

Metropolitan University , Sylhet

GetCodes : Part 1

ACM ICPC Dhaka Regional 2018

Aaman007



18

Table of Contents

Backtracking	3
Generating Permutations.....	3
Data Structure	4-11
Lowest Common Ancestor (LCA)	4-6
Merge Sort Tree	6-7
Segment Tree	7-8
Segment Tree : Lazy Propagation.....	8-9
Trie	10-11
Dynamic Programming	11-16
(0,1) Knapsack	11
Binomial Coefficient nCr	11
Bitmask DP	12
Digit DP	12-13
Edit Distance	13
Longest Common Subsequence (LCS)	14
Longest Increasing Subsequence (LIS) ...	15-16
Coin Change	16
Graph Theory	17-24
Bellman Ford (Single Source)	17-18
Bipartite Matching (BPM) Kuhn's Algo	18
Disjoint Set Union (DSU)	19
Indegree & Outdegree	19
Floyd Warshall (Multiple Source)	20
Minimum Spanning Tree : Kruskal's Algo	21
Minimum Spanning Tree : Prim's Algo	22
Topological Sort	23
Dijkstra (Single Source)	24
Greedy Technique	25
Fractional Knapsack	25

Number Theory	25-31
Big Mod	25
Euclidian Algorithm for GCD	25
Bitwise Sieve	26
Count trailing zeroes in $N!$	26
Count Divisors of $N!$	27
Euler's Totient Function	27
Find Digits of $N!$	28
Finding divisors of a number	28-29
Sieve of Eratosthenes	29
Compute $N!$ under modulo P	30
Prime Factorization	31
Range Minimum Query (RMQ)	31-33
MO's Algorithm	31-33
String Matching	33
Hashing (Rabin-Carp)	33
Counting	33
Combinatorics : Computing nCr	31-33
Others	35-39
Dinic's Maxflow.....	35-36
BPM HopcroftKarp	36-37
Bit Manipulation	38
Direction Array for 2D Array	38
Binary Search Bisection	38
DFS	38
BFS	38
Largest Submatrix in 2D Array	39
Multiplying String by Integer	39
Smallest $N!$ with X Trailing Zeroes	39

***** Generating Permutations *****

Solution with Backtracking :

```
#include<bits/stdc++.h>
#define ll long long
#define FastRead ios_base::sync_with_stdio(false);cin.tie(NULL);
using namespace std;
int n,k;
char ch[27];
bool used[27];
void permutation(int pos)
{
    if(k == 0)
        return;
    else if(pos == n)
    {
        k--;
        for(int i=0; i<n; i++)
            cout << ch[i];
        cout << endl;
        return;
    }
    for(int i=0; i<n; i++)
    {
        if(used[i])
            continue;
        used[i] = 1;
        ch[pos] = i+'A';
        permutation(pos+1);
        used[i] = 0;
    }
}
int main()
{
    FastRead
    int t,cas=1;
    cin >> t;
    while(t--)
    {
        memset(used,0,sizeof used);
        cin >> n >> k;
        cout << "Case " << cas++ << ":\n";
        permutation(0);
    }
}
```

Solution with STL :

```
string characters = "";
for(int i=0;i<n;i++)
    characters += ('A'+i);
do
{
    for(int i=0;i<n;i++)
        cout << characters[i];
    cout << endl;
    k--;
}
while(next_permutation(characters.begin(),characters.end()) && k);
```

***** Lowest Common Ancestor *****

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

#define ll          long long
#define pii         pair<ll,ll>
#define bug(a)      cerr << #a << " : " << a << endl;
#define FastRead    ios_base::sync_with_stdio(false);cin.tie(NULL);

const int MAX = 5e4+10;
struct edge
{
    int u,v,w;
    edge() {}
    edge(int _u,int _v,int _w)
    {
        u = _u;
        v = _v;
        w = _w;
    }
    bool operator<(const edge &b) const
    {
        return w < b.w;
    }
};
vector < pii > adj[MAX];
vector < edge > vv;

int par[MAX];
int Find(int u)
{
    if(par[u] == u)
        return u;
    return par[u] = Find(par[u]);
}
void Kruskal(int n)
{
    for(int i=1;i<=n;i++)
        par[i] = i;

    int cnt = 0;
    sort(vv.begin(),vv.end());

    for(int i=0;i<vv.size();i++)
    {
        int u = vv[i].u , v = vv[i].v , w = vv[i].w;

        if(Find(u) != Find(v))
        {
            par[Find(u)] = Find(v);
            cnt++;

            adj[u].push_back(pii(v,w));
            adj[v].push_back(pii(u,w));

            if(cnt == n-1)
                break;
        }
    }
}

int T[MAX] , dep[MAX] , P[MAX][18] , M[MAX][18];

void DFS(int src,int par,int lev)
{
    T[src] = par;
    dep[src] = lev;
```

```

for(int i=0;i<adj[src].size();i++)
{
    int x = adj[src][i].first;

    if(x == par)
        continue;
    M[x][0] = adj[src][i].second;
    DFS(x,src,lev+1);
}
}
void initLCA(int n)
{
    memset(P,-1,sizeof P);

    for(int i=1;i<=n;i++)
        P[i][0] = T[i];

    for(int j=1;(1<<j)<n;j++)
    {
        for(int i=1;i<=n;i++)
        {
            if(P[i][j-1] != -1)
            {
                P[i][j] = P[P[i][j-1]][j-1];
                M[i][j] = max(M[i][j-1],M[P[i][j-1]][j-1]);
            }
        }
    }
}
int query(int u,int v,int n)
{
    if(u == v)
        return 0;

    if(dep[u] < dep[v])
        swap(u,v);

    int log = 1 , mx = 0;

    while(1)
    {
        int next = log+1;

        if((1<<next) > dep[u])
            break;
        log++;
    }

    for(int i=log;i>=0;i--)
    {
        if(dep[u]-(1<<i) >= dep[v])
            mx = max(mx,M[u][i]) , u = P[u][i];
    }

    if(u == v)
        return mx;

    for(int i=log;i>=0;i--)
    {
        if(P[u][i] != -1 && P[u][i] != P[v][i])
        {
            mx = max(mx,max(M[u][i],M[v][i]));
            u = P[u][i];
            v = P[v][i];
        }
    }
    mx = max(mx,max(M[u][0],M[v][0]));
    return mx;
}

```

```

void init()
{
    vv.clear();
    for(int i=0;i<MAX;i++)
        adj[i].clear();
}
int main()
{
    FastRead

    int t,cas=1;

    cin >> t;

    while(t--)
    {
        init();

        int n,m,u,v,w;

        cin >> n >> m;

        for(int i=0;i<m;i++)
        {
            cin >> u >> v >> w;
            vv.push_back(edge(u,v,w));
        }

        Kruskal(n);

        DFS(1,1,0);
        initLCA(n);

        int q;
        cin >> q;

        cout << "Case " << cas++ << ":\n";
        while(q--)
        {
            cin >> u >> v;

            cout << query(u,v,n) << endl;
        }
    }
}

```

***** Merge Sort Tree *****

