CHEATSHEET NOI

Contents

1. Data Structures
    a. Implicit Treap
    b. treap
    c. Persistent Segment Tree
    d. Dynamic Segment Tree
    e. PBDS
    f. Fenwick_Range
    g. 2D Fenwick
    h. Sparse table
    i. Trie
2. Graph
    a. Tarjan
    b. Kosaraju
    c. Block-cut Tree
    d. Bridge Tree
    e. Heavy-Light Decomposition
    f. LCA O(1) query
    g. LCA Binary Lifting
    h. Centroid Decomposition
    i. Dijkstra’s
    j. SPFA
    k. Bellman Ford
    l. Floyd Warshall
    m. Dinic’s Algorithm
    n. O(VE) Bipartite Matching
3. Math
    a. Matrix Exponentiation
    b. Linear Sieve
    c. Op sieve (nani)
    d. Extended Euclid Algorithm
    e. Miller Rabin Primality Test
4. Geometry
    a. Rumus Rumus
5. DP
    a. Convex Hull Trick
    b. LiChao Tree
    c. LIS
6. VIM SETUP :)

DATA STRUCTURE

1a. Implicit Treap

```
struct node{
	int prior, sz, val;
	bool lz;
	node *l, *r;
	node(int v=0): prior(rnd()), sz(1), lz(0),val(v),
l(NULL), r(NULL){}
};
typedef node* pnode;
int getsz(pnode t){
	return t ? t->sz : 0;
}
void upd_sz(pnode t){
	if (t){
		t->sz = getsz(t->l) + 1 + getsz(t->r);
	}
}

void prop(pnode t){
	if (!t) return;
	if (t->lz){
		//do stuff
	}
	t->lz = 0;
	upd_sz(t);
}
```

```
void split(pnode t, pnode &l, pnode &r, int x, int add =
0){
	if (!t){l = r = NULL;return;}
	prop(t);
	int cur_key = add + getsz(t->l);
	if (cur_key <= x) split(t->r, t->r, r, x,
cur_key+1), l = t;
	else split(t->l, l, t->l, x, add), r = t;
	upd_sz(t);
}

void merge(pnode &t, pnode l, pnode r){
	prop(l); prop(r);
	if (!l || !r){t = l ? l : r;}
	else if (l->prior > r->prior) merge(l->r, l->r,
r), t = l;
	else merge(r->l, l, r->l), t = r;
	upd_sz(t);
}

void rev(pnode t, int l, int r){ // split to left l,
then right r, then swap
	pnode a, b, c;
	split(t, b, c, r);
	split(b, a, b, l-1);
	b->lz ^= 1;//process node b, lz this range
	merge(b, a, b);
	merge(t, b, c);
}
```

1b. Treap

```
struct node{
    LL prior,sz,val;
    LL l, r;
}treap[3000005];
struct dt{
    LL kiri,kanan;
};
LL tridx,root;

void updsz(LL now){
    if(now == 0) return;
    treap[now].sz =
treap[treap[now].l].sz+treap[treap[now].r].sz+1;
}

dt split(LL now, LL tar){
    if(now == 0) return {0,0};
    dt ret;
    if(treap[now].val <= tar){
        dt tmp = split(treap[now].r,tar);
        treap[now].r = tmp.kiri;
        ret = {now,tmp.kanan};
    }else{
        dt tmp = split(treap[now].l,tar);
        treap[now].l = tmp.kanan;
        ret = {tmp.kiri,now};
    }
    updsz(now);
    return ret;
}
```

```
LL merge(LL l, LL r){
    LL ret;
    if(l == 0) ret = r;
    else if(r == 0) ret = l;
    else{
        if(treap[l].prior > treap[r].prior){
            treap[l].r = merge(treap[l].r,r);
            ret = l;
        }else{
            treap[r].l = merge(l,treap[r].l);
            ret = r;
        }
    }
    updsz(ret);
    return ret;
}


bool find(LL now, LL tar){
    while(now){
        if(treap[now].val == tar) return 1;
        if(treap[now].val < tar) now = treap[now].r;
        else now = treap[now].l;
    }
    return 0;
}

void insert(LL tar){
    if(find(root,tar)) return;
    LL cur = ++tridx;
    treap[cur].val = tar;
    treap[cur].prior = rng();
    dt nx = split(root,tar);
```

```
        LL tmp = merge(nx.kiri,cur);
        root = merge(tmp,nx.kanan);
}

void erase(LL tar){
        if(!find(root,tar)) return;
        dt nxa = split(root,tar);
        dt nxb = split(nxa.kiri,tar-1);
        root = merge(nxb.kiri,nxa.kanan);
}
```

1c. Persistent Segment Tree

#nopointersaja

#lebih muda dipahami

```
struct dt{
        LL val;
        LL l,r;
};
LL n,q,cnt;
LL root[100005],isi[100005],RC[100005];
dt segt[3000005];
map <LL,LL> C;


void build(LL kiri, LL kanan, LL idx){
        if(kiri == kanan) return;
        segt[idx].l = ++cnt;
        segt[idx].r = ++cnt;
        LL mid = (kiri+kanan)>>1;
        build(kiri,mid,segt[idx].l);
        build(mid+1,kanan,segt[idx].r);
        segt[idx].val = segt[segt[idx].l].val +
segt[segt[idx].r].val;
}

LL upd(LL kiri, LL kanan, LL idx, LL tar){
        LL curid = ++cnt;
        if(kiri == kanan){
                segt[curid].val = segt[idx].val+1;
                return curid;
        }
        LL mid = (kiri+kanan)>>1;
        segt[curid].l = segt[idx].l;
        segt[curid].r = segt[idx].r;
        if(tar <= mid){
                segt[curid].l =
upd(kiri,mid,segt[idx].l,tar);
        }else{
                segt[curid].r =
upd(mid+1,kanan,segt[idx].r,tar);
        }
        segt[curid].val =
segt[segt[curid].l].val+segt[segt[curid].r].val;
        return curid;
}
```

