ICPC Asia Dhaka Regional 2025

Team: BUBT_Sunday_Monday_Close

Team Members:

1. Md. Tuhin Hasnat
2. Amir Hamza Miraz
3. Rakin Absar Ratul

Generated: December 19, 2025

Contents

Utilities

Template

Code in C++:

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

//For Debugging
#define debug(a...)          {cout<<__LINE__<<" #-->
 ↪  ";dbg,a; cout<<endl;}
struct debugger
{
    template<typename T> debugger& operator , (const T v)
    {
        cout<<v<<" ";
        return *this;
    }
} dbg;

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template<class T>using iset=tree<T,null_type,less<T>,///d⌋
 ↪  escending_order=greater<T>,multi_iset=less_equal<T>
rb_tree_tag,tree_order_statistics_node_update> ;///exmp=
 ↪  iset<double>st
///who is k'th position= *(set.find_by_order(k)),index of
 ↪  v = set.order_of_key(v);

typedef long long ll;/// 1e18;
//typedef __int128_t LL;/// 1e32;
#define deb(a)     cout<<__LINE__<<"# "<<#a<<" ->
 ↪  "<<a<<endl;
#define lbv(vec,x)
 ↪  lower_bound(all(vec),x)-vec.begin()//retrun index
#define lba(ar,n,x) lower_bound(ar,ar+n,x)-&ar[0];//return
 ↪  index

#define LCM(a,b)  (a*b)/__gcd(a,b)
#define deg(n)  n*180/PI /// redian to degree
#define sp(n,d)  fixed << setprecision(d) <<n

template <typename T>
using minHeap = priority_queue<T, vector<T>, greater<T>>;

const double PI = acos(-1);
const double EPS = 1e-7; ///1*10^-7
const int oo = 1e9+10;
const ll MOD = 1e9 +7;// Prime
signed main()
{
#ifndef ONLINE_JUDGE
//     freopen("inputf.in", "r", stdin);  ///To read from
 ↪  a file.
//    freopen("outputf.in", "w", stdout);  ///To write  a
 ↪  file.
#endif
    cin.tie(nullptr)->sync_with_stdio(false);

}
```

Sublime Build System

Code in Bash:

```
{
    "shell_cmd": "g++ -std=c++17 \"${file}\" -o
    ↪  \"${file_path}/${file_base_name}\" &&
    ↪  \"${file_path}/${file_base_name}\" <
    ↪  \"${file_path}/inputf.in\" >
    ↪  \"${file_path}/outputf.out\"",
    "working_dir": "${file_path}",
    "selector": "source.cpp"
}
```

Number Theory

Bigmod

Code in C++:

```
ll mul(ll a, ll b, ll mod) { // a * b % mod
  return __int128(a) * b % mod;
}
ll power(ll a, ll b, ll mod) { // a^b % mod
  ll ans = 1 % mod;
  while (b) {
    if (b & 1) ans = mul(ans, a, mod);
    a = mul(a, a, mod);
    b >>= 1;
  }
  return ans;
}
ll inverse(ll a, ll mod) { // (1 / a) % mod
  return power(a, mod - 2, mod);
}
```

Derangement

Code in C++:

```
ll der[MAXN];
void precalculate_derangements() {
    der[0]=1;der[1]=0;der[2]=1;
    for (int i = 3; i < MAXN; i++) {
        der[i]=(i-1)*(der[i-1]+der[i-2])%MOD;
    }
}
```

Divisors Pre Clc

Code in C++:

```
///nlog(log(n))
vector<int>divisors[1000010];
void Divisor_pre_clc(){
    for(int div=1;div<=1000000;div++)
        for(int num=div;num<=1000000;num+=div)
            divisors[num].push_back(div);
}
```

Prime Factor Sieve

Code in C++:

```
#define SIZE_N 10000000 // 10^7
bool isprime[SIZE_N + 4];
vector<int> prime;
void sieve() {
    for(inti=3;i<=SIZE_N;i+=2)isprime[i]=true;
    isprime[2] = true;
    prime.push_back(2);
    for (ll i = 3; i <= SIZE_N; i += 2) {
        if (isprime[i]) {
            prime.push_back((int)i);
            if ((ll)SIZE_N / i >= i) {
                for (ll j = i * i; j <= SIZE_N; j += 2 *
                ↪  i)
                    isprime[j] = false;
            }
        }
    }
}
//it handle up to 10^14 with current sieve
void prime_factors(ll n) {
    for(int
    ↪   i=0;i<prime.size()&&(ll)prime[i]*prime[i]<=n;i++){
        while (n % prime[i] == 0) {
            cout << prime[i] << " ";
            n /= prime[i];
        }
    }
    if (n > 1) cout << n;
}
int main() {
    sieve();
    prime_factors(100000000000000LL); // 10^14
}
```

Prime Factor Spf

Code in C++:

```
int spf[N];///smallest prime factor
void spf_pre_clc(){///nlog(log(n))
    for(int i=2;i<=N;i++){
        spf[i]=i;
    }
    for(int div=2;div<=N;div++){
        for(int i=div;i<=N;i+=div){
            spf[i]=min(spf[i],div);
        }
    }
}
void prime_factors(int n){///log(n)
    while(n>1){
        cout<<spf[n]<<" ";
            n/=spf[n];
    }
}
int32_t main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    spf_pre_clc();
    int n;
    cin>>n;
    prime_factors(n);
}
```

Sieve

Code in C++:

```
///Nlog(N)
const int N = 1e7 + 9;
bitset<N> not_prime;
vector<int> primes;
void sieve(){
    not_prime[1] = true;
    for (int i = 2; i * i <= N; i++) {
      if (!not_prime[i]) {///prime
        for (int j = i * i; j <= N; j += i) {
          not_prime[j] = true;
        }
      }
    }
    for (int i = 2; i <= N; i++) {
      if (!not_prime[i]) { /// prime
        primes.push_back(i);
      }
    }
}
```

Cumulative Sum 2d

Code in C++:

```
int r,c;
scanf("%d%d",&r,&c);
int ar[r+5][c+5],px[r+5][c+5];
for(int i=1;i<=r;i++)
    for(int j=1;j<=c;j++)
        cin>>ar[i][j];

for(int i=0;i<=r;i++)px[i][0]=0;
for(int j=0;j<=c;j++)px[0][j]=0;

px[1][1]=ar[1][1];
for(int i=2;i<=r;i++){
    px[i][1]=px[i-1][1]+ar[i][1];

}
for(int j=2;j<=c;j++){
    px[1][j]=px[1][j-1]+ar[1][j];
}
for(int i=2;i<=r;i++)
    for(int j=2;j<=c;j++){
        px[i][j]=px[i-1][j]+px[i][j-1]+ar[i][j]-px[i-1][j
         ↪  -1];
    }
    cout<<"\nprefix sum array :"<<endl;
for(int i=1;i<=r;i++){
    for(int j=1;j<=c;j++){
        cout<<px[i][j]<<" ";
    }
    cout<<"\n";
}
cout<<"Range of summation array (i1,j1)->(i2,j2)"<<endl;
int i1,i2,j1,j2,ans;
cin>>i1>>j1>>i2>>j2;
if(i1>i2)swap(i1,i2);
if(j1>j2)swap(j1,j2);

ans=px[i2][j2]-px[i2][j1-1]-px[i1-1][j2]+px[i1-1][j1-1];

cout<<"sum of range = "<<ans<<endl;
```

Divisors

Code in C++:

```
void divisors(int n){
    vector<int> divs;
    for (int i=1;i*i<=n;i++){
        if (n%i==0){
            divs.push_back(i);
            if (i!=n/i)divs.push_back(n/i);
        }
    }
    sort(divs.begin(), divs.end());
    for (auto x: divs) cout << x << ' ';
}
```

Kadanes

Code in C++:

```
cin>>n;
ll ar[n+6];
ll mx=-9999999999;
ll sum=0;
for(int i=1;i<=n;i++){
    cin>>ar[i];
    sum=max(ar[i],sum+ar[i]);
    mx=max(mx,sum);
}
cout<<mx<<"\n";
```

Log B N

Code in C++:

```
int n,b;cin>>n>>b;
double ans;
ans=(log2(n)/log2(b));//logb(n)
cout<<ans<<"\n";
```

Ncr Npr Pre Calculation

Code in C++:

```
template <typename T>
T INV(T base,T mod=1e9+7){//defult mod=1e9+7
    return BIGMOD(base%mod,mod-2,mod)%mod;///base^-1
}
long long  fact[N+10];
long long inv_fact[N+10];
void pre() {
    fact[0] = 1;
    for (long long i = 1; i <= N; i++)
        fact[i] = (fact[i - 1]* i)%MOD;

    inv_fact[N] = INV(fact[N]);
    for (long long i=N-1;i>=0;i--)
        inv_fact[i] = (inv_fact[i + 1]*(i+1))%MOD;

}
long long nCr(long long n, long long r) {
    if (r > n || r < 0) return 0;
    return fact[n] * inv_fact[r]%MOD*inv_fact[n-r]%MOD;
}
long long nPr(long long n, long long r) {
    if (r > n || r < 0) return 0;
    return fact[n] * inv_fact[n-r]%MOD;
}
signed main(){
    pre();
    int n,r;
    cin>>n>>r;
    cout<<nCr(n,r)<<" "<<nPr(n,r)<<"\n";
}

other way NCR
long long ncr(int n, int r){
    int res = 1;
    for (int i=0; i<r; i++){
        res *= (n-i);
        res /= (i+1);
    }
    return res;
}
```

Number Hashing Rng

Code in C++:

```
struct custom_hash {
    static uint32_t splitmix32(uint32_t x) {//uint64_t
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(uint32_t x) const {//uint64_t
        static const uint32_t FIXED_RANDOM =
        chrono::steady_clock::now().time_since_epoch().co
         ↪  unt();
        return splitmix32(x + FIXED_RANDOM);
    }
}rng;///Random number generator
signed main(){
    int a=rng(1);
    int b=rng(2);
    int c=rng(3);
    cout<<a<<" "<<bitset<32>(a)<<"\n";
    cout<<b<<" "<<bitset<32>(b)<<"\n";
    cout<<c<<" "<<bitset<32>(c)<<"\n";
    /// if xor 1,2,3 then generally ans will be
     ↪  zero.because (1^2=3) ^ 3 =0
    /// if we this function then ans will never been zero
     ↪  because
    /// (rng(1)^rng(2) != rng(3)) ^ rng(3) = 0
    /// There can be only one way ans will be zero when we
     ↪  xor the same number with it.
    /// rng(num)^rng(num) = 0 //   rng(num)-rng(num)=0
}
```

Trailingzeroes Of Facturial Number

Code in C++:

```
int findTrailingZeros(int n){
    int count = 0;
    for (int i = 5; n / i >= 1; i *= 5)
        count += n / i;
    return count;
}
```

String Algorithm

Aho Corasick

Code in C++:

```cpp
///Time Complexity: O(n + l + z), where ‘n’ is the length of the text, ‘l’(sum of all ptrns len) is the length of keywords, and ‘z’ is the number of matches.
const int MX_P = 100;/// maximum number of patterns
struct AhoCorasick{
    int nod_no,ptrn_no;
    const int root = 0;
    vector<vector<int>>next;
    vector<int>link;///suffix link/failure link
    vector<bitset<MX_P>>output;///bitset points which which patterns output indicated by this state
    bitset<MX_P>zero;/// zero
    vector<int>occr;

    AhoCorasick(): nod_no(0),ptrn_no(0){node();}

    int node(){
        next.emplace_back(26,0);
        link.emplace_back(root);/// all link initilize by root;
        output.emplace_back(zero);/// each node initilize by 0 set bit
        occr.emplace_back(0);///each pattern occraance initilize by zero
        return nod_no++;/// increase node count
    }
  void add_pattern(const string &s){///trie building
        int currentState=root;
        for(auto c : s){
            int ch=c-'a';
            if(!next[currentState][ch])
                next[currentState][ch]=node();/// node()=create a new node in this state and also next[currentState][ch] set with a state number
            currentState=next[currentState][ch];
        }
        output[currentState][ptrn_no]=1;/// this states end point of  prth_no th pattern
        //output[currentState].set(patn_no,1);
        ptrn_no++;//increse pattern count

    }
    void build_Automaton(){
        queue<int>Q;
        for(int ch=0;ch<26;ch++){
            if(next[root][ch]){
                int stat_lvl1=next[root][ch];/// stat_lvl1=state which connect with root
                link[stat_lvl1]=root;///make level 1 states failure link with root
                Q.push(stat_lvl1);
            }
        }
        while(Q.size()){
            int currentState=Q.front();Q.pop();
            for(int ch =0;ch<26;ch++){
                if(next[currentState][ch]){
                    int child_state=next[currentState][ch];
                    int failure=link[currentState];

                    while(failure!=root && !next[failure][ch])///finding failure node
                        failure=link[failure];
                    failure=next[failure][ch];

                    link[child_state]=failure;
                    output[child_state]|=output[failure];///a state also indicate failure_states all outputs
                    Q.push(child_state);
                }
            }
        }
    }
    int find_NextState(int currentState,int ch){
        while(currentState!=root && !next[currentState][ch])
            currentState=link[currentState];

        return currentState=next[currentState][ch];
    }
    void searchWords(string pattern[],string &text){
        int currentState=root;
        for(int i=0;i<text.size();i++){
            int ch=text[i]-'a';
            currentState=find_NextState(currentState,ch);
            if(output[currentState].any()){// chacking this state point any output
                for(int j=0;j<ptrn_no;j++){
                    if(output[currentState][j]){// if i'th bit is on
                        cout<<pattern[j]<<" appears from "<<i-pattern[j].size()+1<<" to "<<i<<"\n";
                        occr[j]++;/// increse j'th patterns occarence
                    }
                }
            }
        }
    }
};

int main(){
    int n;cin>>n;
    string pattern[n+1];
    string text;
    AhoCorasick aho;
    for(int i=0;i<n;i++){
        cin>>pattern[i];
        aho.add_pattern(pattern[i]);
    }
    cin>>text;
    aho.build_Automaton();
    aho.searchWords(pattern,text);
    for(int i=0;i<n;i++){
        cout<<pattern[i]<<" occurs "<<aho.occr[i]<<" times\n";
    }
}
```

Double Hashing

Code in C++:

```cpp
const int MAX = 1e6 + 10;// string max size
const ll MOD1 = 1e9 + 7;
const ll MOD2 = 1e9 + 9;
const ll base1 = 269;//31,//53
const ll base2 = 277;//31,//53
pair<ll,ll> pw[MAX], inv_pw[MAX];
void pow_clc(){
    ll rev_base1=BIGMOD(base1,MOD1-2,MOD1);///base1^-1
    ll rev_base2=BIGMOD(base2,MOD2-2,MOD2);///base2^-1
    pw[0]={1,1};
    inv_pw[0]={1,1};
    for(int i=1;i<MAX;i++){
        pw[i].F = 1LL * pw[i-1].F * base1 % MOD1;
        inv_pw[i].F = 1LL * inv_pw[i-1].F * rev_base1 % MOD1;

        pw[i].S = 1LL * pw[i-1].S * base2 % MOD2;
        inv_pw[i].S = 1LL * inv_pw[i-1].S * rev_base2 % MOD2;
    }
}
ll compute_prehash(string const &s){///O(string size)
    pair<ll,ll> hash_value={0,0};
    for(int i=0;i<s.size();i++){
        hash_value.F = (hash_value.F + (s[i]*pw[i].F)%MOD1)%MOD1;
        hash_value.S = (hash_value.S + (s[i]*pw[i].S)%MOD2)%MOD2;
    }return (hash_value.F*MOD2 + hash_value.S);
}
vector<pair<ll,ll>>prehsh,sufhsh;
int len;
void hashing(string const &s){///make a hash array in O(string size)
    len=s.size();
    prehsh.resize(len+4);
    sufhsh.resize(len+4);
    for(int i=0;i<len;i++){
        prehsh[i].F= (1LL*s[i]*pw[i].F) %MOD1;
        prehsh[i].S= (1LL*s[i]*pw[i].S) %MOD2;
        if(i){
            prehsh[i].F= (prehsh[i].F + prehsh[i-1].F)
             ↪  %MOD1;
            prehsh[i].S= (prehsh[i].S + prehsh[i-1].S)
             ↪  %MOD2;
        }
        sufhsh[i].F= (1LL*s[i]*pw[len-i-1].F) %MOD1;
        sufhsh[i].S= (1LL*s[i]*pw[len-i-1].S) %MOD2;
        if(i){
            sufhsh[i].F= (sufhsh[i].F + sufhsh[i-1].F)
             ↪  %MOD1;
            sufhsh[i].S= (sufhsh[i].S + sufhsh[i-1].S)
             ↪  %MOD2;
        }
    }
}
ll substring_hash(int i,int j){///O(1)
    assert(i<=j);
    pair<ll,ll>hs({0,0});
    hs.F=prehsh[j].F;
    hs.S=prehsh[j].S;
    if(i){
        hs.F=(hs.F- prehsh[i-1].F +MOD1)%MOD1;
        hs.S=(hs.S- prehsh[i-1].S +MOD2)%MOD2;
    }
    hs.F= (1LL* hs.F * inv_pw[i].F)%MOD1;
    hs.S= (1LL* hs.S * inv_pw[i].S)%MOD2;
    return (hs.F*MOD2 + hs.S);
}
ll GetPrefixHash(int i,int j){
    return substring_hash(i, j);
}
ll GetSuffixHash(int i,int j){
    assert(i<=j);
    pair<ll,ll>hs({0,0});
    hs.F=sufhsh[j].F;
    hs.S=sufhsh[j].S;
    if(i){
        hs.F=(hs.F- sufhsh[i-1].F +MOD1)%MOD1;
        hs.S=(hs.S- sufhsh[i-1].S +MOD2)%MOD2;
    }
    hs.F= (1LL* hs.F * inv_pw[len-j-1].F)%MOD1;
    hs.S= (1LL* hs.S * inv_pw[len-j-1].S)%MOD2;
    return (hs.F*MOD2 + hs.S);
}
bool IsPallindrome(int l , int r) {
    return (GetPrefixHash(l , r) == GetSuffixHash(l , r));
}
void string_matching(string const &txt,string const
 ↪  &pat){///O(N)///Rabin Karp
    hashing(txt);
    ll pat_hsh=compute_prehash(pat);
    int substr_len=pat.size();
    vector<int>idx;
    for(int i=0;i+substr_len-1<txt.size();i++){
        ll substr_hsh=substring_hash(i,i+substr_len-1);
        if(substr_hsh==pat_hsh)idx.push_back(i+1);
    }
    if(idx.size()){
        cout<<"pattern found at index : ";
        for(auto it: idx)cout<<it<<" ";
        cout<<"\n";
    }else{
        cout<<"pattern not found\n";
    }
}
int main() {
    pow_clc();
    string txt,pat;
    while(cin>>txt>>pat){
        hashing(txt);
        string_matching(txt,pat);
    }
    return 0;
}
```

Kmp

Code in C++:

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

}
void KMP(string &txt,string &pat){
    vector<int>failure_idx=failure_function(pat);
    int j=0;
    for(int i=0;i<txt.size();i++){
        while(j>0 && txt[i]!=pat[j])j=failure_idx[j-1];
        if(txt[i]==pat[j]){
            j++;
            if(j==pat.size()){
                cout<<i-pat.size()+1<<" ";
                j=failure_idx[j-1];

            }
        }
    }

}
```

Trie Tree

Code in C++:

```
struct node{
    bool End;
    vector<node*>next;
    node(){
        End=false;
        next.resize(26,nullptr);
    }
};
class Trie {
public:
    node* root;
    Trie() {
        root=new node();
    }
    void insert(string &word) {
        node*cur=root;
        for(auto ch : word){
            if(!cur->next[ch-'a'])cur->next[ch-'a']=new
             ↪  node();
            cur=cur->next[ch-'a'];
        }
        cur->End=true;
    }
    bool search(string &word) {
        node*cur=root;
        for(auto ch : word){
            if(!cur->next[ch-'a'])return false;
            cur=cur->next[ch-'a'];
        }
        return cur->End;
    }
    bool startsWith(string &prefix) {
        node*cur=root;
        for(auto ch : prefix){
            if(!cur->next[ch-'a'])return false;
            cur=cur->next[ch-'a'];
        }
        return true;
    }
};
```

Hashing Longest Common Prefix

Code in C++:

```
void lcp(int i1,int j1,int i2,int j2){
     int l=1,r=min(j1-i1+1,j2-i2+1);/// minimum length of
      ↪  two string
     int ans=0;
     while(l<=r){
        int mid =l+r >>1;
        if(sub_hash(i1,i1+mid-1,txthsh)==sub_hash(i2,i2+
         ↪  mid-1,pathsh)){
            ans=mid;
            l=mid+1;
        }else{
            r=mid-1;
        }
    }
    cout<<ans<<"\n";
    cout<<txt.substr(i1,ans)<<"\n";
}
```

