2-SAT

```
int n,m,id;
vector<int>G[16001],RG[16001],assignment;
stack<int>st;
int vis[16001];

inline int node(int u)
{
   if(u>0) return 2*u-1;
   return -2*u;
}

inline int compliment(int u)
{
   if(u<0) return -2*u-1;
   return 2*u;
}

void dfs1(int s)
{
   vis[s]=1;
   REP(i,G[s].size())
   {
      int u=G[s][i];
      if(!vis[u]) dfs1(u);
   }
   st.push(s);
}

void dfs2(int s)
{
   vis[s]=id;
   REP(i,RG[s].size())
   {
      int u=RG[s][i];
      if(!vis[u]) dfs2(u);
   }
}

bool solve_2SAT()
{
   MEM(vis,0);
   FOR(i,1,2*m) if(!vis[i]) dfs1(i);
   MEM(vis,0);
   id=1;
   while(!st.empty())
   {
      int u=st.top();
      st.pop();
      if(!vis[u]) dfs2(u),id++;
   }
   assignment.clear();
   for(int i=1;i<=2*m;i+=2)
   {
      if(vis[i]==vis[i+1]) return false;
      if(vis[i]>vis[i+1]) assignment.pb(i/2+1);
   }
   return true;
}

int main()
{
   int t;
   scanf("%d",&t);
   FOR(tc,1,t)
   {
      scanf("%d %d",&n,&m);
      FOR(i,1,n)
      {
         int u,v;
         scanf("%d %d",&u,&v);
         G[compliment(u)].pb(node(v));
         G[compliment(v)].pb(node(u));

         RG[node(v)].pb(compliment(u));
         RG[node(u)].pb(compliment(v));
      }
      bool f=solve_2SAT();
      if(f)
      {
         printf("Case %d: Yes\n%d",tc,assignment.size());
         REP(i,assignment.size()) printf(" %d",assignment[i]);
         printf("\n");
      }
      else
      {
         printf("Case %d: No\n",tc);
      }
      FOR(i,1,2*m) G[i].clear(),RG[i].clear();
   }
}
```

Bridge:

```
const int N = 100005;

vector<int>G[N];
bool vis[N];
int discover[N], low[N], pr[N];
vector<pii>br;

void dfs(int u)
{
   static int time = 0;
   vis[u] = 1;
   discover[u] = low[u] = ++time;

   for(int v : G[u]){
      if(!vis[v]){
         pr[v] = u;
         dfs(v);
         low[u] = min(low[u],low[v]);
         if(low[v]>discover[u]){
            br.pb(mk(u,v));
         }
      }
      else if(pr[u]!=v){
         low[u] = min(low[u],discover[v]);
      }
   }
}

int main()
{
   MEM(pr,-1);
   int n,m;
   cin >> n >> m;
```

```
    for(int i = 0;i<m;i++){
        int a,b;
        cin >> a >> b;
        G[a].pb(b);
        G[b].pb(a);
    }
    for(int i = 1;i<=n;i++){
        if(!vis[i])dfs(i);
    }
    for(int i = 0;i<br.size();i++){
        cout << br[i].ff << " " << br[i].ss << endl;
    }
}
```

Dijkstra:

```
struct point
{
    int name,val;
    bool operator <(const point &p) const
    {
        return p.val < val;
    }
};
const int N=100005;
vector<pii>V[N];
int dis[N];
priority_queue<point>Q;

void Dijkstra(int s)
{
    dis[s]=0;
    point get;
    get.name=s;
    get.val=0;
    Q.push(get);
    while(!Q.empty())
    {
        point tmp=Q.top();
        Q.pop();
        int now=tmp.name;
        REP(i,V[now].size())
        {
            int x=V[now][i].ff;
            int y=V[now][i].ss;
            if(dis[now]+y<dis[x])
            {
                dis[x]=dis[now]+y;
                get.name=x;
                get.val=dis[x];
                Q.push(get);
            }
        }
    }
    return;
}
```

Kruskal:

```
struct edge
{
    int u,v,w;
    bool operator < (const edge &p) const
    {
        return w < p.w;
    }
};
const int N=100005;
int pr[N];
vector<edge>e;
int find(int r)
{
    return pr[r]= (pr[r]==r) ? r:find(pr[r]);
}
int kruskal(int n)
{
    sort(e.begin(),e.end());
    FOR(i,1,n) pr[i]=i;
    int cnt=0,sum=0;
    REP(i,e.size())
    {
        int x=find(e[i].u);
        int y=find(e[i].v);
        if(x!=y)
        {
            pr[x]=y;
            cnt++;
            sum+=e[i].w;
            if(cnt==n-1)
                break;
        }
    }
    return sum;
}
```

Bellman Ford:

```
struct edge
{
    int u,v,w;
};
vector<edge>V;
int dis[105];
int n,m;

bool bellmanford(int s)
{
    FOR(i,1,n) dis[i]=INT_MAX;
    dis[s]=0;
    FOR(i,1,n-1) // For V=1000,around 20 iteration works fine.
    {
        REP(j,V.size())
        {
            edge e=V[j];
            if(dis[e.v]>dis[e.u]+e.w)
            {
                dis[e.v]=dis[e.u]+e.w;
            }
        }
    }

    REP(j,V.size())
    {
        edge e=V[j];
        if(dis[e.v]>dis[e.u]+e.w)
        {
            return false;
        }
    }
```

```
  return true;
}

int main()
{
  cin >> n >> m;
  FOR(i,1,m)
  {
    int u,v,w;
    cin >> u >> v >> w;
    V.pb({u,v,w});
  }
  if(bellmanford(1)) cout << "No Negative Cycle\n";
  else cout << "Negative Cycle\n";
}
```

Floyd Warshall:

```
int n,m;
int graph[105][105];

void FloydWarshall()
{
  int i,j,k;
  REP(k,n)
  {
    REP(i,n)
    {
      REP(j,n)
      {
        if(graph[i][k]+graph[k][j]<graph[i][j])
        {
          graph[i][j]=graph[i][k]+graph[k][j];
        }
        //graph[i][j]=graph[i][j] || (graph[i][k] &&
graph[k][j]);
      }
    }
  }
}

int main()
{
  cin >> n >> m;
  REP(i,n)
  {
    REP(j,n)
    {
      if(i==j) graph[i][j]=0;
      else graph[i][j]=1e7;
    }
  }
  REP(i,m)
  {
    int u,v,w;
    cin >> u >> v >> w;
    graph[u][v]=w;
    graph[v][u]=w;
  }
}
```

Building DAG using SCC:

```
const int N=100005;
vector<int>V[N],DAG[N];
int id;
int vis[N];
stack<int>st;

void dfs1(int s)
{
  vis[s]=1;
  for(int i:V[s])
  {
    if(i<0 or vis[i]) continue;
    dfs1(i);
  }
  st.push(s);
}

void dfs2(int s)
{
  vis[s]=id;
  for(int i:V[s])
  {
    i=-i;
    if(i<0 or vis[i]) continue;
    dfs2(i);
  }
}

int main()
{
  int n,m;
  cin >> n >> m;
  REP(i,m)
  {
    int u,v;
    cin >> u >> v;
    V[u].pb(v);
    V[v].pb(-u);    //Reverse Graph
  }
  FOR(i,1,n) if(vis[i]==0 and V[i].size()>0) dfs1(i);
  MEM(vis,0);
  id=0;
  while(!st.empty())
  {
    int u=st.top();
    st.pop();
    if(vis[u]) continue;
    id++;
    dfs2(u);
  }
  FOR(i,1,n)
  {
    for(int j:V[i]){
      if(j<0) continue;
      if(vis[i]!=vis[j]) DAG[vis[i]].pb(vis[j]);
    }
  }

}
```