1d. Dynamic Segment Tree + Lazy Propagation (Sum)

```
struct dt{
        LL lchild, rchild, val;
}segt[1000005];
bool lazy[1000005];
LL q,cntidx;
```

```
void prop(LL kiri, LL kanan, LL idx){
        if(lazy[idx] == 0) return;
        segt[idx].val = kanan-kiri+1;
        if(kiri != kanan){
                if(segt[idx].lchild == 0) segt[idx].lchild =
++cntidx;
                if(segt[idx].rchild == 0) segt[idx].rchild =
++cntidx;
                lazy[segt[idx].lchild] = 1;
                lazy[segt[idx].rchild] = 1;
        }
        lazy[idx] = 0;
        return;
}


void upd(LL kiri, LL kanan, LL idx, LL l, LL r){
        if(kiri > r || kanan < l) return;
        if(segt[idx].val == kanan-kiri+1) return;
        if(l <= kiri && kanan <= r){
                lazy[idx] = 1;
                prop(kiri,kanan,idx);
                return;
        }
        LL mid = (kiri+kanan)/2;
        if(segt[idx].lchild == 0) segt[idx].lchild =
++cntidx;
        if(segt[idx].rchild == 0) segt[idx].rchild =
++cntidx;
        upd(kiri,mid,segt[idx].lchild,l,r);
        upd(mid+1,kanan,segt[idx].rchild,l,r);
        segt[idx].val =
segt[segt[idx].lchild].val+segt[segt[idx].rchild].val;
}

LL get(LL kiri, LL kanan, LL idx, LL l, LL r){
        if(kiri > r || kanan < l) return 0;
        prop(kiri,kanan,idx);
        if(segt[idx].val == kanan-kiri+1) return
min(r,kanan)-max(l,kiri)+1;
        if(l <= kiri && kanan <= r){
                return segt[idx].val;
        }
        LL mid = (kiri+kanan)/2;
        LL ret = 0;
        if(segt[idx].lchild != 0) ret +=
get(kiri,mid,segt[idx].lchild,l,r);
        if(segt[idx].rchild != 0) ret +=
get(mid+1,kanan,segt[idx].rchild,l,r);
        return ret;
}
```

1e. PBDS

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

#define fbo find_by_order
#define ook order_of_key

using namespace __gnu_pbds;
```

```
typedef tree<long long, null_type, less<long long>,
rb_tree_tag, tree_order_statistics_node_update> pbds;
typedef gp_hash_table<long long, null_type> hashset;
//gp_hash_table is a better implementation of
unordered_set/unordered_map but should not be necessary
pbds isi;

int main(){
    isi.insert(val);
    LL tmp = isi.ook(val);
    //ook returns how many numbers less than val
    tmp = (*isi.fbo(k-1));
    //tmp returns pointer to the k-th key, (0-based)
    isi.erase(val);
    tmp = isi.size();
}
```

1f. Fenwick_tree Ranges

```
long ft1[100003];
long ft2[100003];
void update1(int p, int v) { for(; p <= n; p += p&(-p))
ft1[p] += v;}
void update2(int p, int v) { for(; p <= n; p += p&(-p))
ft2[p] += v;}
void update(int a, int b, int v)
{
      update1(a, v);
      update1(b+1, -v);
      update2(a, v*(a-1));
      update2(b+1, -v * b);
}
int query1(int a) { return (a)? ft1[a] + query1(a -
(a&-a)) : 0; }
int query2(int a) { return (a)? ft2[a] + query2(a -
(a&-a)) : 0; }
int query(int b) {return query1(b)*b-query2(b);}
int query(int a, int b) {return query(b)-query(a-1);}
```

1g. 2D Fenwick tree (source : bqi)

```
template <class T, int ...Ns> struct BIT {
   T val = 0;
   void upd(T v) { val += v; }
   T query() { return val; }
};

template <class T, int N, int... Ns> struct BIT<T, N,
Ns...> {
   BIT<T,Ns...> bit[N + 1];
   template<typename... Args> void upd(int pos, Args...
args) {
       for (; pos <= N; pos += (pos&-pos))
bit[pos].upd(args...);
   }
   template<typename... Args> T sum(int r, Args...
args) {
       T res = 0; for (; r; r -= (r&-r)) res +=
bit[r].query(args...);
       return res;
   }
   template<typename... Args> T query(int l, int r,
Args... args) {
       return sum(r,args...)-sum(l-1,args...);
   }
}; // BIT<int,10,10> gives a 2D BIT
```

1h. Sparse Table

```
void precomputeLog(){
  lg[1] = 0;
  for (int i=2; i <= mx; i++) lg[i]=lg[i/2]+1;
}
void buildSparseTable(){
  for (int i = 1; i <= n; i++) spt[0][i] = arr[i];
  for (int i = 1; i <= mxlog; i++){
    for (int j = 1; j <= n && j+(1<<(i-1))<=n; j++){
      spt[i][j] = min(spt[i-1][j],
                      spt[i-1][j+(1<<(i-1))]);
    }
  }
}
```

1i. Dictionary Trie - Binary Trie → trie biasa aja gk sih

```
struct Trie{
        Trie* nx[ALPH_SZ];
        bool has;
        Trie(): has(0){
                for (int i = 0; i < ALPH_SZ; i++) nx[i] = NULL;
        }
};

void insert(Trie* root, int pos, string s){
        if (!root) root = new Trie();
        if (pos == s.length()){
                root->has = 1;
                return;
        }
        insert(root, pos+1, s);
}
```

GRAPH

2a. Tarjan

