Team notebook

Islamic University of Technology

June 3, 2022

Contents

1 Data Structure

1.1 Binary Indexed Tree 2D

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

int mx = 100, my = 100;
BIT[mx][my];

void update(int x, int y, int val)
{
    int y1;
    while (x <= mx)
    {
        y1 = y;
        while (y1 <= my)
        {
            BIT[x][y1] += val;
            y1 += (y1 & -y1);
        }
        x += (x & -x);
    }
}
int query(int x, int y)
{
    int y1, sum = 0;
    while(x)
    {
        y1 = y;
        while(y1)
        {
            sum += BIT[x][y1];
            y1 -= y1&(-y1);
        }
        x -= x&(-x);
    }
    return sum;
}
```

1.2 Binary Indexed Tree

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

int BIT[100005], a[100005], n;

int query(int i)
{
    int sum=0;
    while(i>0)
    {
        sum += BIT[i];
        i -= i & (-i);
    }
    return sum;
}
void update(int i,int d)
{
    while(i<=n)
    {
        BIT[i]+=d;
        i += i & (-i);
    }
}

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

    for(i = 1; i <= n; i++)
    {
        cin >> a[i];
        update(i, a[i]);
    }

    cout << "Sum of First 10 elements: " <<
         query(10) << "\n";
    cout << "Sum of elements in [2, 7]: " <<
         query(7) - query(1) << "\n";
}
```

1.3 DSU

```
int find_set(int v) {
    if (v == parent[v])
        return v;
    return parent[v] = find_set(parent[v]);
         //collapse
}

void union_sets(int a, int b) {
    a = find_set(a);
    b = find_set(b);
    if (a != b) //compare rank, size, or toss with
         rand()
        parent[b] = a;
}

void make_set(int v) {
    parent[v] = v;
    rank[v] = 0;
}
void union_sets(int a, int b) {
    a = find_set(a);
    b = find_set(b);
    if (a != b) {
        if (rank[a] < rank[b])
            swap(a, b);
        parent[b] = a;
        if (rank[a] == rank[b])
            rank[a]++;
    }
}

void union_sets(int a, int b) {
    a = find_set(a);
    b = find_set(b);
    if (a != b) {
        if (size[a] < size[b])
            swap(a, b);
        parent[b] = a;
        size[a] += size[b];
    }
}
```

1.4 HLD

```
/**
 * Heavy Light Decomposition
 * Build Complexity O(n)
 * Query Complexity O(lg^2 n)
 * Call init() with number of nodes
 * It's probably for the best to not do "using
     namespace hld"
**/
namespace hld {
        /**
         * N is the maximum number of nodes
         * g is the adjacency graph
         * par, lev, size corresponds to parent,
              depth, subtree-size
         * head[u] is the starting node of the chain
              u is in
         * in[u] to out[u] keeps the subtree indices
        **/
        const int N = 100000+7;
        vector<int> g[N];
        int par[N], lev[N], head[N], size[N], in[N],
             out[N];
        int cur_pos, n;

        /**
         * returns the size of subtree rooted at u
         * maintains the child with the largest
              subtree at the front of g[u]
         * WARNING: Don't change anything here
              specially with size[] if Jon Snow
        **/
        int dfs(int u, int p) {
                size[u] = 1;
                par[u] = p;
                lev[u] = lev[p] + 1;
                for(auto &v : g[u]) {
                        if(v == p) continue;
                        size[u] += dfs(v, u);
                        if(size[v] >
                             size[g[u].front()]) {
                                swap(v, g[u].front());
                        }
                }
                return size[u];
        }

        /**
         * decomposed the tree in an array
         * note that there is no physical array here
        **/
        void decompose(int u, int p) {
                in[u] = ++cur_pos;
                for(auto &v : g[u]) {
                        if(v == p) continue;
                head[v] = (v == g[u].front()
                     ? head[u] : v);
                decompose(v, u);
        }
        out[u] = cur_pos;
}

/**
 * initializes the structure with _n nodes
**/
void init(int _n, int root = 1) {
        n = _n;
        cur_pos = 0;
        dfs(root, 0);
        head[root] = root;
        decompose(root, 0);
}

/**
 * checks whether p is an ancestor of u
**/
bool isances(int p, int u) {
        return in[p] <= in[u] and out[u] <=
             out[p];
}

/**
 * Returns the maximum node value in the
      path u - v
**/
ll query(int u, int v) {
        ll ret = -INF;
        while(!isances(head[u], v)) {
                ret = max(ret, seg.query(1,
                     1, n, in[head[u]],
                     in[u]));
                u = par[head[u]];
        }
        swap(u, v);
        while(!isances(head[u], v)) {
                ret = max(ret, seg.query(1,
                     1, n, in[head[u]],
                     in[u]));
                u = par[head[u]];
        }
        if(in[v] < in[u]) swap(u, v);
        ret = max(ret, seg.query(1, 1, n,
             in[u], in[v]));
        return ret;
}

/**
 * Adds val to subtree of u
**/
void update(int u, ll val) {
                seg.update(1, 1, n, in[u], out[u],
                     val);
        }
};
```

1.5 LCA

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

int n;
vector<int> adj[400009];
int parent[400009], level[400009], anc[400009][21];

void dfs(int node, int pr, int l)
{
    parent[node] = pr;
    level[node] = l;

    for(auto u : adj[node])
    {
        if(u != parent[node])
            dfs(u, node, l+1);
    }
}

void lca_init()
{
    dfs(1, 1, 0);

    int i, j;
    for(i = 1; i <= n; i++)
    {
        anc[i][0] = parent[i];

        for(j = 1; j <= 20; j++)
            anc[i][j] = 1;
    }

    for(j = 1; (1<<j) <= n; j++)
    {
        for(i = 1; i <= n; i++)
            anc[i][j] = anc[anc[i][j-1]][j-1];
    }
}

int lca(int u, int v)
{
    if(level[u] < level[v])
        swap(u, v);

    int i;
    for(i = log2(n) + 1; i >= 0; i--)
    {
        if(level[anc[u][i]] >= level[v])
        {
            u = anc[u][i];
        }
    }

    if(u == v)
        return u;
    for(i = log2(n) + 1; i >= 0; i--)
    {
        if(anc[u][i] != anc[v][i])
        {
            u = anc[u][i];
            v = anc[v][i];
        }
    }

    return parent[u];
}

int main()
{
    int i, u, v;
    cin >> n;

    for(i = 1; i <= n-1; i++)
    {
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    lca();
    int q;
    cin >> q;
    while(q--)
    {
        cin >> u >> v;
        cout << query(u, v) << endl;
    }
}
```

1.6 Segment tree (Lazy)

```
#include<bits/stdc++.h>
using namespace std;
#define n 100000
int tree[4*n], a[n], lazy[4*n];

void build(int node, int s, int e)
```

```
{
    if(s == e)
    {
        tree[node] = a[s];
        return;
    }

    int mid = (s+e)/2;
    build(2*node, s, mid);
    build(2*node + 1, mid+1, e);

    tree[node] = tree[2*node] + tree[2*node + 1];
}

void updateRange(int node, int s, int e, int l, int r, int val)
{
    if(lazy[node] != 0)
    {
        tree[node] += (e - s + 1) * lazy[node];
        if(s != e)
        {
            lazy[2*node] += lazy[node];
            lazy[2*node + 1] += lazy[node];
        }
        lazy[node] = 0;
    }

    if(s > r || e < l)
        return;

    if(s >= l && e <= r)
    {
        tree[node] += (e - s + 1) * val;
        if(s != e)
        {
            lazy[2*node] += val;
            lazy[2*node + 1] += val;
        }
        return;
    }
    int mid = (s + e) / 2;
    updateRange(2*node, s, mid, l, r, val);
    updateRange(2*node + 1, mid + 1, e, l, r, val);
    tree[node] = tree[2*node] + tree[2*node + 1];
}

int queryRange(int node, int s, int e, int l, int r)
{
    if(lazy[node] != 0)
    {
        tree[node] += (e - s + 1) * lazy[node];
        if(s != e)
        {
            lazy[2*node] += lazy[node];
            lazy[2*node + 1] += lazy[node];
        }
        lazy[node] = 0;
    }

    if(s > r || e < l)
        return 0;

    if(s >= l and e <= r)
        return tree[node];
    int mid = (s + e) / 2;
    int p1 = queryRange(2*node, s, mid, l, r);
    int p2 = queryRange(2*node + 1, mid + 1, e, l, r);
    return (p1 + p2);
}
```

1.7 Segment tree