Heavy Light Decomposition:

```
const int N = 100005;
int n,child[N],pr[N],Lev[N],ara[N];
int P[N][22];

int id = 1,chainID[N],atPos[N],chainHead[N],cNodes[N];
vector<int>G[N];

int dfs(int node,int pre,int dep)
{
   pr[node] = pre;
   Lev[node] = dep;
   int ret = 1;
   for(int i = 0;i < G[node].size();i++){
      int go = G[node][i];
      if(go == pre)continue;
      ret += dfs(go,node,dep + 1);
   }
   return child[node] = ret;
}
void init()
{
   for(int i = 0;i < N;i++)for(int j = 0;j<22;j++)P[i][j] = 1;
   FOR(i,1,N)P[i][0] = pr[i];

   for(int j = 1;(1 << j) < N;j++){
      for(int i = 0;i < N;i++){
         P[i][j] = P[P[i][j-1]][j-1];
      }
   }
}
int lca(int p,int q)
{
   if(Lev[p] < Lev[q])swap(p,q);

   for(int i = 21;i>=0;i--){
      if(Lev[P[p][i]] >= Lev[q])p = P[p][i];
   }
   if(p == q)return p;
   for(int i = 21;i>=0;i--){
      if(P[p][i]!=P[q][i]){
         p = P[p][i]; q = P[q][i];
      }
   }
   return pr[p];
}

struct DATA{
   int l,r,val;
}tree[4 * N];

void update(int node,int L,int R,int pos,int val)
{
   if(pos < L or pos > R)return;
   if(L == R){
      tree[node].val = val;
      return;
   }
   int mid = (L + R)/2;
   if(tree[node].l == 0)tree[node].l = id++;
   if(tree[node].r == 0)tree[node].r = id++;

   int Lnode = tree[node].l;
   int Rnode = tree[node].r;

   update(Lnode,L,mid,pos,val);
   update(Rnode,mid + 1,R,pos,val);

   tree[node].val = max(tree[Lnode].val,tree[Rnode].val);
}
int query(int node,int L,int R,int l,int r)
{
   if(l > r)swap(l,r);
   if(r < L or R < l)return 0;
   if(l <= L and R <= r)return tree[node].val;

   int mid = (L + R)/2;
   int Lnode = tree[node].l;
   int Rnode = tree[node].r;

   int x = query(Lnode,L,mid,l,r);
   int y = query(Rnode,mid+1,R,l,r);
   return max(x,y);
}

queue<int>Q;
void getChain(int node,int cid,int pos,int head)
{
   chainID[node] = cid;
   atPos[node] = pos;
   chainHead[node] = head;
   cNodes[cid]++;

   int heavyChild , sz = 0;
   for(int i = 0;i < G[node].size();i++){
      int go = G[node][i];
      if(go == pr[node])continue;
      if(child[go] > sz){
         sz = child[go];
         heavyChild = go;
      }
   }
   for(int i = 0;i < G[node].size();i++){
      int go = G[node][i];
      if(go == pr[node])continue;
      if(go == heavyChild)getChain(go,cid,pos + 1,head);
      else Q.push(go);
   }
}

void process()
{
   dfs(1,1,0);
   init();

   Q.push(1);
   while(!Q.empty()){
      int node = Q.front();
      Q.pop();
      getChain(node,id++,1,node);
   }
   for(int i = 1;i < n;i++){
      update(chainID[n + i],1,cNodes[chainID[n+i]],atPos[n +
i],ara[n + i]);
   }
}
```

```
int Get(int p,int q)
{
   if(chainID[p] == chainID[q]){
      return
query(chainID[p],1,cNodes[chainID[p]],atPos[p],atPos[q]);
   }else{
      int H = chainHead[p];
      int mx =
query(chainID[p],1,cNodes[chainID[p]],atPos[p],atPos[H]);
      return max(mx,Get(pr[H],q));
   }
}

void input()
{
   scanf("%d",&n);
   for(int i = 1;i < n;i++){
      int a,b,c;
      scanf("%d %d %d",&a,&b,&c);

      ara[n + i] = c;
      G[a].pb(n + i);
      G[b].pb(n + i);
      G[n + i].pb(a);
      G[n + i].pb(b);
   }
}

void answer_me()
{
   char str[105];
   while(1){
      scanf("%s",str);
      if(str[0] == 'D')break;
      if(str[0] == 'Q'){
         int a,b;
         scanf("%d %d",&a,&b);
         int l = lca(a,b);
         int ans = max(Get(a,l),Get(b,l));
         printf("%d\n",ans);
      }else{
         int p,v;
         scanf("%d %d",&p,&v);
         ara[n + p] = v;
         update(chainID[n + p],1,cNodes[chainID[n+p]],atPos[n
+ p],ara[n + p]);
      }
   }
}

int main()
{
   int t;
   scanf("%d",&t);
   while(t--){
      MEM(cNodes,0);
      for(int i = 0;i < N;i++)G[i].clear();
      input();
      process();
      answer_me();
      if(t)printf("\n");
   }
}
```

Dinic Max Flow:

```
struct edge
{
   int to,rev,f,cap;
};

const int maxnodes=10005;
int s,t,lev[maxnodes],q[maxnodes],work[maxnodes];
vector<edge>g[maxnodes];

inline void addEdge(int u,int v,int w)
{
   edge a= {v,g[v].size(),0,w};
   edge b= {u,g[u].size(),0,0};
   g[u].pb(a);
   g[v].pb(b);
}

bool dinic_bfs()
{
   MEM(lev,-1);
   lev[s]=0;
   int idx=0;
   q[idx++]=s;
   REP(i,idx)
   {
      int u=q[i];
      REP(j,g[u].size())
      {
         edge &e=g[u][j];
         if(lev[e.to]<0 and e.f<e.cap)
         {
            lev[e.to]=lev[u]+1;
            q[idx++]=e.to;
         }
      }
   }
   return lev[t]>=0;
}

int dinic_dfs(int u,int f)
{
   if(u==t) return f;
   for(int &i=work[u];i<g[u].size();i++)
   {
      edge &e=g[u][i];
      if(e.cap<=e.f) continue;
      if(lev[e.to]==lev[u]+1)
      {
         int flow=dinic_dfs(e.to,min(f,e.cap-e.f));
         if(flow>0)
         {
            e.f+=flow;
            g[e.to][e.rev].f-=flow;
            return flow;
         }
      }
   }
   return 0;
}

int maxFlow()
{
    int ret=0;
    while(dinic_bfs())
    {
        MEM(work,0);
        while(int flow=dinic_dfs(s,INT_MAX))
            ret+=flow;
    }
    return ret;
}

int main()
{
    int n,m;
    cin >> n >> m >> s >> t;
    REP(i,m)
    {
        int u,v,w;
        cin >> u >> v >> w;
        addEdge(u,v,w);
        addEdge(v,u,w); //If bidirectional
    }
    cout << maxFlow();
    return 0;
}
```

Min Cost Max Flow:

```
const int N = 105;
int cost[N][N],cap[N][N],ara[N][N];
int pr[N],dis[N];

vector<pair<int,int> >e;
void addEdge(int u,int v,int w)
{
   e.push_back(make_pair(u,v));
   e.push_back(make_pair(v,u));
   cap[u][v] += 1;
   cost[u][v] = 1000000-w;
   cost[v][u] = 1000000+w;
}

int bel(int s,int t)
{
   for(int i = 0;i<N;i++)dis[i] = 1e9;
   dis[s] = 0;
   pr[s] = 0;
   for(int i = 0;i<N;i++){
      for(int j = 0;j<e.size();j++){
         int u = e[j].first;
         int v = e[j].second;
         if(cap[u][v] == 0)continue;
         if(dis[u] + cost[u][v] < dis[v]){
            dis[v] = dis[u] + cost[u][v];
            pr[v] = u;
         }
      }
   }
   if(dis[t] == 1e9)return 0;
   else return 1;
}

void init()
{
   memset(cost,0,sizeof(cost));
   memset(cap,0,sizeof(cap));
   memset(pr,0,sizeof(pr));
   memset(dis,0,sizeof(dis));
   memset(ara,0,sizeof(ara));
   e.clear();
}
int main()
{
   int t,cases=0;
   cin >> t;
   while(t--)
   {
     init();
     int n;
     cin >> n;
     for(int i = 1;i<=n;i++){
       for(int j = 1;j<=n;j++){
          cin >> ara[i][j];
          addEdge(i , n + j, ara[i][j]);
       }
     }
     for(int i = 1;i<=n;i++){
       addEdge(0,i,0);
       addEdge(n + i,2*n + 1,0);
     }
     int flow = 0,answer = 0;
     while(bel(0,2*n+1))
     {
       for(int v = 2*n + 1;v!=0;v=pr[v]){
          int u = pr[v];
          cap[u][v]-=1;
          cap[v][u]-=1;
          answer += cost[u][v];
       }
       flow++;
     }
     cout << "Case " << ++cases << ": " << (n*3*1000000 -
answer) << "\n";
   }
}
```

Hopcroft Karp Bipartite Matching:

```
const int MAXN1 = 1505;
const int MAXN2 = 1505;
const int MAXM = 80000;

int n1, n2, edges, last[MAXN1], Prev[MAXM], head[MAXM];
int matching[MAXN2], dist[MAXN1], Q[MAXN1];
bool used[MAXN1], vis[MAXN1];

void init(int _n1, int _n2) {
    n1 = _n1;
    n2 = _n2;
    edges = 0;
    fill(last, last + n1, -1);
}

void addEdge(int u, int v) {
    head[edges] = v;
    Prev[edges] = last[u];
    last[u] = edges++;
}
```

```
void bfs() {
   fill(dist, dist + n1, -1);
   int sizeQ = 0;
   for (int u = 0; u < n1; ++u) {
      if (!used[u]) {
         Q[sizeQ++] = u;
         dist[u] = 0;
      }
   }
   for (int i = 0; i < sizeQ; i++) {
      int u1 = Q[i];
      for (int e = last[u1]; e >= 0; e = Prev[e]) {
         int u2 = matching[head[e]];
         if (u2 >= 0 && dist[u2] < 0) {
            dist[u2] = dist[u1] + 1;
            Q[sizeQ++] = u2;
         }
      }
   }
}

bool dfs(int u1) {
   vis[u1] = true;
   for (int e = last[u1]; e >= 0; e = Prev[e]) {
      int v = head[e];
      int u2 = matching[v];
      if (u2 < 0 || !vis[u2] && dist[u2] == dist[u1] + 1 &&
dfs(u2)) {
         matching[v] = u1;
         used[u1] = true;
         return true;
      }
   }
   return false;
}

int maxMatching() {
   fill(used, used + n1, false);
   fill(matching, matching + n2, -1);
   for (int res = 0;;) {
      bfs();
      fill(vis, vis + n1, false);
      int f = 0;
      for (int u = 0; u < n1; ++u)
         if (!used[u] && dfs(u))
            ++f;
      if (!f)
         return res;
      res += f;
   }
}
pair<int,int> ara[75005];
int main()
{
   FastRead
   int m,n,k;
   cin >> m >> n >> k;
   for(int i = 0;i < k;i++){
      cin >> ara[i].first >> ara[i].second;
      ara[i].first--;
      ara[i].second--;
   }
   int Ans = 0;
   for(int j = 1;j <= m;j++){
      init(m + 2,n);
      for(int i = 0;i < k;i++){
         addEdge(ara[i].first,ara[i].second);
         if(ara[i].first == j){
            addEdge(m,ara[i].second);
            addEdge(m + 1,ara[i].second);
         }
      }
      Ans = max(Ans,maxMatching());
   }
   cout << Ans << "\n";

}
```

