
      file_base_name}\" && \"${file_path}/${file_base_name}\" < input.txt",
    "working_dir": "${file_path}",
    "selector": "source.c++",
}
```

## 2 String

### 2.1 KMP

```
vector<int> preprocess(string p){
    int m = p.size();
    vector<int> fail(m);
    fail[0] = 0; int j = 0;
    for(int i=1;i<m;i++){
        while(j>0&&p[i]!=p[j]) j = fail[j-1];
        if( p[i] == p[j] ){
            fail[i] = j+1; j++;
        }else{
            fail[i] = 0;
        }
    }
    return fail;
}

vector<int> kmp(string s, string p){
    auto fail = preprocess(p);
    vector<int> ans; int n = s.size(), m = p.size();
    int j = 0;
    for(int i=0;i<n;i++){
        while(j>0 && s[i]!=p[j]) j = fail[j-1];
        if( s[i] == p[j] ){
            if( j == m-1 ){
                ans.pb(i-m+1); j = fail[j];
            }else{
                j++;
            }
        }
    }
    return ans;
}
```

### 2.2 Aho Chorasick

```
struct AhoCorasick{
    struct Node{
        int fail;
        vector<int> output;
        int children[26];

        Node(){
            for(int i=0;i<26;i++) children[i] = -1;
            fail = -1;
        }
    };

    vector<Node> trie;
    int new_node(){
        Node x;
        trie.push_back(x);
        return (int)trie.size()-1;
    }

    void add(int node, string &s, int idx, int string_num){
        //cout << node << " " << idx << endl;
        if( idx == s.size() ){
            trie[node].output.push_back(string_num);
            return;
        }
        int c = s[idx] - 'a';
        if( trie[node].children[c] == -1 ){
            int next = new_node();
            trie[node].children[c] = next;
        }

        add(trie[node].children[c], s, idx+1, string_num);
    }

    void build(vector<string> v){
        int root = new_node();
        for(int i=0;i<v.size();i++){
            add(root,v[i],0,i);
        }

        queue<int> q;
        q.push(root); trie[root].fail = root;
        while( !q.empty() ){
            int cur = q.front(); q.pop();
            for(int i=0;i<26;i++){
                int next = trie[cur].children[i];
                if( next == -1 ) continue;

                // build fail
                if( cur == root ){
                    trie[next].fail = root;
                }
                else{
                    int x = trie[cur].fail;
                    while( x != root && trie[x].children[i] == -1 ) x = trie[x].
                      fail;
                    if( trie[x].children[i] != -1 ) x = trie[x].children[i];
                    trie[next].fail = x;
                }
                // build output
                int f = trie[next].fail;
                for(auto e : trie[f].output) trie[next].output.push_back(e);
                q.push(next);
            }
        }
    }

    vector<Pi> find(string s){
        int n = (int) s.size();
        int cur = 0, root = 0;
        vector<Pi> ans;
        for(int i=0;i<n;i++){
            int c = s[i]-'a';
            while( cur != root && trie[cur].children[c] == -1 ) cur = trie[cur].
              fail;
            if( trie[cur].children[c] != -1 ) cur = trie[cur].children[c];

            for(auto e : trie[cur].output){
                ans.push_back({e,i});
            }
        }
        return ans;
    }
};
```

## 2.3 Suffix array

```
namespace Suffix {
    static const int MX = 100010;
    int RA[MX<<1], t[MX], C[MX];

    void build_SA(int N, char A[], int SA[], int LCP[]){
        int cnt = 130;
        for(int i=1;i<=N;i++)RA[i] = A[i];
        for(int i=1;i<=N;i++)C[RA[i]]++;
        for(int i=2;i<=cnt;i++)C[i] += C[i-1];
        for(int i=1;i<=N;i++)SA[C[RA[i]]--] = i;
        for(int i=1;i<=cnt;i++)C[i] = 0;
        for(int L=1;;L<<=1){
            int z = 0;
            for(int i=N-L+1;i<=N;i++)t[++z] = i;
            for(int i=1;i<=N;i++)if(SA[i] > L)t[++z] = SA[i] - L;
            for(int i=1;i<=N;i++)C[RA[i]]++;
            for(int i=2;i<=cnt;i++)C[i] += C[i-1];
            for(int i=N;i;i--)SA[ C[RA[t[i]]]-- ] = t[i];
            for(int i=1;i<=cnt;i++)C[i] = 0;
            cnt = 1;
            for(int i=1;i<=N;i++){
                if(i != 1 && RA[SA[i]] == RA[SA[i-1]] && RA[SA[i] + L] == RA[SA[
                  i-1] + L])C[SA[i]] = cnt-1;
                else C[SA[i]] = cnt++;
            }
            for(int i=1;i<=N;i++)RA[i] = C[i], C[i] = 0;
            if(cnt == N+1)break;
        }
        for(int i=1, L=0;i<=N;i++, L=(L?L-1:0)){
            if(RA[i] == N)continue;
            int t = SA[RA[i]+1];
            while(A[i+L] == A[t+L])++L;
            LCP[RA[i]] = L;
        }
    }
};
```

## 2.4 Manacher's algorithm

```
// finds radius of longest palindrome centered at s[i]
// If you also want to find even-length paindromes, use dummy characters
// baab -> #b#a#a#b#
vector<int> ManacherAlgorithm(string s){
    int n = (int) s.size();
    int p = -1, r = -1;
    vector<int> A(n);
    for(int i=0;i<n;i++){

        if( r < i ){
            A[i] = 0;
            int j = 0;
            while( i + A[i] < n && i - A[i] >= 0 && s[ i+A[i] ] == s[ i-A[i] ] )
               A[i]++;
            A[i]--;
        }
        else{
            A[i] = min( A[2*p - i] , r-i );
            while( i + A[i] < n && i - A[i] >= 0 && s[ i+A[i] ] == s[ i-A[i] ] )
               A[i]++;
            A[i]--;
        }

        // update r
        if( r < i + A[i] ){
            r = i + A[i];
            p = i;
        }
    }
    return A;
}
```

### 2.5 Z algorithm

```
// Calculates LCP[i] for all 0 <= i < n
vector<int> Zalgorithm(string s){
    int l=0, r=0;
    int n = (int) s.size();
    vector<int> Z(n);
    Z[0] = n;
    for(int i=1; i<n; i++){
        // reset and calculate again
        if( i > r ){
            l = r = i;
            while( r<n && s[r] == s[r-l] ) r++;
            r--;
            Z[i] = r-l+1;
        }

        // extend [l,r]
        else{
            int k = i-l;
            // not enough matching at position k
            if( Z[k] < r-i+1 ) Z[i] = Z[k];
            // enough matching. extend [l,r]
            else{
                l = i;
                while( r<n && s[r] == s[r-l] ) r++;
                r--;
                Z[i] = r-l+1;
            }
        }

    }
    return Z;
};
```

## 3 Graph & Flow

### 3.1 BCC

```
int N,M;
int timer = 0;
vector<int> E[300500];
int vis[300500], low[300500];

// dfs1 is to fill vis(discover time) and low array
int dfs1(int x, int pa){
    vis[x] = ++timer;
    low[x] = vis[x];
    for(auto e : E[x])if(e!=pa){
        if( vis[e] ){
            low[x] = min(low[x], vis[e]);
        }
        else{
            dfs1(e,x);
            low[x] = min(low[x], low[e]);
        }
    }
    return low[x] ;
}

int color = 0;
vector<int> colors[300500], E2[300500];
int vis2[300500];

// dfs2 is to color every nodes
// Store node's colors into colors array
// Store new edges into E2
void dfs2(int x, int pa, int c){
    colors[x].pb(c);
    vis2[x] = 1;
    for(auto e : E[x])if(!vis2[e]){
        // x-e is an articulation edge
        if( low[e] > vis[x] ){
            ++color;
            colors[x].pb(color);
            E2[c].pb(color); E2[color].pb(c);
            dfs2(e,x,color);
        }
        // x-e is not an articulation edge
        else dfs2(e,x,c);

    }
}

int main(){
    geti(N,M);
    repp(i,M){
        int a, b; geti(a,b);
        E[a].pb(b); E[b].pb(a);
    }
    // fill vis & low
    dfs1(1,-1);
    // find out articulation edge and color of nodes
    color = 1;
    dfs2(1,-1,color);
}
```

## 3.2 Hopcroft Karp

```
namespace Matching{
//matching [1...n] <-> [1...m]
const int MX = 40040, MY = 40040;
vector <int> E[MX];
int xy[MX], yx[MY];
int n, m;

void addE(int x, int y) { E[x].pb(y); }
void setnm(int sn, int sm) { n = sn; m = sm; }

int tdis[MX], que[MX], *dis = tdis + 1;
int bfs() {
  int *fr = que, *re = que;
  for(int i=1;i<=n;i++) {
    if(xy[i] == -1) *fr++ = i, dis[i] = 0;
    else dis[i] = -1;
  }
  dis[-1] = -1;
  while(fr != re) {
    int t = *re++;
    if(t == -1) return 1;
    for(int e : E[t]) {
      if(dis[yx[e]] == -1) dis[yx[e]] = dis[t] + 1, *fr++ = yx[e];
    }
  }
  return 0;
}

int dfs(int x) {
  for(int e : E[x]) {
    if(yx[e] == -1 || (dis[yx[e]] == dis[x] + 1 && dfs(yx[e]))) {
      xy[x] = e;
      yx[e] = x;
      return 1;
    }
  }
  dis[x] = -1;
  return 0;
}

int Do() {
  memset(xy, -1, sizeof xy);
  memset(yx, -1, sizeof yx);

  int ans = 0;
  while(bfs()) {
    for(int i=1;i<=n;i++) if(xy[i] == -1 && dfs(i)) ++ans;
  }
  return ans;
}
}


void solve(){
  int n, m;
  scanf("%d%d", &n, &m);
  Matching::setnm(n, m);
  for(int i=1;i<=n;i++) {
    int x; scanf("%d", &x);
    while(x--) {
      int y; scanf("%d", &y);
      Matching::addE(i, y);
    }
  }
  printf("%d\n", Matching::Do());
}
```

## 3.3 Dinic

```
struct MaxFlowDinic{
    struct Edge{
        // next, inv, residual
        int to, inv; ll res;
    };

    int n;
    vector<vector<Edge>> graph;

    vector<int> lev,work;

    void init(int x){
        n = x+10;
        graph.resize(x+10);
        lev.resize(n); work.resize(n);
    }

    void make_edge(int s, int e, ll cap, ll caprev = 0){
        Edge forward = {e, (int)graph[e].size(), cap};
        Edge backward = {s, (int)graph[s].size(), caprev};
        graph[s].push_back(forward);
        graph[e].push_back(backward);
    }

    bool bfs(int source, int sink){
        queue<int> q;
        for(auto& e : lev) e = -1;
        lev[source] = 0; q.push(source);
        while(!q.empty()){
            int cur = q.front(); q.pop();
            for(auto e : graph[cur]){
                if(lev[e.to]==-1 && e.res > 0){
                    lev[e.to] = lev[cur]+1;
                    q.push(e.to);
                }
            }
        }
        return lev[sink] != -1;
    }

    ll dfs(int cur, int sink, ll flow){
        if( cur == sink ) return flow;
        for(int &i = work[cur]; i < (int)graph[cur].size(); i++){
            Edge &e =  graph[cur][i];
            if( e.res == 0 || lev[e.to] != lev[cur]+1 ) continue;
            ll df = dfs(e.to, sink, min(flow, e.res) );
            if( df > 0 ){
                e.res -= df;
                graph[e.to][e.inv].res += df;
                return df;
            }
        }
        return 0;
    }


    ll solve( int source, int sink ){
        ll ans = 0;
        while( bfs(source, sink) ){
            for(auto& e : work) e = 0;
            while( true ){
                ll flow = dfs(source,sink,54321987654321LL);
                if( flow == 0 ) break;
                ans += flow;
            }
        }
        return ans;
    }