```
int low[200003], disc[200003];
int rchi
stack<int> stk;
void tarjan(int u)
{
        visited[u] = 1;
        static int tim = 0;//woah ada static
        disc[u] = low[u] = ++ti;
        for(int v : adj[u])
        {
                if(!visited[v])
                {
                        rchi += (disc[u] == 1);
                        parent[v] = u;
                        tarjan(v);
                        low[u] = min(low[u], low[v]);
                        if(low[v] > disc[u])
                                // (u,v) is bridge
                        if((disc[u] == 1 && rchi > 1) || (disc[u] > 1 && low[v] >= disc[u]))
                                // (u) is articulation point
                }
        else if(parent[u] != v)
            low[u] = min(low[u], low[v]);
    }
}
```

2b. Kosaraju
// aku baru sadar kosaraju panjang

```
vector< vector<int> > al_scc;
vector<int> scc;
stack<int> stk; // hati hati stack operplow
void kosa(int now)
{
    visited[now] = 1;
    for(int i : nxt[now])
    {
        if(!visited[i])
        kosa(i);
    }
    stk.push(now);
}

void raju(int now)
{
    visited[now] = 1;
    scc.push_back(now);
    for(int i : bck[now])
    {
        if(!visited[i])
            raju(i);
    }
}

void kosaraju()
{
    for(int i = 1; i <= n; i++)
    {
        if(!visited[i])
            kosa(i);
    }
    memset(visited, 0, sizeof visited);
    while(stk.size())
    {
        int curr = stk.top(); stk.pop();
        scc.clear();
        if(!visited[curr])
        {
            raju(curr);
            al_scc.push_back(scc);
        }
    }
}
```

2c. Tarjan’s SCC

```
LL num[100005],low[100005],isi[100005];
stack <LL> path;
bool vis[100005];
vector <LL> scc[100005];
LL n,dfscnt,scccnt;
vector <LL> edge[100005];

LL tarjan(LL pos){
    num[pos] = low[pos] = ++dfscnt;
    path.push(pos);
    vis[pos] = 1;
    for(int i = 0;i < edge[pos].size();i++){
        LL nx = edge[pos][i];
        if(!num[nx]) tarjan(nx);
```

```
            if(vis[nx]) low[pos] =
min(low[pos],low[nx]);
      }
      if(low[pos] == num[pos]){
            scccnt++;
            while(1){
                  LL now = path.top();
                  path.pop();
                  vis[now] = 0;
                  scc[scccnt].pb(isi[now]);
                  if(now == pos) break;
            }
      }
}

int main(){
    for(int i = 1;i <= n;i++){
      if(num[i]) continue;
      tarjan(i);
    }
}
```

2d. Bridge Tree

```
LL num[100005],low[100005],isi[100005];
struct dt{
      LL to,idx;
};
struct dta{
      LL pos,step;
};
bool isbridge[100005];
vector <dt> edge[100005];
LL low[100005],num[100005];
LL dist[100005],color[100005];
LL curcol,lst;
LL t,dfscnt,cnt;
queue <dta> antri;
vector <LL> newedge[100005];

LL tarjan(LL pos, LL par){
      low[pos] = num[pos] = ++dfscnt;
      for(int i = 0;i < edge[pos].size();i++){
            LL nx = edge[pos][i].to;
            if(nx == par) continue;
            if(!num[nx]){
                  tarjan(nx,pos);
                  if(num[pos] < low[nx])
isbridge[edge[pos][i].idx] = 1;
                  low[pos] = min(low[nx],low[pos]);
            }else{
                  low[pos] = min(low[pos],num[nx]);
            }
      }
}
void build(LL pos, LL curwarna){
      color[pos] = curwarna;
      for(int i = 0;i < edge[pos].size();i++){
            LL nx = edge[pos][i].to;
            if(color[nx]) continue;
            if(isbridge[edge[pos][i].idx]){
                  curcol++;
                  newedge[curwarna].pb(curcol);
                  newedge[curcol].pb(curwarna);
                  build(nx,curcol);
            }else{
                  build(nx,curwarna);
            }
      }
}
```

2e. Shortest Path Count

```
tstruct dt{
      LL pos,step;
};

// vis[i] stores distance from u to i
// cnt[i] stores how many way from u to i
LL vis[5005];
LL cnt[5005];
LD tot[5005];
bool used[5005];
queue <dt> antri;
vector <LL> par[5005];
vector <LL> edge[5005];


void bfs(dt now){
      if(vis[now.pos] < now.step) return;
      if(now.pos == v){
            while(!antri.empty()) antri.pop();
            return;
      }
      for(int i = 0;i < edge[now.pos].size();i++){
            LL nx = edge[now.pos][i];
            if(vis[nx] < now.step+1) continue;
            if(vis[nx] > now.step+1){
                  par[nx].clear();
                  vis[nx] = now.step+1;
                  cnt[nx] = 0;
                  antri.push({nx,now.step+1});
            }
            if(vis[nx] == now.step+1){
                  par[nx].pb(now.pos);
                  cnt[nx] += cnt[now.pos];
            }
      }
}
```

2f. Bellman-Ford Negative Cycle Checking

```
int main(){
    cin >> n >> e;
    for(int i = 1;i <= e;i++){
      LL u,v,c;
      cin >> u >> v >> c;
      edge[u].pb({v,c});
    }
    for(int i = 0;i <= n;i++) dist[i] = LINF;
    dist[0] = 0;
    for(int i = 1;i < n;i++){
      for(int j = 0;j < n;j++){
            if(dist[j] == LINF) continue;
            for(int k = 0;k < edge[j].size();k++){
                  dt now = edge[j][k];
                  dist[now.ke] =
min(dist[now.ke],dist[j]+now.cost);
```