```

const int MAX = 2e5+5;

ll cum[MAX];
vector<ll>tree[4*MAX];

void build(int l,int r,int node)
{
    if(l == r)
    {
        tree[node].push_back(cum[l]);
        return;
    }
    int mid = (l+r)/2;
    int left = 2*node , right = 2*node+1;
    build(l,mid,left);
    build(mid+1,r,right);

    merge(tree[left].begin(),tree[left].end(),tree[right].begin(),tree[right].end(),back_inserter
(tree[node]));
}

```

```
int query(int L,int R,int l,int r,int node,ll t)
{
    if(l > R || r < L)
        return 0;
    else if(l>=L && r<=R)
        return lower_bound(tree[node].begin(),tree[node].end(),t)-tree[node].begin();
    int mid = (l+r)/2;
    return query(L,R,l,mid,2*node,t) + query(L,R,mid+1,r,2*node+1,t);
}

int main()
{
    int n;
    ll t;

    scanf("%d%I64d",&n,&t);

    int a[n+2];

    for(int i=1;i<=n;i++)
    {
        scanf("%d",&a[i]);
        cum[i] = cum[i-1]+a[i];
    }
    build(1,n,1);

    ll ans = 0;
    for(int i=1;i<=n;i++)
        ans += query(i,n,1,n,1,cum[i-1]+t);

    printf("%I64d\n",ans);
}
```

***** Segment Tree *****

```
bool Check(ll n,ll pos) { return (n>>pos)&1; }
const int MAX = 1e5+5;
ll a[MAX];
struct TREE
{
    int bit[65]={0};
}tree[MAX*4];

void build(int l,int r,int pos)
{
    if(l == r)
    {
        for(int i=0;i<=60;i++)
        {
            if(Check(a[l],i))
                tree[pos].bit[i] = 1;
        }
        return;
    }
    int mid = (l+r)/2;
    build(l,mid,pos*2);
    build(mid+1,r,pos*2+1);
    for(int i=0;i<=60;i++)
        tree[pos].bit[i] = tree[2*pos].bit[i]+tree[2*pos+1].bit[i];
    return;
}

void update(int l,int r,int pos,int idx,int x)
{
    if(idx < l || idx > r)
        return;
```

```

    if(l == r)
    {
        a[idx] = x;
        for(int i=0;i<61;i++)
        {
            if(Check(a[l],i))
                tree[pos].bit[i] = 1;
            else
                tree[pos].bit[i] = 0;
        }
        return;
    }
    int mid = (l+r)/2;
    update(l,mid,2*pos,idx,x);
    update(mid+1,r,2*pos+1,idx,x);
    for(int i=0;i<=60;i++)
        tree[pos].bit[i] = tree[2*pos].bit[i]+tree[2*pos+1].bit[i];
    return;
}
ll query(int L,int R,int l,int r,int pos,int k)
{
    if(l>=L && r<=R)
        return tree[pos].bit[k];
    else if(l>R || r<L)
        return 0;
    int mid = (l+r)/2;
    int q1 = query(L,R,l,mid,pos*2,k);
    int q2 = query(L,R,mid+1,r,pos*2+1,k);
    return q1+q2;
}

int main()
{
    int n;
    cin >> n;
    for(int i=1;i<=n;i++)
        cin >> a[i];
    build(1,n,1);
    int q,type,i,x,l,r,k;
    cin >> q;
    while(q--)
    {
        cin >> type;
        if(type == 1)
        {
            cin >> i >> x;
            update(1,n,1,i,x);
        }
        else
        {
            cin >> l >> r >> k;
            cout << query(1,r,1,n,1,k) << endl;
        }
    }
}

```

***** Segment Tree – Lazy Propagation *****

```

const int MAX = 1e5+10;

struct data
{
    ll prop , sum;
};
data tree[4*MAX];

```



```

void update(int l,int r,int L,int R,int pos,ll v)
{
    if(l > R || r < L)
        return;
    else if(l >= L && r <= R)
    {
        tree[pos].prop += v;
        tree[pos].sum += (r-l+1)*v;
        return;
    }

    int left = 2*pos, right = 2*pos+1, mid = (l+r)/2;

    update(l,mid,L,R,left,v);
    update(mid+1,r,L,R,right,v);

    tree[pos].sum = tree[left].sum + tree[right].sum + tree[pos].prop*(r-l+1);
}
ll query(int l,int r,int L,int R,int pos,ll carry)
{
    if(l > R || r < L)
        return 0;
    else if(l >= L && r <= R)
        return tree[pos].sum + (carry*(r-l+1));

    int left = 2*pos, right = 2*pos+1, mid = (l+r)/2;

    ll x = query(l,mid,L,R,left,carry+tree[pos].prop);
    ll y = query(mid+1,r,L,R,right,carry+tree[pos].prop);

    return x+y;
}
void init()
{
    for(int i=1;i<4*MAX;i++)
        tree[i].prop = tree[i].sum = 0;
}
int main()
{
    int t, cas = 1;

    scanf("%d",&t);

    while(t--)
    {
        init();

        int n,q,type,x,y,v;

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

        printf("Case %d:\n",cas++);
        while(q--)
        {
            scanf("%d",&type);
            if(type)
            {
                scanf("%d%d",&x,&y);
                printf("%lld\n",query(1,n,x+1,y+1,1,0));
            }
            else
            {
                scanf("%d%d%d",&x,&y,&v);
                update(1,n,x+1,y+1,1,1LL*v);
            }
        }
    }
}

```

***** Trie – Data Structure *****

```

bool Check(ll n,ll pos) { return (n & (1<<pos)); }
bool Set(ll &n,ll pos) { return n = n | (1<<pos); }
bool Clear(ll &n,ll pos) { return n = n & ~(1<<pos); }
struct trieNode
{
    trieNode *one,*zero;
    int cnt;
    trieNode()
    {
        one = zero = NULL;
        cnt = 0;
    }
};
trieNode *root;
void Insert(ll n)
{
    trieNode *cur = root;
    for(int i=29;i>=0;i--)
    {
        if(Check(n,i))
        {
            if(cur->one == NULL)
                cur->one = new trieNode();
            cur = cur->one;
        }
        else
        {
            if(cur->zero == NULL)
                cur->zero = new trieNode();
            cur = cur->zero;
        }
        cur->cnt++;
    }
}
ll MinimumXor(ll n)
{
    trieNode *cur = root;
    for(int i=29;i>=0;i--)
    {
        if(Check(n,i))
        {
            if(cur->one && cur->one->cnt)
            {
                Clear(n,i);
                cur = cur->one;
            }
            else
                cur = cur->zero;
        }
        else
        {
            if(cur->zero && cur->zero->cnt)
                cur = cur->zero;
            else
            {
                Set(n,i);
                cur = cur->one;
            }
        }
        cur->cnt--;
    }
    return n;
}
int main()
{
    FastRead

```