};
```

### 3.4 MCMF

```
struct MCMF{
    struct edge{
        int to, inv, cap, flow, cost;
        int res(){
            return cap - flow;
        }
    };

    vector<vector<edge>> graph;
    vector<int> pv, pe;
    vector<int> dist, inq;

    void init(int x){
        graph.resize(x+10);
        pv.resize(x+10); pe.resize(x+10);
        dist.resize(x+10);
        inq.resize(x+10);
    }

    void make_edge(int from, int to, int cap, int cost){
        //printf("%d -> %d | cost = %d\n",from,to,cost);
        edge forward = {to, (int)graph[to].size(), cap, 0, cost};
        edge backward = {from, (int)graph[from].size(), 0, 0, -cost};
        graph[from].push_back(forward);
        graph[to].push_back(backward);
    }

    int solve(int source, int sink){
        int ans = 0;
        int totalflow = 0;
        while(true){
            for(auto& e : dist) e = INF;
            for(auto& e : inq) e = 0;
            queue<int> q;
            q.push(source); inq[source] = 1;
            dist[source] = 0;

            while(!q.empty()){
                int cur = q.front(); q.pop();
                inq[cur] = 0;
                for(int i=0;i<(int)graph[cur].size();i++){
                    auto& e = graph[cur][i];
                    if( e.res() > 0 && dist[e.to] > dist[cur] + e.cost ){
                        dist[e.to] = dist[cur] + e.cost;
                        pv[e.to] = cur; pe[e.to] = i;
                        if( inq[e.to] == 0 ){
                            q.push(e.to); inq[e.to] = 1;
                        }
                    }
                }
            }

            if( dist[sink] == INF ) break;

            // add this limit when we don't require maxflow
            //if( dist[sink] > 0 ) break;

            int mnflow = INF;
            for( int v = sink; v != source; v = pv[v] ){
                mnflow = min( mnflow, graph[pv[v]][pe[v]].res() );
            }

            for( int v = sink; v != source; v = pv[v] ){
                int tmp = graph[pv[v]][pe[v]].inv;
                graph[pv[v]][pe[v]].flow += mnflow;
                graph[v][tmp].flow -= mnflow;
            }
            totalflow += mnflow;
            ans += dist[sink] * mnflow;
        }
        return ans;
    }

};
```

## 3.5 Blossom

```
namespace Blossom {
    // from http://codeforces.com/blog/entry/49402
    const int MAX_N = 550;
    const int MAX_M = 130000;
    struct struct_edge{int v;struct_edge* n;};
    typedef struct_edge* edge;
    struct_edge pool[MAX_M*2];
    edge top,adj[MAX_N];
    int V,E,match[MAX_N],qh,qt,q[MAX_N],father[MAX_N],base[MAX_N];
    bool inq[MAX_N],inb[MAX_N],ed[MAX_N][MAX_N];
    void add_edge(int u,int v)
    {
        top->v=v,top->n=adj[u],adj[u]=top++;
        top->v=u,top->n=adj[v],adj[v]=top++;
    }
    int LCA(int root,int u,int v)
    {
        static bool inp[MAX_N];
        rep(i, V) inp[i] = 0;
        while(1)
        {
            inp[u=base[u]]=true;
            if (u==root) break;
            u=father[match[u]];
        }
        while(1)
        {
            if (inp[v=base[v]]) return v;
            else v=father[match[v]];
        }
    }
    void mark_blossom(int lca,int u)
    {
        while (base[u]!=lca)
        {
            int v=match[u];
            inb[base[u]]=inb[base[v]]=true;
            u=father[v];
            if (base[u]!=lca) father[u]=v;
        }
    }
    void blossom_contraction(int s,int u,int v)
    {
        int lca=LCA(s,u,v);
        rep(i, V) inb[i] = 0;
        mark_blossom(lca,u);
        mark_blossom(lca,v);
        if (base[u]!=lca)
            father[u]=v;
        if (base[v]!=lca)
            father[v]=u;
        for (int u=0;u<V;u++)
            if (inb[base[u]])
            {
                base[u]=lca;
                if (!inq[u])
                    inq[q[++qt]=u]=true;
            }
    }
    int find_augmenting_path(int s)
    {
        rep(i, V) father[i] = -1, inq[i] = 0;
        for (int i=0;i<V;i++) base[i]=i;
        inq[q[qh=qt=0]=s]=true;
        while (qh<=qt)
        {
            int u=q[qh++];
            for (edge e=adj[u];e;e=e->n)
            {
                int v=e->v;
                if (base[u]!=base[v]&&match[u]!=v){
                    if ((v==s)||(match[v]!=-1 && father[match[v]]!=-1))
                        blossom_contraction(s,u,v);
                    else if (father[v]==-1)
                    {
                        father[v]=u;
                        if (match[v]==-1)
                            return v;
                        else if (!inq[match[v]])
                            inq[q[++qt]=match[v]]=true;
                    }
                }
            }
        }
        return -1;
    }
    int augment_path(int s,int t)
    {
        int u=t,v,w;
        while (u!=-1)
        {
            v=father[u];
            w=match[v];
            match[v]=u;
            match[u]=v;
            u=w;
        }
        return t!=-1;
    }
    int edmonds()
    {
        int matchc=0;
        rep(i, V) match[i] = -1;
        for (int u=0;u<V;u++)
            if (match[u]==-1)
                matchc+=augment_path(u,find_augmenting_path(u));
        return matchc;
    }
    void solve(int n, vector <pii> Ed, vector <pii> &Mat) { // 1-based
        Mat.clear();
        if(szz(Ed) == 0) return;
        int m = szz(Ed);
        rep(i, n) rep(j, n) ed[i][j] = false;
        top=pool;
        rep(i, m*2) pool[i].v = 0, pool[i].n = NULL;
        rep(i, n) adj[i] = NULL;
        rep(i, n) match[i] = q[i] = father[i] = base[i] = 0;
        rep(i, n) inq[i] = inb[i] = 0;
        qh = qt = 0;
        V = n, E = m;
        rep(i, m) {
            int x = Ed[i].Fi - 1;
            int y = Ed[i].Se - 1;
            add_edge(x, y);
            ed[x][y] = ed[y][x] = true;
        }
        edmonds();
        rep(i, V) if(i < match[i]) Mat.emplace_back(i + 1, match[i] + 1);
    }
}
```

## 3.6 Stoer Wagner

```
namespace stoer_wagner{
    const int MX = 505;
    int G[MX][MX], vst[MX], n;

    void init(int nn){ n = nn; memset(G, 0, sizeof G); }
    void add_edge(int a, int b, int d){ if(a != b) G[a][b] = G[b][a] = d; }

    pii minimum_cut_phase(int st, int &res){
        int dist[MX] = {}, vis[MX];
        int cur = 1e9, s = st, e = -1;
        memcpy(vis, vst, sizeof vst);
        dist[st] = 1e9;
        while(1){
            int mx = 0;
            for(int i=1;i<=n;i++) if(!vis[i] && (!mx || dist[mx] < dist[i])) mx
              = i;
            if(mx == 0) break;
            cur = dist[mx]; e = s; s = mx; vis[mx] = 1;
            for(int i = 1; i <= n; i++) dist[i] += G[mx][i];
        }
        res = min(res, cur);
        return pii(s, e);
    }
    int run(){
        if(n <= 1) return 0;
        memset(vst, 0, sizeof vst);
        int res = 1e9, t = 1, u;
        for(int i = 0; i < n-1; i++){
            tie(t, u) = minimum_cut_phase(t, res);
            vst[u] = 1;
            for(int i = 1; i <= n; i++){
                if(vst[i] || t == i) continue;
                G[t][i] += G[u][i]; G[i][t] += G[u][i];
            }
        }
        return res;
    }
};
```

## 3.7 Dominator Tree

```
#include<vector>
using namespace std;
#define pb(x) push_back(x)
namespace dtree{
const int MAXN = 100010;
vector <int> E[MAXN];
vector <int> RE[MAXN], rdom[MAXN];

int S[MAXN], RS[MAXN], cs;
int par[MAXN], val[MAXN];
int sdom[MAXN], rp[MAXN];
int dom[MAXN];

int Find(int x, int c = 0) {
  if(par[x] == x) return c ? -1 : x;
  int p = Find(par[x], 1);
  if(p == -1) return c ? par[x] : val[x];
  if(sdom[val[x]] > sdom[val[par[x]]]) val[x] = val[par[x]];
  par[x] = p;
  return c ? p : val[x];
}

void Union(int x, int y) {
  par[x] = y;
}

void dfs(int x) {
  RS[ S[x] = ++cs ] = x;
  par[cs] = sdom[cs] = val[cs] = cs;
  for(int e : E[x]) {
    if(S[e] == 0) dfs(e), rp[S[e]] = S[x];
    RE[S[e]].pb(S[x]);
  }
}

int Do(int s, int *up) {
  dfs(s);
  for(int i=cs;i;i--) {
    for(int e : RE[i]) sdom[i] = min(sdom[i], sdom[Find(e)]);
    if(i > 1) rdom[sdom[i]].pb(i);
    for(int e : rdom[i]) {
      int p = Find(e);
      if(sdom[p] == i) dom[e] = i;
      else dom[e] = p;
    }
    if(i > 1) Union(i, rp[i]);
  }
  for(int i=2;i<=cs;i++) if(sdom[i] != dom[i]) dom[i] = dom[dom[i]];
  for(int i=2;i<=cs;i++) {
    up[RS[i]] = RS[dom[i]];
  }
  return cs;
}

void addE(int x, int y) { E[x].pb(y); }
}
```

## 3.8 LR-flow

```
G has a feasible (s,t)-flow iff G' has a saturating (s',t')-flow
in G' total capacity out of s' and into t' are both D (sum of demands)
saturating flow : flow with value exactly D.