Hashing String Divisor

Code in C++:

```
void string_divisors(string const &s){/// nlog(n)
    hashing(s);
    int n=s.size();
    for(int len=1;len<=n;len++){
        bool ok=true;
        for(int i=0;i+len-1<n;i+=len){
            ok &= sub_hash(i,i+len-1)==sub_hash(0,len-1);

            if(i+len+len-1>=n && i+len<=n-1){///partial
             ↪  maching
                ok &= sub_hash(i+len,n-1)==sub_hash(0,n-1
                 ↪  -i-len);
            }
        }
        if(ok==true){
            cout<<s.substr(0,len)<<"\n";
        }
    }
}
```

Backtracking

Combination

Code in C++:

```
int sto[20];
int N, R;
void bt(int start, int depth){
    if(depth == R){
        for(int i=0;i<R;i++)
        cout<<sto[i]<<" ";
        cout<<endl;
        return;
    }
    for(int i = start;i<=N;i++){
        sto[depth] = i;
        bt(i+1, depth+1);
    }
}
int main(){
    while(cin>>N>>R){
        bt(1, 0);
    }
}
```

Nqueen

Code in C++:

```
int n;
vector<vector<string>>boards;
bool UnderAttack(int row,int col,vector<string>&board){
    int duprow = row;
    int dupcol = col;
    while(row>=0 && col>=0){
        if(board[row][col]=='Q')
            return true;
        --row,--col;
    }
    col = dupcol;
    row = duprow;
    while(row<n && col>=0){
        if(board[row][col]=='Q')
            return true;
        ++row,--col;
    }
    row = duprow;
    col = dupcol;
    while(col>=0){
        if(board[row][col]=='Q')
            return true;
        --col;
    }
    return false;
}
void res(int col,vector<string>&board){
    if(col==n){
        boards.push_back(board);
        return;
    }
    for(int row=0;row<n;row++){
        if(UnderAttack(row,col,board))
            continue;
        board[row][col]='Q';
        res(col+1,board);
        board[row][col]='.';
    }
}
int main(){
    cin>>n;
    vector<string>board(n,string(n,'.'));
    res(0,board);
    int w=0;
    for(auto ans : boards){
        cout<<"Possible Way :"<<++w<<el;
        for(auto r : ans){
            cout<<r<<el;
        }
        cout<<el;
    }
    board.clear(),boards.clear();

}
```

Permutation

Code in C++:

```
int n,r;
int color[S],sto[S];
void go(int depth){
    if(depth==r){
        for(int i=0;i<r;i++){
            printf("%d ",sto[i]);
        }
        printf("\n");
        return;
    }
    for(int i=1;i<=n;i++){
        if(color[i]==false){
            color[i]=true;
            sto[depth]=i;
            go(depth+1);
            color[i]=false;
        }
    }
}
int main(){
    while(cin>>n>>r){
        memset(color, false, sizeof color);
        go(0);
    }
}
```

Dynamic Programming

1 Knapsack

Code in C++:

```
int knapSack(int i, int sto) {
   if (sto < 0)return INT_MIN;
   if (i < 0 || sto == 0)return 0;

   if(memo[i][sto]!=-1)return memo[i][sto];

   int in = val[i]+knapSack(i-1, sto-wt[i]);
   int ex = knapSack(i-1, sto);
   return memo[i][sto]=max(in, ex);
}
void path(int i,int sto){
    if(sto<0)return;
    if(i<0 || sto==0) return;
    int in=memo[i-1][sto-wt[i]]+val[i];
    int ex=memo[i-1][sto];
    if(in>ex){
        v.push_back(val[i]);
        path(i-1,sto-wt[i]);
    }
    else{
        path(i-1,sto);
    }
}
```

Edit Distance

Code in C++:

```
int EditDistance(int i,int j){
    if(i<0)return j+1;
    if(j<0)return i+1;

    if(memo[i][j]!=-1)return memo[i][j];

    if(s1[i]==s2[j])
        return memo[i][j]= EditDistance(i-1,j-1);

    /// any move cost 1
    int Insert = 1 + EditDistance(i,j-1);
    int Delete = 1 + EditDistance(i-1,j);
    int Remove = 1 + EditDistance(i-1,j-1);
    return memo[i][j]=min({Insert,Delete,Remove});
}
i
```

Knight Tour

Code in C++:

```
typedef struct{int x,y;}co;
int dis[1006][1006],n;
int dp[1<<17][17];
int N,k;
co ic[20];
void BFS(){
    queue<co>q;
    int r,c,ur,uc,
    dr[] = {2,2,1,1,-1,-1,-2,-2},
    dc[] = {1,-1,2,-2,2,-2,1,-1};
    memset(dis,-1,sizeof(dis));
    dis[2][2] = 0;
    q.push({2,2});

    while(!q.empty())
    {
        ur = q.front().x;
        uc = q.front().y;
        q.pop();
        for(int i=0;i<8;i++){
            r = ur+ dr[i];  c = uc+ dc[i];
            if(r>=0 && c>=0 && r<=1002 && c<=1002 &&
             ↪ dis[r][c]==-1){
                q.push({r,c});
                dis[r][c] = dis[ur][uc]+1;
            }
        }
    }
}
int DIS(co a,co b){
    if(
        (a.x==1 && a.y==1 && b.x==2 && b.y==2) ||
        (a.x==2 && a.y==2 && b.x==1 && b.y==1) ||
        (a.x==n-1 && a.y==n-1 && b.x==n && b.y==n) ||
        (a.x==n && a.y==n && b.x==n-1 && b.y==n-1) ||
        (a.x==1 && a.y==n && b.x==2 && b.y==n-1) ||
        (a.x==2 && a.y==n-1 && b.x==1 && b.y==n) ||
        (a.x==n && a.y==1 && b.x==n-1 && b.y==2) ||
        (a.x==n-1 && a.y==2 && b.x==n && b.y==1)
    )
    return 4;
    return dis[abs(a.x-b.x)+2][abs(a.y-b.y)+2];
}
int go(int msk,int cur){
    if(msk== ((1<<k)-1))return DIS(ic[cur],ic[0]);
    int &rf=dp[msk][cur];
    if(rf!=-1)return rf;
    rf=1<<30;
    for(int i=0;i<k;i++){
        if((msk&(1<<i))==0){
            rf=min(rf,go((msk|(1<<i)),i)+DIS(ic[cur],ic[i
             ↪ ]));
        }
    }
    return rf;
}
int main(){
    co s,a,b;int t,ks=0;
    BFS();//pre
    cin>>t;
    while(t--){
        memset(dp,-1,sizeof dp);
        cin>>n>>k;
        for(int i=0;i<k;i++){
            cin>>ic[i].x>>ic[i].y;
        }
        cout<<go(0,0)<<"\n";
    }
}
```

Lcs

Code in C++:

```
int lcs(int t,int p){
    if(t==n || p==m){
        return 0;
    }
    if(memo[t][p]!=-1)return memo[t][p];
    if(txt[t]==pat[p]){
        memo[t][p]=1+lcs(t+1,p+1);
    }
    else{
        memo[t][p]=max(lcs(t+1,p),lcs(t,p+1));
    }
    return memo[t][p];
}
void path(int t,int p){
    if(t==n || p==m)return;
    if(txt[t]==pat[p]){
        ans+=txt[t];
        path(t+1,p+1);
    }
    else if(memo[t+1][p]>memo[t][p+1]){
        path(t+1,p);
    }
    else {
        path(t,p+1);
    }
}
```

Lcs Lexicography Minimum String

Code in C++:

```
string dp[105][105];
bool vis[105][105];
string  lcs(int i,int j){
    if(i==n || j==m)return "";
    if(vis[i][j])return dp[i][j];

    vis[i][j]=true;
    string ans="";
    if(txt[i]==pat[j]){
        ans=txt[i]+lcs(i+1,j+1);
    }else{
```

```cpp
        string a=lcs(i+1,j);
        string b=lcs(i,j+1);
        if(a.size()>b.size()){
            ans=a;
        }else if(a.size()<b.size()){
            ans=b;
        }else{
            ans=min(a,b);
        }
    }
    dp[i][j]=ans;
    return dp[i][j];
}
```

Lis Lower Bound

Code in C++:

```cpp
int LIS(vector<int>&a){
    vector<int>v;
    int ans=0;
    for(auto x : a){
        auto it =lower_bound(v.begin(),v.end(),x);
        ans=max(ans,(int)(it-v.begin()+1));
        if(it==v.end()){
            v.push_back(x);
        }else{
            *it=x;
        }
    }
    return ans;
}
```

Lis Using Segment Tree

Code in C++:

```cpp
void solve(){
    cin>>n;
    int ar[n+2];
    for(int i=0;i<n;i++){
        cin>>ar[i];
    }

    // segmentree not working above 1e6 // needs to
     ↪ compress values
    // coordinate compression, now all a[i] are 1 <= a[i]
     ↪ <= n which is cute
    /**
        set<int>s;
        for(int i=0;i<n;i++)s.insert(ar[i]);
        int id=0;
        map<int,int>mp;
        for(auto it : s)mp[it]=++id;
        for(int i=0;i<n;i++)ar[i]=mp[ar[i]];
    */
    int max_value=n;// after compression max value will be
     ↪ n or id
    // segment tree on value
    // MAX Segment Tree
    st.build(1,1,max_value);
    vector<int>dp(n+2,1);
    int mx=1;
    for(int i=0;i<n;i++){
        // for(int j=0;j<i;j++){
        //     if(ar[j]<ar[i]){
        //         dp[i]=max(dp[i],1+dp[j]);
        //     }
        // }
        // mx=max(dp[i],mx);
        dp[i]=max(dp[i],st.query(1,1,max_value,1,ar[i]-1)⌋
         ↪ +1);// MAX Segment
         ↪ Tree
        st.update(1,1,max_value,ar[i],dp[i]);// MAX
         ↪ Segment Tree update
        mx=max(mx,dp[i]);
    }
    cout<<mx<<"\n";

}
```

Mcm Burst Balloons

Code in C++:

```cpp
int go(int i,int j){
    if(i>j)return 0;
    int &rf=dp[i][j];
    if(rf!=-1)return rf;
    rf=-9999999;
    for(int idx=i;idx<=j;idx++){
        rf=max(rf,ar[i-1]*ar[idx]*ar[j+1]+go(i,idx-1)+go(⌋
         ↪ idx+1,j));
    }
    return rf;
}
```