Bipartite Matching Another Implementation:

```
const int N = 1505;
int matchR[N],store[N];
bool vis[N];
vector<int>G[N];

bool dfs(int u){
   for(int i = 0;i < G[u].size();i++){
      int v = G[u][i];
      if(vis[v])continue;
      vis[v] = 1;
      if(matchR[v] == 0 || dfs(matchR[v])){
         matchR[v] = u;
         return true;
      }
   }
   return false;
}
int main(){
   FastRead
   int m,n,k;
   cin >> m >> n >> k;
   for(int i = 0;i < k;i++){
      int a,b;
      cin >> a >> b;
      G[a].pb(b);
   }
   int Ans = 0, Extra = 0;
   for(int i = 1;i <= m;i++){
      MEM(vis,0);
      if(dfs(i))Ans++;
   }
   for(int i = 1;i <= n;i++)store[i] = matchR[i];

   for(int i = 1;i <= m;i++){
      int temp = 0;
      for(int j = 1;j <= n;j++)matchR[j] = store[j];
      MEM(vis,0);
      temp += dfs(i);

      MEM(vis,0);
      temp += dfs(i);

      Extra = max(Extra,temp);
      if(Extra == 2)break;
   }
   cout << Ans + Extra << "\n";
}
```

Segment Tree Lazy Propagation:

```
const int N=100005;
ll tree[4*N],lazy[4*N];

void updateRange(int b,int e,int L,int R,int pos,ll val)
{
  if(lazy[pos]!=0)
  {
    tree[pos]+=(R-L+1)*lazy[pos];
    if(L!=R)
    {
      lazy[2*pos+1]+=lazy[pos];
      lazy[2*pos+2]+=lazy[pos];
    }
    lazy[pos]=0;
  }
  if(L>R or L>e or R<b)
    return;
  if(L>=b and R<=e)
  {
    tree[pos]+=(R-L+1)*val;
    if(L!=R)
    {
      lazy[2*pos+1]+=val;
      lazy[2*pos+2]+=val;
    }
    return;
  }
  int mid=(L+R)/2;
  updateRange(b,e,L,mid,2*pos+1,val);
  updateRange(b,e,mid+1,R,2*pos+2,val);
  tree[pos]=tree[2*pos+1]+tree[2*pos+2];
  return;
}

ll getSum(int ql,int qr,int L,int R,int pos)
{
  if(lazy[pos]!=0)
  {
    tree[pos]+=(R-L+1)*lazy[pos];
    if(L!=R)
    {
      lazy[2*pos+1]+=lazy[pos];
      lazy[2*pos+2]+=lazy[pos];
    }
    lazy[pos]=0;
  }
  if(L>R or ql>R or qr<L)
    return 0;
  if(L>=ql and qr>=R)
    return tree[pos];
  int mid=(L+R)/2;
  return
getSum(ql,qr,L,mid,2*pos+1)+getSum(ql,qr,mid+1,R,2*pos+2)
;
}
```

Segment Tree Special And/Or:

```
const int N = 100000;
struct info{
  int val,lazy = -1;
}tree[26][4*N];

void pushDown(int id,int at,int L,int R)
{
  if(tree[id][at].lazy == -1)return;
  if(tree[id][at].lazy == 0){
    tree[id][at].val = 0;
  }else{
    tree[id][at].val = R - L + 1;
  }

  if(L != R){
    tree[id][2*at].lazy = tree[id][at].lazy;
    tree[id][2*at+1].lazy = tree[id][at].lazy;
  }
  tree[id][at].lazy = - 1;
}

void update(int id,int at,int L,int R,int l,int r,int v)
{
  pushDown(id,at,L,R);

  if(L > R or L > r or R < l)return;

  if(L>=l and R<=r){
    tree[id][at].lazy = v;
    pushDown(id,at,L,R);
    return;
  }
  int mid = (L + R)/2;
  update(id,2*at,L,mid,l,r,v);
  update(id,2*at+1,mid+1,R,l,r,v);
  tree[id][at].val = tree[id][2*at].val + tree[id][2*at+1].val;
}

int query(int id,int at,int L,int R,int l,int r)
{
  if(L > R or L > r or R < l)return 0;
  pushDown(id,at,L,R);
  if(L>=l and R<=r)return tree[id][at].val;
  int mid = (L+R)/2;
  int p1 = query(id,2*at,L,mid,l,r);
  int p2 = query(id,2*at+1,mid+1,R,l,r);
  return p1 + p2;
}
```

Persistent Segment Tree:

```
struct data {
  int l, r, c;
  data() {
    l = r = c = 0;
  }
  data(int a, int b, int d) {
    l = a;
    r = b;
    c = d;
  }
}T[N * 20];

int n, in[N], Root[N], id, qr;

int update(int pr, int b, int e, int pos) {
  int node = ++id;
  T[node] = T[pr];
```

```
    if(b == e) {
        T[node].c++;
        return node;
    }
    int mid = b + e >> 1;
    if(pos <= mid) T[node].l = update(T[node].l, b, mid, pos);
    else T[node].r = update(T[node].r, mid + 1, e, pos);
    T[node].c = T[ T[node].l ].c + T[ T[node].r ].c;
    return node;
}

int query(int pr, int cr, int b, int e, int nd) {
    if(b == e) return b;
    int have = T[ T[cr].l ].c - T[ T[pr].l ].c;
//    cout << "from : " << b << " " << e << " , " << T[cr].c - T[pr].c
<< '\n';
    int mid = b + e >> 1;
    if(nd <= have) return query(T[pr].l, T[cr].l, b, mid, nd);
    else return query(T[pr].r, T[cr].r, mid + 1, e, nd - have);
}

int main()
{
    scanf("%d %d", &n, &qr);
    for(int i = 1; i <= n; i++) scanf("%d", &in[i]);
    vi cmp; cmp.push_back(-inf);
    for(int i = 1; i <= n; i++) cmp.push_back(in[i]); Unique(cmp);
    for(int i = 1; i <= n; i++) in[i] = lower_bound(all(cmp), in[i])
- cmp.begin();
    Root[0] = ++id;
    for(int i = 1; i <= n; i++) Root[i] = update(Root[i - 1], 1,
cmp.size(), in[i]);
    while(qr--) {
        int l, r, k; scanf("%d %d %d", &l, &r, &k);
        int p = query(Root[l - 1], Root[r], 1, cmp.size(), k);
        int val = cmp[p];
//        cerr << p << '\n';
        printf("%d\n", val);
    }
    return 0;
}
```

BIT:

```
int tree[100005];
int query(int idx)
{
    int sum=0;
    for(; idx>0; idx-=idx & (-idx))
        sum+=tree[idx];
    return sum;
}
void update(int idx,int val,int n)
{
    for(; idx<=n; idx+=idx & (-idx))
        tree[idx]+=val;
}
```

2D Range BIT:

```
int n;
long long int BIT[2][2][1025][1025];

void update(int x,int y,long long int value){
    int xx=x;
    while(xx<=n){
        int yy=y;
        while(yy<=n){
            BIT[x%2][y%2][xx][yy]^=value;
            yy+=(yy&-yy);
        }
        xx+=(xx&-xx);
    }
}

long long int sum(int x,int y){
    long long int ans=0;
    int xx=x;
    while(xx!=0){
        int yy=y;
        while(yy!=0){
            ans^=BIT[x%2][y%2][xx][yy];
            yy-=(yy&-yy);
        }
        xx-=(xx&-xx);
    }
    return ans;
}

int main(){
    int p,q;
    int a,b,c,d;
    long long int val;
    scanf("%d %d",&n,&q);
    while(q--){
        scanf("%d",&p);
        if(p==1){
            scanf("%d %d %d %d",&a,&b,&c,&d);
            long long int ans=sum(c,d)^sum(a-1,b-1)^sum(c,b-
1)^sum(a-1,d);
            printf("%I64d\n",ans);
        }
        else{
            scanf("%d %d %d %d %I64d",&a,&b,&c,&d,&val);
            update(a,b,val);
            update(a,d+1,val);
            update(c+1,b,val);
            update(c+1,d+1,val);
        }
    }
}
```