1. Make new source, new sink (s', t')

2. for every v:
c'(s'->v) = sum{ d(u->v) } (give demands into v)
c'(v->t') = sum{ d(v->w) } (take demands out of v)

3. for every u->v:
c'(u->v) = c(u->v) - d(u->v)  (difference of cap, demand)

4. make t->s cap:INF

maxflow mf;
lint lsum;
void clear(){
        lsum = 0;
        mf.clear();
}
void add_edge(int s, int e, int l, int r){
        lsum += l;
        mf.add_edge(s + 2, e + 2, r - l);
        mf.add_edge(0, e + 2, l);
        mf.add_edge(s + 2, 1, l);
}
bool solve(int s, int e){
        mf.add_edge(e+2, s+2, 1e9); // to reduce as maxflow with lower bounds,
          in circulation problem skip this line
        return lsum == mf.match(0, 1);
        // to get maximum LR flow, run maxflow from s+2 to e+2 again
}
```

# 4 Query

## 4.1 Splay Tree2

```
void Rotate(node *x) {
    node *p = x->p;
    node *b;
    if (x == p->l) {
        p->l = b = x->r;
        x->r = p;
    } else {
        p->r = b = x->l;
        x->l = p;
    }
    x->p = p->p;
    p->p = x;
    if (b) b->p = p;
    (x->p ? p == x->p->l ? x->p->l : x->p->r : tree) = x;
}

void Splay(node *x) {
    while (x->p) {
        node *p = x->p;
        node *g = p->p;
        if (g) Rotate((x == p->l) == (p == g->l) ? p : x);
        Rotate(x);
    }
}
```

## 4.2 Link Cut Tree

```
#define _CRT_SECURE_NO_WARNINGS
#include<algorithm>
#include<stdio.h>

using namespace std;
const int N_ = 2e5;

struct node{
  void pushup(){
    cnt = (link[0]? link[0]->cnt:0) + (link[1]? link[1]->cnt:0) + 1;
    mx = max( max( link[0]? link[0]->mx:0, link[1]? link[1]->mx:0 ), val);
  }

  int cnt, val, mx; //cnt: number of nodes
  node *link[2], *par, *path_parent;
};

struct linkcuttree{
  node N[ N_ ];

  void clear(int s){
    for(int i=0;i<=s;i++)
      N[i].link[0] = N[i].link[1] = N[i].par = N[i].path_parent = 0, N[i].cnt =
        1;
  }

  inline int dir(node *x){ return x->par->link[0] != x; }
  inline int cnt(node *x){ return x?x->cnt:0; }
  inline int mx(node *x){ return x?x->mx:0; }

  void rotate(node *n) // To
  {
    if( !n->par ) return;
    node *p = n->par;
    int d = dir(n);
    n->path_parent = p->path_parent; p->path_parent = NULL;
    p->link[d] = n->link[!d];   if( n->link[!d] ) n->link[!d]->par = p;
    n->par = p->par;  if( p->par ) p->par->link[ dir(p) ] = n;
    n->link[!d] = p;  p->par = n;
    p->pushup(); n->pushup();
  }

  void splay(node *x){
    while( x->par ){
      if( !x->par->par );
      else if(dir(x) == dir(x->par)) rotate(x->par);
      else rotate(x);
      rotate(x);
    }
  }

  void access(node* x)
  {
    splay(x);
    if( x->link[1] ) x->link[1]->path_parent = x, x->link[1]->par = NULL;
    x->link[1] = NULL; x->pushup();
    while( x->path_parent ){
      node *pp = x->path_parent, *r;
      splay(pp);
      r = pp->link[1];
      if( r ) r->par = NULL, r->path_parent = pp;
      pp->link[1] = x; pp->pushup(); x->par = pp;
      x->path_parent = NULL;
      splay(x);
    }
  }

  void cut(int u)
  {
    access(N+u);
    if( N[u].link[0] ) N[u].link[0]->par = NULL;
    N[u].link[0] = NULL; N[u].pushup();
  }

  void link(int u, int v) // u must be root.
  {
    if( u == v ) return;
    access(N+u);
    access(N+v);
    //assert(!N[u].link[0]);
    N[u].link[0] = N+v; N[v].par = N+u; N[u].pushup();
  }

// recommend: 'dont copy & paste code below.
  int read(int u)
  {
    access( N+u );
    return N[u].cnt;
}

  int root(int u)
  {
    access( N+u );
    node* ans = N+u;
    while( ans->link[0] ) ans = ans->link[0];
    splay(ans);
    return ans - N;
  }

  int mx(int u)
  {
    access( N+u );
    return N[u].max;
  }

  bool chk()
  {
    for(int i=0;i<N_;i++){
      if( N[i].cnt == 0 ) return true;
      if( N[i].cnt != cnt(N[i].link[0]) + cnt(N[i].link[1]) + 1) return false;
      if( N[i].mx != max( max( mx(N[i].link[0]), mx(N[i].link[1]) ), N[i].val) )
         return false;
      if( N[i].par && N+i != N[i].par->link[dir(N+i)] ) return false;
      if( N[i].link[0] && N+i != N[i].link[0]->par) return false;
      if( N[i].link[1] && N+i != N[i].link[1]->par) return false;
    }
    return true;
  }
}LCT;
```

## 4.3 HLD

```
int N,K,M,tc,T;

struct segTree{ //range max query
    vector<int> v; int n;
    void init(int _n){
        _n+=3; v.resize(_n*2+10); n = _n;
    }
    void update(int x, int val){
        for(v[x+=n]=val;x>1;x>>=1) v[x/2] = max(v[x],v[x^1]);
    }
    int query(int l, int r){ // [l,r]
        r++; //to make range as [l,r+1)
        int res = 0;
        for(l+=n,r+=n;l<r;l>>=1,r>>=1){
            if( l&1 ) res = max(res,v[l++]);
            if( r&1 ) res = max(res,v[--r]);
        }
        return res;
    }
};

#define MAXV 100500
#define LOGV 18
// cNo: node# -> hld# mapping
int cNo[MAXV];
```

```
// other arrays are accesed using hld#
int cPos[MAXV], cSize[MAXV], cHead[MAXV], cN; int cLeaf[MAXV];
vector<Pi> E[MAXV]; int pa[LOGV][MAXV]; int sz[MAXV]; int val[MAXV]; int level[
  MAXV];
bool vis[MAXV]; vector<segTree> tree; vector<Pi> edges;
int dfs_build(int x, int p, int v, int lev){
    pa[0][x] = p; sz[x] = 1; val[x] = v; level[x] = lev;
    for(auto e : E[x])if(e.Fi!=p){
        sz[x] += dfs_build(e.Fi,x,e.Se,lev+1);
    }
    return sz[x];
}
void lca_build(){
    for(int k=1;k<LOGV;k++){
        repp(i,N){
            if( pa[k-1][i] != -1 )pa[k][i] = pa[k-1][pa[k-1][i]];
            else pa[k][i] = -1;
        }
    }
}
int lca(int x, int y){
    if( level[x] < level[y] ) swap(x,y);
    int diff = level[x] - level[y];
    for(int k=0;k<LOGV;k++)
        if( diff & (1<<k) )    x = pa[k][x];

    if( x == y ) return x;
    for(int k=LOGV-1;k>=0;k--)
        if( pa[k][x] != pa[k][y] ){
            x = pa[k][x]; y = pa[k][y];
        }
    return pa[0][x];
}

void hld(int cur){
    vis[cur] = true;
    if( cHead[cN] == 0 ) cHead[cN] = cur;
    cLeaf[cN] = cur;
    cNo[cur] = cN;
    cPos[cur] = cSize[cN]; cSize[cN]++;
    int nxt = -1; int mx = -1;
    // get max subtree (special child)
    for(auto e : E[cur])if(!vis[e.Fi]){
        if( sz[e.Fi] > mx ){
            nxt = e.Fi; mx = sz[e.Fi];
        }
    }

    if( mx >= 0 ) hld(nxt);
    for(auto e : E[cur])if(!vis[e.Fi]){
        cN++; hld(e.Fi);
    }
}

void build_hld_segTree(){
    for(int n=1;n<=cN;n++){
        int cur = cLeaf[n];
        tree[n].init(cSize[n]+5);
        while( cur!=-1 && cNo[cur]==n ){
            tree[n].update(cPos[cur],val[cur]);
            cur = pa[0][cur];
        }
    }
}
void update_query(int x, int val){
    tree[cNo[x]].update(cPos[x],val);
}

int query_up(int u, int v){
    int uc = cNo[u], vc = cNo[v]; int ret = 0;
    while(true){
        if( uc == vc ){
            ret = max(ret, tree[uc].query(cPos[v]+1,cPos[u]) );
            break;
        }
        ret = max(ret, tree[uc].query( cPos[cHead[uc]], cPos[u]) );
        u = cHead[uc]; u = pa[0][u]; uc = cNo[u];
    }
    return ret;
}
int query(int u, int v){
    int l = lca(u,v);
    return max(query_up(u,l), query_up(v,l));
}

int main(){
    geti(N);
    rep(i,N-1){
        int a,b,c; geti(a,b,c);
        E[a].push_back({b,c}); E[b].push_back({a,c});
        edges.push_back({a,b});
    }

    dfs_build(1,-1,0,0); lca_build();
    cN = 1;
    hld(1);
    tree.resize(cN+3);
    build_hld_segTree();
    geti(K);
    rep(i,K){
        int a,b,c; geti(a,b,c);
        if( a == 1 ){
            b--; int u = edges[b].Fi; int v = edges[b].Se;
            if( level[u] > level[v] ) swap(u,v);
            update_query(v,c);
        }else{
            printf("%d\n",query(b,c));
        }
    }
}
```

## 4.4 Mo Hilbert Order

```
inline int64_t hilbertOrder(int x, int y, int pow, int rotate) {
        if (pow == 0) {
                return 0;
        }
        int hpow = 1 << (pow-1);
        int seg = (x < hpow) ? (
                (y < hpow) ? 0 : 3
        ) : (
                (y < hpow) ? 1 : 2
        );
        seg = (seg + rotate) & 3;
        const int rotateDelta[4] = {3, 0, 0, 1};
        int nx = x & (x ^ hpow), ny = y & (y ^ hpow);
        int nrot = (rotate + rotateDelta[seg]) & 3;
        int64_t subSquareSize = int64_t(1) << (2*pow - 2);
        int64_t ans = seg * subSquareSize;
        int64_t add = hilbertOrder(nx, ny, pow-1, nrot);
        ans += (seg == 1 || seg == 2) ? add : (subSquareSize - add - 1);
        return ans;
}

struct Query {
        int l, r, idx;
        int64_t ord;

        inline void calcOrder() {
                ord = hilbertOrder(l, r, 21, 0);
        }
};

inline bool operator<(const Query &a, const Query &b) {
        return a.ord < b.ord;
}
```

## 4.5 Lazy Propagation 1