```

root = new trieNode();
int n;
cin >> n;
ll a[n+2] , p;
for(int i=0;i<n;i++)
    cin >> a[i];
for(int i=0;i<n;i++)
{
    cin >> p;
    Insert(p);
}
for(int i=0;i<n;i++)
    cout << MinimumXor(a[i]) << " ";
cout << endl;
}

```

***** (0,1) Knapsack *****

```

int dp[1005][1005],wiegth[1005],cost[1005] , CAP , N;
int knapsack(int i,int w)
{
    if(i==N)
        return 0;
    if(dp[i][w]!=-1)
        return dp[i][w];
    int profit1=0,profit2=0;
    if(w+wiegth[i]<=CAP)
        profit1 = cost[i]+knapsack(i+1,w+wiegth[i]);
    profit2 = knapsack(i+1,w);
    return dp[i][w] = max(profit1,profit2);
}
int main()
{
    int t;
    cin >> t;
    while(t--)
    {
        memset(dp,-1,sizeof(dp));
        cin >> N >> CAP;
        for(int i=0;i<N;i++)
            cin >> wiegth[i] >> cost[i];
        cout << knapsack(0,0) << endl;
    }
}

```

***** Binomial Coefficient nCr *****

```

int dp[700][700];
int nCr(int n,int r)
{
    if(r==n)
        return 1;
    else if(r==1)
        return n;
    if(dp[n][r] != -1)
        return dp[n][r];
    else
        return dp[n][r] = nCr(n-1,r) + nCr(n-1,r-1);
}
int main()
{
    int n,r;
    memset(dp,-1,sizeof(dp));
    cin >> n >> r;
    cout << nCr(n,r) << endl;
}

```

***** Bitmask DP *****

```

bool Check(int n,int pos){ return (n & (1<<pos)); }
int Set(int n,int pos){ return (n | (1<<pos)); }
int w[20][20],n;
int dp[(1<<20)+2];
int call(int mask)
{
    if(mask == (1<<n)-1)
        return 0;
    else if(dp[mask] != -1)
        return dp[mask];

    int ans = 1<<28;
    for(int i=0;i<n;i++)
    {
        if(!Check(mask,i))
        {
            int price = w[i][i];
            for(int j=0;j<n;j++)
            {
                if(Check(mask,j))
                    price += w[i][j];
            }
            price += call(Set(mask,i));
            ans = min(ans,price);
        }
    }
    return dp[mask] = ans;
}
int main()
{
    cin >> n;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cin >> w[i][j];
        }
    }
    memset(dp,-1,sizeof dp);
    cout << call(0) << endl;
}

```

***** Digit DP *****

```

const int MAX = 1e6;

int dp[1015][1005];
string s;
char ans[1015];
bool DP(int cur,int modV,int n)
{
    if(cur == s.size())
        return (modV == 0);

    int &ret = dp[cur][modV];
    if(~ret)
        return ret;
    ret = 0;
    if(s[cur] != '?')
    {
        ret |= DP(cur+1,(modV*10+s[cur]-'0')%n,n);
        ans[cur] = s[cur];
        return ret;
    }
    int st = (cur ? 0 : 1);

```

```

for(int i=st;i<=9;i++)
{
    ret |= DP(cur+1, (modV*10+i)%n,n);
    if(ret)
    {
        ans[cur] = i+'0';
        break;
    }
}
return ret;
}
int main()
{
    FastRead

    int n;

    cin >> s >> n;

    memset(dp,-1,sizeof dp);
    if(!DP(0,0,n))
        cout << "*" << endl;
    else
    {
        for(int i=0;i<s.size();i++)
            cout << ans[i];
        cout << endl;
    }
}

```

***** Edit Distance *****

```

#include<bits/stdc++.h>
using namespace std;
int dp[2005][2005];
string s,s1,ans;
int main()
{
    int t;
    cin >> t;
    while(t--)
    {
        memset(dp,0,sizeof(dp));
        cin >> s >> s1;
        int len1 = s.length() , len2 = s1.length();
        for(int i=0;i<=len1;i++)
            dp[i][0] = i;
        for(int i=0;i<=len1;i++)
            dp[0][i] = i;
        for(int i=1;i<=len1;i++)
        {
            for(int j=1;j<=len2;j++)
            {
                if(s[i-1]==s1[j-1])
                    dp[i][j] = dp[i-1][j-1];
                else
                    dp[i][j] = min(dp[i-1][j-1]+1,min(dp[i-1][j]+1,dp[i][j-1]+1));
            }
        }
        cout << dp[len1][len2] << endl;
    }
}

```

***** Longest Common Subsequence (LCS) *****

```

int dp[1005][1005];

string s,s1,ans;

int LCS(int i,int j)
{
    if(s[i]=='\0' || s1[j]=='\0')
        return 0;
    else if(dp[i][j] != -1)
        return dp[i][j];
    int len1=0,len2=0,mx=0;
    if(s[i]==s1[j])
        len1 = 1 + LCS(i+1,j+1);
    else
    {
        len1 = LCS(i+1,j);
        len2 = LCS(i,j+1);
    }
    return dp[i][j] = max(len1,len2);
}

void findLCS(int i,int j)
{
    if(s[i]=='\0' || s1[j]=='\0')
    {
        cout << ans << endl;
        return;
    }
    if(s[i]==s1[j])
    {
        ans += s[i];
        findLCS(i+1,j+1);
    }
    else
    {
        if(dp[i+1][j]>dp[i][j+1])
            findLCS(i+1,j);
        else
            findLCS(i,j+1);
    }
}

int main()
{
    memset(dp,-1,sizeof(dp));
    cin >> s >> s1;
    cout << LCS(0,0) << endl;
    ans = "";
    findLCS(0,0);
}

```

***** Longest Increasing Subsequence *****

Code 1 :

```
const int MAX = 1005;
int dp[MAX], dr[MAX], N, a[MAX];
int LIS(int u)
{
    if(dp[u] != -1)
        return dp[u];
    int maxi = 0;
    for(int i = u+1; i < N; i++)
    {
        if(a[u] < a[i] && LIS(i) > maxi)
        {
            maxi = LIS(i);
            dr[u] = i;
        }
    }
    return dp[u] = 1 + maxi;
}

void printLIS(int start)
{
    while(dr[start] != -1)
    {
        cout << "Index : " << start+1 << "\t";
        cout << "Value : " << a[start] << endl;
        start = dr[start];
    }
    cout << "Index : " << start+1 << "\t";
    cout << "Value : " << a[start] << endl;
}

int main()
{
    cin >> N;

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

    memset(dp, -1, sizeof(dp));
    memset(dr, -1, sizeof(dr));

    int maxLIS = 0, start = -1;

    for(int i = 0; i < N; i++)
    {
        if(LIS(i) > maxLIS)
        {
            maxLIS = LIS(i);
            start = i;
        }
    }

    cout << "Size : " << maxLIS << " " << endl;
    cout << "Starting Index : " << start+1 << " " << endl;

    cout << "LIS ->\n";
    printLIS(start);
}
```

Code 2 :

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

    int n;
    cin >> n;

    ll a[n+2], mx = 0, ans;
    map < ll, ll > f;
```

```

for(int i=0;i<n;i++)
{
    cin >> a[i];
    f[a[i]] = f[a[i]-1]+1;
    if(f[a[i]] > mx)
    {
        mx = f[a[i]];
        ans = a[i];
    }
}
vector < ll > v;
for(int i=n-1;i>=0;i--)
{
    if(a[i] == ans)
    {
        ans--;
        v.push_back(i+1);
    }
}
cout << v.size() << endl;
for(int i=v.size()-1;i>=0;i--)
    cout << v[i] << " ";
cout << endl;
}