Coin Change Distinct Way

Code in C++:

```cpp
memset(dp,0,sizeof dp);
dp[0]=1;
for(int i=0;i<n;i++){
    for(int j=0;j<=s;j++){
        if(j-c[i]>=0){
            dp[j]=dp[j]+dp[j-c[i]]%MOD;
        }
    }
}
cout<<dp[s]%MOD<<endl;
```

Coin Change Use One More Time Arbitrary Order

Code in C++:

```cpp
int way(int W) {
    if (W < 0) return INF;
    if (W == 0) return 0;

    int &rf=memo[W];
    if (rf != -1) {
        return rf;
    }
    rf = INF;
    for (int i = 0;i < NC;i++) {
        rf = min(rf, 1 + way(W - C[i]));
    }
    return rf;
}
```

Longest Common Substring

Code in C++:

```cpp
int lcs(string &s1, string &s2)
{
    int l1=s1.size(),l2=s2.size();
    vector<vector<int>>dp(l1+3,vector<int>(l2+3,0));
    int ans=0;
    for(int i=l1-1;i>=0;i--){
        for(int j=l2-1;j>=0;j--){
            if(s1[i]==s2[j]){
                dp[i][j]=1+dp[i+1][j+1];
            }else{
                dp[i][j]=0;
            }
            ans=max(ans,dp[i][j]);
        }
    }
    return  ans;
}
```

Number Of Palindrome In Range Query

Code in C++:

```cpp
string s;int q;
int ispal[5010][5010];
int IsPallindrome(int l , int r) {
    if(l>r)return 1;
    if(ispal[l][r]!=-1)return ispal[l][r];
    return ispal[l][r]= s[l]==s[r] &
    ↪  IsPallindrome(l+1,r-1);
}
int dp[5010][5010];
int num_of_palin(int l,int r){
    if(l>r)return 0;
    if(dp[l][r]!=-1)return dp[l][r];
    return dp[l][r]=IsPallindrome(l,r)+
        num_of_palin(l+1,r)+num_of_palin(l,r-1)-num_of_pa⌟
        ↪  lin(l+1,r-1);
}
int main() {
    cin>>s>>q;
    memset(dp,-1,sizeof dp);
    memset(ispal,-1,sizeof ispal);
    while(q--){
        int l,r;
        cin>>l>>r;
        l--,r--;
        cout<<num_of_palin(l,r)<<"\n";
    }
}
```

Two Player Optimal Moves

Code in C++:

```
ll dp1[3005][3005];
bool vis1[3005][3005];
ll dp2[3005][3005];
bool vis2[3005][3005];
ll  ar[3005];
ll first_player_optimal(int i,int j);///function prototype
ll second_player_optimal(int i,int j){

    if(i>j)return 0;

    if(vis2[i][j])return dp2[i][j];
    vis2[i][j]=true;

    ll left =first_player_optimal(i+1 ,j)-ar[i];
    ll right =first_player_optimal(i ,j-1)-ar[j];
    return dp2[i][j] = min(left,right);
}

ll first_player_optimal(int i,int j){

    if(i>j)return 0;

    if(vis1[i][j])return dp1[i][j];
    vis1[i][j]=true;

    ll left =second_player_optimal(i+1,j)+ar[i];
    ll right =second_player_optimal(i,j-1)+ar[j];
    return dp1[i][j] = max(left,right);
}
```

Digit DP

Code in C++:

```
ll dp[10][10][2][2];
ll go(ll pos,ll st,ll ever_sm,ll val,int len,string &s){
    if(pos==len)return val;
    ll &rf=dp[pos][val][ever_sm][st];
    if(rf!=-1)return rf;
    rf=0;
    if(st){
        if(ever_sm){
            for(ll i=0;i<=9;i++){
                rf+=go(pos+1,1,1,val+(i==0),len,s);
            }
        }else{
            for(ll i=0;i<=s[pos]-'0';i++){
                rf+=go(pos+1,1,ever_sm|(i<(s[pos]-'0')),v⌟
                ↪  al+(i==0),len,s);
            }
        }
    }else{
        rf+=go(pos+1,0,1,0,len,s);
        if(pos==0){
            for(ll i=1;i<=s[0]-'0';i++){
                rf+=go(pos+1,1
                ↪  ,ever_sm|(i<(s[pos]-'0')),0,len,s);
            }
        }else{
            for(ll i=1;i<=9;i++){
                rf+=go(pos+1,1 ,1,0,len,s);
            }
        }
    }
    return rf;
}
```

Geometry

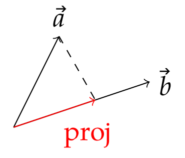
Formulas

Vector Algebra

Magnitude / Length: $|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$
ভেক্টরের দৈর্ঘ্য বা মান (Magnitude) নির্ণয় করতে এটি ব্যবহৃত হয়।

Dot Product: $\vec{a} \cdot \vec{b} = a_x b_x + a_y b_y + a_z b_z = |\vec{a}||\vec{b}| \cos\theta$
দুটি ভেক্টর লম্ব (Perpendicular) কিনা তা চেক করতে ($\vec{a} \cdot \vec{b} = 0$) অথবা তাদের মধ্যবর্তী কোণ θ বের করতে ব্যবহৃত হয়।

Angle Between Vectors: $\cos\theta = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|}$



Projection

Projection of $\vec{a}$ onto $\vec{b}$:
$\text{proj}_{\vec{b}}\vec{a} = \frac{\vec{a}\cdot\vec{b}}{|\vec{b}|^2}\vec{b}$
ভেক্টর $\vec{a}$ এর ছায়া বা উপাংশ ভেক্টর $\vec{b}$ এর ওপর কতটুকু তা বের করার জন্য।

Cross Product: $\vec{a} \times \vec{b} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$
দুটি ভেক্টরের লম্ব ভেক্টর (Normal Vector) পেতে অথবা তাদের দ্বারা গঠিত সামান্তরিকের ক্ষেত্রফল (Area of Parallelogram) এবং ত্রিভুজের ক্ষেত্রফল ($0.5 \times |\vec{a} \times \vec{b}|$) বের করতে।

Coordinate Geometry Rotation

একটি বিন্দু (x, y) কে মূলবিন্দুর (Origin) সাপেক্ষে θ কোণে ঘড়ির কাঁটার বিপরীতে (Counter-Clockwise) ঘোরাতে:

2D / Rotation around Z-axis:
$x' = x\cos\theta - y\sin\theta, \quad y' = x\sin\theta + y\cos\theta$

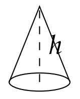
Rotation around X-axis (x remains fixed):
$y' = y\cos\theta - z\sin\theta, \quad z' = y\sin\theta + z\cos\theta$

Rotation around Y-axis (y remains fixed):
$x' = x\cos\theta + z\sin\theta, \quad z' = -x\sin\theta + z\cos\theta$

Rodrigues' Formula: (Rotation of vector $\vec{v}$ by angle θ around axis unit vector $\vec{u}$)
$\vec{v}_{rot} = \vec{v}\cos\theta + (\vec{u} \times \vec{v})\sin\theta + \vec{u}(\vec{u} \cdot \vec{v})(1 - \cos\theta)$
যেকোনো 3D ভেক্টরকে যেকোনো অক্ষ (Axis) সাপেক্ষে ঘোরাতে এটি সবচেয়ে পাওয়ারফুল সূত্র।

Solid Geometry

Sphere (□□□□):
Volume $= \frac{4}{3}\pi r^3$
Surface Area $= 4\pi r^2$



Cone

Cone (□□□□):
Volume $= \frac{1}{3}\pi r^2 h$
Surface Area $= \pi r(r + \sqrt{h^2 + r^2})$

Pyramid:
Volume $= \frac{1}{3} \times$ Base Area $\times$ Height
পিরামিডের আয়তন বের করতে ভূমির ক্ষেত্রফল জানা থাকতে হবে।

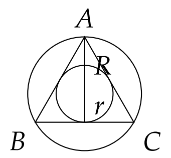
Triangle Properties

Equilateral Triangle Area: $\frac{\sqrt{3}}{4}a^2$

Inradius (r): $r = \frac{\Delta}{s}$ (Δ = ত্রিভুজের এরিয়া, s = অর্ধপরিসীমা)
Circumradius (R): $R = \frac{abc}{4\Delta}$ (a, b, c হলো তিন বাহু)

Sine Rule: $\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C} = 2R$
ত্রিভুজের বাহু এবং বিপরীত কোণের সম্পর্ক। পরিবৃত্তের ব্যাসার্ধ বের করতেও লাগে।

Cosine Rule: $\cos A = \frac{b^2+c^2-a^2}{2bc}$
তিনটি বাহু জানা থাকলে কোণ বের করতে, অথবা দুটি বাহু ও অন্তর্ভুক্ত কোণ জানা থাকলে তৃতীয় বাহু বের করতে।



Number Theory

Divisors of $N = p_1^{a_1} p_2^{a_2} \dots$ **:**
Count $= (a_1+1)(a_2+1)\dots$
Sum $= \prod \frac{p_i^{a_i+1}-1}{p_i-1}$

Logarithm: $\log_b x = k \iff b^k = x$
Number of Digits in Base $b = \lfloor \log_b(N) \rfloor + 1$
যেকোনো সংখ্যার ডিজিট সংখ্যা বের করার শর্টকাট।

Linear Algebra (Cramer's Rule)

দুই চলক বিশিষ্ট সরল সমীকরণ সমাধান করতে:
System: $ax + by = e,\ cx + dy = f$

$$D = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

$$D_x = \begin{vmatrix} e & b \\ f & d \end{vmatrix} = ed - bf, \quad D_y = \begin{vmatrix} a & e \\ c & f \end{vmatrix} = af - ce$$

Answer: $x = \frac{D_x}{D}, \quad y = \frac{D_y}{D} \quad$ (Valid if $D \neq 0$)