```
struct segTree{
    struct Node{
        ll d, lazy;
    };
    vector<Node> data;
    int n;
    void init(int x){
        n = 1; while( n < x ) n *= 2;
        data.resize(n*2+10);
    }
    void propagate(int node, int nodeL, int nodeR){
        if( data[node].lazy == 0 ) return;
        ll len = nodeR - nodeL + 1;
        data[node].d += len*data[node].lazy;
        if( len > 1 ){
            data[node*2].lazy += data[node].lazy;
            data[node*2+1].lazy += data[node].lazy;
        }
        data[node].lazy = 0;
    }

    void update(int l, int r, ll val, int node, int nodeL, int nodeR){
        propagate(node, nodeL, nodeR);
        if( l > nodeR || r < nodeL ) return;
        if( l <= nodeL && nodeR <= r ){
            data[node].lazy += val;
            propagate(node,nodeL,nodeR);
            return;
        }
        update(l,r,val,node*2,nodeL,(nodeL+nodeR)/2);
        update(l,r,val,node*2+1,(nodeL+nodeR)/2+1,nodeR);
        data[node].d = data[node*2].d + data[node*2+1].d;
    }

    ll query(int l, int r, int node, int nodeL, int nodeR){
        propagate(node, nodeL, nodeR);
        if( l > nodeR || r < nodeL ) return 0;
        if( l <= nodeL && nodeR <= r ){
            return data[node].d;
        }
        ll sum = 0;
        sum += query(l,r,node*2,nodeL,(nodeL+nodeR)/2);
        sum += query(l,r,node*2+1,(nodeL+nodeR)/2+1,nodeR);
        return sum;
    }

};
```

## 4.6 Dynamic Convex Hull Trick

```
using line_t = double;
const line_t is_query = -1e18;

struct Line {
    line_t m, b;
    mutable function<const Line*()> succ;
    bool operator<(const Line& rhs) const {
        if (rhs.b != is_query) return m < rhs.m;
        const Line* s = succ();
        if (!s) return 0;
        line_t x = rhs.m;
        return b - s->b < (s->m - m) * x;
    }
};

struct HullDynamic : public multiset<Line> { // will maintain upper hull for
  maximum
    bool bad(iterator y) {
        auto z = next(y);
        if (y == begin()) {
            if (z == end()) return 0;
            return y->m == z->m && y->b <= z->b;
        }
        auto x = prev(y);
        if (z == end()) return y->m == x->m && y->b <= x->b;
        return (x->b - y->b)*(z->m - y->m) >= (y->b - z->b)*(y->m - x->m);
    }
    void insert_line(line_t m, line_t b) {
        auto y = insert({ m, b });
        y->succ = [=] { return next(y) == end() ? 0 : &*next(y); };
        if (bad(y)) { erase(y); return; }
        while (next(y) != end() && bad(next(y))) erase(next(y));
        while (y != begin() && bad(prev(y))) erase(prev(y));
    }
    line_t query(line_t x) {
        auto l = *lower_bound((Line) { x, is_query });
        return l.m * x + l.b;
    }
}H;
```

## 5 Geometry

### 5.1 Voronoi

```
typedef pair<double, double> pdd;

int get_det(pii a, pii b, pii c, pii d) {
    // 1 : out, -1 : in, 0 : boundary
    b.Fi -= a.Fi; c.Fi -= a.Fi; d.Fi -= a.Fi;
    b.Se -= a.Se; c.Se -= a.Se; d.Se -= a.Se;
    __int128 vb = (ll)b.Fi * b.Fi + (ll)b.Se * b.Se;
    __int128 vc = (ll)c.Fi * c.Fi + (ll)c.Se * c.Se;
    __int128 vd = (ll)d.Fi * d.Fi + (ll)d.Se * d.Se;
    __int128 res = b.Fi * (c.Se * vd - d.Se * vc) - c.Fi * (b.Se * vd - d.Se *
      vb) + d.Fi * (b.Se * vc - c.Se * vb);
    if(res > 0) return 1;
    else if(res == 0) return 0;
    else return -1;
}

const int MAXN = 2020;
const int INF = 10010;
int C[MAXN+5][MAXN+5];
void get_delaunay(vector <pii> pts, vector <t3> &L) {
    vector <pii> P;
    P.pb(pii(-INF*INF*4, -INF*INF*4));
    P.pb(pii(INF*INF*4, -INF*INF*4));
    P.pb(pii(0, INF*INF));
    rep(i, szz(pts)) P.pb(pts[i]);
    int N = szz(P);
    rep(i, N) rep(j, N) C[i][j] = 0;
    vector <t3> Tri;
    Tri.pb(t3(0, 1, 2));
    auto get_c = [&](int x, int y) { return C[x][y] + C[y][x]; };
    for(int i=3;i<N;i++) {
        int m = szz(Tri);
        vector <int> bad_t;
        rep(j, m) {
            int a, b, c;
            tie(a, b, c) = Tri[j];
            if(get_det(P[a], P[b], P[c], P[i]) == -1) {
                bad_t.pb(j);
                C[a][b]++; C[b][c]++; C[c][a]++;
            }
        }
        for(int e : bad_t) {
            int a, b, c;
            tie(a, b, c) = Tri[e];
            if(get_c(a, b) == 1) Tri.pb(t3(a, b, i));
            if(get_c(b, c) == 1) Tri.pb(t3(b, c, i));
            if(get_c(c, a) == 1) Tri.pb(t3(c, a, i));
        }

        for(int e : bad_t) {
            int a, b, c;
            tie(a, b, c) = Tri[e];
            C[a][b] = C[b][c] = C[c][a] = 0;
        }

        reverse(all(bad_t));
        for(int e : bad_t) {
            swap(Tri[e], Tri.back());
            Tri.pop_back();
        }
    }
    for(t3 e : Tri) {
        int a, b, c;
        tie(a, b, c) = e;
        if(min({a, b, c}) >= 3) L.pb(t3(a-3, b-3, c-3));
    }
}
```

## 6 Math

### 6.1 FFT

```
#include <cmath>
#include <complex>
using namespace std;
typedef pair<int,int> pii;
typedef complex<double> base;

void fft(vector<base> &a, bool invert){
    int n = a.size();
    for(int i=1,j=0;i<n;i++){
        int bit = n >> 1;
        for (;j>=bit;bit>>=1)j -= bit;
        j += bit;
        if (i < j) swap(a[i], a[j]);
    }
    for(int len=2;len<=n;len<<=1){
        double ang = 2*acos(-1)/len*(invert?-1:1);
        base wlen(cos(ang),sin(ang));
        for(int i=0;i<n;i+=len){
            base w(1);
            for(int j=0;j<len/2;j++){
                base u = a[i+j], v = a[i+j+len/2]*w;
                a[i+j] = u+v;
                a[i+j+len/2] = u-v;
                w *= wlen;
            }
        }
    }
    if (invert) {
        for(int i=0;i<n;i++) a[i] /= n;
    }
}

void multiply(const vector<int> &a, const vector<int> &b, vector<int> &res){
    vector<base> fa(a.begin(), a.end()), fb(b.begin(),b.end());
    int n = 1;
    while(n < max(a.size(), b.size())) n <<= 1;
    n <<= 1;
    fa.resize(n); fb.resize(n);
    fft(fa,false);fft(fb,false);
    for(int i=0;i<n;i++) fa[i] *= fb[i];
    fft(fa,true);
    res.resize(n);
    for(int i=0;i<n;i++) res[i] = int(fa[i].real() + (fa[i].real() > 0 ? 0.5 :
      -0.5));
}
```

## 6.2 Kirchhoff Theorem

```
Find number of MST in given graph G.
m[i][j] := -( number of i<->j edges ) (i != j)
m[i][i] := degree of vertex i
(ans) = (det of (n-1)x(n-1) matrix obtained from m with first row&col deleted )
```

## 6.3 Berlekamp Massey

```
#include<cstdio>
#include<algorithm>
#include<vector>
#include<cassert>
#include<tuple>
typedef long long lint;

lint mod = 1000000007;
using namespace std;

lint ipow(lint a, lint b) {
    lint r = 1;
    while (b) {
        if (b & 1)r = r*a%mod;
        b >>= 1, a = a*a%mod;
    }
    return r;
}
```

```
vector<lint> berlekamp_massey(vector<lint> x) {
    vector<lint> ls, cur;
    lint lf, ld;
    for (lint i = 0; i<x.size(); i++) {
        lint t = 0;
        for (lint j = 0; j<cur.size(); j++) {
            t = (t + 1ll * x[i - j - 1] * cur[j]) % mod;
        }
        if ((t - x[i]) % mod == 0) continue;
        if (cur.empty()) {
            cur.resize(i + 1);
            lf = i;
            ld = (t - x[i]) % mod;
            continue;
        }
        lint k = -(x[i] - t) * ipow(ld, mod - 2) % mod;
        vector<lint> c(i - lf - 1);
        c.push_back(k);
        for (auto &j : ls) c.push_back(-j * k % mod);
        if (c.size() < cur.size()) c.resize(cur.size());
        for (lint j = 0; j<cur.size(); j++) {
            c[j] = (c[j] + cur[j]) % mod;
        }
        if (i - lf + (lint)ls.size() >= (lint)cur.size()) {
            tie(ls, lf, ld) = make_tuple(cur, i, (t - x[i]) % mod);
        }
        cur = c;
    }
    for (auto &i : cur) i = (i % mod + mod) % mod;
    return cur;
}
lint get_nth(vector<lint> rec, vector<lint> dp, lint n) {
    lint m = rec.size();
    vector<lint> s(m), t(m);
    s[0] = 1;
    if (m != 1) t[1] = 1;
    else t[0] = rec[0];
    auto mul = [&rec](vector<lint> v, vector<lint> w) {
        lint m = v.size();
        vector<lint> t(2 * m);
        for (lint j = 0; j<m; j++) {
            for (lint k = 0; k<m; k++) {
                t[j + k] += 1ll * v[j] * w[k] % mod;
                if (t[j + k] >= mod) t[j + k] -= mod;
            }
        }
        for (lint j = 2 * m - 1; j >= m; j--) {
            for (lint k = 1; k <= m; k++) {
                t[j - k] += 1ll * t[j] * rec[k - 1] % mod;
                if (t[j - k] >= mod) t[j - k] -= mod;
            }
        }
        t.resize(m);
        return t;
    };
    while (n) {
        if (n & 1) s = mul(s, t);
        t = mul(t, t);
        n >>= 1;
    }
    lint ret = 0;
    for (lint i = 0; i<m; i++) ret += 1ll * s[i] * dp[i] % mod;
    return ret % mod;
}
lint guess_nth_term(vector<lint> x, lint n) {
    if (n < x.size()) return x[n];
    vector<lint> v = berlekamp_massey(x);
    if (v.empty()) return 0;
    return get_nth(v, x, n);
}
```

## 6.4 Simplex