```

***** Coin Change : Number of Ways *****

```

const int MOD = 100000007;
int main()
{
    int t;
    cin >> t;
    while(t--)
    {
        int k,n;
        cin >> k >> n;
        int coin[k+2];
        for(int i=1;i<=k;i++)
            cin >> coin[i];
        int ways[n+2] = {0};
        ways[0] = 1;
        for(int i=1;i<=k;i++)          /// K = number of Coins    /// N = Amount
            for(int j=coin[i];j<=n;j++)
                ways[j] = (ways[j]+ways[j-coin[i]])%MOD;
        cout << "Case X: " << ways[n] << endl;
    }
}

```

Code 2 :

```

int dp[10][105] , coin[105] , N;
int coinChange(int i,int amount)
{
    if(i==N)
    {
        if(amount==0)
            return 1;
        else
            return 0;
    }
    else if(dp[i][amount]!=-1)
        return dp[i][amount];
    int p1 = 0 , p2 = 0;
    if(amount-coin[i]>=0)
        p1 = coinChange(i,amount-coin[i]);
    p2 = coinChange(i+1,amount);

    return dp[i][amount] = p1+p2;
}

```


***** Bellman Ford *****

```

struct edge
{
    int u,v,w;
    edge(int _u,int _v,int _w)
    {
        u = _u;
        v = _v;
        w = _w;
    }
};

const int MAX = 1e5+7 , INF = 1e7+7;
vector < edge > adj;
long long dist[MAX];
int par[MAX];
int V,E;
void bellman_ford(int src)
{
    for(int i=1;i<=V;i++)
        dist[i] = INF;
    dist[src] = 0;
    par[src] = -1;
    for(int i=1;i<V;i++)
    {
        int flag = 0;
        for(auto j : adj)
        {
            if(dist[j.v] > dist[j.u]+j.w)
            {
                dist[j.v] = dist[j.u]+j.w;
                par[j.v] = j.u;
                flag = 1;
            }
        }
        if(!flag)
            break;
    }
}

void print_path(int src,int node)
{
    vector < int > path;
    int i = node;
    while(i!=-1)
    {
        path.push_back(i);
        i = par[i];
    }
    for(i=path.size()-1;i>=0;i--)
        cout << path[i] << " ";
    cout << endl;
}

bool negative_cycle(int src)
{
    bellman_ford(src);
    for(auto i : adj)
    {
        if(dist[i.v] > dist[i.u]+i.w)
        {
            return true;
        }
    }
    return false;
}

int main()
{
    ios_base::sync_with_stdio(false);cin.tie(NULL);
    int uu,vv,ww;
    cin >> V >> E;

```

```

for(int i=0;i<E;i++)
{
    cin >> uu >> vv >> ww;
    adj.push_back(edge(uu,vv,ww));
}
if(negetive_cycle(1))
    cout << "Negetive Cycle is found!!\n";
else if(dist[V]<INF)
    print_path(1,V);
else
    cout << -1 << endl;
}

```

***** BPM *****

```
const int MAX = 1e3+10;
```

```

vector < int > adj[MAX];
bool vis[MAX];
int match[MAX];
char grid[MAX][MAX];

bool DFS(int src)
{
    for(int i=0; i<adj[src].size(); i++)
    {
        int x = adj[src][i];

        if(vis[x])
            continue;

        vis[x] = 1;

        if(match[x] == -1 || DFS(match[x]))
        {
            match[x] = src;
            return 1;
        }
    }
    return 0;
}

int BPM(int n)
{
    int cnt = 0;
    for(int i=1; i<=n; i++)
    {
        memset(vis,0,sizeof vis);
        if(DFS(i))
            cnt++;
    }
    return cnt;
}

void init()
{
    for(int i=0; i<MAX; i++)
        adj[i].clear();
    memset(match,-1,sizeof match);
}

int main()
{
    init();

    int n;

    cin >> n;

    cout << BPM(n) << endl;
}

```

***** DSU *****

```
int par[130];
int Find(int u)
{
    if(par[u] == u)
        return u;
    else return par[u] = Find(par[u]);
}
void Union(int u,int v)
{
    u = Find(u) , v = Find(v);
    par[u] = v;
}
int main()
{
    for(int i=1;i<=28;i++)
        par[i] = i;
    int n;
    cin >> n;
    vector < string > ans;
    string s,s1,temp;
    cin >> s >> s1;
    for(int i=0;i<n;i++)
    {
        if(Find(s[i]-'a') != Find(s1[i]-'a'))
        {
            Union(s[i]-'a',s1[i]-'a');
            temp = s[i] , temp += " " , temp += s1[i];
            ans.push_back(temp);
        }
    }
    cout << ans.size() << endl;
    for(auto i : ans)
        cout << i << endl;
}
```

***** Indegree & Outdegree *****

```
const int mx = 1005;
vector < int > adj[mx];
int in[mx] , out[mx];
int main()
{
    int n,e,u,v;
    cin >> n >> e;
    for(int i=0;i<e;i++)
    {
        cin >> u >> v;
        in[v]++; out[u]++;
    }
    for(int i=1;i<=n;i++)
    {
        cout << "In and out degree of " << i << " is : ";
        cout << in[i] << " " << out[i] << endl;
    }
}
```

***** Floyd Warshal *****

```

const int mx = 1005 , inf = INFINITY;
int N,E;
int adj[mx][mx],path[mx][mx];
void floydWarsh()
{
    for(int k=1; k<=N; k++)
    {
        for(int i=1; i<=N; i++)
        {
            for(int j=1; j<=N; j++)
            {
                if(adj[i][j]>adj[i][k]+adj[k][j] && adj[i][k]!=inf && adj[k][j]!=inf)
                {
                    adj[i][j] = adj[i][k]+adj[k][j];
                    path[i][j] = path[i][k];
                }
            }
        }
    }
}

void findPath(int i,int j)
{
    vector < int > vec;
    while(i!=j)
    {
        if(i== -1)
        {
            cout << "No Path Found!!\n";
            return;
        }
        vec.push_back(i);
        i = path[i][j];
    }
    vec.push_back(i);
    cout << "Path : ";
    for(auto i:vec)
    {
        cout << i << "->";
    }
    cout << endl;
}

int main()
{
    int u,v,w;
    cin >> N >> E;
    for(int i=1; i<=N; i++)
    {
        for(int j=1; j<=N; j++)
        {
            path[i][j] = -1;
            adj[i][j] = inf;
        }
    }
    for(int i=1; i<=N; i++)
        adj[i][i] = 0, path[i][i] = i;

    for(int i=0; i<E; i++)
    {
        cin >> u >> v >> w;
        adj[u][v] = w;
        path[u][v] = v;
    }
    floydWarsh();
    while(1)
    {
        int a,b;
        cin >> a >> b;
        findPath(a,b);
        cout << "Distance : " << adj[a][b] << endl; } }