```
#include<bits/stdc++.h>
using namespace std;
#define sz 100005

int n, k;
int x[sz];
int Tree[4*sz];

void build(int node, int s, int e)
{
    if(s == e)
    {
        Tree[node] = x[s];
        return;
    }

    int mid = (s+e)/2, left = 2*node+1, right = 2*node+2;

    build(left, s, mid);
    build(right, mid+1, e);

    Tree[node] = Tree[left] * Tree[right];
}

void update(int node, int s, int e, int idx, int val)
{
    if(s == e)
    {
        Tree[node] = val;
        x[idx] = val;
        return;
    }

    int mid = (s+e)/2, left = 2*node+1, right = 2*node+2;

    if(idx <= mid)
        update(left, s, mid, idx, val);
    else
        update(right, mid+1, e, idx, val);

    Tree[node] = Tree[left] * Tree[right];
}

int query(int node, int s, int e, int l, int r)
{
    if(l > e || r < s)
        return 1;

    if(l <= s && r >= e)
        return Tree[node];

    int mid = (s+e)/2, left = 2*node+1, right = 2*node+2;
    int p1 = query(left, s, mid, l, r);
    int p2 = query(right, mid+1, e, l, r);

    return p1*p2;
}
```

1.8 Sparse Table RMQ

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

ll a[100005], sparse[100005][20], Log[100005];

ll findMin(ll st, ll ed)
{
    k = Log[ed-st+1];
    return min(sparse[st][k], sparse[ed - (1 << k) + 1][k]);
}

int main()
{
    Log[1] = 0;
    for(i = 2; i <= 100005; i++)
        Log[i] = Log[i>>1] + 1;

    ll n, i, j;
    for(i = 0; i <= n; i++)
    {
        cin >> a[i];
        sparse[i][0] = a[i];
    }

    for(j = 1; j < 20; j++)
    {
        for(i = 0; i + (1LL << j) - 1 < n; i++)
            sparse[i][j] = min(sparse[i][j-1],
                 sparse[i + (1LL << (j-1))][j-1]);
    }

    ll q, k;
    cin >> q;

    while(q--)
    {
        cin >> i >> j;

        cout << findMin(i, j) << endl;
    }
}
```

1.9 Square Root Decomposition

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

int main()
{
    int n;
    vector<int> a (n);

    int i, len = (int) sqrt (n + .0) + 1;
    vector<int> b (len);
    for (i = 0; i < n; i++)
        b[i/len] += a[i];

    while(1)
    {
        int l, r;

        int sum = 0;
        for (i = l; i <= r;)
        {
            if (i % len == 0 && i + len - 1 <= r)
            {
                sum += b[i / len];
                i += len;
            }
            else
            {
                sum += a[i];
                i++;
            }
        }
    }
}
```

2 DP

2.1 0-1 knapsack iterative

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n, m, i, j;
    cin >> n >> m;
    int w[n+1], p[n+1], dp[n+1][m];

    for(i = 1; i <= n; i++)
        cin >> w[i] >> p[i];

    for(i = 0; i <= m; i++)
        dp[0][i] = 0;

    for(i = 1; i <= n; i++)
    {
        for(j = 0; j <= m; j++)
        {
            if(j < w[i])
                dp[i][j] = dp[i-1][j];
            else
                dp[i][j] = max(dp[i-1][j], p[i] +
                     dp[i-1][j-w[i]]);
        }
    }

    cout << dp[n][m];
}
```

2.2 0-1 knapsack recursive

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

int n, m, w[1009], p[1009], dp[1009][1009];

int call(int i, int j)
{
    if(i == n)
        return 0;
    if(dp[i][j] != -1)
        return dp[i][j];
    if(j + w[i] > m)
        dp[i][j] = call(i+1, j);
    else
        dp[i][j] = max(call(i+1, j), p[i] +
             call(i+1, j+w[i]));
    return dp[i][j];
}

int main()
{
    int i, j;

    memset(dp, -1, sizeof dp);

    cin >> n >> m;
    for(i = 0; i < n; i++)
        cin >> w[i] >> p[i];
    cout << call(0, 0);
}
```

2.3 Coin change

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n, m, i, j;

    scanf("%d %d", &n, &m);

    int dp[n+10], c[m];

    for(i = 0; i <= n + 5; i++)
        dp[i] = INT_MAX;

    for(i = 0; i < m; i++)
        scanf("%d", &c[i]);

    dp[0] = 0;

    for(i = 1; i <= n; i++)
    {
        for(j = 0; j < m; j++)
        {
            if(i - c[j] >= 0)
                dp[i] = min(dp[i], dp[i - c[j]] + 1);
        }
    }
    if(dp[n] != INT_MAX)
        printf("%d", dp[n]);
    else
        printf("%d is not possible", n);
}
```

2.4 Edit Distance

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int i, j, n, m;
    string x, y;

    cin >> x >> y;

    n = x.size(), m = y.size();
    int d[n+1][m+1] = {0};

    for(i = 1; i <= n; i++)
        d[i][0] = i;
    for(i = 1; i <= m; i++)
        d[0][i] = i;

    for(i = 1; i <= n; i++)
    {
        for(j = 1; j <= m; j++)
        {
            d[i][j] = min(d[i-1][j] + 1, d[i][j-1] +
                 1);
            if(x[i] == y[j])
                d[i][j] = min(d[i][j], d[i-1][j-1]);
            else
                d[i][j] = min(d[i][j], d[i-1][j-1] +
                     2);
        }
    }

    cout << d[n][m];
}
```

2.5 Longest Common Subsequence iterative

```
#include<bits/stdc++.h>
using namespace std;
int lcs[3009][3009];

int main()
{
    string a, b;
    int i, j, n, m;
    cin >> a >> b;

    n = a.size();
    m = b.size();

    for(i = 0; i <= n; i++)
        lcs[i][0] = 0;
    for(i = 0; i <= m; i++)
        lcs[0][i] = 0;

    for(i = 1; i <= n; i++)
    {
        for(j = 1; j <= m; j++)
        {
            if(a[i-1] == b[j-1])
                lcs[i][j] = lcs[i-1][j-1] + 1;
            else
                lcs[i][j] = max(lcs[i-1][j],
                     lcs[i][j-1]);
        }
    }

    i = n, j = m;
    string ans;
    while(i > 0 && j > 0)
    {
        if(a[i-1] == b[j-1])
        {
            ans = a[i-1] + ans;
            i--;
            j--;
        }
        else if(lcs[i-1][j] >= lcs[i][j-1])
            i--;
        else
            j--;
    }

    cout << lcs[n][m] << endl << ans;
}
```

2.6 Longest Common Subsequence recursive

```
#include<bits/stdc++.h>
using namespace std;
int lcs[3009][3009];
string a, b, ans;

int call(int i, int j)
{
    if(i == (int)a.size() || j == (int)b.size())
        return 0;
    if(lcs[i][j] != -1)
        return lcs[i][j];

    if(a[i] == b[j])
        return lcs[i][j] = 1 + call(i+1, j+1);
    else
        return lcs[i][j] = max(call(i+1, j), call(i,
             j+1));
}

string call2(int i, int j)
{
    if(i == (int)a.size() || j == (int)b.size())
        return "";
    if(a[i] == b[j])
        return a[i] + call2(i+1, j+1);
    else if(lcs[i+1][j] >= lcs[i][j+1])
        return call2(i+1, j);
    else
        return call2(i, j+1);
}

int main()
{
    cin >> a >> b;
    memset(lcs, -1, sizeof lcs);

    cout << call(0, 0) << endl << call2(0, 0);
}
```

2.7 Longest Increasing Subsequence

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

int a[100], p[100], n, last;

int lis()
{
    int i, j;

    for(i = 0; i < n; i++)
        dp[i] = 1;
    for(i = 1; i < n; i++)
    {
        for(j = 0; j < i; j++)
        {
            if(a[j] <= a[i] && dp[i] < dp[j] + 1)
            {
                dp[i] = dp[j];
                p[i] = j;
            }
        }
    }

    int ret = 0, last = -1;
    for(i = 0; i < n; i++)
    {
        if(ret < dp[i])
        {
            ret = dp[i];
            last = i;
        }
    }

    return ret;
}

int main()
{
    cin >> n;

    for(int i = 0; i < n; i++)
        cin >> a[i];

    cout << lis();
}
```