```
            }
      }
    }
    bool ncyc = 0;
    for(int j = 0;j < n;j++){
      if(dist[j] == LINF) continue;
      for(int k = 0;k < edge[j].size();k++){
            dt now = edge[j][k];
            if(dist[now.ke] > dist[j]+now.cost){
                  ncyc = 1;
                  cout << "Ada negative cycle" << endl;
                  break;
            }
      }
      if(ncyc) break;
    }
}
```

2g. Floyd-Warshall's APSP

```
for(int i = 1;i <= n;i++)
      for(int j = 1;j <= n;j++)
            dist[i][j] = LINF;
    for(int i = 1;i <= n;i++) dist[i][i] = 0;
    for(int i = 1;i <= m;i++){
      cin >> u >> v >> c;
      dist[u][v] = dist[v][u] = c;
    }
    for(int k = 1;k <= n;k++)
      for(int i = 1;i <= n;i++)
            for(int j = 1;j <= n;j++)
                  dist[i][j] =
min(dist[i][j],dist[i][k]+dist[k][j]);
```

2h. Kruskal's MST

```
struct dt{
      LL u,v,w;
      bool operator<(const dt& MyStruct)const{
            return w < MyStruct.w;
      }
}isi[100005];

struct dta{
      LL pos,cost;
};

LL par[100005];
bool inmst[100005];
LL n,m,cnt,mst,ans;
vector <dta> edge[100005];

LL findpar(LL now){
      if(now == par[now]) return now;
      return par[now] = findpar(par[now]);
}

int main(){
      fasterios();
      ans = LINF;
    cin >> n >> m;
    for(int i = 1;i <= m;i++) cin >> isi[i].u >>
isi[i].v >> isi[i].w;
    for(int i = 1;i <= n;i++) par[i] = i;
    sort(isi+1,isi+1+m);
      for(int i = 1;i <= m;i++){
            if(findpar(isi[i].u) == findpar(isi[i].v))
continue;
            par[par[isi[i].v]] = par[isi[i].u];
            edge[isi[i].u].pb({isi[i].v,isi[i].w});
            edge[isi[i].v].pb({isi[i].u,isi[i].w});
            mst += isi[i].w;
            cnt++;
            inmst[i] = 1;
            if(n == cnt+1) break;
      }
      for(int i = 1;i <= m;i++){
            if(inmst[i]) continue;
            ans = min(ans,mst+isi[i].w);
      }
      cout << mst << endl;
    return 0;
}
```

2h. Prim Jarnik's MST

```
struct dt{
      LL pos,cost;
      bool operator<(const dt &now)const{
            return cost > now.cost;
      }
};
LL n,m,u,v,c,ans;
bool vis[100005];
vector<dt> edge[100005];
priority_queue <dt> antri;

void prim(dt now){
      if(vis[now.pos]) return;
      vis[now.pos] = 1;
      ans += now.cost;
      for(int i = 0;i < edge[now.pos].size();i++){
            dt nx = edge[now.pos][i];
            if(vis[nx.pos]) continue;
            antri.push({nx.pos,nx.cost});
      }
      return;
}

int main(){
    cin >> n >> m;
    for(int i = 1;i <= m;i++){
      cin >> u >> v >> c;
      edge[u].pb({v,c});
      edge[v].pb({u,c});
    }
      prim({1,0});
    while(!antri.empty()){
      dt tmp = antri.top();
      antri.pop();
      prim(tmp);
    }
    cout << ans << endl;
    return 0;
}
```

2i. Dinic's Maxflow (Hopcroft-Karp's)

```
struct dt{
    LL to,cap,backidx;
};
struct dtb{
    LL pos,step;
};
LL maxflow;
LL n,m,s,e;
LL st[5005];
LL level[5005];
queue <dtb> antri;
vector <dt> edge[5005];

void bfs(dtb now){
    for(int i = 0;i < edge[now.pos].size();i++){
        dt nx = edge[now.pos][i];
        if(level[nx.to]) continue;
        if(nx.cap == 0) continue;
        level[nx.to] = now.step+1;
        antri.push({nx.to,now.step+1});
    }
    return;
}

LL dfs(LL now, LL bcap){
    if(now == e) return bcap;
    for(;st[now] < edge[now].size();st[now]++){
        dt nx = edge[now][st[now]];
        if(level[nx.to] != level[now]+1) continue;
        if(nx.cap == 0) continue;
        LL ret = dfs(nx.to,min(bcap,nx.cap));
        if(ret == 0) continue;
        edge[now][st[now]].cap -= ret;
        edge[nx.to][nx.backidx].cap += ret;
        return ret;
    }
    return 0;
}

int main(){
    fastll(n); fastll(m);
    for(int i = 1;i <= m;i++){
        LL u,v,c; fastll(u); fastll(v); fastll(c);
        LL szu = edge[u].size();
        LL szv = edge[v].size();
        edge[u].pb({v,c,szv});
        edge[v].pb({u,c,szu});
    }
    s = 1; e = n;
    // LL cnt = 0;
    while(1){
        // if(++cnt == 2) break;
        for(int i = 1;i <= n;i++) level[i] = 0;
        antri.push({s,1});
        level[s] = 1;
        while(!antri.empty()){
            dtb tmp = antri.front();
            antri.pop();
            bfs(tmp);
        }
        if(level[e] == 0) break;
        LL curflow;
        memset(st,0,sizeof(st));
        while(curflow = dfs(s,LINF)) maxflow += curflow;
    }
    printf("%lld\n",maxflow);
    return 0;
}
```

2j. O(VE) Bipartite Matching

```
bool bpm(LL u){
      int nx;
      for(int i = 0;i < edge[u].size();i++){
            nx = edge[u][i];
            if(vis[nx]) continue;
            vis[nx] = 1;
            if(pas[nx] == -1 || bpm(pas[nx])){
                  pas[nx] = u;
                  return 1;
            }
      }
      return 0;
}