```

***** Minimum Spanning Tree (Kruskal) *****

```

const int mx = 1005;

struct edge
{
    int u,v,w;
};

vector < edge > adj;

int N,E,par[mx];

bool cmp(edge a,edge b)
{
    return a.w < b.w;
}

int Find(int n)
{
    if(par[n]==n)
        return n;
    return par[n] = Find(par[n]);
}

int kruskal()
{
    int ans = 0 , cnt = 0 , uu , vv;
    for(int i=1;i<=N;i++)
        par[i] = i;
    for(int i=0; i<E; i++)
    {
        uu = Find(adj[i].u) , vv = Find(adj[i].v);
        if(uu!=vv)
        {
            par[uu] = vv;
            cnt++;
            ans += adj[i].w;
            if(cnt==N-1)
                break;
        }
    }
    return ans;
}

int main()
{
    int uu,vv,ww;

    edge get;

    cin >> N >> E;

    for(int i=0; i<E; i++)
    {
        cin >> uu >> vv >> ww;

        get.u = uu;
        get.v = vv;
        get.w = ww;

        adj.push_back(get);
    }

    sort(adj.begin(),adj.end(),cmp);
    cout << kruskal() << endl;
}

```

***** Minimum Spanning Tree Prim's Algo *****

```

const int MAX = 100005;
struct node
{
    int u,w;
    node() {}
    node(int _u,int _w)
    {
        u = _u;
        w = _w;
    }
};
bool operator<(node a,node b)
{
    return a.w > b.w;
}

int cost[MAX],vis[MAX],n,e;
priority_queue<node> pq;
vector < pii > adj[MAX];

int prim(int src)
{
    for(int i=1;i<=n;i++)
        cost[i] = INFINITY;
    int ans = 0;
    cost[src] = 0;
    pq.push(node(src,cost[src]));
    while(!pq.empty())
    {
        node temp = pq.top();
        pq.pop();
        if(vis[temp.u])
            continue;
        vis[temp.u] = 1;
        ans += temp.w;
        for(auto v : adj[temp.u])
        {
            if(vis[v.first])
                continue;
            else if(v.second < cost[v.first])
            {
                cost[v.first] = v.second;
                pq.push(node(v.first,v.second));
            }
        }
    }
    return ans;
}

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

    int u,v,w;

    for(int i=0;i<e;i++)
    {
        cin >> u >> v >> w;

        adj[u].push_back(pii(v,w));
        adj[v].push_back(pii(u,w));
    }

    cout << "Minimum Cost : " << prim(1) << endl;
}

```

***** Topological Sort *****

```

const int MAX = 105;
struct Node
{
    int first,second;
    Node(){}    /// Default Constructor
    Node(int _f,int _s)
    {
        first = _f;
        second = _s;
    }
};
bool operator<(Node A,Node B)
{
    return A.second > B.second;
}
vector < int > adj[MAX];
int in[MAX];
int node,edge;
bool vis[MAX];
priority_queue < Node> q;
void topSort_BFS()
{
    while(!q.empty())
    {
        int v,u = q.top().second , w = q.top().first;
        q.pop();
        cout << u << endl;
        for(int i=0;i<adj[u].size();i++)
        {
            v = adj[u][i];
            in[v]--;
            if(!vis[v] && in[v]==0)
            {
                vis[v] = 1;
                q.push(Node(in[v],v));
            }
        }
    }
}

int main()
{
    int u,v;

    cin >> node >> edge;

    for(int i=0;i<edge;i++)
    {
        cin >> u >> v;

        adj[u].push_back(v);

        in[v]++;
    }

    int mn_node , mn = INT_MAX;

    for(int i=1;i<=node;i++)
    {
        if(in[i]==0)
            q.push(Node(in[i],i));
    }

    topSort_BFS();
}

```

***** Dijkstra Algorithom Code *****

```

const int mx = 1e5+5;

vector < pii > cost[mx];
int vis[mx],par[mx],dist[mx],N,E;

bool dijkstra(int st,int en)
{
    priority_queue< pii,vector<pii>,greater<pii> > pq;
    for(int i=1;i<=N;i++)
        dist[i] = INFINITY;
    pq.push(pii(0,st));
    par[st] = -1;
    dist[st] = 0;
    while(!pq.empty())
    {
        int h = pq.top().second;
        pq.pop();
        if(h==en)
            return true;
        vis[h] = 1;
        for(auto i : cost[h])
        {
            int w = i.second , v = i.first;
            if(!vis[v] && dist[h]+w<dist[v])
            {
                dist[v] = dist[h]+w;
                pq.push(pii(dist[v],v));
                par[v] = h;
            }
        }
    }
    return false;
}

int main()
{
    int u,v,w;
    cin >> N >> E;
    for(int i=0;i<E;i++)
    {
        cin >> u >> v >> w;
        cost[u].push_back(pii(v,w));
        cost[v].push_back(pii(u,w));
    }
    if(dijkstra(1,N))
    {
        vector < int > path;
        int i = N;
        while(i!=-1)
        {
            path.push_back(i);
            i = par[i];
        }
        for(int i=path.size()-1;i>=0;i--)
            cout << path[i] << " ";
        cout << endl;
    }
    else
        cout << -1 << endl;
}

```


***** Fractional Knapsack Code *****

```
vector < pii > v;
bool cmp(pii a,pii b)
{
    return a.second*b.first > b.second*a.first;
}
int main()
{
    int N,W,weight,price,temp,ans = 0;
    cin >> N >> W;
    for(int i=0;i<N;i++)
    {
        cin >> weight >> price;
        v.push_back(pii(weight,price));
    }
    sort(v.begin(),v.end(),cmp);
    for(int i=0;i<N;i++)
    {
        temp = min(W,v[i].first);
        W -= temp;
        ans += temp*v[i].second;
    }
    cout << "Maximum Cost : " << ans << endl;
}
```

***** Big Mod Code *****

```
#define ll long long
using namespace std;
int bigMod(int a,int b,int M)
{
    if(b==0)
        return 1;
    ll x = bigMod(a,b/2,M);
    x = (x*x)%M;
    if(b&1)
        x = (x*a)%M;
    return x;
}
int main()
{
    ll b,p,m;
    while(cin >> b >> p >> m)
    {
        cout << bigMod(b,p,m) << endl;
    }
}
```

***** Euclidian Algorithm for GCD *****

```
long long gcd(long long a,long long b)
{
    long long rem;
    while(b>0)
    {
        rem = a%b;
        a = b;
        b = rem;
    }
    return a;
}
int main()
{
    long long int a,b,rem;
    cin >> a >> b;
    cout << gcd(a,b) << endl;
}
```