3D BIT:

```
long long matrix[101][101][101];
void update(long long n,long long x,long long y,long long
z,long long  val) {
    long long y1,x1;

    while(z <= n) {
        x1 = x;
        while(x1 <= n) {
            y1 = y;
            while(y1 <= n) {
                matrix[x1][y1][z] += val;
                y1 += (y1 & -y1 );
            }
            x1 += (x1 & -x1);
        }
        z += (z & -z);
    }

}

long long calculate_sum(long long  x,long long y,long long z) {
    long long y1,x1,sum=0;
    while (z>0) {
        x1=x;
        while(x1>0) {
            y1=y;
            while(y1>0) {
                sum += matrix[x1][y1][z];
                y1-= (y1 & -y1);
            }
            x1 -= (x1 & -x1);
        }
        z -= (z & -z);

    }
    return sum;
}

void process(long long n,long long m) {

    long long x,y,z,x0,y0,z0;
    long long value1,value2,val;
    char command[10];

    memset(matrix,0,sizeof(matrix));

    while(m--) {
        scanf("%s",command);

        if(!strcmp(command,"QUERY")) {
            scanf("%lld %lld %lld %lld %lld
%lld",&x0,&y0,&z0,&x,&y,&z);

            value1 = calculate_sum(x,y,z)- calculate_sum(x0-1,y,z)
                    - calculate_sum(x,y0-1,z) + calculate_sum(x0-1,y0-
1,z);

            value2 = calculate_sum(x,y,z0-1) - calculate_sum(x0-
1,y,z0-1)
                    - calculate_sum(x,y0-1,z0-1)  + calculate_sum(x0-
1,y0-1,z0-1);

            printf("%lld\n",value1 - value2);
            //PrintMatrix(n);

        }

        if(!strcmp(command,"UPDATE")) {

            scanf("%lld %lld %lld %lld",&x,&y,&z,&val);
            x0 = x;
            y0 = y;
            z0 = z ;

            value1 = calculate_sum(x,y,z)- calculate_sum(x0-1,y,z)
                    - calculate_sum(x,y0-1,z) + calculate_sum(x0-1,y0-
1,z);
            value2 = calculate_sum(x,y,z0-1) - calculate_sum(x0-
1,y,z0-1)
                    - calculate_sum(x,y0-1,z0-1)  + calculate_sum(x0-
1,y0-1,z0-1);

            update(n,x,y,z,val -(value1 - value2 ));

        }

    }
}
int main() {
    long long cases; scanf("%lld",&cases);
    while(cases--) {

        long long n,m; scanf("%lld %lld",&n,&m);
        process(n,m);
    }
    return 0;
}
```

MO Basic:

```
int n,q,block_size,ans;
int arr[MAX],cnt[MAX],answer[MAX];
pair<pii,int>qry[MAX];

bool mo_cmp(pair<pii,int>x,pair<pii,int>y)
{
    int blk_x=x.ff.ff/block_size;
    int blk_y=y.ff.ff/block_size;
    if(blk_x!=blk_y)
        return blk_x<blk_y;
    return x.ff.ss < y.ff.ss;
}

void add(int x)
{
    if(cnt[x]==0) ans++;
    cnt[x]++;
}

void Remove(int x)
{
    cnt[x]--;
    if(cnt[x]==0) ans--;
}

int main()
{
    cin >> n;
    REP(i,n) cin >> arr[i];
    cin >> q;
    REP(i,q)
    {
        cin >> qry[i].ff.ff >> qry[i].ff.ss;
        qry[i].ss=i;
    }
    block_size=sqrt(n);
    sort(qry,qry+q,mo_cmp);
    int ml=0,mr=-1;
    REP(i,q)
    {
        int l=qry[i].ff.ff;
    int r=qry[i].ff.ss;
    while(mr<r)
    {
      mr++;
      add(arr[mr]);
    }
    while(mr>r)
    {
      Remove(arr[mr]);
      mr--;
    }
    while(ml<l)
    {
      Remove(arr[ml]);
      ml++;
    }
    while(ml>l)
    {
      ml--;
      add(arr[ml]);
    }
    answer[qry[i].ss]=ans;
  }
  REP(i,q) cout << answer[i] << '\n';
  return 0;
}
```

MO With Updates:

```
const int MAX=100005;
int id,n,q,block_size,ans;
int arr[MAX],answer[MAX],freq[MAX],cnt[2*MAX];
pii update[MAX];
pair<pii,pii>qry[MAX];

map<int,int>mp;`

bool mo_cmp(pair<pii,pii>x,pair<pii,pii>y)
{
  if(x.ff.ff/block_size!=y.ff.ff/block_size)
    return x.ff.ff/block_size<y.ff.ff/block_size;
  if(x.ff.ss/block_size!=y.ff.ss/block_size)
    return x.ff.ss/block_size<y.ff.ss/block_size;
  return x.ss.ff < y.ss.ff;
}

void add(int x)
{
  freq[cnt[x]]--;
  cnt[x]++;
  freq[cnt[x]]++;
}

void Remove(int x)
{
  freq[cnt[x]]--;
  cnt[x]--;
  freq[cnt[x]]++;
}

void _update(int i,int u,int v)
{
  int idx=update[i].ff;
  int val=update[i].ss;
  if(idx>v or idx<u) swap(arr[idx],update[i].ss);
  else
  {
    Remove(arr[idx]);
    add(val);
    swap(arr[idx],update[i].ss);
  }
}

int main()
{
  FastRead
  cin >> n >> q;
  FOR(i,1,n)
  {
    cin >> arr[i];
    if(mp[arr[i]]==0)
    {
      mp[arr[i]]=++id;
      arr[i]=id;
    }
    else arr[i]=mp[arr[i]];
  }
  int up=0,qr=0;
  REP(i,q) {
    int u,v,w;
    cin >> u >> v >> w;
    if(u==1)
    {
      qry[qr]=mk(pii(v,w),pii(up,qr));
      qr++;
    }
    else
    {
      if(mp[w]==0) mp[w]=++id;
      update[++up]=pii(v,mp[w]);
    }
  }
  block_size=cbrt(n)*cbrt(n);
  sort(qry,qry+qr,mo_cmp);
  int ml=1,mr=0,mu=0;
  REP(i,qr)
  {
    int l=qry[i].ff.ff;
    int r=qry[i].ff.ss;
    int u=qry[i].ss.ff;
    while(mu<u)
    {
      mu++;
      _update(mu,ml,mr);
    }
    while(mu>u)
    {
      _update(mu,ml,mr);
      mu--;
    }
    while(mr<r)
    {
      mr++;
      add(arr[mr]);
    }
    while(mr>r)
    {
      Remove(arr[mr]);
      mr--;
    }
    while(ml<l)
    {
      Remove(arr[ml]);
      ml++;
    }
    while(ml>l)
    {
      ml--;
      add(arr[ml]);
    }
    FOR(j,1,700)
    {
      if(freq[j]==0 and answer[qry[i].ss.ss]==0)
      {
        answer[qry[i].ss.ss]=j;
        break;
      }
    }
  }
  REP(i,qr) cout << answer[i] << '\n';
  return 0;
}
```

MO on Tree:

```
const int N=40005;
int n,m,id,block_size=290,ans,tmp;
int
arr[N],ST[N],EN[N],Pr[N],L[N],P[N][22],flag[2*N],answer[100
005],cnt[N],inrange[N];
vector<int>V[N];
pair<pii,pii>qry[100005];
map<int,int>mp;

void dfs(int s,int p,int d)
{
  Pr[s]=p;
  L[s]=d;
  ST[s]=++id;
  flag[id]=s;
  for(int i:V[s])
    if(i!=p)
      dfs(i,s,d+1);
  EN[s]=++id;
  flag[id]=s;
}

void lca()
{
  dfs(1,0,1); //Source,Prev_Node(0/-1),Depth
  REP(i,N) REP(j,22) P[i][j]=1;
  FOR(i,1,N-1) P[i][0]=Pr[i];
  for(int j=1; (1<<j)<N; j++)
  {
    REP(i,N)
    {
      P[i][j]=P[P[i][j-1]][j-1];
    }
  }
}

int query(int p,int q)
{
  if(L[p]<L[q])
    swap(p,q);
  ROF(i,21,0) if(L[P[p][i]]>=L[q])
    p=P[p][i];
  if(p==q)
    return p;
  ROF(i,21,0)
  {
    if(P[p][i]!=P[q][i])
    {
      p=P[p][i];
      q=P[q][i];
    }
  }
  return Pr[p];
}

bool mo_cmp(pair<pii,pii>x,pair<pii,pii>y)
{
  int blk_x=x.ff.ff/block_size;
  int blk_y=y.ff.ff/block_size;
  if(blk_x!=blk_y)
    return blk_x<blk_y;
  return x.ff.ss < y.ff.ss;
}

void add(int i)
{
  if(inrange[i])
  {
    cnt[arr[i]]--;
    if(cnt[arr[i]]==0) ans--;
  }
  else
  {
    if(cnt[arr[i]]==0) ans++;
    cnt[arr[i]]++;
  }
  inrange[i]^=1;
}

int main()
{
  scanf("%d %d",&n,&m);
  FOR(i,1,n)
  {
    scanf("%d",&arr[i]);
    if(mp[arr[i]]==0)
      mp[arr[i]]=++tmp;
    arr[i]=mp[arr[i]];
  }
  FOR(i,1,n-1)
  {
    int u,v;
    scanf("%d %d",&u,&v);
    V[u].pb(v);
    V[v].pb(u);
  }
  lca();
  FOR(i,1,m)
  {
    int u,v;
    scanf("%d %d",&u,&v);
    if(ST[u]>ST[v])
      swap(u,v);
    int p=query(u,v);
    if(u==p or v==p)
      qry[i]=mk(pii(ST[u],ST[v]),pii(i,0));
    else
      qry[i]=mk(pii(EN[u],ST[v]),pii(i,ST[p]));
  }
  sort(qry+1,qry+m+1,mo_cmp);
  int ml=1,mr=0;
  FOR(i,1,m)
  {
    int l=qry[i].ff.ff;
    int r=qry[i].ff.ss;
    while(mr<r)
    {
      mr++;
      add(flag[mr]);
    }
    while(mr>r)
    {
      add(flag[mr]);
      mr--;
    }
    while(ml<l)
    {
      add(flag[ml]);
      ml++;
    }
    while(ml>l)
    {
      ml--;
      add(flag[ml]);
    }
    if(qry[i].ss.ss!=0) add(flag[qry[i].ss.ss]);
    answer[qry[i].ss.ff]=ans;
    if(qry[i].ss.ss!=0) add(flag[qry[i].ss.ss]);
  }
  FOR(i,1,m) printf("%d\n",answer[i]);
  return 0;
}
```