//int main function
          for(int i = 1;i <= n;i++){
            memset(vis,0,sizeof(vis));
            cnt += bpm(i);
          }
```

2k. Heavy light decomposition

```
// require LCA to use this HLD
// this algorithm is freaking OP
vector<int> vec[200003];
int sz;
int no[200003], head[200003], id[200003], dp[200003],
lup[200003];
int T[200003][20];
int chain[200003], arr[200003];

// insert range query DS here
```

```
// HLD + segment tree = GEGE HAHAHAHA
void build()
{
      for(int i = 0; i <= 18; i++)
            for(int j = 1; j+(1<<i)-1 <= sz; j++)
                  T[j][i] = (i == 0)? chain[j] :
max(T[j][i-1],T[j+(1<<(i-1))][i-1]);
}

int RMQ(int kir, int kan)
{
      int x = (int)log2(kan-kir+1);
      return max(T[kir][x], T[kan-(1<<x)+1][x]);
}

void count_subtree(int u)
{
      dp[u] = 1;
      for(int v : vec[u])
      {
            if(!dp[v])
            {
                  count_subtree(v);
                  dp[u] += dp[v];
            }
      }
}

// create HLD array
void HLD(int u, int H)
{
      ++sz;
```

```
        head[u] = H; id[u] = sz; no[sz] = u; chain[sz] =
arr[u];

        int mx = 0, hev = 0;
        for(int v : vec[u])
        {
                if(!id[v] && dp[v] > mx)
                {
                        mx = dp[v];
                        hev = v;
                }
        }

        if(hev) HLD(hev, H);
        for(int v : vec[u])
        {
                if(!id[v] && v != hev)
                {
                        lup[v] = u;
                        HLD(v,v);
                }
        }
}

int  qtohead(int u, int H)
{
        int mx = 0;
        while(head[u] != head[H])
        {
                mx = max(mx,RMQ(id[head[u]], id[u]));
                u = lup[head[u]];
        }
        if(u != H) mx = max(mx,RMQ(id[H]+1, id[u]));
        return mx;
}
int query(int u, int v) { return
max(qtohead(u,LCA(u,v)), qtohead(v,LCA(u,v))); }
```

MATH

3a. Matrix Exponentiation // sama ky fast expo sih tbh

```
struct Matrix{
  LL m[103][103]; // typedef LL dulu
  Matrix(){memset(m, 0, sizeof(m));}
  Matrix operator * (Matrix const &rhs) const {
    Matrix ret;
    for (int i = 1; i <= 100; i++){
      for (int j = 1; j <= 100; j++){
        ret.m[i][j] = 0;
        for (int k = 1; k <= 100; k++){
          ret.m[i][j] += ((m[i][k] * rhs.m[k][j]) %
mod);
          ret.m[i][j] %= mod;
        }
      }
    }
    return ret;
  }
};

Matrix fastpow(Matrix b, LL e){
  Matrix ret;
  for (int i = 1; i <= 100; i++) ret.m[i][i] = 1;
  while (e > 0){
    if (e&1) ret = ret * b;
    b = b * b;
    e >>= 1;
  }
  return ret;
}
```

3b. Linear Sieve

```
void sieve(){
      memset(is_prime,1,sizeof(is_prime));
      is_prime[0] = is_prime[1] = 0;
      for (int i = 2; i <= 1000000; i++){
            if (is_prime[i])
                  prime.push_back(i);
            for (int j = 0; j < prime.size() && i
* prime[j] <= 1000000; j++){
                  is_prime[i * prime[j]] = false;
                  if (i % prime[j] == 0) break;
            }
      }
}
```

3c. OP Sieve

```
#define MAX 100000000
#define SQ 10000
#define check(n) (isPrime[n>>6]&(1<<((n&63)>>1)))
#define set(n) isPrime[n>>6]|=(1<<((n&63)>>1))

LL t,n;
int isPrime[MAX>>6];
vector <LL> prime;

int fastersieve(){
      for(int i=3;i<=SQ;i+=2){
            if (!check(i)){
                  int inc = 2*i;
                  for(int j=i*i;j<=MAX;j+=inc){
                        set(j);
                  }
            }
      }
      prime.pb(2);
      for(int i = 3;i <= MAX;i += 2)
            if(!check(i)) prime.pb(i);
}
```

3d. Extended Euclid Algorithm

```
//returns g = gcd(a, b); finds x, y such that d =
ax + by
int extended_euclid(int a, int b, int &x, int
&y){
      int xx = y = 0;
      int yy = x = 1;
      while (b){
            int q = a/b;
            int t = b; b = a%b; a = t;
            t = xx; xx = x - q * xx; x = t;
            t = yy; yy = y - q * yy; y = t;
      }
      return a;
}
```

// Further Implementation

```
struct dt{
      LL x,y;
};
dt nol;
LL n,ca,na,cb,nb,curgcd;

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

void exgcd(LL a, LL b){
      if(b == 0){
            nol = {1,0};
            return;
      }
      exgcd(b,a%b); dt tmp;
      tmp = {nol.y,nol.x-(a/b)*nol.y};
      nol = tmp;
}

int main(){
      cin >> n;
      cin >> na >> nb;
      curgcd = gcd(na,nb);
      if(n%curgcd != 0){
            //No solution
      }
      exgcd(na,nb);
      LL aa = na/curgcd;
      LL bb = nb/curgcd;
    nol.x *= n/curgcd;
    nol.y *= n/curgcd;
    LL curk = //any number is a solution for na*x + nb*y = n
      LL curx = nol.x+curk*bb;
      LL cury = nol.y-curk*aa;
    return 0;
}
```

GEOMETRI

```
//Titip dulu Geometri
     // can we struct point pls
```

4a. Angle of ABC in °, or simply return alpha in radian