***** Bitwise Sieve *****

```
bool Check(int N,int pos){ return (bool) (N & (1<<pos));}
void Set(int &N,int pos){ N=N | (1<<pos);}
const int MAX = 1e8+5;
int prime[(MAX>>5)+2];
vector< ll > primes;
void sieve()
{
    int lim = sqrt(MAX);
    for(int i=3; i<=lim; i+=2)
    {
        if(!Check(prime[i>>5],i&31))
        {
            for(int j=i*i; j<=MAX; j+=(i<<1))
            {
                Set(prime[j>>5],j&31);
            }
        }
    }
    primes.push_back(0);
    primes.push_back(2);
    int last = 1;
    for(int i=3; i<=MAX; i+=2)
    {
        if(!Check(prime[i>>5],i&31))
        {
            primes.push_back(i+primes[last]);
            last++;
        }
    }
}
int main()
{
    sieve();
    int n;
    cin >> n;
    while(n-->0)
    {
        int x,y;
        cin >> x >> y;
        cout << primes[y]-primes[x-1] << endl;
    }
}
```

*****Count trailing zeroes in factorial of a number*****

```
#define ll long long
int trailingZeroes(int n)
{
    int cnt = 0 , f = 5;
    while(f <= n)
    {
        cnt += n/f;
        f *= 5;
    }
    return cnt;
}
int main()
{
    int n;
    while(cin >> n)
    {
        cout << trailingZeroes(n) << endl;
    }
}
```

***** Count divisors of N factorial *****

```
vector < ll > primes;
const int MAX = 1000005;
bool prime[MAX];
void sieve(int n)
{
    for(int i=2; i<=n; i++)
        prime[i] = 1;
    for(int i=2; i<=n; i++)
    {
        if(prime[i]==1)
        {
            primes.push_back(i);
            for(int j=2; i*j<=n; j++)
                prime[i*j] = 0;
        }
    }
}
ll factorialDivisors(ll n)
{
    ll res = 1;
    for(int i=0;primes[i]<=n;i++)
    {
        ll exp = 0;
        ll p = primes[i];
        while(p <= n)
        {
            exp += (n/p);
            p *= primes[i];
        }
        res *= (exp+1);
    }
    return res;
}
int main()
{
    sieve(MAX);
    ll n;
    while(cin >> n)
    {
        cout << factorialDivisors(n) << endl;
    }
    return 0;
}
```

***** Euler's Totient Function *****

```
const int MAX = 1000005;
int phi[MAX],mark[MAX];
void totient()
{
    /// Initializing
    for(int i=1;i<=MAX;i++)
        phi[i] = i;
    /// 1 is not prime
    mark[1] = 1;
    /// Finding all number's phi
    for(int i=2;i<=MAX;i++)
    {
        if(!mark[i])          /// If i is prime
        {
            for(int j=i;j<=MAX;j+=i)
            {
                mark[j] = 1;      /// Marking j as not prime
                phi[j] = phi[j]/i*(i-1);  /// Formula phi(p) = p/n * (n-1) where n is p's prime
            }
        }
    }
}
```

***** Find Digits of factorial of N *****

```
#define ll    long long
int findDigits(int n)
{
    if(n<=1)
        return n;
    double digits = 0;
    for(int i=2; i<=n; i++)
        digits += log10(i);
    return floor(digits)+1;
}
int main()
{
    int n;
    while(cin >> n)
    {
        cout << findDigits(n) << endl;
    }
}
```

***** Finding Divisors of a number *****

Code for Finding Number of divisors and then printing them :

```
const int MAX = 100005;
vector < int > divisors[MAX];
void Divisors(int n)
{
    for(int i=1; i<n; i++)
    {
        for(int j=i; j<n; j+=i)
        {
            divisors[j].push_back(i);
        }
    }
}
int main()
{
    Divisors(MAX);
    int x;
    while(cin >> x)
    {
        cout << "Number of Divisors of X is : " << divisors[x].size() << endl;
        cout << "The Divisors are :\n";
        for(int i=0; i
<divisors[x].size(); i++)
            cout << divisors[x][i] << " ";
        cout << endl;
    }
}
```

Code for Sum of Divisors (SOD) :

```
long long sod[MAX];
void SOD(int n)    ///SOD = Sum of Divisors
{
    for(int i=1; i<=n; i++)
    {
        for(int j=i; j<=n; j+=i)
        {
            sod[j] += i;
        }
    }
}
int main()
{
    SOD(MAX);
}
```

Code for Number of Divisors (NOD) :

```

const int MAX = 1000005;
long long nod[MAX];
void NOD(int n)    ///NOD = Number of Divisors
{
    for(int i=1; i<n; i++)
    {
        for(int j=i; j<n; j+=i)
        {
            nod[j]++;
        }
    }
}
int main()
{
    NOD(MAX);
}

```

Sum of Divisors of numbers in range A to B :

```

ll super(ll n)
{
    if(n&1)
        return n*((n+1)>>1);
    else
        return (n>>1)*(n+1);
}
ll divisorSum(ll n)
{
    ll sum = 0 , lim = sqrt(n) , x = super(lim);
    for (ll i = 1; i <= lim; i++)
    {
        sum += (n/i)*i;
        if(i*i != n)
        {
            sum += super(n/i);
            sum -= x;
        }
    }
    return sum;
}
int main()
{
    FastRead
    ll a,b,sum;
    cin >> a >> b;
    sum = divisorSum(b)-divisorSum(a-1);
    cout << sum << endl;
}

```

******* Sieve of Erathosthenes *******

```

bool prime[10000005];
void sieve(int n)
{
    for(int i=2; i<n; i++)
        prime[i] = 1;
    for(int i=2; i<n; i++)
    {
        if(prime[i]==1)
        {
            for(int j=2; i*j<n; j++)
            {
                prime[i*j] = 0;
            }
        }
    }
}

```

***** Compute N Factorial under modulo P *****

```

vector < int > primes;
const int MAX = 1000005;
bool prime[MAX];
void sieve(int n)
{
    for(int i=2; i<=n; i++)
        prime[i] = 1;
    for(int i=2; i<=n; i++)
    {
        if(prime[i]==1)
        {
            primes.push_back(i);
            for(int j=2; i*j<=n; j++)
                prime[i*j] = 0;
        }
    }
}
int bigMod(int a,int b,int M)
{
    if(b==0)
        return 1;
    ll x = bigMod(a,b/2,M);
    x = (x*x)%M;
    if(b&1)
        x = (x*a)%M;
    return x;
}
int largestPower(int n,int p)
{
    int cnt = 0;
    while(n)
    {
        n /= p;
        cnt += n;
    }
    return cnt;
}
int fact(int n,int p)
{
    int res = 1;
    for(int i=0;primes[i]<=n;i++)
    {
        int k = largestPower(n,primes[i]);
        res = (res*bigMod(primes[i],k,p))%p;
    }
    return res;
}
int main()
{
    sieve(MAX);
    int n,p;
    while(cin >> n >> p)
    {
        cout << fact(n,p) << endl;
    }
    return 0;
}