```
/*
LP Duality
tableu 를대각선으로뒤집고음수부호를붙인답        =  -( 원문제의답   )
ex) n = 2, m = 3, a = [[0.5, 2, 1], [1, 2, 4]], b = [24, 60], c = [6, 14, 13]
<=> n = 3, m = 2, a = [[-0.5, -1], [-2, -2], [-1, -4]], b = [-6, -14, -13], c =
  [-24, -60]

n := number of variables
m := number of constraints
a[1~m][1~n] := constraints
b[1~m] := constraints value (b[i] can be negative)
c[1~n] := maximum coefficient
v := results
sol[i] := 등호조건, i 번째변수의값
ex) Maximize p = 6x + 14y + 13z
    Constraints: 0.5x + 2y + z ≤ 24
                 x + 2y + 4z ≤ 60
    n = 2, m = 3, a = [[0.5, 2, 1], [1, 2, 4]], b = [24, 60], c = [6, 14, 13]
*/

namespace simplex {
  using T = long double;
  const int N = 410, M = 30010;
  const T eps = 1e-7;
  int n, m;
  int Left[M], Down[N];
  T a[M][N], b[M], c[N], v, sol[N];

  bool eq(T a, T b) { return fabs(a - b) < eps;  }
  bool ls(T a, T b) { return a < b && !eq(a, b); }

  void init(int p, int q) {
    n = p; m = q; v = 0;
    for(int i = 1; i <= m; i++){
      for(int j = 1; j <= n; j++) a[i][j]=0;
    }
    for(int i = 1; i <= m; i++) b[i]=0;
    for(int i = 1; i <= n; i++) c[i]=sol[i]=0;
  }

  void pivot(int x,int y) {
    swap(Left[x], Down[y]);
    T k = a[x][y]; a[x][y] = 1;
    vector<int> nz;
    for(int i = 1; i <= n; i++){
      a[x][i] /= k;
      if(!eq(a[x][i], 0)) nz.push_back(i);
    }
    b[x] /= k;

    for(int i = 1; i <= m; i++){
      if(i == x || eq(a[i][y], 0)) continue;
      k = a[i][y]; a[i][y] = 0;
      b[i] -= k*b[x];
      for(int j : nz) a[i][j] -= k*a[x][j];
    }
    if(eq(c[y], 0)) return;
    k = c[y]; c[y] = 0;
    v += k*b[x];
    for(int i : nz) c[i] -= k*a[x][i];
  }

  // 0: found solution, 1: no feasible solution, 2: unbounded
  int solve() {
    for(int i = 1; i <= n; i++) Down[i] = i;
    for(int i = 1; i <= m; i++) Left[i] = n+i;
    while(1) { // Eliminating negative b[i]
      int x = 0, y = 0;
      for(int i = 1; i <= m; i++) if (ls(b[i], 0) && (x == 0 || b[i] < b[x])) x
        = i;
      if(x == 0) break;
      for(int i = 1; i <= n; i++) if (ls(a[x][i], 0) && (y == 0 || a[x][i] < a[x
        ][y])) y = i;
      if(y == 0) return 1;
      pivot(x, y);
    }
    while(1) {
      int x = 0, y = 0;
      for(int i = 1; i <= n; i++)
        if (ls(0, c[i]) && (!y || c[i] > c[y])) y = i;
      if(y == 0) break;
      for(int i = 1; i <= m; i++)
        if (ls(0, a[i][y]) && (!x || b[i]/a[i][y] < b[x]/a[x][y])) x = i;
      if(x == 0) return 2;
      pivot(x, y);
    }
    for(int i = 1; i <= m; i++) if(Left[i] <= n) sol[Left[i]] = b[i];
    return 0;
  }
}
```

## 6.5 Gaussian Elimination

```
#define MAX_N 300       // adjust this value as needed
struct AugmentedMatrix { double mat[MAX_N][MAX_N + MAX_N + 10]; };
struct ColumnVector { double vec[MAX_N]; };

// 0 indexed row and column
AugmentedMatrix GaussianElimination(int N, AugmentedMatrix Aug) {
    // input: N X 2N matrix  [A I], output: [I invA]

    // forward eliminataion phase
    for(int i=0;i<N;i++){
        int l = i;
        // which row has largest column value
        for(int j=i+1;j<N;j++)
            if( fabs(Aug.mat[j][i]) > fabs(Aug.mat[l][i]) )
                l = j;
        // swap this pivot row to minimize error
        for(int k=i;k<2*N;k++)
            swap(Aug.mat[i][k],Aug.mat[l][k]);
        // calculate forward elimination
        for(int j=i+1;j<N;j++)
            for(int k=2*N-1;k>=i;k--)
                Aug.mat[j][k] -= Aug.mat[i][k] * Aug.mat[j][i] / Aug.mat[i][i];
    }

    // normalize pivots
    for(int i=0;i<N;i++)
        for(int j=2*N-1;j>=i;j--)
            Aug.mat[i][j] /= Aug.mat[i][i];

    // backward elimination
    for(int i=N-1;i>0;i--)
        for(int j=i-1;j>=0;j--)
            for(int k=2*N-1;k>=i;k--)
                Aug.mat[j][k] -= Aug.mat[i][k] * Aug.mat[j][i] / Aug.mat[i][i];


    return Aug;
}

int main() {

    AugmentedMatrix Aug;
    int N; geti(N);
    rep(i,N) rep(j,N) scanf("%lf",&Aug.mat[i][j]);
    for(int i=N;i<2*N;i++) Aug.mat[i-N][i] = 1;

    AugmentedMatrix res = GaussianElimination(N, Aug);

    // Print inversion of A
    for(int i=0;i<N;i++){
        for(int j=N;j<2*N;j++) printf("%f ",res.mat[i][j]);
        printf("\n");
    }


    return 0;
}
```

## 6.6 Prime Algorithms

```
typedef long long ll;
using namespace std;

ll gcd(ll a, ll b) {
    if (b == 0)
        return a;
    return gcd(b, a%b);
}

namespace miller_rabin {
    ll mul(ll x, ll y, ll mod) { return (__int128)x * y % mod; }
    //ll mul(ll x, ll y, ll mod) { return x * y % mod; }
    ll ipow(ll x, ll y, ll p) {
        ll ret = 1, piv = x % p;
        while (y) {
            if (y & 1) ret = mul(ret, piv, p);
            piv = mul(piv, piv, p);
            y >>= 1;
        }
        return ret;
    }
    bool miller_rabin(ll x, ll a) {
        if (x % a == 0) return 0;
        ll d = x - 1;
        while (1) {
            ll tmp = ipow(a, d, x);
            if (d & 1) return (tmp != 1 && tmp != x - 1);
            else if (tmp == x - 1) return 0;
            d >>= 1;
        }
    }
    bool isprime(ll x) {
        for (auto &i : { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37 }) {
            if (x == i) return 1;
            if (x > 40 && miller_rabin(x, i)) return 0;
        }
        if (x <= 40) return 0;
        return 1;
    }
}
namespace pollard_rho {
    ll f(ll x, ll n, ll c) {
        return (c + miller_rabin::mul(x, x, n)) % n;
    }
    void rec(ll n, vector<ll> &v) {
        if (n == 1) return;
        if (n % 2 == 0) {
            v.push_back(2);
            rec(n / 2, v);
            return;
        }
        if (miller_rabin::isprime(n)) {
            v.push_back(n);
            return;
        }
        ll a, b, c;
        while (1) {
            a = rand() % (n - 2) + 2;
            b = a;
            c = rand() % 20 + 1;
            do {
                a = f(a, n, c);
                b = f(f(b, n, c), n, c);
            } while (gcd(abs(a - b), n) == 1);
            if (a != b) break;
        }
        ll x = gcd(abs(a - b), n);
        rec(x, v);
        rec(n / x, v);
    }
    vector<ll> factorize(ll n) {
        vector<ll> ret;
        rec(n, ret);
        sort(ret.begin(), ret.end());
        return ret;
    }
};

int main() {
    vector<ll> res;
    ll num;
    scanf("%lld", &num);
    res = pollard_rho::factorize(num);
    for (int i = 0; i < res.size(); ++i)
        printf("%lld\n", res[i]);
}

/////////////
// Shanks-Tonelli, Square Root Modulo P
/////////////
long long get_sqrt(long long a, long long Mod) {
    long long tp = Mod - 1, S = 0;
    if (Mod == 2) return a;
    if (Pow(a, (Mod - 1) / 2, Mod) != 1) {
        puts("No square");
        return -1;
    }
    while (tp % 2 == 0) {
        S++;
        tp /= 2;
    }
    long long Q = tp, z;
    for (int i = 2;; i++) {
        if (Pow(i, (Mod - 1) / 2, Mod) != 1) {
            z = i;
            break;
        }
    }
    long long M = S, c = Pow(z, Q, Mod), t = Pow(a, Q, Mod), R = Pow(a, (Q + 1)
      / 2, Mod);
    while (1) {
        if (t == 0)return 0;
        if (t == 1)return R;
        long long tt = t, i = 0;
        while (tt != 1) {
            tt = tt * tt%Mod; i++;
        }
        long long b = c;
        for (int j = 0; j < M - i - 1; j++) {
            b = b * b%Mod;
        }
        M = i;
        c = b * b%Mod;
        t = t * b%Mod*b%Mod;
        R = R * b%Mod;
    }
}
```

## 7 Miscelleneous

### 7.1 Hungarian

```
/*
Tests
http://www.spoj.com/problems/GREED/
https://www.acmicpc.net/problem/8992
SRM 506 mid

Time complexity O(n^3)

Usage
MinWeightBipartiteMatch matcher(n);
for (int i = 0; i < n; i++) for (int j = 0; j < n; j++) matcher.weights[i][j] =
  SOMETHING;
cost_t total = matcher.solve();

See matcher.match(row -> col) and matcher.matched(col -> row) for actual match
*/

struct MinWeightBipartiteMatch
{
  typedef long long cost_t;

  cost_t max_cost() const { return numeric_limits<cost_t>::max(); }

  // input
  int n;
  vector<vector<cost_t>> weights;
  // output
  vector<int> match, matched;

  MinWeightBipartiteMatch(int n) :
    n(n), match(n), matched(n), weights(n, vector<cost_t>(n))
  {

  }
  void resize(int n) {
    this->n = n;
    match.resize(n);
    matched.resize(n);
    weights.resize(n);
    for (int i = 0; i < n; i++) {
      weights[i].resize(n);
    }
  }