```
struct point{
      LD x,y;
}

LD getangle(point a, point b, point c){
  point ab = { b.x - a.x, b.y - a.y };
  point cb = { b.x - c.x, b.y - c.y };
    LD dot = (ab.x * cb.x + ab.y * cb.y); // dot product
    LD cross = (ab.x * cb.y - ab.y * cb.x); // cross product
    LD alpha = atan2(cross,dot);
    LD rs = (alpha*180.0)/PI;
  return rs;
}
```

4b. Get a point with distance r, given angle in ° curang, and center p

```
point get(LD curang){
  LD piang = (curang*PI)/180.0;
  point curin = {p.x+(cos(piang)*r),p.y+(sin(piang)*r)};
  return curin;
}
```

4c. Shoelace Formula, area of a polygon (sets of points)

```
int main(){
    cin >> n;
    for(int i = 1;i <= n;i++) cin >> isi[i].x >> isi[i].y;
    for(int i = 1;i <= n;i++){
      ans += isi[i].x*isi[(i%n)+1].y;
      ans -= isi[(i%n)+1].x*isi[i].y;
    }
    ans /= 2.0;
}
```

4d. Point-Segment Distance, point inters is the projection of smth. IDK. imani saja

```
int main(){
        point a,b,c;
        cin >> a.x >> a.y;
        cin >> b.x >> b.y >> c.x >> c.y;
        LD A = a.x-b.x; LD B = a.y-b.y;
        LD C = c.x-b.x; LD D = c.y-b.y;
        LD dot = A*C+B*D;
        LD len_sq = C*C+D*D;
        if(b.x == c.x && b.y == c.y){
                cout << dist(a,b) << endl;
                return 0;
        }
        LD param = dot/len_sq;
        if(param >= 0 && param <= 1){
                point inters;
                inters.x = b.x+param*C;
                inters.y = b.y+param*D;
                cout << dist(inters,a) << endl;
                return 0;
        }
        cout << min(dist(a,b),dist(a,c)) << endl;

}
```

4e. Clockwise check

```
Given (x0,y0),(x1,y1),(x2,y2)
        D = (x1-x0)*(y2-y0)-(x2-y0)*(y1-y0)
        D = 0, collinear
```

```
    D > 0, counter clockwise, left
    D < 0, clockwise, right
```

4f. Segment-segment Intersection, maaf bad variable naming.

```
int main(){
    cin >> x1 >> yy1 >> x2 >> y2;
    cin >> x3 >> y3 >> x4 >> y4;
    LD x =
((x1*y2-yy1*x2)*(x3-x4)-(x1-x2)*(x3*y4-x4*y3))/((x1-x2)*
(y3-y4)-(yy1-y2)*(x3-x4));
    LD y =
((x1*y2-yy1*x2)*(y3-y4)-(yy1-y2)*(x3*y4-x4*y3))/((x1-x2)
*(y3-y4)-(yy1-y2)*(x3-x4));
    LD xst = min(x1,x2);
    LD xed = max(x1,x2);
    if(x < xst || x > xed) cant();
    xst = min(x3,x4);
    xed = max(x3,x4);
    if(x < xst || x > xed) cant();
    LD yst = min(yy1,y2);
    LD yed = max(yy1,y2);
    if(y < yst || y > yed) cant();
    yst = min(y3,y4);
    yed = max(y3,y4);
    if(y < yst || y > yed) cant();
    cout << fixed << setprecision(10) << x << " " << y
<< endl;
}
```

4g. Graham Scan

```
struct point{
    LD x,y;
    bool operator <(const point &other)const{
        if(x == other.x) return y < other.y;
        return x < other.x;
    }
};
LL n;
point pivot;
point isi[105];
vector <point> ch;
LL solve(point p, point q,point r){
    LD cross =
(r.x-q.x)*(p.y-q.y)-(r.y-q.y)*(p.x-q.x);
    if(fabs(cross) < eps) return 0; //Collinear
    else if(cross < 0) return -1; //Clockwise
    else return 1; //counter-clockwise
}
LL dist(point a,point b){
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}
bool cmp(point a, point b) {
    LL dir = solve(pivot,a,b);
    if(dir == 0) return dist(pivot,a) < dist(pivot,b);
    return (dir == -1);
}
int main(){
    cin >> n;
    for(int i = 1;i <= n;i++){
    cin >> isi[i].x >> isi[i].y;
    if(i == 1) pivot = isi[i];
    pivot = min(pivot,isi[i]);
    }
    sort(isi+1,isi+1+n,cmp);
    for(int i = 1;i <= n;i++){
```

```
    LL ntmp = ch.size();
    while(ntmp >= 2){
        if(solve(ch[ntmp-2],ch[ntmp-1],isi[i]) <= 0)
break;
        else ch.popb();
        ntmp = ch.size();
    }
    ch.pb(isi[i]);
    }
}
```

4h. Monotone Chain

```
LL n;
point pivot;
point isi[105];
vector <point> ch;
vector <point> upperhull;
vector <point> lowerhull;
int main(){
    cin >> n;
    for(int i = 1;i <= n;i++)
        cin >> isi[i].x >> isi[i].y;
    sort(isi+1,isi+1+n);
    for(int i = 1;i <= n;i++){
        LL ntmp = upperhull.size();
        while(ntmp >= 2){
            if(solve(upperhull[ntmp-2],upperhull[n
        tmp-1],isi[i]) <= 0) break;
            else upperhull.popb();
            ntmp = upperhull.size();
        }
        upperhull.pb(isi[i]);
    }
    for(int i = 1;i <= n;i++){
        LL ntmp = lowerhull.size();
        while(ntmp >= 2){
            if(solve(lowerhull[ntmp-2],lowerhull[n
        tmp-1],isi[i]) <= 0) break;
            else lowerhull.popb();
            ntmp = lowerhull.size();
        }
        lowerhull.pb(isi[i]);
    }
}
```

4i. O(N) Inverse Modulo