Convex Hull

Code in C++:

```
struct P{
    double x,y;
};
P pvt;
P vec(P a,P b){return {b.x-a.x,b.y-a.y};}
double Cross(P a,P b){return a.x*b.y-a.y*b.x;}
double eDis(P a,P b){return
 ↪  sqrt(sqr(a.x-b.x)+sqr(a.y-b.y));}
bool comp(P a,P b)
{
    double c = Cross(vec(pvt,a),vec(pvt,b));

    if(c) return c>0;
    return eDis(pvt,a)<eDis(pvt,b);

}
vector<P> makeConvexHull(vector<P> p){
    int nConvex=p.size(),i,j;
    if(nConvex<2) return p;
/**     IF ALL POINTS ARE CO-LINER   */
//    j=0;
//    for(i=2;i<nConvex;i++)
//        if(Cross(vec(p[0],p[1]),vec(p[0],p[i]))==0)
//            j++;
//    if(j+2==nConvex)  return P
    pvt=p[0];
    for(int i=1;i<nConvex;i++)
        if(pvt.y>p[i].y)    pvt=p[i];
        else if(pvt.y==p[i].y && pvt.x>p[i].x)
         ↪  pvt=p[i];

    sort(all(p),comp);
    j=2;
    for(int i=2;i<nConvex;i++)
    {
        //while(j>1 &&
         ↪  Cross(vec(p[j-2],p[j-1]),vec(p[j-2],p[i]))<=0)
         ↪  skip same line
        while(j>1 &&
         ↪  Cross(vec(p[j-2],p[j-1]),vec(p[j-2],p[i]))<0)
         ↪  j--;
        p[j++]=p[i];
    }
    p.resize(j);
    return p;
}
void solve(int n) {
    vector<P>v;
    for(int i=0;i<n;i++){
        double x,y;cin>>x>>y;
        v.push_back({x,y});
    }
    double Perimeter=0;
    if(n==1){
        cout<<"("<<sp(v[0].x,1)<<","<<sp(v[0].y,1)<<")\n";
        cout<<"Perimeter length =
         ↪  "<<sp(Perimeter,2)<<"\n";
        return;
    }
    v=makeConvexHull(v);
    reverse(all(v));
    for(int i=0;i<v.size();i++){
        cout<<"("<<sp(v[i].x,1)<<","<<sp(v[i].y,1)<<")-";
        Perimeter+=(eDis(v[i],v[(i+1)%v.size()]));
    }
    cout<<"("<<sp(v[0].x,1)<<","<<sp(v[0].y,1)<<")\n";
    cout<<"Perimeter length = "<<sp(Perimeter,2)<<"\n";
}
```

Segment Intersect

Code in C++:

```
struct P{
    double x,y;
};
P vec2d(P &a,P &b){
    return{b.x - a.x, b.y - a.y};
}
double cross(P &A, P &B) {
    return A.x * B.y - A.y * B.x;
}
bool onSegment(P &a,P &b, P &p) {
    return min(a.x,b.x)-1e-9<=p.x&&p.x<=max(a.x,b.x)+1e-9
        && min(a.y,b.y)-1e-9<=p.y&&p.y<=max(a.y,b.y)+1e-9;
}
P getIntersectionPoint(P a, P b, P c, P d) {
    P AB = vec2d(a, b);
    P CD = vec2d(c, d);
    P AC = vec2d(a, c);
    double det = cross(AB, CD);
    double t = cross(AC, CD) / det;
    return {a.x + t * AB.x, a.y + t * AB.y};
}
bool segmentIntersect(P &a,P &b, P &c, P&d){
    P AB=vec2d(a,b);
    P AC=vec2d(a,c);
    P AD=vec2d(a,d);
    P CD=vec2d(c,d);
    P CB=vec2d(c,b);
    P CA=vec2d(c,a);

    if(cross(AB,AC)*cross(AB,AD)<0 &&
     ↪  cross(CD,CB)*cross(CD,CA)<0)
        return true;
    if(cross(AB,AC)==0 && onSegment(a,b,c))return true;
    if(cross(AB,AD)==0 && onSegment(a,b,d))return true;
    if(cross(CD,CB)==0 && onSegment(c,d,b))return true;
    if(cross(CD,CA)==0 && onSegment(c,d,a))return true;
    return false;

}
void solve() {
    P a,b,c,d;
    cin>>a.x>>a.y>>b.x>>b.y>>c.x>>c.y>>d.x>>d.y;

    if(segmentIntersect(a,b,c,d)){
        YES
        // Special case: check if lines are collinear
        P AB = vec2d(a, b);
        P CD = vec2d(c, d);
        if(abs(cross(AB, CD))<1e-9){
            cout<<"Segments are collinear"<<endl;
        } else {
            P it = getIntersectionPoint(a, b, c, d);
            cout << "Point: " << it.x << " " << it.y <<
             ↪  endl;
        }
    }else{
        NO
    }


}
```

Segment Intersect 3D

Code in C++:

```
const double EPS = 1e-9;
struct P {
    double x, y, z;
};
P vec3d(const P &a, const P &b) {
    return {b.x - a.x, b.y - a.y, b.z - a.z};
}
P cross(P A, P B) {
    return {
        A.y * B.z - A.z * B.y,
        A.z * B.x - A.x * B.z,
        A.x * B.y - A.y * B.x
    };
}
double dot(P A, P B) {
    return A.x * B.x + A.y * B.y + A.z * B.z;
}
double magSq(P A) {
    return dot(A, A);
}
bool onSegment(const P &a, const P &b, const P &p){
    return p.x>=min(a.x,b.x)-EPS&&p.x<=max(a.x,b.x)+EPS&&
           p.y>=min(a.y,b.y)-EPS&&p.y<=max(a.y,b.y)+EPS&&
           p.z>=min(a.z,b.z)-EPS&&p.z<=max(a.z,b.z)+EPS;
}
P getIntersectionPoint(P a, P b, P c, P d){
    P AB = vec3d(a, b);
    P CD = vec3d(c, d);
    P AC = vec3d(a, c);

    double t=dot(cross(AC,CD),cross(AB,CD))/dot(cross(AB,
     ↪  CD),cross(AB,CD));
    return {a.x + t * AB.x, a.y + t * AB.y, a.z + t *
     ↪  AB.z};
}
bool segmentIntersect(const P &a, const P &b, const P &c,
 ↪  const P &d) {
    P AB = vec3d(a, b);
    P CD = vec3d(c, d);
    P AC = vec3d(a, c);

    P cp = cross(AB, CD);
    double cp_mag2 = magSq(cp);

    // Case 1: Parallel or Collinear Lines
    if (cp_mag2 < 1e-18) {
        // Check if they lie on the same infinite line by
         ↪  checking if AC is parallel to AB
        if (magSq(cross(AC, AB)) > 1e-18) return false;

        // If they are on the same line, check if they
         ↪  overlap
        return onSegment(a, b, c) || onSegment(a, b, d)
         ↪  ||
               onSegment(c, d, a) || onSegment(c, d, b);
    }

    // Case 2: Skew Lines Check (Coplanarity)
    // In 3D, lines must be in the same plane to
     ↪  intersect.
    // The volume of the parallelepiped formed by AC, AB,
     ↪  and CD must be 0.
    if (abs(dot(AC, cp)) > EPS) return false;

    // Case 3: Calculate point and check if it is within
     ↪  both segments
    P intersect = getIntersectionPoint(a, b, c, d);
    return onSegment(a, b, intersect) && onSegment(c, d,
     ↪  intersect);
}
void solve() {
    P a, b, c, d;
    cin >> a.x >> a.y >> a.z >> b.x >> b.y >> b.z
        >> c.x >> c.y >> c.z >> d.x >> d.y >> d.z;
    if (segmentIntersect(a, b, c, d)) {
        cout << "YES" << endl;
        P AB = vec3d(a, b);
        P CD = vec3d(c, d);
        if (magSq(cross(AB, CD)) < 1e-18) {
            cout << "Segments are collinear and
             ↪  overlapping." << endl;
        } else {
            P res = getIntersectionPoint(a, b, c, d);
            cout << fixed << setprecision(10) << "Point:
             ↪  "
                 << res.x << " " << res.y << " " << res.z
                  ↪  << endl;
        }
    } else {
        cout << "NO" << endl;
    }
}
```

Tree Algos

Diameter Of A Tree Dfs

Code in C++:

```
int depth[Size];
vector<int>graph[Size];
int max_depth;
int max_depth_node;
void init(int V){
    for(int i=0;i<V+5;i++){
        graph[i].clear();depth[i]=0;
    }
    max_depth=0;
}
int dfs(int u,int par=-1){
    if(depth[u]>max_depth){
        max_depth=depth[u];
        max_depth_node=u;
    }
    for(auto v : graph[u]){
        if(v==par)continue;
        depth[v]=depth[u]+1;
        dfs(v,u);
    }
    return max_depth_node;
}
int main(){
    int V,u,v;cin>>V;
    init(V);
    int E=V-1;
    for(int i=0;i<E;i++){
        cin>>u>>v;
        graph[u].push_back(v);
        graph[v].push_back(u);
    }
    max_depth_node=dfs(1);///one based
    memset(depth,0,sizeof depth);
    max_depth=0;
    max_depth_node=dfs(max_depth_node);
    cout<<"Diameter of this Tree =
     ↪  "<<depth[max_depth_node]<<"\n";

}
```

Lowest Common Ancestor Sparse Table

Code in C++:

```
///Complexity: O(NlgN,lgN)
int E,V;
int LVL[Size];
int par[Size];
int A[Size][20];
vector<int>adj[Size];
/// finding nodes tree level and parent
void leveling_dfs(int u){
    for(auto v : adj[u]){
        if(v==par[u])continue;
        LVL[v]=LVL[u]+1;
        par[v]=u;
        leveling_dfs(v);
    }
}
void Sparse_Table(){
    for(int p=0;p<=log2(V)+1;p++){
        for(int i=1;i<=V;i++){
            if(p==0)
                A[i][p] = par[i];///2^p = 2^0 = 1'th
                 ↪  parent
            else
                A[i][p] = A[A[i][p-1]][p-1];///  A[i][p] =
                 ↪  i'th nodes 2^p'th parant
        }
    }
}
int LCA(int u,int v){
    if(LVL[u]>LVL[v])
        swap(u,v);
    //Bring u and v in same level
    for(int i=log2(V)+1;i>=0;i--){
        int x = A[v][i];
        if(LVL[u]==LVL[x]){
            v=x;
            break;
        }
        if(LVL[u]<LVL[x])
            v = x;
    }
    if(u==v)return u;
    for(int i=log2(V)+1;i>=0;i--){
        if(A[u][i] != -1 && A[u][i] != A[v][i]){
            u = A[u][i];
            v = A[v][i];
        }
    }
    return par[u];
}
void build_LCA(int source){
    LVL[source]=1,par[source]=source;
    leveling_dfs(source);
    Sparse_Table();
}
int main(){
    ///one based code
    int i,j,u,v,q;
    scanf("%d",&V);
    for(i=0;i<V+2;i++){
        adj[i].clear();
        for(j=0;j<=log2(V+1);j++)
            A[i][j] = -1;
    }
    for(i=1;i<V;i++){
        scanf("%d%d",&u,&v);
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    int source=1;
    build_LCA(source);
    scanf("%d%d",&u,&v);
    printf("%d\n",LCA(u,v));
}
```