  /* for solve() */
  vector<cost_t> slack;
  vector<cost_t> potential_row, potential_col;
  vector<int> reach_row, reach_col;
  int rcnt;
  vector<int> from;
  void found_match(int r, int c) {
    do {
      int old_match = match[r];
      match[r] = c;
      matched[c] = r;
      tie(r, c) = make_pair(from[r], old_match);
    } while (r >= 0 && c >= 0);
  }

  void augment(int row_to_match) {
    slack.resize(n);
    for (int c = 0; c < n; c++) {
      slack[c] = weights[row_to_match][c] - potential_row[row_to_match] -
        potential_col[c];
    }
    ++rcnt;
    vector<int> q; q.reserve(n);
    int h = 0;
    q.push_back(row_to_match);
    reach_row[row_to_match] = rcnt;
    from[row_to_match] = -1;
    for (;;) {
      while (h < q.size()) {
        int r = q[h++];
        for (int c = 0; c < n; c++) {
          cost_t gap = weights[r][c] - potential_row[r] - potential_col[c];
          slack[c] = min(slack[c], gap);
          if (gap != cost_t()) continue;
          int next = matched[c];
          if (next < 0) {
            found_match(r, c);
            return;
          }
          reach_col[c] = rcnt;
          if (reach_row[next] == rcnt) continue;
          q.push_back(next);
          reach_row[next] = rcnt;
          from[next] = r;
        }
      }
      cost_t delta = max_cost();
      for (int c = 0; c < n; c++) {
        if (reach_col[c] == rcnt) continue; // non-covered -> continue
        delta = min(delta, slack[c]);
      }
      for (int r = 0; r < n; r++) {
        if (reach_row[r] == rcnt) continue;
        potential_row[r] -= delta;
      }
      for (int c = 0; c < n; c++) {
        if (reach_col[c] == rcnt) continue;
        potential_col[c] += delta;
        slack[c] -= delta;
      }
      int lastsize = q.size();
      for (int c = 0; c < n; c++) {
        if (reach_col[c] == rcnt) continue;
        if (slack[c] != cost_t()) continue;
        int next = matched[c];
        if (next >= 0 && reach_row[next] == rcnt) continue;
        for (int qi = 0; qi < lastsize; qi++) {
          int r = q[qi];
          cost_t gap = weights[r][c] - potential_row[r] - potential_col[c];
          if (gap != cost_t()) continue;
          if (next < 0) {
            found_match(r, c);
            return;
          }
          reach_col[c] = rcnt;
          q.push_back(next);
          reach_row[next] = rcnt;
          from[next] = r;
          break;
        }
      }
    }
  }

  void initialize() {
    potential_row.assign(n, cost_t());
    potential_col.assign(n, cost_t());
    match.assign(n, -1);
    matched.assign(n, -1);
    reach_row.assign(n, 0);
    reach_col.assign(n, 0);
    from.resize(n);
    rcnt = 1;
    for (int i = 0; i < n; i++) {
      cost_t row_min_weight = *min_element(weights[i].begin(), weights[i].end())
        ;
      potential_row[i] = row_min_weight;
    }
    for (int i = 0; i < n; i++) {
      cost_t col_min_weight = weights[0][i] - potential_row[0];
      for (int j = 1; j < n; j++) col_min_weight = min(col_min_weight, weights[j
        ][i] - potential_row[j]);
      potential_col[i] = col_min_weight;
    }
  }

  cost_t solve() {
    initialize();
    for (int row_to_match = 0; row_to_match < n; row_to_match++) {
      augment(row_to_match);
    }
    cost_t ans = cost_t();
    for (auto v : potential_row) ans += v;
    for (auto v : potential_col) ans += v;
    return ans;
  }
};
```

## 7.2 LiChao Tree

```
/* This is MAX Lichao Tree*/
struct LiChaoTree{
    ll f(Line l, ll x){
        return l.first * x + l.second;
    }
    struct Node{
        int lnode, rnode;
        ll xl, xr;
        Line l;
    };
    vector<Node> nodes;
    void init(ll xmin, ll xmax){
        nodes.push_back({-1,-1,xmin,xmax,{0,-1e18}});
    }

    void insert(int n, Line newline){
        ll xl = nodes[n].xl, xr = nodes[n].xr;
        ll xm = (xl + xr) >> 1;

        Line llow = nodes[n].l, lhigh = newline;
        if( f(llow, xl) >= f(lhigh,xl) ) swap(llow, lhigh);

        if( f(llow, xr) <= f(lhigh, xr) ){
            nodes[n].l = lhigh;
            return;
        }

        else if( f(llow, xm) <= f(lhigh, xm) ){
            nodes[n].l = lhigh;
            if( nodes[n].rnode == -1 ){
                nodes[n].rnode = nodes.size();
                nodes.push_back({-1,-1,xm+1,xr,{0,-1e18}});
            }
            insert(nodes[n].rnode, llow);
        }
        else{
            nodes[n].l = llow;
            if( nodes[n].lnode == -1 ){
                nodes[n].lnode = nodes.size();
                nodes.push_back({-1,-1,xl,xm,{0,-1e18}});
            }
            insert(nodes[n].lnode, lhigh);
        }
    }
    ll get(int n, ll xq){
        if( n == -1 ) return -1e18;
        ll xl = nodes[n].xl, xr = nodes[n].xr;
        ll xm = (xl + xr) >> 1;

        if( xq <= xm ) return max(f(nodes[n].l, xq), get(nodes[n].lnode, xq));
        else return max(f(nodes[n].l, xq), get(nodes[n].rnode, xq));
    }
};

int main() {
    LiChaoTree tree;
    tree.init(-2e12, 2e12);

    int Q; scanf("%d",&Q);
    for(int q=0;q<Q;q++){
        ll op, a, b;
        scanf("%lld",&op);
        if( op == 1 ){
            scanf("%lld%lld",&a,&b);
            tree.insert(0, {a,b});
        }
        if( op == 2 ){
            scanf("%lld",&a);
            printf("%lld\n",tree.get(0, a));
        }
    }

}

// This is MIN Lichao tree //
struct LiChaoTree{
    ll f(Line l, ll x){
        return l.first * x + l.second;
    }
    struct Node{
        int lnode, rnode;
        ll xl, xr;
        Line l;
    };
    vector<Node> nodes;
    void init(ll xmin, ll xmax){
        nodes.push_back({-1,-1,xmin,xmax,{0,1e18}});
    }

    void insert(int n, Line newline){
        ll xl = nodes[n].xl, xr = nodes[n].xr;
        ll xm = (xl + xr) >> 1;

        Line llow = nodes[n].l, lhigh = newline;
        if( f(llow, xl) >= f(lhigh,xl) ) swap(llow, lhigh);

        if( f(llow, xr) <= f(lhigh, xr) ){
            nodes[n].l = llow;
            return;
        }

        else if( f(llow, xm) <= f(lhigh, xm) ){
            nodes[n].l = llow;
            if( nodes[n].rnode == -1 ){
                nodes[n].rnode = nodes.size();
                nodes.push_back({-1,-1,xm+1,xr,{0,1e18}});
            }
            insert(nodes[n].rnode, lhigh);
        }

        else{
            nodes[n].l = lhigh;
            if( nodes[n].lnode == -1 ){
                nodes[n].lnode = nodes.size();
                nodes.push_back({-1,-1,xl,xm,{0,1e18}});
            }
            insert(nodes[n].lnode, llow);
        }
    }
    ll get(int n, ll xq){
        if( n == -1 ) return 1e18;
        ll xl = nodes[n].xl, xr = nodes[n].xr;
        ll xm = (xl + xr) >> 1;

        if( xq <= xm ) return min(f(nodes[n].l, xq), get(nodes[n].lnode, xq));
        else return min(f(nodes[n].l, xq), get(nodes[n].rnode, xq));
    }
};
```

## 7.3 Persistence Segment Tree

```
int n, cnt;
int root[MAXN];

struct node {
    int sum, left, right;
} tree[3 * MAXN * LOGN];

int build(int l = 0, int r = n) {
    int idx = ++cnt;
    if(r - l <= 1) {
        tree[idx] = {0, 0, 0};
        return idx;
    }
    int mid = (l + r) >> 1;
    tree[idx] = {0, build(l, mid), build(mid, r)};
    return idx;
}

int update(int x, int prev, int l = 0, int r = n) {
    if(x < l || r <= x) return prev;
    int idx = ++cnt;
    if(r - l <= 1) {
        tree[idx] = {1, 0, 0};
        return idx;
    }

    int mid = (l + r) >> 1;
    int L = update(x, tree[prev].left, l, mid);
    int R = update(x, tree[prev].right, mid, r);
    tree[idx] = {tree[L].sum + tree[R].sum, L, R};
    return idx;
}

int query(int x, int y, int k, int l = 0, int r = n) {
    if(r - l <= 1) return l;
    int mid = (l + r) >> 1;
    int leftSum = tree[tree[y].left].sum - tree[tree[x].left].sum;
    if(leftSum >= k)
        return query(tree[x].left, tree[y].left, k, l, mid);
    else
        return query(tree[x].right, tree[y].right, k - leftSum, mid, r);
}



int a[MAXN], rev[MAXN];
map<int, int> M;

int main() {
    int q;
    geti(n, q);
    for(int i = 1; i <= n; i++) {
        geti(a[i]);
        rev[i-1] = a[i];
    }
    sort(rev, rev + n);
    for(int i = 0; i < n; i++)
        M[rev[i]] = i;
    for(int i = 1; i <= n; i++)
        a[i] = M[a[i]];

    root[0] = build();
    for(int i = 1; i <= n; i++)
        root[i] = update(a[i], root[i-1]);

    while(q--) {
        int i, j, k;
        geti(i, j, k);
        printf("%d\n", rev[query(root[i-1], root[j], k)]);
    }
}
```

## 7.4 Various FFTs (Bitwise Convolution)

```
// OR convolution
vector<ll> FFT_OR(vector<ll> A, int rev) {
    int N = A.size(); // A.size() must be power of 2
    for(int len=1;len<N;len<<=1) {
        for(int i=0;i<N;i+=len*2) {
            for(int j=0;j<len;j++) {
                ll x = A[i+j];
                ll y = A[i+j+len];
                if(rev == 0) A[i+j+len] = x + y;
                else A[i+j+len] = y - x;
            }
        }
    }
    return A;
}

// AND convolution
vector<ll> FFT_AND(vector<ll> A, int rev) {
    int N = A.size(); // A.size() must be power of 2
    for(int len=1;len<N;len<<=1) {
        for(int i=0;i<N;i+=len*2) {
            for(int j=0;j<len;j++) {
                ll x = A[i+j];
                ll y = A[i+j+len];
                if(rev == 0) A[i+j] = x + y;
                else A[i+j] = x - y;
            }
        }
    }
    return A;
}

// XOR convolution
vector<ll> FFT_XOR(vector<ll> A, int rev) {
    int N = A.size(); // A.size() must be power of 2
    for(int len=1;len<N;len<<=1) {
        for(int i=0;i<N;i+=len*2) {
            for(int j=0;j<len;j++) {
                ll x = A[i+j];
                ll y = A[i+j+len];
                A[i+j] = x + y;
                A[i+j+len] = x - y;
            }
        }
    }
    if(rev) rep(i, N) A[i] /= N;
    return A;
}

const int L = 10;
const int N = 1<<L;
int main() {
    vector<ll> A(N),B(N);
    rep(i, N) A[i] = rand() % 1000;
    rep(i, N) B[i] = rand() % 1000;
    ll EX[N] = {};
    rep(i, N) rep(j, N) EX[i&j] = EX[i&j] + A[i] * B[j];
    A = FFT_AND(A, 0);
    B = FFT_AND(B, 0);
    rep (i, N) A[i] = A[i] * B[i];
    A = FFT_AND(A, 1);
    rep(i, N) printf("%lld %lld\n", A[i], EX[i]);
    return 0;
}
```

## 7.5 NTT