```
const int mxsz = 2e5 + 10;
const int MX = 2e5 + 1;
LL fac[mxsz], inv[mxsz], finv[mxsz];
int n, m;

void prec(){ // factorial / inverse
    fac[0] = inv[0] = finv[0] = 1;
    fac[1] = inv[1] = finv[1] = 1;
    for (int i = 2; i <= MX; i++){
        fac[i] = (fac[i-1] * i) % mod;
        inv[i] = mod - (mod/i) * inv[mod%i] % mod;
        finv[i] = (finv[i-1] * inv[i]) % mod;
    }
}
```

DP

5a. DP Knuth

```
//Define dp(i,j) = minimum cost to cut string from i to j
//Define opt(i,j) = first occurence of the answer of dp(i,j)
//In this problem, the property opt(i,j-1) <= opt(i,j) <= opt(i+1,j) works

LL isi[1005];
LL memo[1005][1005];
LL opt[1005][1005];
int main(){
      fasterios();
      LL n,m;
    while(cin >> n >> m){
      for(int i = 1;i <= m;i++) cin >> isi[i];
      isi[++m] = n;
      //Make the base case for 2 segment, we can only cut the one in the middle;
      for(int i = 0;i+2 <= m;i++){
            memo[i][i+2] = isi[i+2]-isi[i];
            opt[i][i+2] = i+1;
      }
      for(int i = 3;i <= m;i++){
            //Compute dp for i segment, we shrink down the search by using the opt property
            //Knuth here
            for(int j = 0;j+i <= m;j++){
                  LL cur = LINF;
                  for(int k = opt[j][j+i-1];k <= opt[j+1][j+i];k++){
                        LL now = memo[j][k]+memo[k][j+i];
                        if(cur > now){
                              cur = now;
                              opt[j][j+i] = k;
                        }
                  }
                  memo[j][j+i] = cur+isi[j+i]-isi[j];
            }
      }
      cout << memo[0][m] << endl;
    }
    return 0;
}
```

5b. IOI 2016 Alien Solution (Convex Hull + Monge Property BS)

```
struct dt{
    LL a,b;
    bool operator<(const dt &other)const{
        if(a == other.a) return b > other.b;
        return a < other.a;
    }
};
struct line{
    LL m,c,idx;
};

double getintersection(line a,line b){
    return (double)(a.c-b.c)/(double)(a.m-b.m);
```

```
}

LL gety(line a, LL x){
    return a.m*x+a.c;
}

LL N;
LL curk;
LL par[100005];
dt isi[100005];
deque <line> hull;
dt isiold[100005];
LL memo[100005],cnt;

void cari(LL curslope){
    hull.clear();
    //dp ch, g(x) = f(x)-c(x)
    // where f(x) is a function which has non
decreasing slope
    cnt = 0;
    for(int i = 1;i <= N;i++){
        LL tmp;
        if(isi[i-1].b < isi[i].a) tmp = 0;
        else tmp =
(isi[i-1].b-isi[i].a+1)*(isi[i-1].b-isi[i].a+1);
        line nx =
{2*isi[i].a,memo[i-1]-(2*isi[i].a)+(isi[i].a*isi[i].a)+1
-tmp,i-1};
        while(hull.size() > 1){
            LL blkg = hull.size()-1;
            if(getintersection(hull[blkg],hull[blkg-1])
< getintersection(hull[blkg],nx)) break;
            hull.popb();
```

```
        }
        hull.pb(nx);
        while(hull.size() > 1){
            if(gety(hull[0],-isi[i].b) <
gety(hull[1],-isi[i].b)) break;
            hull.popf();
        }
        memo[i] =
gety(hull[0],-isi[i].b)+(isi[i].b*isi[i].b)+(2*isi[i].b)
+curslope;
        par[i] = hull[0].idx;
    }
    LL now = N;
    while(now){
        // cout << "here is " << now << endl;
        cnt++;
        now = par[now];
    }
}

long long take_photos(int n, int m, int k, vector<int>
r, vector<int> c) {

    for(int i = 0;i < n;i++){
        if(r[i] > c[i]) swap(r[i],c[i]);
        isiold[i+1] = {r[i],c[i]};
    }
    sort(isiold+1,isiold+1+n);
    LL cur = -1;
    LL newn = 0;
    isi[0] = {-1,-1};
    for(int i = 1;i <= n;i++){
        if(cur == -1 || isiold[cur].b < isiold[i].b){
            isi[++newn] = isiold[i];
            cur = i;
        }else continue;
    }

    N = newn;
    LL kiri = 0; LL kanan = 1000000000000000000LL;
    //Makin dibesarin midnya, k nya akan makin kecil
    //Makin kecil midnya, k nya semakin besar
    //Cari cnt pertama yang sama dengan k
    while(kiri < kanan){
        LL mid = (kiri+kanan+1)>>1;
        cari(mid);
        if(cnt >= k) kiri = mid;
        else kanan = mid-1;
    }
    cari(kiri);
    return memo[N]-k*kiri;
}
```

5c. DP Deque

```
struct dt{
    LL val,idx;
};
LL n,k,x;
LL isi[5005];
LL memo[5005][5005];

int main(){
    fasterios();
    cin >> n >> k >> x;
    for(int i = 1;i <= n;i++){
        cin >> isi[i];
    }
    for(int i = 0;i <= n;i++){
        for(int j = 0;j <= x;j++){
            memo[i][j] = -LINF;
        }
    }
    memo[0][0] = 0;
    for(int i = 1;i <= x;i++){
        deque <dt> antri;
        antri.pb({0,0});
        for(int j = 1;j <= n;j++){
            while(antri.front().idx < j-k) antri.popf();
            memo[j][i] = antri.front().val+isi[j];
            while(!antri.empty() && antri.back().val <=
memo[j][i-1]) antri.popb();
            antri.pb({memo[j][i-1],j});
        }
    }
    LL ans = -1;
    for(int i = (n-k+1);i <= n;i++){
        ans = max(ans,memo[i][x]);
    }
    cout << ans << endl;
    return 0;
}
```

5d. DP SOS