Centroid Decomposition:

```
const int N=100005;
int n,q;
set<int>G[N];
int sub[N],par[N];

void dfs(int node,int pr)
{
  sub[node]=1;
  for(int i:G[node])
  {
    if(i==pr) continue;
    dfs(i,node);
    sub[node]+=sub[i];
  }
}

int centroid(int node,int pr,int sz)
{
  for(int i:G[node])
  {
    if(i==pr) continue;
    if(sub[i]>sz) return centroid(i,node,sz);
  }
  return node;
}

void decompose(int node,int pr)
{
  dfs(node,-1);
  int c=centroid(node,-1,(sub[node]+1)/2);
  par[c]=pr;
  for(int i:G[c])
  {
    G[i].erase(c);
    decompose(i,c);
  }
}

int main()
{
  cin >> n >> q;
  for(int i=1;i<n;i++)
  {
    int u,v;
    cin >> u >> v;
    G[u].insert(v);
    G[v].insert(u);
  }
  decompose(1,-1);
  return 0;
}
```

DSU On Tree:

```
string str[MAX];
vector<int>G[MAX];
vector<pii>Q[MAX];
int L[MAX],ans[MAX];

void dfs(int v,int d)
{
  L[v]=d;
  for(int i:G[v])
  {
    dfs(i,d+1);
  }
  return;
}

void dsu(int v,map<int,set<string>>&mp)
{
  for(int i:G[v])
  {
    map<int,set<string>>s;
    dsu(i,s);
    if(s.size()>mp.size()) swap(mp,s);
    for(auto it:s)
    {
      mp[it.ff].insert(all(it.ss));
    }
  }
```

```
    if(v!=0) mp[L[v]].insert(str[v]);
    for(pii p:Q[v])
    {
        ans[p.ss]=mp[p.ff].size();
    }
    return;
}

int main()
{
    //FastRead
    int n;
    cin >> n;
    FOR(i,1,n)
    {
        int u;
        cin >> str[i] >> u;
        G[u].pb(i);
    }
    dfs(0,0);
    int q;
    cin >>q;
    FOR(i,1,q)
    {
        int v,k;
        cin >> v >> k;
        Q[v].pb(pii(k+L[v],i));
    }
    map<int,set<string>>mp;
    dsu(0,mp);
    FOR(i,1,q)
    {
        cout << ans[i] << '\n';
    }
    return 0;
}
```

SOS DP:

```
const int N = 22;
int F[1 << N],n;
int ara[1000006];
int lim = (1 << N) - 1;

int main()
{
   MEM(F,-1);
   int n;
   scanf("%d",&n);
   FOR(i,1,n)scanf("%d",&ara[i]) , F[ara[i]] = ara[i];
   LL ans = 0;

   for(int i = 0;i < N; ++i) {
      for(int mask = 0; mask < (1<<N); ++mask){
         if(mask & (1<<i) and F[mask^(1<<i)] > 0){
            F[mask] = F[mask^(1<<i)];
         }
      }
   }
   for(int i = 1;i<=n;i++){
      printf("%d ",F[ara[i]^lim]);
   }
}
```

Divide and Conquer Dp Optimization:

```
const int N = 5005;
int n,k,ara[N],A[N][N];
LL dp[N][N];

inline void solve(int L,int R,int x,int y,int id)
{
    if(L > R)return;
    int mid = (L + R)/2;
    pair<LL,int> best = mk(-1,-1);
    for(int i = x;i <= min(y,mid);i++){
        best = max(best,{dp[id-1][i-1] + A[i][mid],i});
    }
    dp[id][mid] = best.first;
    solve(L,mid-1,x,best.ss,id);
    solve(mid+1,R,best.ss,y,id);
}
int main()
{
    int t;
    scanf("%d",&t);
    while(t--){

        scanf("%d %d",&n,&k);
        for(int i = 1;i <= n;i++)scanf("%d",&ara[i]);

        for(int i = 1;i <= n;i++){
            A[i][i] = ara[i];
            for(int j = i + 1;j <= n;j++){
                A[i][j] = A[i][j-1] | ara[j];
            }
        }
        for(int i = 1;i <= n;i++){
            dp[0][i] = A[1][i];
        }
        for(int i = 1;i < k;i++){
            solve(1,n,1,n,i);
        }
        printf("%lld\n",dp[k-1][n]);
    }
}
```

Convex Hull Trick Dp Optimization:

```
const int N = 100005;

LL n,A[N],B[N],dp[N],Q[N];

LL compute(int i,int j)
{
    return dp[j] + B[j] * A[i];
}

double secant(int x,int y)
{
    return (double)(dp[y] - dp[x])/(B[x] - B[y] + 0.0);
}
int main()
{
    cin >> n;
    for(int i = 1;i <= n;i++)cin >> A[i];
    for(int i = 1;i <= n;i++)cin >> B[i];

    LL sz = 0,p = 1;
```

```
  for(int i = 1;i <= n;i++){
    while(p < sz and compute(i,Q[p]) >=
compute(i,Q[p+1]))p++;
    dp[i] = compute(i,Q[p]);
    while(p < sz and secant(Q[sz-1],Q[sz]) >=
secant(Q[sz],i))sz--;
    Q[++sz] = i;
  }
  cout << dp[n] << "\n";
}
```

Manacher Algorithm:

```
#define SIZE 100000 + 1
int P[SIZE * 2];

// Transform S into new string with special characters
inserted.
string convertToNewString(const string &s) {
  string newString = "@";

  for (int i = 0; i < s.size(); i++) {
    newString += "#" + s.substr(i, 1);
  }

  newString += "#$";
  return newString;
}

string longestPalindromeSubstring(const string &s) {
  string Q = convertToNewString(s);
  int c = 0, r = 0;           // current center, right limit

  for (int i = 1; i < Q.size() - 1; i++) {
    // find the corresponding letter in the palidrome
subString
    int iMirror = c - (i - c);
    if(r > i) {
      P[i] = min(r - i, P[iMirror]);
    }

    // expanding around center i
    while (Q[i + 1 + P[i]] == Q[i - 1 - P[i]]){
      P[i]++;
    }

    // Update c,r in case if the palindrome centered at i
expands past r,
    if (i + P[i] > r) {
      c = i;          // next center = i
      r = i + P[i];
    }
  }
  // Find the longest palindrome length in p.

  int maxPalindrome = 0;
  int centerIndex = 0;

  for (int i = 1; i < Q.size() - 1; i++) {
    if (P[i] > maxPalindrome) {
      maxPalindrome = P[i];
      centerIndex = i;
    }
  }
  cout << maxPalindrome << "\n";
  return s.substr( (centerIndex - 1 - maxPalindrome) / 2,
maxPalindrome);
}
int main() {
  string s = "kiomaramol\n";
  cout << longestPalindromeSubstring(s);
  return 0;
}
```

KMP:

```
int lps[2000006];
string txt,pat;

void failure_table()
{
  int i=1,j=0,len=pat.size();
  lps[0]=0;
  while(i<len)
  {
    if(pat[i]==pat[j]){
      j++;
      lps[i]=j;
      i++;
    }
    else{
      if(j!=0) j=lps[j-1];
      else{
        lps[i]=0;
        i++;
      }
    }
  }
  return;
}

int KMP()
{
  int m=pat.size();
  int n=txt.size();
  failure_table();
  int i=0,j=0;
  while(i<n){
    if(pat[j]==txt[i]){
      i++;
      j++;
    }
    if(j==m) return i-j;
    else if(i<n and pat[j]!=txt[i]){
      if(j!=0) j=lps[j-1];
      else i++;
    }
  }
  return -1;
}

int main()
{
  cin >> txt >> pat;
  cout << KMP();
  return 0;
}
```