```

***** Prime Factorization Code *****

```
vector < int > v;
void primeFact(int n)
{
    while(n%2==0 && n>0)
    {
        v.push_back(2);
        n /= 2;
    }
    for(int i=3;i<=sqrt(n);i+=2)
    {
        while(n%i==0 && n>0)
        {
            v.push_back(i);
            n /= i;
        }
    }
    if(n>2)
        v.push_back(n);
}
int main()
{
    int n;
    while(cin >> n)
    {
        v.clear();
        int k = n;
        if(n == 0)
            break;
        if(n<0)
            v.push_back(-1) , n *= -1;
        primeFact(n);
        cout << k << " = ";
        for(int i=0;i<v.size();i++)
        {
            cout << v[i];
            if(i == v.size()-1)
            {
                cout << endl;
                break;
            }
            cout << " x ";
        }
    }
}
```

***** MO's Algo *****

```
const int MAX = 1e5+10;

int BLOCK_SIZE;

struct data
{
    int l,r,idx;
    bool operator<(const data &b) const
    {
        int x = l/BLOCK_SIZE, y = b.l/BLOCK_SIZE;
        if(x != y)
            return x < y;
        return r < b.r;
    }
};

data Q[MAX];
```

```

int a[MAX], ans[MAX], freq[MAX];
unordered_set < int > s;

void add(int x)
{
    if(!freq[x])
        s.insert(x);
    freq[x]++;
}
void del(int x)
{
    freq[x]--;
    if(!freq[x])
        s.erase(x);
}
void MO(int n,int q)
{
    BLOCK_SIZE = sqrt(n);
    sort(Q,Q+q);

    int st = 0, en = -1;

    for(int i=0; i<q; i++)
    {
        int l = Q[i].l-1, r = Q[i].r-1, idx = Q[i].idx;

        if(i)
        {
            if(l == Q[i-1].l-1 && r == Q[i-1].r-1)
            {
                ans[idx] = ans[Q[i-1].idx];
                continue;
            }
        }
        while(en < r)
        {
            en++;
            add(a[en]);
        }
        while(en > r)
        {
            del(a[en]);
            en--;
        }

        while(st > l)
        {
            st--;
            add(a[st]);
        }
        while(st < l)
        {
            del(a[st]);
            st++;
        }

        int mx = 0, mn = 1e9;
        for(auto j : s)
        {
            mx = max(mx,freq[j]);
            mn = min(mn,freq[j]);
        }
        ans[idx] = mx-mn;
    }
    for(int i=0; i<q; i++)
        printf("%d\n",ans[i]);
}

```



```

int main()
{
    int n,q;

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

    for(int i=0; i<n; i++)
        scanf("%d",&a[i]);
    for(int i=0; i<q; i++)
    {
        scanf("%d%d",&Q[i].l,&Q[i].r);
        Q[i].idx = i;
    }

    MO(n,q);
}

***** Hashing *****

ll base = 1331;
ll pw[MAX];
void preCalc()
{
    pw[0] = 1;
    for(int i=1;i<MAX;i++)
        pw[i] = pw[i-1]*base;
}
ll H[MAX];
void setHash(string s)
{
    H[0] = 0;
    for(int i=1;i<s.size();i++)
        H[i] = H[i-1]*base+s[i];
}
ll getHash(int l,int r)
{
    return H[r]-(H[l-1]*pw[r-l+1]);
}
ll Hasher(string s)
{
    ll hashValue = 0;
    for(int i=0;i<s.size();i++)
        hashValue = hashValue*base+s[i];
    return hashValue;
}
int main()
{
    FastRead
    preCalc();
    int t,cas=1;
    cin >> t;
    while(t--)
    {
        string a,b;
        cin >> a >> b;

        a = "#" + a;
        setHash(a);
        int l1 = a.size() , l2 = b.size();

        ll hashValue = Hasher(b);
        int cnt = 0;
        for(int i=1;i+l2<=l1;i++)
        {
            int l = i , r = i+l2-1;
            if(getHash(l,r) == hashValue)
                cnt++;
        }
        cout << "Case " << cas++ << ": " << cnt << endl;
    }
}

```

***** Combinatorics *****

```

const int MAX = 2e5+10;
const int MOD = 1e9+7;
ll f[MAX];
void calcFact()
{
    f[0] = 1;
    for(int i=1;i<MAX;i++)
        f[i] = (f[i-1]*i)%MOD;
}
ll bigMod(ll a,ll b)
{
    if(b == 0)
        return 1;
    ll x = bigMod(a,b/2);
    x = (x*x)%MOD;

    if(b&1)
        x = (x*a)%MOD;
    return x;
}
ll nCr(ll n,ll r)
{
    ll x = (f[n]*bigMod((f[r]*f[n-r])%MOD,MOD-2))%MOD;

    return x;
}
int main()
{
    FastRead

    calcFact();

    string s;

    cin >> s;

    int n = s.size() , lf[n+2] = {} , rt[n+2] = {};
    s = "$"+s;

    for(int i=1;i<=n;i++)
    {
        lf[i] = lf[i-1];
        if(s[i] == '(')
            lf[i]++;
    }
    for(int i=n;i>=1;i--)
    {
        rt[i] = rt[i+1];
        if(s[i] == ')')
            rt[i]++;
    }

    ll ans = 0 , x , y;
    for(int i=1;i<=n;i++)
    {
        x = nCr(lf[i]+rt[i] ,
min(lf[i],rt[i]));
        y = nCr(lf[i-1]+rt[i] , min(lf[i-
1],rt[i]));

        ans = (ans+x-y)%MOD;
        if(ans<0)
            ans += MOD;
    }

    cout << ans << endl;
}

```

****Dinic's MaxFlow****

```

//V^2*E Complexity

//number of augment path * (V+E)

//Base doesn't matter

const int INF = 2000000000;

const int MAXN = 100;///total nodes

const int MAXM = 10000;///total edges

int N,edges;

int last[MAXN],Prev[MAXM],head[MAXM];

int Cap[MAXM],Flow[MAXM];

int dist[MAXN];

int nextEdge[MAXN];///used for keeping track
of next edge of ith node

queue<int> Q;

void init(int N)

{

    edges=0;

    memset(last,-1,sizeof(int)*N);

}

//cap=capacity of edges , flow = initial
flow

inline void addEdge(int u,int v,int cap,int
flow)

{

    head[edges]=v;

    Prev[edges]=last[u];

    Cap[edges]=cap;

    Flow[edges]=flow;

    last[u]=edges++;

    head[edges]=u;

    Prev[edges]=last[v];

    Cap[edges]=0;

```

```

Flow[edges]=0;

    last[v]=edges++;

}

inline bool dinicBfs(int S,int E,int N)

{

    int from=S,to,cap,flow;

    memset(dist,0,sizeof(int)*N);

    dist[from]=1;

    while(!Q.empty()) Q.pop();

    Q.push(from);

    while(!Q.empty())

    {

        from=Q.front();Q.pop();

        for(int e=last[from];e>=0;e=Prev[e])

        {

            to=head[e];

            cap=Cap[e];

            flow=Flow[e];

            if(!dist[to] && cap>flow)

            {

                dist[to]=dist[from]+1;

                Q.push(to);

            }

        }

    }

    return (dist[E]!=0);

}

inline int dfs(int from,int minEdge,int E)

{

    if(!minEdge) return 0;

    if(from==E) return minEdge;

    int to,e,cap,flow,ret;