2.8 Longest Palidromic Subsequence Iterative

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

int lps[3009][3009];

int main()
{
    string a, ans;
    int i, j, n;
    cin >> a;
    n = a.size();

    for(i = 1; i <= n; i++)
        lps[i][i] = 1;
    for(i = 2; i <= n; i++)
    {
        for(j = 1; j <= n - i + 1; j++)
        {
            if(a[j-1] == a[j+i-2])
                lps[j][j+i-1] = lps[j+1][j+i-2] + 2;
            else
                lps[j][j+i-1] = max(lps[j+1][j+i-1],
                     lps[j][j+i-2]);
        }
    }


    for(i = 0; i <= n; i++)
    {
        for(j = 0; j <= n; j++)
            cout << lps[i][j] << " ";
        cout << "\n";
    }
}
```

2.9 Longest Palidromic Subsequence Recursive

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

string s;
int dp[1009][1009];

int lps(int i, int j)
{
    if(i > j)
        return 0;
    if(dp[i][j] != -1)
        return dp[i][j];

    int ret = 0;
    if(s[i] == s[j])
    {
        if(i == j)
            ret = 1;
        else
            ret = 2 + lps(i+1, j-1);
    }

    ret = max(ret, lps(i+1, j));
    ret = max(ret, lps(i, j-1));

    return dp[i][j] = ret;
}

int main()
{
    cin >> s;

    memset(dp, -1, sizeof dp);

    cout << lps(0, (s.size() - 1)) << "\n";
}
```

2.10 Matrix Chain Multiplication iterative

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

int dims[1000], n, mcm[1000][1000];

int main()
{
    int i, j, k;
    cin >> n;
    for(i = 0; i < n; i++)
        cin >> dims[i];

    for (i = 1; i <= n; i++)
                mcm[i][i] = 0;


    int len, cost;
        for (len = 2; len <= n; len++)
        {
                for (i = 1; i + len - 1 < n; i++)
                {
                        j = i + len - 1;
                        mcm[i][j] = INT_MAX;

                        for (k = i; j < n && k <= j -
                             1; k++)
                        {
                                cost = mcm[i][k] +
                                     mcm[k+1][j] +
                                     dims[i-1]*dims[k]*dims[j];
                mcm[i][j] = min(mcm[i][j], cost);
                        }
                }
        }

        cout << mcm[1][n-1];
}
```

2.11 Matrix Chain Multiplication recursive

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

int dims[1000], n, mcm[1000][1000];

int call(int i, int j)
{
    if(i+1 >= j)
        return 0;

    if(mcm[i][j] != 0)
        return mcm[i][j];

    int ret = INT_MAX;

    for(int k = i+1; k <= j-1; k++)
    {
        int cost = call(i, k);
        cost += dims[i]*dims[k]*dims[j];
        cost += call(k, j);

        ret = min(ret, cost);
    }

    return mcm[i][j] = ret;
}

int main()
{
    int i;
    cin >> n;
    for(i = 0; i < n; i++)
        cin >> dims[i];
    cout << call(0, n-1);
}
```

2.12 Subset Sum iterative

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

int main()
{
    int i, j, n, s;
    cin >> n >> s;
    int a[n+1];
    a[0] = 0;
    for(i = 1; i <= n; i++)
        cin >> a[i];
    bool dp[n+1][s+1];

    for(i = 1; i <= s; i++)
        dp[0][i] = 0;
    for(i = 0; i <= n; i++)
        dp[i][0] = 1;

    for(i = 1; i <= n; i++)
    {
        for(j = 1; j <= s; j++)
        {
            dp[i][j] = dp[i-1][j];
            if(j >= a[i])
                dp[i][j] = dp[i][j] | dp[i-1][j -
                    a[i]];
        }
    }

    for(i = 0; i <= n; i++)
    {
        for(j = 0; j <= s; j++)
        {
            cout << dp[i][j] << " ";
        }
        cout << endl;
    }
}
```

2.13 Subset Sum recursive

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

bool dp[109][100009], vis[109][100009];
int n, s, a[109];

bool call(int i, int j)
{
    if(vis[i][j])
        return dp[i][j];

    vis[i][j] = 1;

    if(i == 0 && a[i] == j || j == 0)
        return dp[i][j] = 1;
    else if(i == 0)
        return dp[i][j] = 0;

    if(j >= a[i])
        return dp[i][j] = call(i-1, j) | call(i-1,
            j-a[i]);
    else
        return dp[i][j] = call(i-1, j);
}

int main()
{
    int i, j;
    cin >> n >> s;
    for(i = 0; i < n; i++)
        cin >> a[i];

    cout << call(n-1, s) << endl;

    for(i = 0; i < n; i++)
    {
        for(j = 0; j <= s; j++)
            cout << vis[i][j] << "," << dp[i][j] <<
                " ";
        cout << endl;
    }
}
```

3 Flow or Matching

3.1 Edmonds Karp

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define pb push_back
#define ff first
#define ss second

ll n;
ll cap[109][109], par[109];
vector<ll> adj[109];

ll bfs(ll s, ll t)
{
    memset(par, -1, sizeof par);
    par[s] = -2;

    queue<pair<ll, ll>> q;
    q.push({s, INT_MAX});

    while(!q.empty())
    {
        ll cur = q.front().ff;
        ll flow = q.front().ss;
        q.pop();

        for(auto next : adj[cur])
        {
            if(par[next] == -1 && cap[cur][next])
            {
                par[next] = cur;
                ll new_flow = min(flow,
                     cap[cur][next]);
                if (next == t)
                    return new_flow;
                q.push({next, new_flow});
            }
        }
    }
    return 0;
}

ll maxflow(ll s, ll t)
{
    ll flow = 0, new_flow;

    while(new_flow = bfs(s, t))
    {
        flow += new_flow;

        ll prev, cur;
        for(cur = t; cur != s; cur = prev)
        {
            prev = par[cur];
            cap[prev][cur] -= new_flow;
            cap[cur][prev] += new_flow;
        }
    }

    return flow;
}

int main()
{
    ll T, i, s, t, c, caseno = 0;
    cin >> T;
    while(T--)
    {
        cin >> n >> s >> t >> c;
        memset(cap, 0, sizeof cap);
        for(i = 0; i < n; i++)
            adj[i].clear();

        while(c--)
        {
            ll a, b, w;
            cin >> a >> b >> w;
            cap[a][b] += w;
            cap[b][a] += w;
            adj[a].pb(b);
            adj[b].pb(a);
        }
        cout << "Case " << ++caseno << ": " <<
             maxflow(s, t) << endl;
    }
}
```

3.2 Hopcroft Karp

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

const int N = 3e5 + 9;

struct HopcroftKarp
{
    static const int inf = 1e9;
    int n;
    vector<int> l, r, d;
    vector<vector<int>> g;
    HopcroftKarp(int _n, int _m)
    {
        n = _n;
        int p = _n + _m + 1;
        g.resize(p);
        l.resize(p, 0);
        r.resize(p, 0);
        d.resize(p, 0);
    }
    void add_edge(int u, int v)
    {
        g[u].push_back(v + n); //right id is
             increased by n, so is l[u]
    }
    bool bfs()
    {
        queue<int> q;
        for (int u = 1; u <= n; u++)
        {
            if (!l[u]) d[u] = 0, q.push(u);
            else d[u] = inf;
        }
        d[0] = inf;
        while (!q.empty())
        {
            int u = q.front();
            q.pop();
            for (auto v : g[u])
            {
                if (d[r[v]] == inf)
                {
                    d[r[v]] = d[u] + 1;
                    q.push(r[v]);
                }
            }
        }
        return d[0] != inf;
    }
    bool dfs(int u)
    {
        if (!u) return true;
        for (auto v : g[u])
        {
            if(d[r[v]] == d[u] + 1 && dfs(r[v]))
            {
                l[u] = v;
                r[v] = u;
                return true;
            }
        }
        d[u] = inf;
        return false;
    }
    int maximum_matching()
    {
        int ans = 0;
        while (bfs())
        {
            for(int u = 1; u <= n; u++) if (!l[u] &&
                 dfs(u)) ans++;
        }
        return ans;
    }
};
int32_t main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int n, m, q;
    cin >> n >> m >> q;
    HopcroftKarp M(n, m);
    while (q--)
    {
        int u, v;
        cin >> u >> v;
        M.add_edge(u, v);
    }
    cout << M.maximum_matching() << '\n';
    return 0;
}
```