Z Algorithm:
```
const int N = 1000005;
int Z[N];
int n,m;
string str;
void Function()
{
  int L = 0, R = 0 , k ,n = str.size();
  for(int i = 1;i < n;i++){
    if(i > R){
      L = R = i;
      while (R<n && str[R-L] == str[R])R++;
      Z[i] = R-L;
      R--;
    }else{
      k = i - L;
       if (Z[k] < R-i+1)Z[i] = Z[k];
       else{
         L = i;
         while (R<n && str[R-L] == str[R])R++;
         Z[i] = R-L;
         R--;
       }
    }
  }
}
```

Trie:
```
int tri[1000005][26]; //Total char in input file,Number of distinct char
bool flag[1000005]; //Indicate where string finishes
int id=1;

int main()
{
  string str;
  cin >> str;
  int r=1;
  REP(i,str.size())
  {
    int x=str[i]-'a'; // It maybe '0'/'A'/both
    if(!tri[r][x])
    {
      tri[r][x]=++id;
    }
    r=tri[r][x];
  }
  flag[r]=true;
}
```

Suffix Array:
```
const int N = 2000006;
const int M = 22;

int n, stp, sfxMv, sfx[N], tmp[N];
int sfxSum[N], sfxCnt[N], Rank[M][N];
int lcp[N], rnk[N];
char in[N];

char a[N], b[N];

inline bool Equal(const int &u, const int &v){
  if(!stp) return in[u] == in[v];
  if(Rank[stp-1][u] != Rank[stp-1][v]) return false;
  int a = u + sfxMv < n ? Rank[stp-1][u+sfxMv] : -1;
  int b = v + sfxMv < n ? Rank[stp-1][v+sfxMv] : -1;
  return a == b;
}

void update(){
  int i, rnk;
  for(i = 0; i < n; i++) sfxSum[i] = 0;
  for(i = rnk = 0; i < n; i++) {
    sfx[i] = tmp[i];
    if(i && !Equal(sfx[i], sfx[i-1])) {
      Rank[stp][sfx[i]] = ++rnk;
      sfxSum[rnk+1] = sfxSum[rnk];
    }
    else Rank[stp][sfx[i]] = rnk;
    sfxSum[rnk+1]++;
  }
}

void Sort() {
  int i;
  for(i = 0; i < n; i++) sfxCnt[i] = 0;
  memset(tmp, -1, sizeof tmp);
  for(i = 0; i < sfxMv; i++){
    int idx = Rank[stp - 1][n - i - 1];
    int x = sfxSum[idx];
    tmp[x + sfxCnt[idx]] = n - i - 1;
    sfxCnt[idx]++;
  }
  for(i = 0; i < n; i++){
    int idx = sfx[i] - sfxMv;
    if(idx < 0)continue;
    idx = Rank[stp-1][idx];
    int x = sfxSum[idx];
    tmp[x + sfxCnt[idx]] = sfx[i] - sfxMv;
    sfxCnt[idx]++;
  }
  update();
  return;
}

inline bool cmp(const int &a, const int &b){
  if(in[a]!=in[b]) return in[a]<in[b];
  return false;
}

void print(){
  for(int i=0;i<n;i++) { for(int j=sfx[i];j<n;j++) printf("%c", in[j]); printf("\n"); }
}

void suffixArray() {
  int i;
  for(i = 0; i < n; i++) tmp[i] = i;
  sort(tmp, tmp + n, cmp);
  stp = 0;
  update();
  ++stp;
  for(sfxMv = 1; sfxMv < n; sfxMv <<= 1) {
    Sort();
    stp++;
  }
```

```
    stp--;
    for(i = 0; i <= stp; i++) Rank[i][n] = -1;
}

void kasai() {
    for(int i=0;i<n;i++) rnk[ sfx[i] ] = i;
    for(int i = 0, k = 0; i < n; i++, k ? k-- : 0) {
        if(rnk[i] == n - 1) {
            k = 0;
            continue;
        }
        int j = sfx[ rnk[i] + 1 ];
        while(i + k < n && j + k < n && in[i + k] == in[j + k]) k++;
        lcp[ rnk[i] ] = k;
    }
}

int main(){
    scanf("%s",in);
    n=strlen(in);

    suffixArray();
    print();
    kasai();

    for(int i=0;i<n;i++) cout << lcp[i] << '\n';

    return 0;
}
```

Palindromic Tree:

```
const int N=100005;
int tree[N][26],len[N],link[N],idx,t;
char str[N]; // 1-indexed

void extend(int p)
{
    while(str[p-len[t]-1]!=str[p]) t=link[t];
    int x=link[t];
    while(str[p-len[x]-1]!=str[p]) x=link[x];
    int c=str[p]-'a';
    if(!tree[t][c])
    {
        tree[t][c]=++idx;
        len[idx]=len[t]+2;
        link[idx]=len[idx]==1?2:tree[x][c];
    }
    t=tree[t][c];
}

void build()
{
    len[1]=-1,link[1]=1;
    len[2]=0,link[2]=1;
    idx=t=2;
    int l=strlen(str+1);
    for(int i=1; i<=l; i++) extend(i);
}

int main(){
    scanf("%s",str+1);
    build();
}
```

Fast Fourier Transform:

```
struct complx{
    long double real, img;

    inline complx(){
        real = img = 0.0;
    }

    inline complx(long double x){
        real = x, img = 0.0;
    }

    inline complx(long double x, long double y){
        real = x, img = y;
    }

    inline void operator += (complx &other){
        real += other.real, img += other.img;
    }

    inline void operator -= (complx &other){
        real -= other.real, img -= other.img;
    }

    inline complx operator + (complx &other){
        return complx(real + other.real, img + other.img);
    }

    inline complx operator - (complx &other){
        return complx(real - other.real, img - other.img);
    }

    inline complx operator * (complx& other){
        return complx((real * other.real) - (img * other.img), (real
* other.img) + (img * other.real));
    }
};
;

void FFT(vector <complx> &ar, int n, int inv){
    int i, j, l, len, len2;
    const long double p = 4.0 * inv * acos(0.0);

    for (i = 1, j = 0; i < n; i++){
        for (l = n >> 1; j >= l; l >>= 1) j -= l;
        j += l;
        if (i < j) swap(ar[i], ar[j]);
    }

    for(len = 2; len <= n; len <<= 1) {
        long double ang = 2 * PI / len * inv;
        complx wlen(cos(ang), sin(ang));
        for(i = 0; i < n; i += len) {
            complx w(1);
            for(j = 0; j < len / 2; j++) {
                complx u = ar[i + j];
                complx v = ar[i + j + len / 2] * w;
                ar[i + j] = u + v;
                ar[i + j + len / 2] = u - v;
                w = w * wlen;
            }
        }
    }
    if (inv == -1){
        long double tmp = 1.0 / n;
        for (i = 0; i < n; i++) ar[i].real *= tmp;
    }
}


vector <complx> Mul(const vector <complx> &x, const vector
<complx> &y) {
    int n = 1;
    while(n <= x.size() + y.size()) n = n * 2;
    vector <complx> A(n), B(n);
    REP(i, x.size()) A[i] = x[i];
    REP(i, y.size()) B[i] = y[i];
    FFT(A, n, 1);
    FFT(B, n, 1);
    REP(i, n) A[i] = A[i] * B[i];
    FFT(A, n, -1);
    return A;
}

int main()
{
    int t;
    cin >> t;
    while(t--){
        string a,b;
        cin >> a >> b;
        vector<complx>v1,v2;

        int sign = 0;
        if(a[0] == '-'){
            sign = 1 - sign;
            a.erase(a.begin());
        }
        if(b[0] == '-'){
            sign = 1 - sign;
            b.erase(b.begin());
        }

        for(int i = 0;i < a.size();i++){
            int d = a[i] - '0';
            v1.push_back(complx(d));
        }
        for(int i = 0;i < b.size();i++){
            int d = b[i] - '0';
            v2.push_back(complx(d));
        }

        reverse(all(v1)),reverse(all(v2)); //Reverse needed if v1
is in x^n+x^n-1+.....+x^1+1 form
        vector<complx>v = Mul(v1,v2);

        int carry = 0;
        vector<int>answer;
        for(int i = 0;i < v.size();i++){
            int temp = round(v[i].real);
            temp += carry;
            answer.push_back(temp % 10);
            carry = temp/10;
        }
        while(answer.size() > 1 and answer.back() ==
0)answer.pop_back();
        reverse(all(answer));

        for(int i : answer)cout << i;
        cout << "\n";

    }
}
```