```

```

for(;nextEdge[from]>=0;nextEdge[from]=Prev[e
])
{
    e=nextEdge[from];

    to=head[e];

    cap=Cap[e];

    flow=Flow[e];

    if(dist[to]!=dist[from]+1) continue;

    ret=dfs(to,min(minEdge, cap-flow),E);

    if(ret)
    {
        Flow[e]+=ret;

        Flow[e^1]-=ret;

        return ret;
    }
}

return 0;
}

int dinicUpdate(int S,int E)
{
    int flow=0;

    while(int minEdge = dfs(S,INF,E))
    {
        if(minEdge==0) break;

        flow+=minEdge;
    }

    return flow;
}

int maxFlow(int S,int E,int N)
{
    int totFlow=0;

    while(dinicBfs(S,E,N))

```

```

{
    for(int i=0;i<=N;i++)
    nextEdge[i]=last[i];/// update last edge of
    ith node

    totFlow+=dinicUpdate(S,E);

}

return totFlow;
}

int main()
{
    return 0;
}

```

BPM HopCroftKarp

```

//*****My Code Starts Here*****

//Esqrt(V) Complexity

//O Based

//Edge from set a to set b

const int MAXN1 = 50010; //nodes in set a
const int MAXN2 = 50010; //nodes in set b
const int MAXM = 150010; //number of edges

int n1, n2, edges, last[MAXN1], prev[MAXM],
head[MAXM];

int matching[MAXN2], dist[MAXN1], Q[MAXN1];

bool used[MAXN1], vis[MAXN1]; //vis is
cleared in each dfs

// n1 = number of nodes in set a, n2 =
number of nodes in set b

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 augmentPath() {
    bfs();
    fill(vis, vis + n1, false);
    int f = 0;
    for (int u = 0; u < n1; ++u)
        if (!used[u] && dfs(u))
            ++f;
    return f;
}

int maxMatching() {
    fill(used, used + n1, false);
    fill(matching, matching + n2, -1);
    for (int res = 0;;) {
        int f = augmentPath();
        if (!f)
            return res;
        res += f;
    }
}

int main() {
    return 0;
}

```

Bit Manipulation

```
int Set(int N,int pos){return N=N |
(1<<pos);}

int reset(int N,int pos){return N= N &
~(1<<pos);}

bool check(int N,int pos){return (bool) (N &
(1<<pos));}
```

****Direction Array******For 8 sides :**

```
int fx[]={+0,+0,+1,-1,-1,+1,-1,+1};
int fy[]={-1,+1,+0,+0,+1,+1,-1,-1};
```

For 4 sides :

```
int fx[]={1,-1,0,0};
int fy[]={0,0,1,-1};
```

****BS Bisection****

```
double Sqrt(int n)
{
    double high = (double)n , low = 0.0 ,
    mid;
    while(high-low > 0.000001)
    {
        mid = (high+low)/2.0;
        if(mid*mid > n)
            high = mid;
        else if(mid*mid < n)
            low = mid;
    }
    return mid;
}
int main()
{
    int n;
    cin >> n;
    cout << Sqrt(n) << endl;
}
```

******* DFS *******

```
const int mx = 1005;
int N,E;
bool vis[mx];
vector < int > adj[mx];
void DFS(int src)
{
```

```
    vis[src] = 1;
    cout << src << endl;
    for(auto i : adj[src])
    {
        if(!vis[i])
            DFS(i);
    }
}

int main()
{
    int u,v;
    cin >> N >> E;
    for(int i=0;i<E;i++)
    {
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    DFS(1);
}
```

******* BFS *******

```
using namespace std;

const int MAX = 1e5+10;

vector < int > adj[MAX];
bool vis[MAX];

void BFS(int src)
{
    queue < int > q;

    vis[src] = true;
    q.push(src);

    while(!q.empty())
    {
        int u = q.front();
        q.pop();

        cout << u << endl;

        for(int i=0;i<adj[u].size();i++)
        {
            int x = adj[u][i];
            if(!vis[x])
            {
                q.push(x);
                vis[x] = true;
            }
        }
    }
}

int main()
{
    int n,m,u,v;

    cin >> n >> m;

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

    BFS(1);
}
```

****Find submatrix with largest sum in a given 2D matrix of integers****

```
const int MAX = 1e6;

int n , a[205][205] , cum[205];

int kadane()
{
    int mx = -1e9 , sum = 0;
    for(int i=1;i<=n;i++)
    {
        sum += cum[i];
        mx = max(mx,sum);
        if(sum < 0)
            sum = 0;
    }
    return mx;
}

int findMaxSubMatrix()
{
    int mx = -1e9;

    for(int i=1;i<=n;i++)
    {
        memset(cum,0,sizeof cum);

        for(int j=i;j<=n;j++)
        {
            for(int k=1;k<=n;k++)
                cum[k] += a[k][j];
            mx = max(mx,kadane());
        }
    }
    return mx;
}

int main()
{
    cin >> n;

    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            cin >> a[i][j];

    cout << findMaxSubMatrix() << endl;
}
```

String Multiplied by Integer

```
string multy(string a,ll b)
{
    int carry = 0 , len = a.size();
    for(int i=0;i<len;i++)
    {
        carry += (a[i]-'0')*b;
        a[i] = (carry%10)+'0';
        carry /= 10;
    }
    while(carry){
        a += (carry%10)+'0';
        carry /= 10;
    }
    return a;
}
```

**** Smallest number N such that , factorial of N ends with exactly x zeroes ****

```
ll counter(ll p)
{
    ll cnt = 0, f = 5;
    while (f <= p)
    {
        cnt += p/f;
        f *= 5;
    }
    return cnt;
}

ll findNum(ll n)
{
    if(n==1)
        return 5;
    int st = 0;
    int en = 5*n;
    ll ans = -1;
    while (st <= en)
    {
        int mid = (st+en)/2;
        ll x = counter(mid);
        if(x == n)
        {
            ans = mid;
            en = mid-1;
        }
        else if(x > n)
            en = mid-1;
        else
            st = mid+1;
    }
    return ans;
}

int main()
{
    int t,cas=1;
    cin >> t;
    while(t-->0)
    {
        int n;
        cin >> n;
        int ans = findNum(n);
        cout << "Case " << cas++ << ": ";
        if(ans>=0)
            cout << ans << endl;
        else
            cout << "Go Home! You are
drunk!\n";
    }
}

ll triangle(ll a, ll b) { return (a + b + 1) * (b - a) / 2 ; }

ll divSum(ll a, ll b) // Sum of divisors between a to b
{
    ll n = sqrt(b);
    ll sum = 0;

    for (ll i = 1; i <= n; i++)
        sum += i * (b / i - a / i) + triangle(max(n, a / i), max(n, b
/ i));
    return sum;
}
```