3.3 Kuhn

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

```
int n, m;//left group size n. right group size n.
vector<int> adj[102];
vector<int> mt;
bool[102] vis;

bool try_kuhn(int v)
{
    if (vis[v])
        return false;
    vis[v] = true;
    for (int to : adj[v])
    {
        if (mt[to] == -1 || try_kuhn(mt[to]))
        {
            mt[to] = v;
            return true;
        }
    }
    return false;
}

int main()
{
    int i, u, v, edges;

    cin >> n >> m >> edges;

    for(i = 1; i <= n; i++)
        adj[i].clear();

    for(i = 0; i < edges; i++)
    {
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    mt.assign(m, -1);
    for(i = 1; i <= n; i++)
    {
        memset(vis, 0, sizeof vis);
        try_kuhn(i);
    }

    for(i = 1; i <= m; i++)
    {
        if(mt[i] != -1)
            cout << mt[i] << " " << i;
    }
}
```

4 Geometry

4.1 0D Geo

```
#include<bits/stdc++.h>
using namespace std;
#define EPS 1e-9
#define PI 2*acos(0.0)

struct point
{
    double x, y;
    point() { x = y = 0.0; }
    point(double _x, double _y) : x(_x), y(_y) {}

    bool operator == (point other) const
    {
        return abs(x - other.x) < EPS && abs(y -
             other.y) < EPS;
    }

    bool operator < (point other) const
    {
        if(abs(x - other.x) > EPS)
            return x < other.x;
        return y < other.y;
    }

    double dist(point p1, point p2)
    {
        return sqrt((p1.x - p2.x)*(p1.x - p2.x) +
             (p1.y - p2.y)*(p1.y - p2.y));
    }

    //rotate point p by theta degrees CCW w.r.t
         origin (0, 0)
    point Rotate(point p, double theta)
    {
         double rad = theta * PI / 180;
         return point(p.x*cos(rad) - p.y*sin(rad),
                      p.x*sin(rad) + p.y*cos(rad));
    }

};
```

4.2 2D Geo

```
#include<bits/stdc++.h>
using namespace std;
#define EPS 1e-9
#define PI 2*acos(0.0)

struct point
{
    double x, y;
    point() { x = y = 0.0; }
    point(double _x, double _y) : x(_x), y(_y) {}

    bool operator == (point other) const
    {
        return abs(x - other.x) < EPS && abs(y -
             other.y) < EPS;
    }

    bool operator < (point other) const
    {
        if(abs(x - other.x) > EPS)
            return x < other.x;
        return y < other.y;
    }

    double dist(point p1, point p2)
    {
        return sqrt((p1.x - p2.x)*(p1.x - p2.x) +
             (p1.y - p2.y)*(p1.y - p2.y));
    }

    //rotate p by theta degrees CCW w.r.t origin
         (0, 0)
    point Rotate(point p, double theta)
    {
         double rad = theta * PI / 180;
         return point(p.x*cos(rad) - p.y*sin(rad),
                      p.x*sin(rad) + p.y*cos(rad));
    }
};

struct line
{
    //ax + by = c
    double a, b, c;

    //the answer is stored in third parameter (pass
         by reference)
    void pointsToLine(point p1, point p2, line &l)
    {
        if(abs(p1.x - p2.x) < EPS)
        {
            l.a = 1;
            l.b = 0;
            l.c = -p1.x;
        }
        else
        {
            double delx, dely;
            delx = p2.x - p1.x;
            dely = p2.y - p1.x;

            l.a = -dely / delx;
            l.b = 1; //we fix the value of b to 1.0
            l.c = -(p1.x*dely - p1.y*delx) / delx;
        }
    }

    bool areParallel(line l1, line l2)
    {
        return (abs(l1.a-l2.a) < EPS) &&
             (abs(l1.b-l2.b) < EPS);
    }

    bool areSame(line l1, line l2)
    {
        return areParallel(l1, l2) && (abs(l1.c -
             l2.c) < EPS);
    }
};
```

4.3 Closest pair

```
#include<bits/stdc++.h>
using namespace std;
long long ClosestPair(vector<pair<int, int>> pts)
{
    int n = pts.size();
    sort(pts.begin(), pts.end());
    set<pair<int, int>> s;

    long long best_dist = 1e18;
    int j = 0;
    for (int i = 0; i < n; ++i)
    {
        int d = ceil(sqrt(best_dist));
        while (pts[i].first - pts[j].first >=
             best_dist)
        {
            s.erase({pts[j].second, pts[j].first});
            j += 1;
        }

        auto it1 = s.lower_bound({pts[i].second - d,
             pts[i].first});
        auto it2 = s.upper_bound({pts[i].second + d,
             pts[i].first});

        for (auto it = it1; it != it2; ++it)
        {
            int dx = pts[i].first - it->second;
            int dy = pts[i].second - it->first;
            best_dist = min(best_dist, 1LL * dx * dx
                 + 1LL * dy * dy);
        }
        s.insert({pts[i].second, pts[i].first});
    }
    return best_dist;
}

int main()
{
    vector<pair<int, int> > v;
    v.push_back({0, 2});
    v.push_back({0, 0});

    cout << ClosestPair(v);
}
```

4.4 Convex Hull

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define pii pair<ll, ll>
#define ff first
#define ss second

vector<pii> v;

bool cmp(pii a, pii b)
{
    return a.ff < b.ff || (a.ff == b.ff && a.ss <
         b.ss);
}

bool clockWise(pii a, pii b, pii c)
{
    return
         a.ff*(b.ss-c.ss)+b.ff*(c.ss-a.ss)+c.ff*(a.ss-b.ss)
         <= 0;
    //being !clockWise and being anticlockWise
         aren't same. look at "<="
}

bool anticlockWise(pii a, pii b, pii c)
{
    return
         a.ff*(b.ss-c.ss)+b.ff*(c.ss-a.ss)+c.ff*(a.ss-b.ss)
         >= 0;
    //being !clockWise and being anticlockWise
         aren't same. look at ">="
}

void convex_hull()
{
    if(v.size() == 1)
        return;

    sort(v.begin(), v.end(), cmp);

    pii p1 = v[0], p2 = v.back();

    vector<pii> up, down;
    up.push_back(p1);
    down.push_back(p1);

    for (ll i = 1; i < (ll)v.size(); i++)
    {
        if (i == v.size() - 1 || clockWise(p1, v[i],
             p2))
        {
            while (up.size() >= 2 &&
                 !clockWise(up[up.size()-2],
                 up[up.size()-1], v[i]))
                up.pop_back();
            up.push_back(v[i]);
        }
        if (i == v.size() - 1 || anticlockWise(p1,
             v[i], p2))
        {
            while(down.size() >= 2 &&
                 !antiClockWise(down[down.size()-2],
                 down[down.size()-1], v[i]))
                down.pop_back();
            down.push_back(v[i]);
        }
    }

    v.clear();
    for (ll i = 0; i < (ll)down.size(); i++)
        v.push_back(down[i]);
    for (ll i = up.size() - 2; i > 0; i--)
        v.push_back(up[i]);
}
```

4.5 Line Intersection

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

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

```
bool intersect(point p1, point p2, point p3, point
     p4)
{
    ll a1, b1, c1;
    a1 = p1.y - p2.y;
    b1 = p2.x - p1.x;
    c1 = p2.x*p1.y - p1.x*p2.y;

    ll a2, b2, c2;
    a2 = p3.y - p4.y;
    b2 = p4.x - p3.x;
    c2 = p4.x*p3.y - p3.x*p4.y;

    ll det = a1*b2 - b1*a2;

    if(!det)
        return 0;

    ll px = (b2*c1 - b1*c2);
    ll py = (a1*c2 - a2*c1);

    if(px < min(p1.x*det, p2.x*det) || px >
         max(p1.x*det, p2.x*det) || py <
         min(p1.y*det, p2.y*det) || py >
         max(p1.y*det, p2.y*det))
        return 0;
    if(px < min(p3.x*det, p4.x*det) || px >
         max(p3.x*det, p4.x*det) || py <
         min(p3.y*det, p4.y*det) || py >
         max(p3.y*det, p4.y*det))
        return 0;

    return 1;
}

int main()
{
    point p1{10, 0}, p2{0, 20}, p3{5, 5}, p4{10009,
         10009};
    cout << intersect(p1, p2, p3, p4);
}
```

5 Graph

5.1 Articulation Point

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

int n, m;
bool vis[10009];
int tin[10009], low[10009], timer;
vector<int> adj[10009];
set<int> AP;

void dfs(int v, int p = -1)
{
    vis[v] = 1;
    timer++;
    tin[v] = low[v] = timer;

    int child = 0;
    for(auto to : adj[v])
    {
        if(to == p)
            continue;
        if(!vis[to])
        {
            child++;
            dfs(to, v);
            low[v] = min(low[v], low[to]);
            if(low[to] >= tin[v] && p != -1)
                AP.insert(v);
        }
        else
            low[v] = min(low[v], tin[to]);
    }

    if(p == -1 && child > 1)
        AP.insert(v);
}

void findAP()
{
    AP.clear();
    timer = 0;
    int i;
    for(i = 1; i <= n; i++)
    {
        vis[i] = 0;
        tin[i] = -1;
        low[i] = -1;
    }

    for(i = 1; i <= n; i++)
    {
        if(!vis[i])
            dfs(i);
    }
}

int main()
{
    int i, j, u, v;
    cin >> n >> m;
    for(i = 0; i < m; i++)
    {
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    findAP();

    cout << AP.size();
}
```