NTT:

```
const LL mod = 163577857;
const LL frd_root = 121532577;
const LL inv_root = 100122727; // inverse of mod
const LL limit = 1 << 22;


#define MAX 1048625

LL wlen_P[MAX >> 1], A[MAX], B[MAX];

void NTT(LL *ar, int n, int inv){
    int i, j, l, len, len2;

    for (i = 1, j = 0; i < n; i++){
        for (l = n >> 1; j >= l; l >>= 1) j -= l;
        j += l;
        if (i < j) swap(ar[i], ar[j]);
    }

    for (len = 2; len <= n; len <<= 1){
        LL w_ml = inv == -1 ? inv_root : frd_root;

        for(i = len; i < limit; i <<= 1) w_ml = w_ml * w_ml % mod;

        for(i = 0; i < n; i += len) {
            LL w = 1;
            for(j = 0; j < len / 2; j++) {
                LL u = ar[i + j];
                LL v = ar[i + j + len / 2] * w % mod;
                ar[i + j] = u + v < mod ? u + v : u + v - mod;
                ar[i + j + len / 2] = u - v >= 0 ? u - v : u - v + mod;
                w = w * w_ml % mod;
            }
        }
    }

    if (inv == -1){
        LL inv_ml = InvMod((LL)n, mod);
        for(i = 0; i < n; i++) ar[i] = ar[i] * inv_ml % mod;
    }
}

char a[N], b[N];
int res[N];
int na, nb;

int main()
{
    int t; scani(t);
    while(t--) {
        int ma, mb; ma = mb = 1;
        na = scans(a);
        nb = scans(b);

        if(a[0] == '-') ma = -1, a[0] = '0';
        if(b[0] == '-') mb = -1, b[0] = '0';

        reverse(a, a + na);
        reverse(b, b + nb);

        for(int i = 0; i < na; i++) A[i] = (a[i] - '0');
        for(int i = 0; i < nb; i++) B[i] = (b[i] - '0');

        int n = 1; while(n < na * 2 || n < nb * 2) n = n << 1;

        for(int i = na; i < n; i++) A[i] = 0;
        for(int i = nb; i < n; i++) B[i] = 0;

        NTT(A, n, 1);
        NTT(B, n, 1);
        for(int i = 0; i < n; i++) A[i] = A[i] * B[i] % mod;
        NTT(A, n, -1);
//      for(int i = 0; i < n; i++) cout << A[i] << ' '; cout << '\n';
        for(int i = 0; i < n; i++) res[i] = A[i];

        for(int i = 0; i < n; i++) {
            res[i + 1] += res[i] / 10;
            res[i] %= 10;
        }
        n = na + nb - 1;
        while(res[n] <= 0 && n > 0) n--;

        if(ma * mb < 0) pc('-');
        for(int i = n; i >= 0; i--) write(res[i], false); pc('\n');
    }
    return 0;
}
```

Walsh Hadamar:

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
//#define bitwiseXOR 1
#define bitwiseAND 2
//#define bitwiseOR 3
const LL MOD = 1000000007;

void FWHT(vector< LL >&p, bool inverse)
{
    int n = p.size();
    assert((n&(n-1))==0);

    for (int len = 1; 2*len <= n; len <<= 1) {
        for (int i = 0; i < n; i += len+len) {
            for (int j = 0; j < len; j++) {
                LL u = p[i+j];
                LL v = p[i+len+j];

                #ifdef bitwiseXOR
                p[i+j] = u+v;
                p[i+len+j] = u-v;
                #endif // bitwiseXOR

                #ifdef bitwiseAND
                if (!inverse) {
                    p[i+j] = v;
                    p[i+len+j] = u+v;
                } else {
                    p[i+j] = -u+v;
                    p[i+len+j] = u;
                }
                #endif // bitwiseAND

                #ifdef bitwiseOR
                if (!inverse) {
                    p[i+j] = u+v;
                    p[i+len+j] = u;
                } else {
                    p[i+j] = v;
                    p[i+len+j] = u-v;
                }
                #endif // bitwiseOR
            }
        }
    }

    #ifdef bitwiseXOR
    if (inverse) {
        for (int i = 0; i < n; i++) {
            assert(p[i]%n==0);
            p[i] /= n;
        }
    }
    #endif // bitwiseXOR
}

LL pw(LL a, LL b){
    if (b==0) return 1;
    LL r = pw(a, b/2);
    r = (r*r)%MOD;
    if (b%2) r = (r*a)%MOD;
    return r;
}

int main(){
    int n;
    cin >> n;

    int sz = 1<<20;
    vector< LL >p(sz, 0);

    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        p[x] = 1;
    }

    p[0] = 1;

    FWHT(p, false);
    for (int i = 0; i < sz; i++) p[i] = pw(p[i], n);
    FWHT(p, true);

    int ans = 0;
    for (int i = 0; i < sz; i++) {
        ans += p[i]!=0;
//      if (p[i]) cout << i << endl;
    }
    cout << ans << endl;
    return 0;
}

poly FWHT(poly P, bool inverse) {
    for (len = 1; 2 * len <= degree(P); len <<= 1) {
        for (i = 0; i < degree(P); i += 2 * len) {
            for (j = 0; j < len; j++) {
                u = P[i + j];
                v = P[i + len + j];
                P[i + j] = u + v;
                P[i + len + j] = u - v;
            }
        }
    }

    if (inverse) {
        for (i = 0; i < degree(P); i++)
            P[i] = P[i] / degree(P);
    }

    return P;
}
```

Mobius:

```
const int N=1000001;
int mu[N];
void mobius()
{
    MEM(mu,-1);
    mu[1]=1;
    for(int i = 2; i<N; i++)
    {
        if(mu[i])
        {
            for(int j = i+i; j<N; j += i)
                mu[j] -= mu[i];
        }
    }
    return;
}
```

Gobius Function:

```
int mobius[N], gobius[N], isP[N];
int prime_size, prime[N];

void pre() {
    mobius[1] = 1;
    isP[0] = isP[1] = 1;

    for(int i = 2; i < N; i++) {
        if(!isP[i]) {
            prime[prime_size++] = i;
            mobius[i] = -1;
        }
        for(int j = 0; j < prime_size && i * prime[j] < N; j++) {
            isP[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                mobius[i * prime[j]] = 0;
                break;
            }
            mobius[i * prime[j]] = -mobius[i];
        }
    }

    for(int i = 1; i < N; i++) if(!isP[i]) for(int j = i; j < N; j += i)
gobius[j] += mobius[j / i];
    for(int i = 1; i < N; i++) gobius[i] += gobius[i - 1];
}
```

Modular Inverse Using Extended Euclid:

```
int egcd(int a,int b, int &x,int &y)
{
    if(a==0){
        x = 0;
        y = 1;
        return b;
    }
    int x1,y1;
    int d = egcd(b%a,a,x1,y1);
    x = y1 -(b/a)*x1;
    y = x1;
    return d;
}

int main()
{
    int x,y;
    int g = egcd(7,10,x,y); /// we will get modular inverse of 7
with mod 10
                        /// If x < 0 , x += 10;
    cout << g << " " << x << " " << y << endl;
}
```

AxBy Solution:

```
long long solve(long long a, long long b, long long n) {
    if (a > b) {
        swap(a, b);
    }
    if (a == b){
        return n / a;
    }
    long long k = min((b - 1) / (b - a), n / a);
    long long res = ((2 * (a - 1) - (b - a) * (k - 1)) * k) >> 1;
    if (a * (k + 1) > n && k * b < n) {
        res += (n - k * b);
    }
    return n + 1 - res;
}

int main()
{
    long long A, B, N;
    while(cin >> A >> B >> N) {
        cout << solve(A, B, N) << '\n';
    }
    return 0;
}
```

Chinese Remainder Theorem:

```
namespace crt{
   long long extended_gcd(long long a, long long b, long long&
x, long long& y){
      if (!b){
         y = 0, x = 1;
         return a;
      }
      long long g = extended_gcd(b, a % b, y, x);
      y -= ((a / b) * x);
      return g;
   }

   long long mod_inverse(long long a, long long m){
      long long x, y, inv;
      extended_gcd(a, m, x, y);
      inv = (x + m) % m;
      return inv;
   }

   long long chinese_remainder(vector <long long> ar, vector
<long long> mods){
      int i, j;
      long long x, y, res = 0, M = 1;

      for (i = 0; i < ar.size(); i++) M *= mods[i];
      for (i = 0; i < ar.size(); i++){
         x = M / mods[i];
         y = mod_inverse(x, mods[i]);
         res = (res + (((x * ar[i]) % M) * y)) % M;
      }
      return res;
   }
}


namespace bin{
   int dp[MAXP];
   long long mod = 0;

   long long trailing(long long x, long long p){
      long long res = 0;
      while (x){
         x /= p;
         res += x;
      }
      return res;
   }

   long long expo(long long x, long long n, long long m){
      if (!n) return 1;
      else if (n & 1) return ((expo(x, n - 1, m) * x) % m);
      else{
         long long r = expo(x, n >> 1, m);
         return ((r * r) % m);
      }
   }

   long long factorial(long long x, long long p){
      long long res = expo(dp[mod - 1], x / mod, mod);
      if (x >= p) res = res * factorial(x / p, p) % mod;
      return res * dp[x % mod] % mod;
   }

   long long binomial(long long n, long long k, long long p, long
long q){
      if (k > n) return 0;
      if (n == k || k == 0) return 1;