```
template< int mod, int primitiveroot >
struct NumberTheoreticTransform {
  vector< vector< int > > rts, rrts;

  void ensure_base(int N) {
    if(rts.size() >= N) return;
    rts.resize(N), rrts.resize(N);
    for(int i = 1; i < N; i <<= 1) {
      if(rts[i].size()) continue;
      int w = mod_pow(primitiveroot, (mod - 1) / (i * 2));
      int rw = inverse(w);
      rts[i].resize(i), rrts[i].resize(i);
      rts[i][0] = 1, rrts[i][0] = 1;
      for(int k = 1; k < i; k++) {
        rts[i][k] = mul(rts[i][k - 1], w);
        rrts[i][k] = mul(rrts[i][k - 1], rw);
      }
    }
  }

  inline int mod_pow(int x, int n) {
    int ret = 1;
    while(n > 0) {
      if(n & 1) ret = mul(ret, x);
      x = mul(x, x);
      n >>= 1;
    }
    return ret;
  }

  inline int inverse(int x) {
    return mod_pow(x, mod - 2);
  }

  inline int add(int x, int y) {
    x += y;
    if(x >= mod) x -= mod;
    return x;
  }

  inline int mul(int a, int b) {
    return int(1LL * a * b % mod);
  }

  void DiscreteFourierTransform(vector< int > &F, bool rev) {
    const int N = (int) F.size();
    ensure_base(N);
    for(int i = 0, j = 1; j + 1 < N; j++) {
      for(int k = N >> 1; k > (i ^= k); k >>= 1);
      if(i > j) swap(F[i], F[j]);
    }
    for(int i = 1; i < N; i <<= 1) {
      for(int j = 0; j < N; j += i * 2) {
        for(int k = 0; k < i; k++) {
          int s = F[j + k], t = mul(F[j + k + i], rev ? rrts[i][k] : rts[i][k]);
          F[j + k] = add(s, t), F[j + k + i] = add(s, mod - t);
        }
      }
    }
    if(rev) {
      int temp = inverse(N);
      for(int i = 0; i < N; i++) F[i] = mul(F[i], temp);
    }
  }

  vector< int > Multiply(const vector< int > &A, const vector< int > &B) {
    int sz = 1;
    while(sz < A.size() + B.size() - 1) sz <<= 1;
    vector< int > F(sz), G(sz);
    for(int i = 0; i < A.size(); i++) F[i] = A[i];
    for(int i = 0; i < B.size(); i++) G[i] = B[i];
    DiscreteFourierTransform(F, false);
    DiscreteFourierTransform(G, false);
    for(int i = 0; i < sz; i++) F[i] = mul(F[i], G[i]);
    DiscreteFourierTransform(F, true);
    F.resize(A.size() + B.size() - 1);
    return F;
  }
};

// https://codeforces.com/contest/1096/problem/G

int N,K;
vector<int> powv[500];
int main() {
    NumberTheoreticTransform<mod, 3> ntt;
    scanf("%d%d",&N,&K);
    vector<int> v;
    rep(i,10) v.pb(0);
    repp(i,K){
        int x; scanf("%d",&x); v[x] = 1;
    }
    powv[0] = v;
    for(int k=1;k<18;k++){
        v = ntt.Multiply(v,v);
        powv[k] = v;
    }
    vector<int> ansv;
    rep(i,10) ansv.pb(0);
    ansv[0] = 1;
    for(int k=0;k<25;k++){
        if( (N/2)&(1<<k) ) ansv = ntt.Multiply(ansv, powv[k]);
    }
    ll ans = 0;
    for(auto e:  ansv){
        ans = (ans + (ll)e*e)%mod;
    }
    cout << ans << endl;
    //Debug(ansv);


}
```

## 7.6 Order Statistic Tree

```
#include <ext/pb_ds/assoc_container.hpp> // Common file
#include <ext/pb_ds/tree_policy.hpp> // Including
  tree_order_statistics_node_update

// Need this
// We can run this code on codeforces
// http://codeforces.com/blog/entry/11080
using namespace __gnu_pbds;

typedef tree<
int,
null_type,
less<int>,
rb_tree_tag,
tree_order_statistics_node_update>
ordered_set;

int main(){
    ordered_set X;
    X.insert(1);
    X.insert(2);
    X.insert(4);
    X.insert(8);
    X.insert(16);

    cout<<*X.find_by_order(1)<<endl; // 2
    cout<<*X.find_by_order(2)<<endl; // 4
    cout<<*X.find_by_order(4)<<endl; // 16
    cout<<(end(X)==X.find_by_order(6))<<endl; // true

    cout<<X.order_of_key(-5)<<endl;  // 0
    cout<<X.order_of_key(1)<<endl;   // 0
    cout<<X.order_of_key(3)<<endl;   // 2
    cout<<X.order_of_key(4)<<endl;   // 2
    cout<<X.order_of_key(400)<<endl; // 5

}
```

## 7.7 BITSET

```
#define M 32
int main()
{
    // default constructor initializes with all bits 0
    bitset<M> bset1;

    // bset2 is initialized with bits of 20
    bitset<M> bset2(20);

    // bset3 is initialized with bits of specified binary string
    bitset<M> bset3(string("1100"));

    // cout prints exact bits representation of bitset
    cout << bset1 << endl;  // 00000000000000000000000000000000
    cout << bset2 << endl;  // 00000000000000000000000000010100
    cout << bset3 << endl;  // 00000000000000000000000000001100
    cout << endl;

    // declaring set8 with capacity of 8 bits

    bitset<8> set8;    // 00000000

    // setting first bit (or 6th index)
    set8[1] = 1;    // 00000010
    set8[4] = set8[1];   //  00010010
    cout << set8 << endl;

    // count function returns number of set bits in bitset
    int numberof1 = set8.count();

    // size function returns total number of bits in bitset
    // so there difference will give us number of unset(0)
    // bits in bitset
    int numberof0 = set8.size() - numberof1;
    cout << set8 << " has " << numberof1 << " ones and "
         << numberof0 << " zeros\n";

    // test function return 1 if bit is set else returns 0
    cout << "bool representation of " << set8 << " : ";
    for (int i = 0; i < set8.size(); i++)
        cout << set8.test(i) << " ";

    cout << endl;

    // any function returns true, if atleast 1 bit
    // is set
    if (!set8.any())
        cout << "set8 has no bit set.\n";

    if (!bset1.any())
        cout << "bset1 has no bit set.\n";

    // none function returns true, if none of the bit
    // is set
    if (!bset1.none())
        cout << "bset1 has all bit set\n";

    // bset.set() sets all bits
    cout << set8.set() << endl;

    //  bset.set(pos, b) makes bset[pos] = b
    cout << set8.set(4, 0) << endl;

    // bset.set(pos) makes bset[pos] = 1  i.e. default
    // is 1
    cout << set8.set(4) << endl;

    // reset function makes all bits 0
    cout << set8.reset(2) << endl;
    cout << set8.reset() << endl;

    // flip function flips all bits i.e.  1 <-> 0
    // and  0 <-> 1
    cout << set8.flip(2) << endl;
    cout << set8.flip() << endl;

    // Converting decimal number to binary by using bitset
    int num = 100;
    cout  << "\nDecimal number: " << num
         << "  Binary equivalent: " << bitset<8>(num);

    return 0;
}

int main()
{
    bitset<4> bset1(9);      // bset1 contains 1001
    bitset<4> bset2(3);      // bset2 contains 0011

    // comparison operator
    cout << (bset1 == bset2) << endl;  // false 0
    cout << (bset1 != bset2) << endl;  // true  1

    // bitwise operation and assignment
    cout << (bset1 ^= bset2) << endl;  // 1010
    cout << (bset1 &= bset2) << endl;  // 0010
    cout << (bset1 |= bset2) << endl;  // 0011

    // left and right shifting
    cout << (bset1 <<= 2) << endl;     // 1100
    cout << (bset1 >>= 1) << endl;     // 0110

    // not operator
    cout << (~bset2) << endl;          // 1100

    // bitwise operator
    cout << (bset1 & bset2) << endl;    // 0010
    cout << (bset1 | bset2) << endl;    // 0111
    cout << (bset1 ^ bset2) << endl;    // 0101
}
```

## 7.8 Bit Hacks

```
int __builtin_clz(int x);// number of leading zero
int __builtin_ctz(int x);// number of trailing zero
int __builtin_clzll(long long x);// number of leading zero
int __builtin_ctzll(long long x);// number of trailing zero
int __builtin_popcount(int x);// number of 1-bits in x
int __builtin_popcountll(long long x);// number of 1-bits in x

lsb(n): (n & -n); // last bit (smallest)
floor(log2(n)): 31 - __builtin_clz(n | 1);
floor(log2(n)): 63 - __builtin_clzll(n | 1);

// compute next perm. ex) 00111, 01011, 01101, 01110, 10011, 10101..
long long next_perm(long long v){
    long long t = v | (v-1);
    return (t + 1) | (((~t & -~t) - 1) >> (__builtin_ctz(v) + 1));
}

// submask enumeration
// Be aware that s=0 is not processed
// Check for s=0 case
for (int s=m; s; s=(s-1)&m){
  // ... you can use s ...