5.2 Bellman Ford

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

struct edge
{
    int a, b, cost;
};

int n, m, v;
vector<edge> e;
const int INF = 1000000000;

void solve()
{
    vector<int> d (n, INF);
    vector<int> p (n, -1);
    d[v] = 0;
    bool any = 1;
    while(any)
    {
        any = 0;

        for (int j=0; j<m; j++)
        {
            if (d[e[j].a] < INF)
            {
                if (d[e[j].b] > d[e[j].a] +
                     e[j].cost)
                {
                    d[e[j].b] = d[e[j].a] + e[j].cost;
                    p[e[j].b] = e[j].a;
                    any = 1;
                }
            }
        }
    }

    if (d[t] == INF)
```

```
        cout << "No path from " << v << " to " << t
             << ".";
    else
    {
        vector<int> path;
        for (int cur = t; cur != -1; cur = p[cur])
            path.push_back (cur);
        reverse (path.begin(), path.end());

        cout << "Path from " << v << " to " << t <<
             ": ";
        for (size_t i = 0; i < path.size(); i++)
            cout << path[i] << ' ';
    }
}
```

5.3 Bridge

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

int n, m;
bool vis[10009];
int tin[10009], low[10009], timer;
vector<int> adj[10009];
vector<int> bridge[10009];

void dfs(int v, int p = -1)
{
    vis[v] = 1;
    timer++;
    tin[v] = low[v] = timer;

    int child = 0;
    for(auto to : adj[v])
    {
        if(to == p)
            continue;
        if(!vis[to])
        {
            child++;
            dfs(to, v);
            low[v] = min(low[v], low[to]);
            if(low[to] > tin[v])
            {
                bridge[v].push_back(to);
                bridge[to].push_back(v);
            }
        }
        else
            low[v] = min(low[v], tin[to]);
    }
}

void findBR()
{
    bridge.clear();
    timer = 0;
    int i;
    for(i = 1; i <= n; i++)
    {
        vis[i] = 0;
        tin[i] = -1;
        low[i] = -1;
    }

    for(i = 1; i <= n; i++)
    {
        if(!vis[i])
            dfs(i);
    }
}

int main()
{
    int i, j, u, v;
    cin >> n >> m;
    for(i = 0; i < m; i++)
    {
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    findBR();

    cout << bridge.size();
}
```

5.4 Centroid Decomposition

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

const int maxn = 200010;

int n;
vector <int> adj[maxn];
int subtree_size[maxn];

int get_subtree_size(int node, int par = -1)
{
        int ret = 1;
        for (auto next : adj[node])
    {
                if (next != par)
            ret += get_subtree_size(next, node);
        }
        return subtree_size[node] = ret;
}

int get_centroid(int node, int par = -1)
{
        for (auto next : adj[node])
    {
                if (next != par &&
                     subtree_size[next] * 2 > n)
                        return get_centroid(next,
                             node);
        }
        return node;
}

int main()
{
    int i, a, b;
        cin >> n;
        for (i = 1; i < n; i++)
    {
                a, b;
                cin >> a >> b;

                adj[a].push_back(b);
                adj[b].push_back(a);
        }

        get_subtree_size(1);
        cout << get_centroid(1) << endl;
}
```

5.5 Dijkstra

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define ff first
#define ss second

ll n;
vector <pair<ll, ll>> adj[505];
ll dis[505], par[505];

void init()
{
    for(ll i = 1; i <= n; i++)
    {
        dis[i] = INT_MAX;
        par[i] = -1;
    }
}

void dijkstra(ll s)
{
    init();

    set <pair<ll, ll> > q;

    dis[s] = 0;
    q.insert({0, s});

    while(!q.empty())
    {
        pair<ll, ll> p = *q.begin();
        q.erase(q.begin());

        ll node = p.ss;
        if(p.ff > dis[node])
            continue;

        for (auto u : adj[node])
        {
            ll len = u.ff;
            ll to = u.ss;
            if (dis[node] + len < dis[to])
            {
                dis[to] = dis[node] + len;
                q.insert({dis[to], to});
                par[to] = node;
            }
        }
    }
}

int main()
{
    ll i, m, s;

    cin >> n >> m;
    for(i = 0; i < m; i++)
    {
        ll a, b, c;
        cin >> a >> b >> c;
        adj[a].push_back({c, b});
        adj[b].push_back({c, a});
    }

    cin >> s;
    dijkstra(s);

    for(i = 1; i <= n; i++)
        cout << i << ": " << dis[i] << endl;
}
```

5.6 Finding Cycle

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

int n;
vector<vector<int>> adj;
vector<int> color, parent;
int cycle_start, cycle_end;

bool dfs(int v)
{
    color[v] = 1;
    for (int u : adj[v])
    {
        if (color[u] == 0)
        {
            parent[u] = v;
            if (dfs(u))
                return true;
        }
        else if (color[u] == 1)
        {
            cycle_end = v;
            cycle_start = u;
            return true;
        }
    }
    color[v] = 2;
    return false;
}

void find_cycle()
{
    color.assign(n, 0);
    parent.assign(n, -1);
    cycle_start = -1;

    for (int v = 0; v < n; v++)
    {
        if (color[v] == 0 && dfs(v))
            break;
    }

    if (cycle_start == -1)
    {
        cout << "Acyclic" << endl;
    }
    else
    {
        vector<int> cycle;
        cycle.push_back(cycle_start);
        for (int v = cycle_end; v != cycle_start; v
             = parent[v])
            cycle.push_back(v);
        cycle.push_back(cycle_start);
        reverse(cycle.begin(), cycle.end());

        cout << "Cycle found: ";
        for (int v : cycle)
            cout << v << " ";
        cout << endl;
    }
}
```

5.7 Floyd warshall

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

#define ll long long

ll n;
vector <pair<ll, ll>> adj[10009];
ll dis[10009][10009];

int main()
{
    ll i, j, k, m, u, v, w;

    cin >> n >> m;

    for(i = 1; i <= n; i++)
    {
        for(j = 1; j <= n; j++)
            dis[i][j] = INT_MAX;
    }

    for(i = 1; i <= n; i++)
        dis[i][i] = 0;

    for(i = 0; i < m; i++)
    {
        cin >> u >> v >> w;
        adj[u].push_back({w, v});
        dis[u][v] = w;
    }

    for(k = 1; k <= n; k++)
    {
        for(i = 1; i <= n; i++)
        {
            for(j = 1; j <= n; j++)
                dis[i][j] = min(dis[i][j], dis[i][k]
                     + dis[k][j]);
        }
    }

    for(i = 1; i <= n; i++)
    {
        for(j = 1; j <= n; j++)
            cout << dis[i][j] << " ";
        cout << endl;
    }
}
```

5.8 Kruskal

```
#include<bits/stdc++.h>
using namespace std;
#define ff first
#define ss second

int parent[10009], n, m;
vector<pair<int, pair<int, int> > >edges;

int Find(int a)
{
    if(parent[a] == a)
        return a;
    parent[a] = Find(parent[a]);
    return parent[a];
}

void Union(int a, int b)
{
    a = Find(a);
    b = Find(b);
    parent[a] = b;
}

int kruskal()
{
    int i, u, v, w, ret = 0;
    for(i = 0; i <= n; i++)
        parent[i] = i;

    sort(edges.begin(), edges.end());
    int cnt = 0;
    for(auto e : edge)
    {
        u = e.ss.ff;
        v = e.ss.ss;
        w = e.ff;

        if(Find(u) != Find(v))
        {
            Union(u, v);
            ret += w;
            cnt++;
            if(cnt == n-1)
                return ret;
        }
    }
}

int main()
{
    int i, j;

    cin >> n >> m;
    for(i = 0; i < m; i++)
    {
        int a, b, c;
        cin >> a >> b >> c;
        edge.push_back({c, {a, b}});
    }

    kruskal();
}
```