      int i, j;
      for (i = 0, mod = 1; i < q; i++) mod *= p;
      long long t = trailing(n, p) - trailing(k, p) - trailing(n - k, p);
      if (t >= q) return 0;

      assert(mod < MAXP);
      for (dp[0] = 1, i = 1; i < mod; i++){
         dp[i] = (long long)dp[i - 1] * ((i % p) ? i : 1) % mod;
      }

      long long res = factorial(n, p) * expo(factorial(k, p) *
factorial(n - k, p) % mod, (mod / p) * (p - 1) - 1, mod) % mod;
      while (t--) res = res * p % mod;
      return res;
   }

   long long binomial(long long n, long long k, long long m){
      if (k > n || m == 1) return 0;
      if (n == k || k == 0) return 1;

      vector <pair<int, int>> factors;
      for (long long i = 2; i * i <= m; i++){
         int c = 0;
         while (m % i == 0){
            c++;
            m /= i;
         }
         if (c) factors.push_back(make_pair(i, c));
      }
      if (m > 1) factors.push_back(make_pair(m, 1));

      vector <long long> ar, mods;
      for (int i = 0; i < factors.size(); i++){
         long long x = 1;
         for (int j = 0; j < factors[i].second; j++) x *=
factors[i].first;
         mods.push_back(x), ar.push_back(binomial(n, k,
factors[i].first, factors[i].second));
      }
      return crt::chinese_remainder(ar, mods);
   }
}

const long long MOD = 142857; // MOD can be non prime

int main(){
   int t, n, k;

   scanf("%d", &t);
   while (t--){
      scanf("%d %d", &n, &k);
      printf("%lld\n", bin::binomial(n, k, MOD));
   }
   return 0;
}
```

Milar Robin:

```
//Complexity O(k log^3 n)

#include <bits/stdc++.h>
using namespace std;
#define LL long long

LL ModularMultiplication(LL a, LL b, LL m)
{
   LL ret=0, c=a;
   while(b)
   {
      if(b&1) ret=(ret+c)%m;
      b>>=1;
      c=(c+c)%m;
   }
   return ret;
}

LL ModularExponentiation(LL a, LL n, LL m)
{
   LL ret=1, c=a;
   while(n)
   {
      if(n&1) ret=ModularMultiplication(ret, c, m);
      n>>=1;
      c=ModularMultiplication(c, c, m);
   }
   return ret;
}

bool Witness(LL a, LL n)
{
   LL u=n-1;
   int t=0;
   while(!(u&1))
   {
      u>>=1;
      t++;
   }

   LL x0=ModularExponentiation(a, u, n), x1;
   for(int i=1; i<=t; i++)
   {
      x1=ModularMultiplication(x0, x0, n);
      if(x1==1 && x0!=1 && x0!=n-1) return true;
      x0=x1;
   }
   if(x0!=1) return true;
   return false;
}

LL Random(LL n)
{
   LL ret=rand();
   ret*=32768;
   ret+=rand();
   ret*=32768;
   ret+=rand();
   ret*=32768;
   ret+=rand();
   return ret%n;
}

bool IsPrimeFast(LL n, int TRIAL)
{
   if(n == 1) return false;
   if(n == 2) return true;
   while(TRIAL--)
   {
      LL a=Random(n-2)+1;
      if(Witness(a, n)) return false;
   }
   return true;
}

LL SQRT(LL n)
{
   LL lo = 0,hi = 1e9,mid,ans;
   while(lo <= hi)
   {
      mid = (lo + hi)/2;
      if(mid * mid <= n)
      {
         lo = mid + 1;
         ans = mid;
      }
      else
      {
         hi = mid - 1;
      }
   }
   return ans;
}
int main()
{
   srand(time(NULL));
   LL n;
   cin >> n;
   LL ret = 1;
   for(int i = 2; i <= 2e6; i++)
   {
      LL cnt = 0;
      while(n % i == 0)
      {
         cnt++;
         n/=i;
      }
      if(cnt > 0) ret = ret * (cnt + 1);
   }

   if(n == 1);
   else if(IsPrimeFast(n,1))
   {
      ret = ret * 2;
   }
   else
   {
      LL sq = SQRT(n);
      if(sq * sq == n) ret = ret * 3;
      else ret = ret * 2 * 2;
   }
   cout << ret << "\n";
}
```

Matrix Expo:

```
ll mod;
const ll N=6;

void MatMul(ll A[N][N], ll B[N][N])
{
   ll R[N][N];
   MEM(R,0);
   REP(i, N) REP(j, N) REP(k, N) R[i][j] = (R[i][j]%mod +
(A[i][k] * B[k][j])%mod)%mod;
   REP(i, N) REP(j, N) B[i][j] = R[i][j];
   return;
}


void MatPow(ll R[N][N],ll M[N][N],ll P)
{
   while(P)
   {
      if(P & 1)
         MatMul(M,R);
      MatMul(M,M);
      P = P >> 1;
   }
}

int main()
{

   ll n,M[N][N],R[N][N]; // M is Co-efficient Matrix,R is Base
case Matrix
   //Take input values of M and R matrix
   //Input n,We have to find f(n)
   MatPow(R,M,n-2); // Here n-2 may changes in diffrent
problems
   //value of f(n) is in R[0][0] position
   return 0;
}
```

Convex Hull:

```
struct point
{
   ll x,y;
   bool operator < (const point &p) const
   {
      return x<p.x || (x==p.x && y<p.y);
   }
} P[MAX],C[MAX];

inline ll Cross(point &o,point &a,point &b)
{
   return (a.x-o.x)*(b.y-o.y)-(a.y-o.y)*(b.x-o.x);
}

void ConvexHull(int np,int &nc)
{
   sort(P,P+np);
   REP(i,np)
   {
      while(nc>=2 and Cross(C[nc-2],C[nc-1],P[i])<=0)
         nc--;
      C[nc++]=P[i];
   }
   int t=nc+1;
   ROF(i,np-1,1)
   {
      while(nc>=t and Cross(C[nc-2],C[nc-1],P[i-1])<=0)
         nc--;
      C[nc++]=P[i-1];
   }
   nc--;
   return;
}

int main()
{
   int nc=0,np;
   scanf("%d",&np);
   REP(i,np)
   {
      scanf("%lld %lld",&P[i].x,&P[i].y);
   }
   ConvexHull(np,nc);
   REP(i,nc)
   {
      printf("%lld %lld\n",C[i].x,C[i].y);
   }
   return 0;
}
```

Ternary Search:

```
struct point{
   double x,y,z;
   double dis(const point a,const point b,double t)
   {
      point p;
      p.x=a.x+(b.x-a.x)*t;
      p.y=a.y+(b.y-a.y)*t;
      p.z=a.z+(b.z-a.z)*t;
      return SQ(x-p.x)+SQ(y-p.y)+SQ(z-p.z);
   }
}A,B,P;

double ternary(){
   double l=0.0,h=1.0;
   int s=49;
   while(s--)
   {
      double t1=(2.0*l+h)/3.0;
      double t2=(l+2.0*h)/3.0;
      double d1=P.dis(A,B,t1);
      double d2=P.dis(A,B,t2);
      if(d1<d2) h=t2;
      else l=t1;
   }
   double d=P.dis(A,B,l);
   return sqrt(d);
}
```

Ordered Set:

```
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
template <typename T> using orderset = tree <T, null_type,
less<T>, rb_tree_tag,tree_order_statistics_node_update>;
 // find_by_order, order_of_key
```

Himel Templete:

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

#define MAX           100005
#define MOD           1000000007
#define eps           1e-6
int fx[] =            {1,-1,0,0};
int fy[] =            {0,0,1,-1};

#define FastRead       ios_base::sync_with_stdio(0);cin.tie(0);
#define fRead          freopen("in.txt","r",stdin);
#define fWrite         freopen ("out.txt","w",stdout);

#define ll            long long
#define ull            unsigned long long
#define ff            first
#define ss             second
#define pb             push_back
#define PI            acos(-1.0)
#define mk             make_pair
#define pii           pair<int,int>
#define pll           pair<ll,ll>
#define all(a)         a.begin(),a.end()

#define min3(a,b,c)     min(a,min(b,c))
#define max3(a,b,c)     max(a,max(b,c))
#define min4(a,b,c,d)   min(a,min(b,min(c,d)))
#define max4(a,b,c,d)   max(a,max(b,max(c,d)))

#define FOR(i,a,b)      for(int i=a;i<=b;i++)
#define ROF(i,a,b)      for(int i=a;i>=b;i--)
#define REP(i,b)        for(int i=0;i<b;i++)
#define IT(it,x)       for(it=x.begin();it!=x.end();it++)

#define MEM(a,x)        memset(a,x,sizeof(a))
#define TC             int t;cin >> t;FOR(tc,1,t)
#define ABS(x)          ((x)<0?-(x):(x))
#define SQ(x)          ((x)*(x))
#define SP(x)          fixed << setprecision(x)


#define Make(x,p)       (x | (1<<p))
#define DeMake(x,p)     (x & ~(1<<p))
#define Check(x,p)      (x & (1<<p))
#define popcount(x)     __builtin_popcount(x)
```