}
```

## 7.9 Fast IO

```
static char buf[1 << 19]; // size : any number geq than 1024
static int idx = 0;
static int bytes = 0;
static inline int _read() {
        if (!bytes || idx == bytes) {
                bytes = (int)fread(buf, sizeof(buf[0]), sizeof(buf), stdin);
                idx = 0;
        }
        return buf[idx++];
}
static inline int _readInt() {
        int x = 0, s = 1;
        int c = _read();
        while (c <= 32) c = _read();
        if (c == '-') s = -1, c = _read();
        while (c > 32) x = 10 * x + (c - '0'), c = _read();
        if (s < 0) x = -x;
        return x;
}
```

Jungle 100%

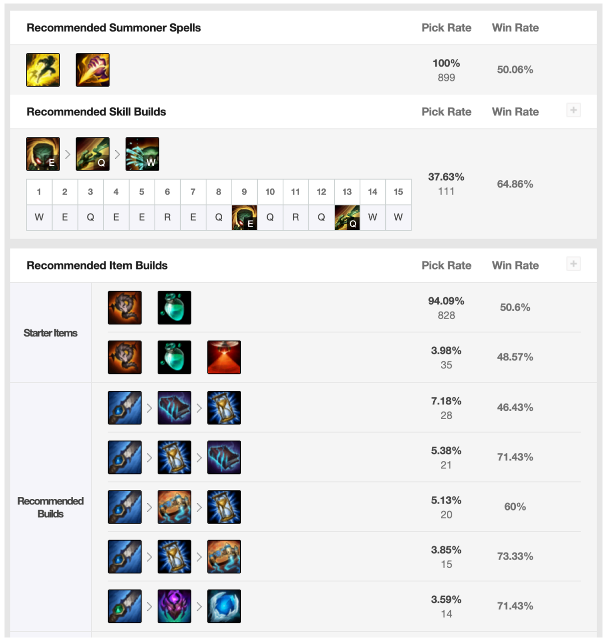
Korea - Version : 9.22

Amumu

Champion Tier: Tier 5

Counter Champion | Strong against

| | | |
|---|---|---|
| Rek'Sai | Win Ratio 33.96% | Counter |
| Warwick | Win Ratio 35.82% | Counter |
| Nocturne | Win Ratio 40.98% | Counter |

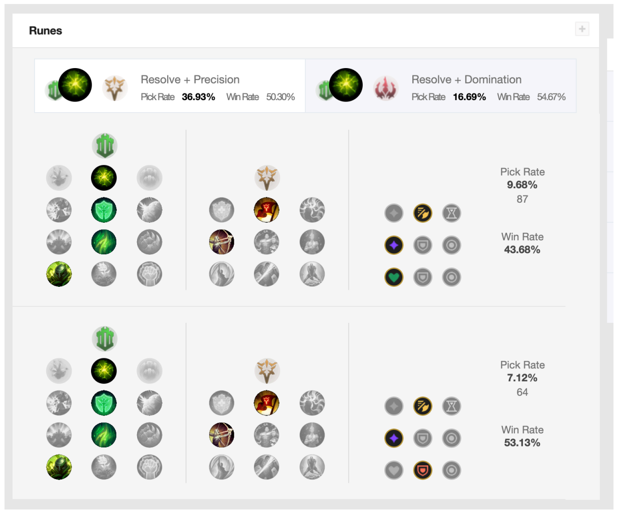


| Recommended Summoner Spells | Pick Rate | Win Rate |
|---|---|---|
| | 100% 899 | 50.06% |

| Recommended Skill Builds | Pick Rate | Win Rate |
|---|---|---|
| E > Q > W | 37.63% 111 | 64.86% |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | E | Q | E | E | R | E | Q | E | Q | R | Q | Q | W | W |

| Recommended Item Builds | | Pick Rate | Win Rate |
|---|---|---|---|
| Starter Items | | 94.09% 828 | 50.6% |
| | | 3.98% 35 | 48.57% |
| Recommended Builds | | 7.18% 28 | 46.43% |
| | | 5.38% 21 | 71.43% |
| | | 5.13% 20 | 60% |
| | | 3.85% 15 | 73.33% |
| | | 3.59% 14 | 71.43% |

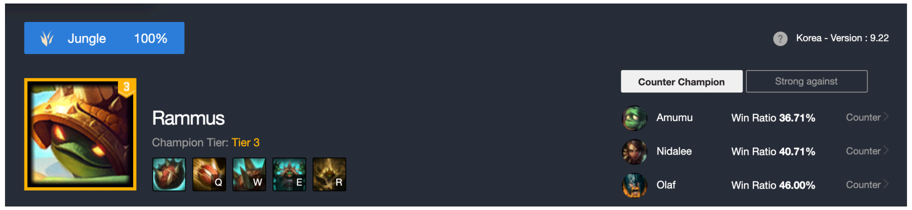


Runes

| Resolve + Precision | Resolve + Domination |
|---|---|
| Pick Rate 36.93% Win Rate 50.30% | Pick Rate 16.69% Win Rate 54.67% |

Pick Rate 9.68% 87
Win Rate 43.68%

Pick Rate 7.12% 64
Win Rate 53.13%


Jungle 100%

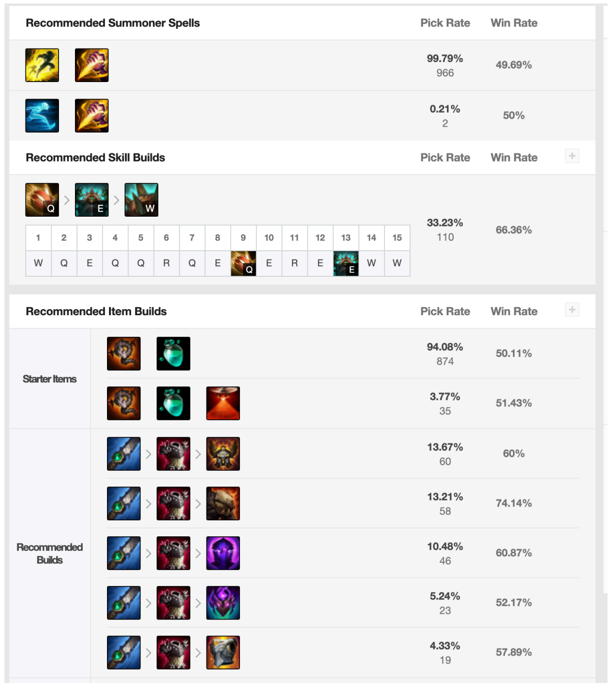
Korea - Version : 9.22

Rammus

Champion Tier: Tier 3

Counter Champion | Strong against

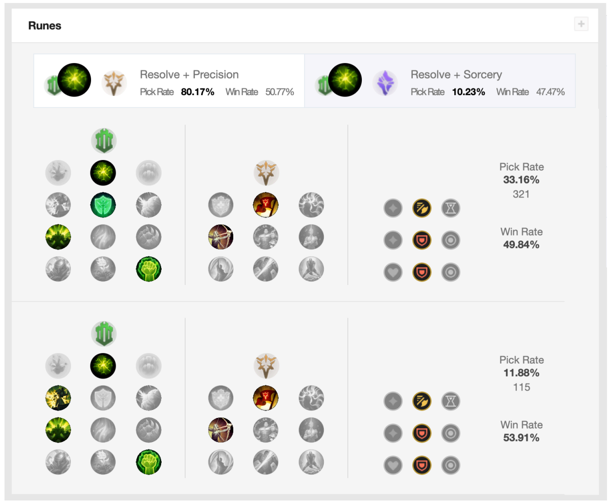
| | | |
|---|---|---|
| Amumu | Win Ratio **36.71%** | Counter |
| Nidalee | Win Ratio **40.71%** | Counter |
| Olaf | Win Ratio **46.00%** | Counter |


| Recommended Summoner Spells | Pick Rate | Win Rate |
|---|---|---|
| | 99.79% 966 | 49.69% |
| | 0.21% 2 | 50% |

| Recommended Skill Builds | Pick Rate | Win Rate |
|---|---|---|
| Q > E > W | 33.23% 110 | 66.36% |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | Q | E | Q | Q | R | Q | E | Q | E | R | E | E | W | W |

| Recommended Item Builds | | Pick Rate | Win Rate |
|---|---|---|---|
| Starter Items | | 94.08% 874 | 50.11% |
| | | 3.77% 35 | 51.43% |
| Recommended Builds | | 13.67% 60 | 60% |
| | | 13.21% 58 | 74.14% |
| | | 10.48% 46 | 60.87% |
| | | 5.24% 23 | 52.17% |
| | | 4.33% 19 | 57.89% |


Runes

Resolve + Precision PickRate **80.17%** WinRate 50.77%

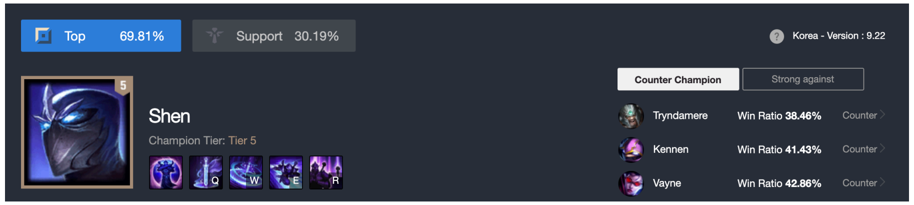
Resolve + Sorcery PickRate **10.23%** WinRate 47.47%

Pick Rate **33.16%** 321

Win Rate **49.84%**

Pick Rate **11.88%** 115

Win Rate **53.91%**


Top 69.81% | Support 30.19%

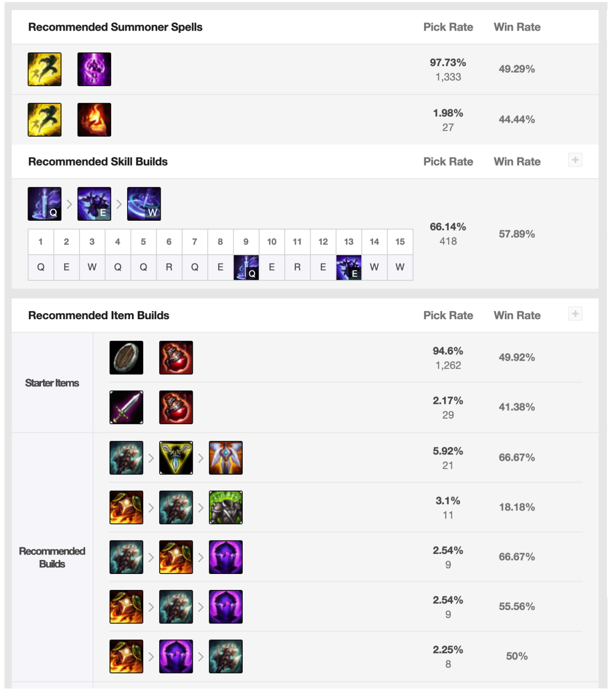
Korea - Version : 9.22

Shen

Champion Tier: Tier 5

Counter Champion | Strong against

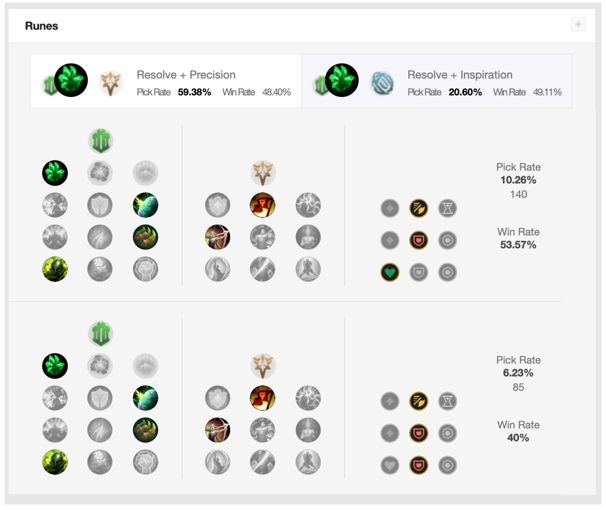
| | | |
|---|---|---|
| Tryndamere | Win Ratio **38.46%** | Counter |
| Kennen | Win Ratio **41.43%** | Counter |
| Vayne | Win Ratio **42.86%** | Counter |


| Recommended Summoner Spells | Pick Rate | Win Rate |
|---|---|---|
| | 97.73% 1,333 | 49.29% |
| | 1.98% 27 | 44.44% |

| Recommended Skill Builds | Pick Rate | Win Rate |
|---|---|---|
| Q > E > W | 66.14% 418 | 57.89% |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Q | E | W | Q | Q | R | Q | E | Q | E | R | E | E | W | W |

| Recommended Item Builds | | Pick Rate | Win Rate |
|---|---|---|---|
| Starter Items | | 94.6% 1,262 | 49.92% |
| | | 2.17% 29 | 41.38% |
| Recommended Builds | | 5.92% 21 | 66.67% |
| | | 3.1% 11 | 18.18% |
| | | 2.54% 9 | 66.67% |
| | | 2.54% 9 | 55.56% |
| | | 2.25% 8 | 50% |


Runes

Resolve + Precision PickRate **59.38%** WinRate 48.40%

Resolve + Inspiration PickRate **20.60%** WinRate 49.11%

Pick Rate **10.26%** 140

Win Rate **53.57%**

Pick Rate **6.23%** 85

Win Rate **40%**