5.9 Strongly Connected Components

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

const int N = 10009;
vector <int> g[N], gr[N];
bool vis[N];
vector<int> order, component;

void dfs1(int v)
{
    vis[v] = 1;
    for (auto u : g[v])
    {
        if (!vis[u])
            dfs1(u);
    }
    order.push_back (v);
}

void dfs2(int v)
{
    vis[v] = 1;
    component.push_back(v);
    for (auto u : gr[v])
    {
        if(!vis[u])
            dfs2(u);
    }
}

int main()
{
    int n, m, i, cnt = 0;
    cin >> n >> m;
    for (i = 0; i < m; i++)
    {
        int a, b;
        cin >> a >> b;
        g[a].push_back (b);
        gr[b].push_back (a);
    }

    memset(vis, 0, sizeof vis);
    for (i = 1; i <= n; i++)
    {
        if (!vis[i])
            dfs1(i);
    }

    memset(vis, 0, sizeof vis);
    for (i = 1; i <= n; i++)
    {
        int v = order[n-i];
        if (!vis[v])
        {
            dfs2 (v);
            cout << "Component No. " << ++cnt << ":
                ";
            for(auto u : component)
                cout << u << " ";
            cout << endl;
            component.clear();
        }
    }
}
```

5.10 Topsort with DFS

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

const int N = 100005;
vector<int> adj[N];
stack<int> st;
```

```
int col[N];

bool dfs(int s)
{
    int ret = 1;
    col[s] = 1;
    for(auto u : adj[s])
    {
        if(col[u] == 0)
            ret = ret & dfs(u);
        else if(col[u] == 1)
            return 0;
    }
    col[s] = 2;
    st.push(s);
    return ret;
}

int main()
{
    int i, node, edge;
    cin >> node >> edge;
    for(i = 1; i <= edge; i++)
    {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
    }
    for(i = 1; i <= node; i++)
    {
        if(col[i] == 0 && dfs(i) == 0)
        {
            cout << "impossible";
            return 0;
        }
    }
    while(!st.empty())
    {
        cout << st.top() << " ";
        st.pop();
    }
}
```

5.11 TopSort with Indegree

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

const int N = 100005;
int n;
vector<int> adj[N];
vector<int> path;
int in[N];

void topsort()
{
    queue<int> Q;
    int i;
    for(i = 1; i <= n; i++)
    {
        if(in[i] == 0)
            Q.push(i);
    }
    while(!Q.empty())
    {
        int node = Q.front();
        Q.pop();
        path.push_back(node);
        for(auto u : adj[node])
        {
            in[u]--;
            if(in[u] == 0)
                Q.push(u);
        }
    }
}

int main()
{
    int i, m;
    cin >> n >> m;
    for(i = 0; i < m; i++)
    {
        int u,v;
        cin >> u >> v;
        adj[u].push_back(v);
        in[v]++;
    }

    topsort();

    if(path.size() != n)
        cout << "impossible";
    else
    {
        for(i = 0; i < path.size(); i++)
            cout << path[i] << " ";
    }

}
```

6 Number Theory

6.1 Bigmod

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

#define ll long long

ll bigmod(ll a, ll b, ll mod)
{
    if(b == 0)
        return 1%mod;
    if(b == 1)
        return a%mod;

    ll res = bigmod(a, b>>1, mod);
    res = (res*res)%mod;
    if(b&1)
        return (a*res)%mod;
    return res;
}

ll bigmod2(ll a, ll b, ll mod)
{
    ll res = 1%mod;
    while (b)
    {
        if (b & 1)
            res = (res * a)%mod;
        a = (a * a)%mod;
        b >>= 1;
    }
    return res;
}

int main()
{
    ll a, b, mod;
    cout << bigmod2(7, 5*29, 91);
}
```

6.2 Euler's Totient

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define N 100009
bool flag[N];
vector<int> primes;
ll phi[N];

void sieve()
{
    int i, j;
    flag[0] = flag[1] = 1;
    for(i = 4; i < N; i += 2)
        flag[i] = 1;
    for(i = 3; i * i < N; i += 2)
    {
        if(!flag[i])
        {
            for(j = i * i; j < N; j += 2 * i)
                flag[j] = 1;
        }
    }

    for(i = 2; i < N; i++)
    {
        if(!flag[i])
            primes.push_back(i);
    }
}

ll findPhi(ll n)
{
    if(phi[n] != 0)
        return phi[n];

    ll i, cnt, ret = n, temp = n;
    for(i = 0; primes[i] * primes[i] <= n; i++)
    {
        for(cnt = 0; n % primes[i] == 0; cnt++)
            n /= primes[i];

        if(cnt > 0)
            ret = ret / primes[i] * (primes[i] - 1);
    }

    if(n > 1)
        ret = ret / n * (n - 1);

    return phi[temp] = ret;
}

void sievephi()
{
    ll i, j;
    for(i = 1; i < N; i++)
        phi[i] = i;

    for(i = 2; i < N; i++)
    {
        if(phi[i] == i)
        {
            for(j = i; j < N; j += i)
                phi[j] = phi[j] / i * (i - 1);
        }
    }
}
```

```
void segsievephi(ll a, ll b)
{
    ll i, j, cnt;

    for(i = a; i <= b; i++)
    {
        phi[i-a] = i;
        val[i-a] = i;
    }

    for(auto p : primes)
    {
        if(p * p > b)
            break;

        for(i = (a + p - 1) / p * p; i <= b; i += p)
        {
            for(cnt = 0; val[i - a] % p == 0; cnt++)
                val[i - a] /= p;

            if(cnt)
                phi[i - a] = phi[i - a] / p * (p -
                     1);
        }
    }

    for(i = a; i <= b; i++)
    {
        if(val[i - a] > 1)
            phi[i - a] = phi[i - a] / val[i - a] *
                 (val[i - a] - 1);
    }
}

int main()
{
    sieve();

    int n;
    cin >> n;
    cout << findPhi(n) << endl;

    sievephi();
    for(int i = 1; i <= n; i++)
        cout << phi[i] << " ";
}
```

6.3 Extended GCD

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

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

int egcd2(int a, int b, int& x, int& y)
{
    x = 1, y = 0;
    int x1 = 0, y1 = 1, a1 = a, b1 = b;
    while (b1)
    {
        int q = a1 / b1;
        tie(x, x1) = make_tuple(x1, x - q * x1);
        tie(y, y1) = make_tuple(y1, y - q * y1);
        tie(a1, b1) = make_tuple(b1, a1 - q * b1);
    }
    return a1;
}

bool LDE(int a, int b, int c, int &x0, int &y0, int
     &d) {
    d = egcd(abs(a), abs(b), x0, y0);
    if (c % d)
        return 0;

    x0 *= c / d;
    y0 *= c / d;
    x0 = (a < 0? -1 : 1) * x0;
    y0 = (b < 0? -1 : 1) * y0;
    return 1;
}

bool LDEall(int a, int b, int c, int t, int &x, int
     &y)
{
    int d;
    if(LDE(a, b, c, x, y, d))
    {
        x = x + b*t;
        y = y - a*t;
        return 1;
    }

    return 0;
}
```

6.4 Invmod

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define N 100007
ll inv[N];

ll bigmod(ll a, ll p, ll m)
{
    if(p == 0)
        return 1;
    if(p == 1)
        return a%m;
    ll x = 1;
    if(p&1)
        x = a%m;
    ll y = bigmod(a, p/2, m);
    return ((y*y)%m*x)%m;
}
ll bigmod2(ll a, ll b, ll mod)
{
    ll res = 1%mod;
    while (b)
    {
        if (b & 1)
            res = (res * a)%mod;
        a = (a * a)%mod;
        b >>= 1;
    }
    return res;
}

ll invmod(ll a, ll m) //only if m is prime and
     gcd(a, m) = 1
{
    return bigmod(a, m-2, m);
}

ll egcd(ll a, ll m, ll& x, ll& y)
{
    if(m == 0)
    {
        x = 1;
        y = 0;
        return a;
    }

    ll x1, y1;
    ll d = egcd(m, a%m, x1, y1);

    x = y1;
    y = x1 - y1*(a/m);
    return d;
}

ll invmod2(ll a, ll m) //when gcd(a, m) = 1
{
    ll x, y;
    egcd(a, m, x, y);

    return (x%m + m) % m;
}

void allinvmod() //when N is prime
{
    ll i;
    inv[1] = 1;
    for(i = 2; i < N; i++)
        inv[i] = ((-N/i*inv[N%i]) % N + N) % N;
}
```

6.5 Linear Sieve