```
LL n;
LL isi[10005];
LL dp[17000000];
```

```
int main(){
    cin >> n;
    for(int i = 1;i <= n;i++){
        string s; cin >> s;
        for(int j = 0;j < 3;j++){
            isi[i] |= (1LL<<(s[j]-'a'));
        }
        dp[isi[i]]++;
        // cout << isi[i] << endl;
    }
    //Dp[mask] contain all numbers of submask
    for(int i = 0;i <= 23;i++){
        for(int mask = (1LL<<24)-1;mask >= 0;mask--){
            if(mask&(1LL<<i))
                dp[mask] += dp[mask^(1LL<<i)];
        }
    }
    LL ans = 0;
    for(int mask = (1LL<<24)-1;mask >= 0;mask--){
        dp[mask] = n-dp[mask];
        ans ^= (dp[mask]*dp[mask]);
    }
    cout << ans << endl;
    return 0;
}
```

5e. Slope Trick

```
//Menyebrangi Sungai - TLX.

LL n,h,l[100005],r[100005],len[100005];

priority_queue<LL> lower;
//Grafik yang slopenya negatif

priority_queue<LL,vector<LL>,greater<LL>> upper;
//Grafik yang slopenya positif

LL lazylow, lazyhigh;

int main(){
    cin >> n;
    for(int i = 1;i <= n;i++){
        cin >> l[i] >> r[i];
        len[i] = r[i]-l[i];
    }
    //dp(n,k) = minimum cost untuk kayu ke n, apabila
kepala kayu ke n berada pada posisi k
    //define len[i] = r[i]-l[i]
    // dp(n,k) = allmin[ k-len[i-1] <= K' <= k+len[i] ]
dp(n-1,K') + |k-l[i]|
    // dp(n,k) (baru) pengen dari dp(n-1,k-len[i-1])
    // dp(n,k) (baru) pengen dari dp(n-1,k+len[i])
    LL ans = 0;
    upper.push(l[1]); lower.push(l[1]);
    for(int i = 2;i <= n;i++){
        lazylow -= len[i];
        //Lazylow would be negative
        // as we translate it more to the left
        lazyhigh += len[i-1];
        // Lazyup would be positive
        // Change lazy values based on question
        //Kasus 1
        if(l[i] <= lower.top()+lazylow){
            //The update point is on the negative slope
of our graph
            LL shifted = lower.top()+lazylow;
            ans += (shifted-l[i]);
            lower.push(l[i]-lazylow);
            lower.push(l[i]-lazylow);
            lower.pop();
            upper.push(shifted-lazyhigh);
        }
        // kasus 2
        else if(l[i] >= upper.top()+lazyhigh) {
            //The update point is on the positive slope
of our graph
            LL shifted = upper.top()+lazyhigh;
            ans += (l[i]-shifted);
            upper.push(l[i]-lazyhigh);
            upper.push(l[i]-lazyhigh);
            upper.pop();
            lower.push(shifted-lazylow);
        }

        // kasus 3
        else {
            // cout << "Kasus 3 " << endl;
            lower.push(l[i]-lazylow);
            upper.push(l[i]-lazyhigh);
        }
    }
    cout << ans << endl;
    return 0;
}
```

5f. LCRS

```
void lcrs(LL pos,LL par){
    LL be = -1;
    for(int i = 0;i < edge[pos].size();i++){
        LL nx = edge[pos][i];
        if(par == nx) continue;
        lcrs(nx,pos);
        if(isi[pos].kiri == -1) isi[pos].kiri = nx;
        if(be != -1) isi[be].kanan = nx;
        be = nx;
    }
    return;
}
```

5G - LI CHAO TREE

Can use dynamic segtree. Node for range [L, R) stores best line for that range.

```
const int maxn = 2e5; // max range

Line st[4 * maxn]; // initialize to {0, inf}

void add_line(Line nw, int v = 1, int l = 0, int r =
maxn) {
    int m = (l + r) / 2;
    bool lef = nw.get(l) < st[v].get(l);
    bool mid = nw.get(m) < st[v].get(m);
    if(mid) {
        swap(st[v], nw);
    }
    if(r - l == 1) {
        return;
    } else if(lef != mid) {
        //update [l, m)
        add_line(nw, 2 * v, l, m);
    } else {
        //update [m, r)
        add_line(nw, 2 * v + 1, m, r);
    }
}

int get(int x, int v = 1, int l = 0, int r = maxn) {
    int m = (l + r) / 2;
    if(r - l == 1) {
        return line[v].get(x);
    } else if(x < m) {
        return min(line[v].get(x), get(x, 2 * v, l, m));
    } else {
        return min(line[v].get(x), get(x, 2 * v + 1, m,
r));
    }
}
```

MISCELLANEOUS:

6A. Vim template - configure ~/.vimrc

```
syntax on
set ruler
set number
set laststatus=2
set tabstop=4
set softtabstop=4
set shiftwidth=4
set foldmethod=indent
set nofoldenable
filetype indent on
```

6B. Cpp template

```
#include <bits/stdc++.h>
#define fi first
#define se second
#define mp make_pair
#define pb push_back
typedef long long ll;
using namespace std;
ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);
```

6C. Fast Input for integers

```
inline int readInt(){
  int num = 0; bool sgn = 0;
  char c = getchar();
  for (; c < '0' || c > '9'; c = getchar()){
    if (c == '-') sgn = 1;
  }
  for (; c >= '0' && c <= '9'; c = getchar()){
    num = (num<<3) + (num<<1) + c - '0';
  }
  return sgn ? -num : num;
}
```