Merge Sorttree Number Of Element Greater Then K

Code in C++:

```
vector<ll>ar;
vector<vector<ll>>tree;
vector<ll> marge(vector<ll>&a,vector<ll>&b){
    int n=a.size(),m=b.size();
    vector<ll>c;
    int i=0,j=0;
    while(i<n && j<m){
        if(a[i]<=b[j]){
            c.push_back(a[i]);
            i++;
        }else{
            c.push_back(b[j]);
            j++;
        }
    }
    while(i<n)c.push_back(a[i]),i++;
    while(j<m)c.push_back(b[j]),j++;
    return c;
}
void build(int node,int left,int right){
    if(left==right){
        tree[node].push_back(ar[left]);
        return ;
    }
    int mid=(left+right)/2;
    build(node*2,left,mid);
    build(node*2+1,mid+1,right);
    tree[node]=marge(tree[node*2],tree[node*2+1]);
}
int query(int node,int left,int right,int ql,int qr,ll
 ↪  k){///query left=ql,right=qr
    if(left>=ql && right<=qr){
        int ans= (int)tree[node].size()
        -(upper_bound(tree[node].begin(),tree[node].end()
        ↪  ,k)-tree[node].begin());
        return ans;
    }
    int mid=(left+right)/2;
    if(qr<=mid){
        return query(2*node,left,mid,ql,qr,k);
    }
    else if(mid<ql){
        return query(2*node+1,mid+1,right,ql,qr,k);
    }
    else{
        int left_node=query(2*node,left,mid,ql,mid,k);
        int right_node=query(2*node+1,mid+1,right,mid+1,q
        ↪  r,k);
        return left_node+right_node;
    }
}
int main(){
    int n,ql,qr,pos,new_val;
        cin>>n;
        ar.resize(n+5);
        tree.resize(4*(n+5));
        for(int i=1;i<=n;i++){
            cin>>ar[i];
        }
        build(1,1,n);
        // int q;cin>>q;
        // while(q--){
        //     ll k;
        //     cin>>ql>>qr>>k;
        //     cout<<query(1,1,n,ql,qr,k)<<"\n";

        // }

    return 0;
}
```

Segmenttree Lazypropagation

Code in C++:

```
/// sl = segment left,sr = segment right, tre[nod] contain
 ↪  [sl-sr] range
#define ll long long
const int MAX_N =100007;
const int oo = 2e9+10;
int ar[MAX_N];
struct LazyTree{
    vector<int>tre,lazy;
    LazyTree(int sz){
        tre.assign((sz*4)+10,0);
        lazy.assign((sz*4)+10,0);
    }
    inline void lazyUpdate(int nod,int sl,int sr){
        if(lazy[nod]==0)return;
        //tre[nod] += lazy[nod];// change += or = // if we
         ↪  chaking for max,min
        tre[nod] += lazy[nod]*(sr-sl+1);// change += or =
        if(sl!=sr){
            int left_child = 2*nod , right_child =
             ↪  2*nod+1;
            lazy[left_child] += lazy[nod],
             ↪  lazy[right_child] += lazy[nod];// change
             ↪  += or =
        }
        lazy[nod]=0;
    }
    void build(int nod,int sl,int sr){
        lazy[nod]=0;
        if(sl==sr){
            tre[nod]=ar[sr];// root node
            return;
        }
        int mid = (sl+sr)/2;
        int left_child = 2*nod , right_child = 2*nod+1;
        build(left_child , sl , mid);
        build(right_child , mid+1, sr);
        tre[nod] = tre[left_child] +
         ↪  tre[right_child];///change
    }
    ll query(int nod,int sl,int sr,int ql,int qr){
        lazyUpdate(nod,sl,sr);
        if(ql<=sl && sr<=qr){///fully overlaped
            return tre[nod];
        }
        if(qr<sl || sr<ql)return 0;/// out of the range
         ↪  ,0/-oo/oo;
        int mid = (sl+sr)/2;
        int left_child = 2*nod , right_child = 2*nod+1;
        return query(left_child,sl,mid,ql,qr)+query(right
         ↪  _child,mid+1,sr,ql,qr);///change
    }
    void update(int nod,int sl,int sr,int ql,int qr,ll
     ↪  val){
        lazyUpdate(nod,sl,sr);
        if(ql<=sl && sr<=qr){///fully overlaped
            lazy[nod]+=val;
            lazyUpdate(nod,sl,sr);
            return;
        }
        if(qr<sl || sr<ql)return;/// position is out of
         ↪  the range
        int mid = (sl+sr)/2;
        int left_child = 2*nod , right_child = 2*nod+1;
        update(left_child,sl,mid,ql,qr,val);
        update(right_child,mid+1,sr,ql,qr,val);
        tre[nod]=tre[left_child]+tre[right_child];///chan
         ↪  ge
    }
};
```

```
signed main(){
    int n,q,ql,qr,ty;
    int val;
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>ar[i];
    }
    LazyTree lt(n);
    lt.build(1,1,n);//lt.build(1,0,n-1);
    while(1){
        cin>>ty;
        if(ty==1){
            cin>>ql>>qr>>val;
            lt.update(1,1,n,ql,qr,val);
        }else if(ty==2){
            cin>>ql>>qr;
            cout<<lt.query(1,1,n,ql,qr)<<"\n";
        }else{
            break;
        }
    }

}
```

Segment Tree

Code in C++:

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

const int Size=10000;
int ar[Size];
int tree[4*Size];
void build(int node,int left,int right){
    if(left==right){
        tree[node]=ar[left];
        return ;
    }
    int mid=(left+right)/2;
    build(node*2,left,mid);
    build(node*2+1,mid+1,right);
    tree[node]=tree[node*2]+tree[node*2+1];
    //tree[node]=max(tree[node*2],tree[node*2+1]);
    //tree[node]=min(tree[node*2],tree[node*2+1]);
}
int query(int node,int left,int right,int ql,int
 ↪  qr){///query left=ql,right=qr
    if(left>=ql && right<=qr){
        return tree[node];
    }
    int mid=(left+right)/2;
    if(qr<=mid){/// range is left of mid
        return query(2*node,left,mid,ql,qr);
    }
    else if(mid<ql){/// range is right of mid
        return query(2*node+1,mid+1,right,ql,qr);
    }
    else{/// range is partially overlap
        int left_node=query(2*node,left,mid,ql,mid);
        int right_node=query(2*node+1,mid+1,right,mid+1,q
         ↪  r);
        return left_node+right_node;
        //return max(left_node,right_node);
        //return min(left_node,right_node);
    }
}
void update(int node,int left,int right,int pos)
{
    if(left==pos && right==pos){
        tree[node]= ar[pos];
        return;
    }
    int mid = (left+right)/2;
    if(pos<=mid && pos>=left)///if pos in lower left
        update(node*2,left,mid,pos);
    else
        update(node*2+1,mid+1,right,pos);/// if pos in
         ↪  lower right
    tree[node] = tree[node*2] + tree[(node*2)+1];
//    tree[node] = min(tree[node*2],tree[(node*2)+1]);
//    tree[node] = max(tree[node*2],tree[(node*2)+1]);
}

int main(){
    //ios_base::sync_with_stdio(0); cin.tie(0);
    int n,ql,qr,pos,new_val;;
    while(cin>>n){
        for(int i=1;i<=n;i++){
            cin>>ar[i];
        }
        build(1,1,n);
        while(1){
            int qtype;cin>>qtype;
            if(qtype==1){///query
                cin>>ql>>qr;
                cout<<query(1,1,n,ql,qr)<<"\n";
            }else if(qtype==2){///update
                cin>>pos>>new_val;
                ar[pos]=new_val;
                update(1,1,n,pos);
            }else{
                break;
            }
        }
    }
    return 0;
}
```

Graph

Articulation Point

Code in C++:

```cpp
void art_point_dfs(int node,int parent,int vis[],int
 ↪  &timer,int tin[],
        int tlow[],vector<int>adj[],set<int>&art_points){
    vis[node]=1;
    tin[node]=tlow[node]=timer;
    timer++;
    int child=0;
    for(auto v : adj[node]){
        if(v==parent)continue;
        if(vis[v]==0){
            art_point_dfs(v,node,vis,timer,tin,tlow,adj,a⌋
             ↪  rt_points);
            tlow[node]=min(tlow[v],tlow[node]);
            if(tlow[v]>=tin[node] && parent!=-1){
                art_points.insert(node);
            }
            child++;
        }else{
            tlow[node]=min(tin[v],tlow[node]);
        }
    }
    if(child>1 && parent ==-1){
        art_points.insert(node);
    }
}

int main(){
    int V,E;
    while(cin>>V>>E){
        int vis[V+5]={0},tin[V+5],tlow[V+5];
        vector<int>adj[V+5];
        set<int>art_points;
        for(int i=0;i<E;i++){
           int u,v;
           cin>>u>>v;
           adj[u].push_back(v);
           adj[v].push_back(u);///underected graph

        }
        int timer=1;
        for(int i=0;i<V;i++){///zero based;
            if(vis[i]==0)
                art_point_dfs(i,-1,vis,timer,tin,tlow,adj⌋
                 ↪  ,art_points);
        }
        for(auto it : art_points){
            cout<<it<<", ";
        }
        if(art_points.size()==0)
            cout<<"There is no Articulation Points";
        cout<<"\n";
    }
}
```

Bellman Ford

Code in C++:

```cpp
struct Edge{
    int u,v,w;
};
vector<Edge>edgeList;
int dist[V_SZ];
int par[V_SZ];
int V,E,Source;
const int oo = (1<<25);

void init(){
    for(int i =1;i<=V;i++)
    {
        dist[i] = oo;
        par[i] = -1;
    }
    edgeList.clear();
}

bool bellmanFord_IsNegCyc(int Source){
    dist[s] = 0;
    bool isUpdated;
    /// bellmanford needs meximum V-1 itaration for update
     ↪  all
    ///nodes minimum distance from source .but why there
    /// is V iteration , becouse if its update a nodes
     ↪  distance
    /// after V-1 iteration that means it have nagative
     ↪  cycle
    for(int i=1;i<=V;i++)
    {
        isUpdated = false;

        for(auto edg: edgeList){
            if(dist[edg.v] > dist[edg.u] + edg.w){
                dist[edg.v] = dist[edg.u] + edg.w;
                par[edg.v] = edg.u;
                isUpdated = true;/// if its update V'th
                 ↪  itaration
                                /// thats means it has
                                 ↪  nagative cycle
            }
        }
    }
    return isUpdated;
}
```

Bipartite Graph Dfs

Code in C++:

```cpp
vector<int>adj[SIZE];
int color[SIZE],V,E,u,v;
bool DFS(int u,int col){
    color[u]=col;
    for(auto v : adj[u]){
        if(color[v]==-1){
            if(DFS(v,!col)==false)/// color[v]= reverse
             ↪  color[u]
                return false;
        }
        else if(color[v]==col){
            return false;
        }
    }
    return true;
}
signed main(){
    cin>>V>>E;
    for(int i=0;i<V+5;i++){
        color[i]=-1;
        adj[i].clear();
    }
    for(int i=0;i<E;i++){
        cin>>u>>v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    int flg=1;
    for(int i=1;i<=V;i++){///1 based
        if(color[i]==-1){
            if(DFS(i,0)==false){
                flg=0;
                break;
            }
        }
    }
    ///if graph has a cycle which is contain odd number of
     ↪  nodes
    ///in that case graph is not  Bipartite Graph
    if(flg){
        cout<<"Yes\n";
    }
    else{
        cout<<"No\n";
    }
}
```

Bridges

Code in C++:

```cpp
const int nodes=105;
int timer;
int vis[nodes],tin[nodes],tlow[nodes];
vector<int>adj[nodes];
```

```cpp
vector<pair<int,int>>bridge;
void bridge_dfs(int node,int parent){
    vis[node]=1;
    tin[node]=tlow[node]=timer;
    timer++;
    for(auto v : adj[node]){
        if(v==parent)continue;
        if(vis[v]==0){
            bridge_dfs(v,node);
            tlow[node]=min(tlow[v],tlow[node]);
            if(tlow[v]>tin[node]){
                bridge.push_back({node,v});
            }
        }else{
            tlow[node]=min(tin[v],tlow[node]);
        }
    }
}
void init(int V){
    for(int i=0;i<V+5;i++){
        vis[i]=0;
        adj[i].clear();
    }
    timer=1;bridge.clear();
}

int main(){
    ///zero based;
    int V,E;
    cin>>V>>E
    init(V);
    for(int i=0;i<E;i++){
       int u,v;cin>>u>>v;
       adj[u].push_back(v);
       adj[v].push_back(u);///underected graph
    }
    for(int i=0;i<V;i++){///zero based;
        if(vis[i]==0)bridge_dfs(i,-1);
    }
    for(auto it : bridge){
        cout<<it.first<<"->"<<it.second<<"\n";
    }
    if(bridge.size()==0)cout<<"No Bridges\n";
    cout<<"\n";
    return 0;
}
```

Dfs

Code in C++:

```cpp
const int WHITE = 0;
const int GRAY = 1;
const int BLACK = 2;
vector<int>g[V_SZ];
int col[V_SZ];
int par[V_SZ];
int startTime[V_SZ];
int finishTime[V_SZ];
int Flattening_tree[2*V_SZ];
int depth[V_SZ];
int height[V_SZ];
int subtree_sum[V_SZ];
vector<int>order;
int Time;
int V,E;
void init(){
    for(int i =1;i<=V;i++){
        col[i] = WHITE;
        par[i] = -1;
        g[i].clear();

        startTime[i] = finishTime[i] = -1;
        Flattening_tree[i]=Flattening_tree[V-i+1]=-1;

        height[i]=0;
        depth[i]=0;
        subtree_sum[i]=i;
    }
    Time = 1;
    order.clear();
}

void dfs(int u){
    startTime[u] = Time;
    Flattening_tree[Time]=u;
    Time++;
    col[u] = GRAY;

    for(auto v: g[u]){
        /// if we solve tree problem then we can use
         ↪  if(v!=par[u])
        ///condition in this line becouse tree do not
         ↪  contain cycle
        if(col[v]==WHITE){
            depth[v]=depth[u]+1;
            par[v] = u;
            dfs(v);
            height[u]=max(height[u],height[v]+1);
            subtree_sum[u]+=subtree_sum[v];
        }
    }
    col[u] = BLACK;
    order.push_back(u);
    finishTime[u] = Time;
    Flattening_tree[Time]=u;
    Time++;
}
int main(){
    /// in tree problem E = V-1
    cin>>V>>E;
    init();
    int u,v;
    for(int i=0;i<E;i++){
        cin>>u>>v;
        g[u].push_back(v);
        g[v].push_back(u); ///then it will be nonderected
         ↪  graph
    }
    for(int i=1;i<=V;i++)
        if(col[i]==WHITE)dfs(i);

    puts("Parent:");
    for(int i=1;i<=V;i++)printf("[%d:%d], ",i,
     ↪  par[i]);printf("\n");
    puts("\nTime:");
    for(int i=1;i<=V;i++)printf("%d:[%d-%d], ",i,
     ↪  startTime[i],finishTime[i]);
    puts("\nFlattening Tree :");
    for(int i=1;i<=2*V;i++)printf("%d,
     ↪  ",Flattening_tree[i]);
    puts("\ndepth of all vertex:");
    for(int i=1;i<=V;i++){
        cout<<i<<" -> "<<depth[i]<<" ,";
    }
    puts("\n\nheight of all vertex:");
    for(int i=1;i<=V;i++){
        cout<<i<<" -> "<<height[i]<<" ,";
    }
    puts("\n\nSub tree sum:");
    for(int i=1;i<=V;i++){
        cout<<"["<<i<<":"<<subtree_sum[i]<<"] ,";
    }
    puts("\n\nFinal complite  visited Order:");
    for(auto v: order)  printf("%d, ", v);  puts("");

}
```

Dfs Cycle Finder

Code in C++:

```cpp
const ll mxn=2e5+10;
vector<ll>g[mxn+10],cyl[mxn+10];
ll par[mxn+10];
ll vis[mxn+10],cnt;
void cycle(ll snode,ll enode){
    cyl[cnt].push_back(enode);
    while(snode!=enode){
        cyl[cnt].push_back(snode);
        snode=par[snode];
    }
    cyl[cnt].push_back(enode);
    ///reverse(cyl[cnt].begin(),cyl[cnt].end());///for
     ↪  directed graph
}
void DFS(ll u){
    vis[u]=1;
    for(auto v:g[u]){
        if(vis[v]==0){
            par[v]=u;
            DFS(v);
        }
        else if(vis[v]==1&&v!=par[u]){///if(vis[v]==1) for
         ↪ directed graph
            ll snode=u;
            ll enode=v;
            cycle(snode,enode);
            cnt++;///number of cycle
        }
    }
    vis[u]=2;/// u nodes all adj nodes r visited
}

int main(){
    ///for undirected = minimum three nodes make a cycle
     ↪ for this code
    ///for directed = also one node can be make a cycle
     ↪ for this code
    ll V,E,u,v;
    while(cin>>V>>E){
        for(int i=0;i<=V+5;i++){
            vis[i]=0;
            par[i]=-1;
            g[i].clear();
            cyl[i].clear();
        }
        for(ll i=0; i<E; i++){
            cin>>u>>v;
            g[u].push_back(v);
            g[v].push_back(u);///off the line for directed
             ↪ graph
        }
        cnt=0;
        for(ll i=1;i<=V;i++){
            if(vis[i]==0)
            DFS(i);
        }
        for(ll i=0; i<cnt; i++){
            for(auto v:cyl[i])
                cout<<v<<" ";
            cout<<"\n";
        }
        if(cnt==0)cout<<"IMPOSSIBLE"<<"\n";
    }
}
```

Dijkstra Using Pq

Code in C++:

```
struct Nod{
    int u, dis;
    Nod(int iU, int iDis){
        u = iU;
        dis = iDis;
    }
    bool operator<(const Nod& b) const{
        return dis > b.dis;
    }
};
const int Vertex_N = 101;
const int oo = 1e8+0.5;
int dist[Vertex_N];
int par[Vertex_N];
vector<int>graph[Vertex_N];
vector<int>weight[Vertex_N];
void init(int n){
    for(int i=1;i<=n;i++){
        dist[i] = oo;
        par[i] = -1;
        graph[i].clear();
        weight[i].clear();
    }
}
int dijkstra(int source, int destination){
    priority_queue<Nod>pq;
    dist[source] = 0;
    pq.push(Nod(source, 0));///pq.push({source, 0});
    while(!pq.empty()){
        Nod cur = pq.top();
        pq.pop();

        int u = cur.u;
        int uDist = cur.dis;
        if(dist[u] < uDist) {
            continue;
        }
        for(int i=0;i<graph[u].size();i++){
            int v = graph[u][i];
            int edgeWeight = weight[u][i];
            if(dist[v] > uDist + edgeWeight){
                dist[v] = uDist + edgeWeight;
                par[v] = u;
                pq.push({v, dist[v]});
            }
        }
    }
    return dist[destination];
}
vector<int> getPaht(int source, int destination){
    int v = destination;
    vector<int>path;
    while(source != v){
        path.push_back(v);
        v = par[v];
    }
    path.push_back(source);
    reverse(path.begin(), path.end());
    return path;
}

int main(){
    int V, E, S, D;///S=Point vartex,D=terget vartex
     ↪ distence
    cin>>V>>E>>S>>D;
    init(V);
    for(int i=0;i<E;i++){
        int u,v,w;
        cin>>u>>v>>w;
        graph[u].push_back(v);
        weight[u].push_back(w);
        ///for undirected graph
        graph[v].push_back(u);
        weight[v].push_back(w);
    }
    int distance = dijkstra(S, D);
    printf("Distace: %d\n", distance);

    vector<int>path = getPaht(S, D);
    printf("Path: ");
    for(auto v: path) cout<<v<<" ";cout<<endl;
}
```

Floyd Warshall

Code in C++:

```
const int oo = 1e8;
const int Size = 100;
int dis[Size][Size],N,E;
cin>>N>>E;/// N=number of nodes,E=number of edges
for(int i=1;i<=N;i++){
    for(int j=1;j<=N;j++){
        dis[i][j]=oo;
        if(i==j)dis[i][j]=0;
    }
}
for(int i=0;i<E;i++){
    int u,v,w;
    cin>>u>>v>>w;
    dis[u][v]=w;/// directed graph;
}

for(int via=1;via<=N;via++){
    for(int u=1;u<=N;u++){
        for(int v=1;v<=N;v++){
            dis[u][v]=min(dis[u][v] ,
             ↪ dis[u][via]+dis[via][v]);
        }
    }
}
cout<<"All nodes distance matrix:\n";
for(int u=1;u<=N;u++){
    for(int v=1;v<=N;v++){
        if(dis[u][v]==oo){
            cout<<"oo ";
        }
        else
        cout<<dis[u][v]<< " ";
    }
    el;
}
/// if any node dist[u][u]<0 then
/// we call it has nagtive cycle
```

Kruskal Union By Size

Code in C++:

```
struct Edge{
    int u,v,w;
    Edge(int ui,int vi,int wi){
        u=ui;v=vi;w=wi;
    }
};
vector<Edge>edgeList;
int parant[sz];
int compoSize[sz];
void disjoint(int V){
    edgeList.clear();
    for(int i=0;i<=V+3;i++){
        parant[i]=i;///call by make func in DSU
        compoSize[i]=1;
    }
}
/// finding root node of this component and
/// make root node is parant of the all nodes of this
 ↪  component
int FindRootParant(int node){
    if(node == parant[node])
        return node;
    return parant[node]=FindRootParant(parant[node]);
}
/// joining two components
void join_components(int u,int v){///union/makelink  u
 ↪  nodes component
                                /// to v nodes component
                                 ↪  and
                                 /// make them as a same
                                  ↪  component

    int u_parant=FindRootParant(u); /// finding u 's root
     ↪  node
    int v_parant=FindRootParant(v); /// finding v 's root
     ↪  node

    if(u_parant==v_parant)///both r alrady joined
        return ;
    if(compoSize[u_parant]>compoSize[v_parant]){
        parant[v_parant]=u_parant;
        compoSize[u_parant]+=compoSize[v_parant];
    }
    else{
        parant[v_parant]=u_parant;
        compoSize[v_parant]+=compoSize[u_parant];
    }
}
bool com_by_waight(Edge a,Edge b){
    return a.w<b.w;
}
int kruskal(){
    int cost=0;
    sort(edgeList.begin(),edgeList.end(),com_by_waight);

    for(int i=0;i<edgeList.size();i++){
        ///if two vertex or groups root parent r same
        ///then it will be creat a cycle
        ///Then we won't add them
        ///else ↓
        if(FindRootParant(edgeList[i].u) !=
         ↪  FindRootParant(edgeList[i].v)){
            join_components(edgeList[i].u ,
             ↪  edgeList[i].v);
            cost+=edgeList[i].w;
            ///connecting edges
            cout<<edgeList[i].u<<"<->"<<edgeList[i].v<<" =
             ↪  "<<edgeList[i].w;el;
        }
    }
    return cost;
}

int main(){
    int V,E;cin>>V>>E;
    disjoint(V);///init
    for(int i=0;i<E;i++){
        int u,v,w;
        cin>>u>>v>>w;
        edgeList.push_back({u,v,w});
    }
    int MST_Cost = kruskal();
    cout<<"MST COST = "<<MST_Cost;el;

}
```

Max Flow Dinics Algorithm

Code in C++:

```
//O(V^2 E)
const long long inf = 1LL << 61;
struct Dinic {
  struct edge {
    int to, rev;
    long long flow, w;
    int id;
  };
  int n, s, t, mxid;
  vector<int> d, flow_through;
  vector<int> done;
  vector<vector<edge>> g;
  Dinic() {}
  Dinic(int _n) {
    n = _n + 10;
    mxid = 0;
    g.resize(n);
  }
  void add_edge(int u, int v, long long w, int id = -1) {
    edge a = {v, (int)g[v].size(), 0, w, id};
    edge b = {u, (int)g[u].size(), 0, 0, -2};//for
     ↪  bidirectional edges cap(b) = w
    g[u].emplace_back(a);
    g[v].emplace_back(b);
    mxid = max(mxid, id);
  }
  bool bfs() {
    d.assign(n, -1);
    d[s] = 0;
    queue<int> q;
    q.push(s);
    while (!q.empty()) {
      int u = q.front();
      q.pop();
      for (auto &e : g[u]) {
        int v = e.to;
        if (d[v] == -1 && e.flow < e.w) d[v] = d[u] + 1,
         ↪  q.push(v);
      }
    }
    return d[t] != -1;
  }
  long long dfs(int u, long long flow) {
    if (u == t) return flow;
    for (int &i = done[u]; i < (int)g[u].size(); i++) {
      edge &e = g[u][i];
      if (e.w <= e.flow) continue;
      int v = e.to;
      if (d[v] == d[u] + 1) {
        long long nw = dfs(v, min(flow, e.w - e.flow));
        if (nw > 0) {
          e.flow += nw;
          g[v][e.rev].flow -= nw;
          return nw;
        }
      }
    }
    return 0;
  }
```

```cpp
  long long max_flow(int _s, int _t) {
    s = _s;
    t = _t;
    long long flow = 0;
    while (bfs()) {
      done.assign(n, 0);
      while (long long nw = dfs(s, inf)) flow += nw;
    }
    flow_through.assign(mxid + 10, 0);
    //for(int i = 0; i <=n; i++) for(auto e : g[i]) if(e.id >= 0) flow_through[e.id] = e.flow;
    return flow;
  }
};
int main() {
  int n, m;
  cin >> n >> m;
  Dinic F(n + 1);
  for (int i = 1; i <= m; i++) {
    int u, v, w;/// onse based
    cin >> u >> v >> w;
    F.add_edge(u, v, w);
    //F.add_edge(v, u, w);// bi directional
  }
  cout << F.max_flow(1, n) << '\n';
  return 0;
}
```

Stronglyconnectedcomponents

Code in C++:

```cpp
struct node{
    int idx,st,fin;
};
node Time[Size];
vector<int>adj[Size];
vector<int>radj[Size];
vector<int>component[Size];
int vis[Size],scc[Size],ti,compo_no;
bool com(node a,node b){
    return a.fin>b.fin;
}
bool comidx(node a,node b){
    return a.idx<b.idx;
}
void dfs(int u){
    Time[u].st=ti++;
    vis[u]=1;
    for(int i=0;i<adj[u].size();i++){
        int v=adj[u][i];
        if(vis[v]==0){
            dfs(v);
        }
    }
    Time[u].fin=ti++;
}
void rdfs(int u,int compo_no){
    vis[u]=1;
    scc[u]=compo_no;///scc[u] is compo_no'th component who is carring nod u
    component[compo_no].push_back(u);
    for(int i=0;i<radj[u].size();i++){
        int v=radj[u][i];
        if(vis[v]==0){
            rdfs(v,compo_no);
        }
    }
}
int main(){
    int V,E,u,v;cin>>V>>E;
    /// SCC works only for directional graph
    for(int i=0;i<=V+5;i++){
        adj[i].clear();
        radj[i].clear();
        component[i].clear();
        vis[i]=0;
    }
    for(int i=1;i<=E;i++){
        cin>>u>>v;
        adj[u].push_back(v);
        radj[v].push_back(u);/// for reverce edges direction
    }
    ti=1;
    for(int i=1;i<=V;i++){/// 1 based graph
        Time[i].idx=i;
        if(vis[i]==0){
            dfs(i);
        }
    }
    // cout<<"[Start time,Finish Time]:\n";
    // for(int i=1;i<=V;i++){
    //     cout<<"Nod "<<i<<" :["<<Time[i].st<<","<<Time[i].fin<<"]\n";
    // }
    memset(vis,0,sizeof vis),compo_no=0;///compo_no = n'th component[1 based]
    sort(&Time[1],&Time[V+1],com);///precedency by finish time
    for(int i=1;i<=V;i++){
        if(vis[Time[i].idx]==0){
            compo_no++;          /// compo_no also compo index which is 1 based
            rdfs(Time[i].idx,compo_no);///dfs traverce for reverse direction
        }
    }
    //sort(&Time[1],&Time[V+1],comidx);
    for(int i=1;i<=compo_no;i++){/// number of component is compo_no
        cout<<"SCC("<<scc[component[i][0]]<<")-> ";///component[i][0] is the 1st node of i'th component
        for(auto v : component[i]){
            cout<<v<<" ";
            //cout<<"-> SCC("<<scc[v]<<") ,";
        }
        cout<<"\n";
    }
    cout<<"\n";

}
```

Topologicalsort Khans

Code in C++:

```cpp
const int Size=105;
vector<int>adj[Size];
vector<int>TS;///Topological Sort
int indegree[Size],V,E;
queue<int>Q;
void init(){
    for(int i=0;i<V+5;i++){
        indegree[i]=0;
        adj[i].clear();
    }
    TS.clear();
}
void topo_BFS(){
    while(!Q.empty()){
        int u=Q.front();Q.pop();
        for(auto v : adj[u]){
            --indegree[v];
            if(indegree[v]==0){
                TS.emplace_back(v);
                Q.push(v);
            }
        }
    }
}
int main(){
    cin>>V>>E;
    init();int u,v;
    for(int i=0;i<E;i++){
        cin>>u>>v;
        adj[u].emplace_back(v);
        ++indegree[v];
    }
    for(int i=1;i<=V;i++){///1 based
        if(indegree[i]==0){
            TS.emplace_back(i);
            Q.push(i);
        }
    }
    topo_BFS();
    if(TS.size()!=V){
        cout<<"CycleExist\n";
    }
    for(auto it : TS){
        cout<<it<<" ";
    }
    cout<<"\n";
}
```

Matrix

Matrix Exponential

Code in C++:

```
vector<vector<ll>> matMulti(vector<vector<ll>>&a,
 ↪  vector<vector<ll>>&b, ll Mod){
    int r = a.size();
    int c = b[0].size();
    vector<vector<ll>> ans(r, vector<ll>(c, 0));
    for(int i = 0; i < r; i++) {
        for(int j = 0; j < c; j++) {
            for(int k = 0; k < a[0].size(); k++) {
                ans[i][j]=((ans[i][j]+(a[i][k]*b[k][j])%M⌋
                 ↪  od)%Mod+Mod)%Mod;
            }
        }
    }
    return ans;
}
vector<vector<ll>> matExpo(vector<vector<ll>>& b, ll pw,
 ↪  ll Mod){
    int r = b.size();
    vector<vector<ll>> ans(r, vector<ll>(r, 0));
    for(int i = 0; i < r; ++i) {
        ans[i][i] = 1;
    }
    while(pw) {
        if(pw & 1) {
            ans = matMulti(ans, b, Mod);
        }
        b = matMulti(b, b, Mod);
        pw >>= 1;
    }
    return ans;
}
```