```
#include<bits/stdc++.h>
using namespace std;
#define N 10000007

int leastFactor[N];
bool flag[N];
vector<int> primes;
void linSieve()
{
    int i, j;
    for(i = 2; i < N; i++)
    {
        if (!flag[i])
            primes.push_back(i);

        for(j = 0; j < (int)primes.size() &&
             i*primes[j] < N; j++)
        {
            flag[i * primes[j]] = 1;
            if(i % primes[j] == 0)
                break;
        }
    }
}

void linSieve2()
{
    int i, j;
    for (i = 2; i < N; ++i)
    {
        if (leastFactor[i] == 0)
        {
            leastFactor[i] = i;
            primes.push_back(i);
        }
        for (j = 0; j < (int)primes.size() &&
             primes[j] <= leastFactor[i] &&
             i*primes[j] < N; ++j)
        {
            leastFactor[i * primes[j]] = primes[j];
        }
    }
}

int main()
{
    linSieve();

    int mx = 0;
    for(int i = 0; i < 10; i++)
        cout << primes[i] << " ";
}
```

6.6 Matrix Exponentiation

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define mod 1000000007
typedef vector<vector<ll>> Mat;

Mat mul(Mat A, Mat B)
{
    Mat ret(A.size(), vector<ll>(B[0].size()));
    ll i, j, k;

    for(i = 0; i < ret.size(); i++)
    {
        for(j = 0; j < ret[0].size(); j++)
        {
            for(k = 0; k < A[0].size(); k++)
                ret[i][j] = (ret[i][j] +
                      (A[i][k]*B[k][j])%mod)%mod;
        }
    }

    return ret;
}

Mat power(Mat A, ll p)
{
    Mat ret(A.size(), vector<ll>(A[0].size()));
    for(ll i = 0; i < ret.size(); i++)
        ret[i][i] = 1;

    while(p)
    {
        if(p&1)
            ret = mul(ret, A);
        A = mul(A, A);
        p >>= 1;
    }
    return ret;
}

int main()
{
    Mat A(3, vector<ll>(3));
    Mat B;
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
            A[i][j] = i+j;
    }

    B = power(A, 0);
    for(int i = 0; i < 3; i++)
    {
        for(int j = 0; j < 3; j++)
            cout << B[i][j] << " ";
        cout << endl;
    }
}
```

6.7 Mulmod

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

#define ll long long

ll mulmod(ll a, ll b, ll mod)
{
    if(b == 0)
        return 0;
    ll res = mulmod(a, b>>1, mod);
    res = (res<<1)%mod;
    if(b&1)
        return (res+a)%mod;
    else
        return res;
}

ll mulmod2(ll a, ll b, ll mod)
{
    ll res = 0;
    while(b)
    {
        if(b&1)
            res = (res + a)%mod;
        res = (res << 1)%mod;
        b >>= 1;
    }
    return res;
}
```

6.8 nCr

```
/*
We need actual value assuming answer fits in long long
    1. O(r): Multiply by (n-i) and divide by i, in each step. C(n, r) = C(n-1, r-1)*n/r

Prime mod M
    2. O(n): Precalculate factorial and inverse factorial array.
       O(1): Answer each query from these arrays
    3. O(M): Use Lucas Theorem

Non-prime mod M
    4. O(n*n): Use Pascal's Triangle
    5. O(M): Use Chinese Remainder Theorem
*/

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

ll fact[2000006];
ll inv[2000006];
ll dp[500][500];

ll findFact(ll n, ll mod);
ll bigmod(ll a, ll p, ll mod);
ll invmod(ll a, ll mod);

ll nCr1(ll n, ll r)
{
    if(n < r)
        return 0;

    r = min(r, n-r);

    if(r == 0)
        return 1;
    return n * nCr1(n-1, r-1) / r;
}

ll nCr2(ll n, ll r, ll mod)
{
    if(n < r)
        return 0;
    return ((findFact(n, mod) * invmod(findFact(r, mod), mod))%mod * invmod(findFact(n-r, mod), mod))%mod;
}

ll nCr3(ll n, ll r, ll mod)
{
    if(n < r)
        return 0;

    ll ret = 1;
    while(r)
    {
        ret = (ret * nCr2(n%mod, r%mod))%mod;
        n /= mod;
        r /= mod;
    }

    return ret;
}

ll nCr4(ll n, ll r, ll mod)
{
    if(n < r)
        return 0;

    if(dp[n][r] != 0)
        return dp[n][r];

    if(!r)
        return dp[n][r] = 1;

    return dp[n][r] = (nCr4(n-1, r-1, mod) + nCr4(n-1, r, mod)) % mod;
}

ll nCr5(ll n, ll r, ll mod)
{
    return -1;
}

int main()
{
    cout << nCr1(5, 3) << "\n";
    cout << nCr2(5, 3, 101) << "\n";
    cout << nCr4(5, 3, 101) << "\n";
}
```

```
ll findFact(ll n, ll mod)
{
    if(fact[n])
        return fact[n];
    if(n == 0 || n == 1)
        return 1;

    fact[n] = (n*findFact(n-1, mod))%mod;
    return fact[n];
}

ll bigmod(ll a, ll p, ll mod)
{
    if(p == 0)
        return 1;
    if(p == 1)
        return a%mod;

    ll res = bigmod(a, p>>1, mod);
    res = (res*res)%mod;
    if(p&1)
        return (a*res)%mod;
    return res;
}

ll invmod(ll a, ll mod)
{
    if(inv[a])
        return inv[a];

    return inv[a] = bigmod(a, mod-2, mod);
}
```

6.9 Number of Divisions (sqrt)

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define N 1000009

bool flag[N];
vector<ll> primes;

void sieve()
{
    ll i, j;

    flag[2] = 1;
    for(i = 3; i < N; i += 2)
        flag[i] = 1;

    for(i = 3; i * i < N; i+=2)
    {
        if(flag[i])
        {
            for(j = i*i; j < N; j += 2*i)
                flag[j] = 0;
        }
    }

    primes.push_back(2);
    for(i = 3; i < N; i += 2)
    {
        if(flag[i])
            primes.push_back(i);
    }
}

ll NOD(ll n)
{
    ll i, c, ret = 1;

    for(i = 0; primes[i]*primes[i] <= n; i++)
    {
        for(c = 0; n % primes[i] == 0; c++)
            n /= primes[i];
        ret *= (c+1);
    }

    if(n > 1)
        ret = ret << 1;

    return ret;
}
```

6.10 Shanks' Baby Step, Giant Step

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

int bigmod(int b, int p, int m)
{
    if(p == 0)
        return 1;

    int ret = bigmod(b, p/2, m);
    ret = (ret*ret)%m;

    if(p&1)
        ret = (ret*b)%m;

    return ret;
}

int babyStepGiantStep(int a, int b, int p)
{
    int i, j, c, sq = sqrt(p);
    map<int, int> babyTable;

    for(j = 0, c = 1; j <= sq; j++, c = (c*a)%p)
        babyTable[c] = j;

    int giant = bigmod(a, sq*(p-2), p);

    for(i = 0, c = 1; i <= sq; i++, c = (c*giant)%p)
    {
        if(babyTable.find((c*b)%p) !=
             babyTable.end())
            return i*sq+babyTable[(c*b)%p];
    }

    return -1;
}

int main()
{
    int a, b, p;

    cin >> a >> b >> p;

    cout << babyStepGiantStep(a, b, p);
}
```

6.11 Sieve of Eratosthenes

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define N 10000007
bool flag[N];
vector<ll> primes;
void sieve()
{
    ll i, j;

    flag[2] = 1;
    for(i = 3; i < N; i += 2)
        flag[i] = 1;

    for(i = 3; i * i < N; i+=2)
    {
        if(flag[i])
        {
            for(j = i*i; j < N; j += 2*i)
                flag[j] = 0;
        }
    }

    for(i = 2; i < N; i++)
    {
        if(flag[i])
            primes.push_back(i);
    }
}
```

6.12 Sum of Divisors

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define N 1000009

bool flag[N];
int leastFactor[N];
vector<ll> primes;
ll sod[N];

void linSieve()
{
    int i, j;

    for(i = 2; i < N; i++)
    {
        if(leastFactor[i] == 0)
        {
            leastFactor[i] = i;
            primes.push_back(i);

        }
        for(j = 0; j < (int)primes.size() &&
             primes[j] <= leastFactor[i] &&
             i*primes[j] < N; j++)
            leastFactor[i*primes[j]] = primes[j];
    }
}


void sieve()
{
    ll i, j;

    flag[0] = flag[1] = 1;
    for(i = 4; i < N; i += 2)
        flag[i] = 1;

    for(i = 3; i * i < N; i+=2)
    {
        if(!flag[i])
        {
            for(j = i*i; j < N; j += 2*i)
                flag[j] = 1;
        }
    }

    for(i = 2; i < N; i++)
    {
        if(!flag[i])
            primes.push_back(i);
    }
}

ll linSOD(ll n)
{
    ll lf, c, p, ret = 1;

    while(n > 1)
    {
        lf = leastFactor[n];
        p = 1;

        for(c = 0; n%lf == 0; c++)
        {
            n /= lf;
            p *= lf;
        }

        ret *= (p*lf - 1)/(lf - 1);
    }

    return ret;
}

ll SOD(ll n)
{
    ll i, c, ret = 1;

    for(i = 0; primes[i]*primes[i] <= n; i++)
    {
        ll p = 1;
        for(c = 0; n % primes[i] == 0; c++)
        {
            n /= primes[i];
            p = p * primes[i];
        }
        ret *= (p * primes[i] - 1) / (primes[i] - 1);
    }

    if(n > 1)
        ret *= (n*n - 1) / (n - 1);
    return ret;
}
void allSOD()
{
    ll i, j;

    for(i = 1; i < N; i++)
    {
        for(j = i; j < N; j += i)
            sod[j] += i;
    }
}
```

7 String

7.1 Hashing

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
ll base = 31, mod = 1e9+7;

ll n, p[100009];
ll preHash[100009];
ll sufHash[100009];

int main()
{
    ll i;
    string s;

    p[0] = 1;
    for(i = 1; i < 100009; i++)
        p[i] = (p[i-1] * base) % mod;

    cin >> s;

    n = s.size();

    for(i = 1; i <= n; i++)
    {
        preHash[i] = (preHash[i-1] * base) % mod;
        preHash[i] = (preHash[i] + s[i-1] - 'a' + 1)
             % mod;
    }

    for(i = n; i > 0; i--)
    {
        sufHash[i] = (sufHash[i+1] * base) % mod;
        sufHash[i] = (sufHash[i] + s[i-1] - 'a' + 1)
             % mod;
    }
}
```

7.2 KMP

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

int LPS[200009], n;
string s;

void KMP()
{
    int i, j, n = s.length();

    LPS[0]=0;

    for (i = 1; i < n; i++)
    {
        j = LPS[i-1];

        while (j > 0 && s[i] != s[j])
            j = LPS[j-1];

        if (s[i] == s[j])
            j++;

        LPS[i] = j;
    }
}

int main()
{
    string p, t;
    cin >> p >> t;
    s = p + '#' + t;

    KMP();

    int i, cnt = 0;

    for(i = p.size() + 1; i < s.size(); i++)
    {
        if(LPS[i] == p.size())
            cnt++;
    }
    cout << cnt;
}
```

7.3 Manacher's Algo

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

int pal[300009], n;
string s = "#";

void manacher()
{
    int i, l, r, k;

    for(i = 0, l = 0, r = -1; i < n; i++)
    {
        if(i > r)
            k = 1;
        else
            k = min(pal[l + r - i], r - i + 1);

        while(i-k >= 0 && i+k < n && s[i-k] ==
             s[i+k])
            k++;

        pal[i] = k;
        k--;

        if(i+k > r)
        {
            l = i-k;
            r = i+k;
        }
    }
}

int main()
{
    int i, ans = 0;
    string stemp;

    cin >> n >> stemp;

    for(i = 0; i < stemp.size(); i++)
    {
        s.push_back(stemp[i]);
        s.push_back('#');
    }

    n = s.size();

    manacher();

    for(i = 0; i < n; i++)
        cout << pal[i] << " ";
}
```

7.4 Trie

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

ll trie[6800009][2], len[6800009];
ll id;

void Add(ll x)
{
    ll r = 0;

    for(ll i = 34; i >= 0; i--)
    {
        ll bit = ((x & (1LL << i)) >> i);

        if(trie[r][bit] == 0)
            trie[r][bit] = ++id;

        r = trie[r][bit];
        len[r]++;
    }
}

void Erase(ll x)
{
    ll r = 0;

    for(ll i = 34; i >= 0; i--)
    {
        ll bit = ((x & (1LL << i)) >> i);

        r = trie[r][bit];
        len[r]--;
    }
}

int main()
{
    ll q, x;
    string s;

    Add(0); //Majhemoddhe 0 dhukano lagbe

    cin >> q;

    while(q--)
    {
        cin >> s >> x;

        if(s == "+")
            Add(x);
        else if(s == "-")
            Erase(x);
    }
}
```

7.5 Z algorithm

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

int Z[100005];

void z_function(string s)
{
    int i, l, r, n = s.size();
    Z[0] = 0;
    for(i = 1, l = 0, r = 0; i < n; i++)
    {
        if(i <= r) //This condition is false when i=1
            Z[i] = min(r-i+1, Z[i-l]);
        while(i+Z[i] < n && s[Z[i]] == s[i+Z[i]])
            Z[i]++; //if Z[1] has previous value, it
                 will cause problem here

        if(i+Z[i] - 1 > r)
        {
            l = i;
            r = i+Z[i]-1;
        }
    }
}
```

8 Templates

8.1 Nafis Template

```
/*
Check and remove this section while coding
1. Get rid of toolbars except compiler and main.
     Enable only logs and status.
2. Use C++17 in global compiler settings.
3. Turn on Wall, Wextra, Wshadow in warnings.
4. Make tab spout 4 spaces
5. Settings -> Compiler -> Linker Settings -> Other
     Linker Options: -Wl,--stack,268435456
6. Settings -> Environment -> General Settings ->
     Terminal to launch console programs -> select
     gnome
*/

/*
ID: nafis.f1
TASK:
LANG: C++
*/
#include<bits/stdc++.h>
using namespace std;
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template<typename T>
using ordered_set = tree<T, null_type,less<T>,
     rb_tree_tag,
     tree_order_statistics_node_update>;
#define ll long long
#define show(x) cout << #x << ": " << x << "; ";
#define pll pair<ll, ll>
#define ff first
#define ss second
#define pb push_back

#define maxN 200005
#define mod 1000000007

void solve()
{

}

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    ll T;

    cin >> T;

    while(T--);
    {
        solve();
    }
}
```

8.2 Rifat Template

```
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
#define ordered_set          tree<long long int,
     null_type, less<long long int>, rb_tree_tag,
     tree_order_statistics_node_update>
// find_by_order(k) returns iterator to kth element
     starting from 0;
// order_of_key(k) returns count of elements
     strictly smaller than k;
template<class T> using
     min_heap=priority_queue<T,vector<T>,greater<T>
     >;
#define bin_least_sig_onebit(x) __builtin_ffs(x)
#define bin_leading_zeros(x) __builtin_clz(x)
#define bin_trailing_zeros(x) __builtin_ctz(x)
#define bin_total_ones(x)    __builtin_popcount(x)
#define getbit(n,i)          (((n)&(1<<(i)))!=0)
#define setbit0(n,i)         ((n)&(~(1<<(i))))
#define setbit1(n,i)         ((n)|(1<<(i)))
#define togglebit(n,i)       ((n)^(1<<(i)))
#define Lower_bound(v, x)    distance(v.begin(),
     lower_bound(v.begin(), v.end(), x))
#define Upper_bound(v, x)    distance(v.begin(),
     upper_bound(v.begin(), v.end(), x))
#define what_is(x)           cerr << __LINE__ <<":
     " << #x << " is " << x << endl;

#define error(args...) { string _s = #args;
     replace(_s.begin(), _s.end(), ',', ' ');
     stringstream _ss(_s); istream_iterator<string>
     _it(_ss); err(_it, args); }
void err(istream_iterator<string> it) {}
template<typename T, typename... Args>
void err(istream_iterator<string> it, T a, Args...
     args)
{
   cerr << *it << " = " << a << endl;
   err(++it, args...);
}
```