KUET_BreakDown

Contents

1 Include

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#define ull unsigned long long
#define ll long long
#define endl '\n'
#define line cout<<"--------------------------"<<endl
#define dd(x) cout<<#x<<" = "<<x<<' '
#define sz(v) (ll)v.size()
#define srt(v) sort(v.begin(),v.end())
#define rsrt(v) sort(v.rbegin(),v.rend())
#define all(v) v.begin(),v.end()
#define pb push_back
#define pi acos(-1)
#define ff first
#define ss second
#define pii pair<int,int>
#define mp make_pair
#define mod 1000000007
#define fast ios_base::sync_with_stdio(false); cin.tie(NULL)
#define filein freopen("input.txt","r",stdin)
#define fileout freopen("output.txt","w",stdout)
using namespace std;
using namespace __gnu_pbds;
template <typename T> using os=tree < T, null_type
     ,less<T>,
     rb_tree_tag,tree_order_statistics_node_update >;
///find_by_order(k) : returns an iterator of the k-th index
///order_of_key(k) : returns the number of elements less
     than k
mt19937
     rng(chrono::steady_clock::now().time_since_epoch().count());
const ll mx=100009;
os<int> s;
//python:
//a=list(map(int,input().split()))
//merge function of merge sort tree
//merge(all(tree[node*2]),all(tree[node*2+1]),back_inserter(tree[noc
```

2 Math

2.1 Sieve

```
const int mx=1000009;
bitset<mx>prime;
int pr[mx];
vector<int> v;
void sieve() {
    prime[1]=1;
    for(int i=4;i<mx;i+=2) prime[i]=1;
    for(int i=3;i*i<mx;i+=2) {
        if(prime[i]==false) {
            for(int j=i*i;j<mx;j+=2*i) {
                prime[j]=true;
            }
        }
    }
    v.pb(2);
    for(int i=3;i<mx;i+=2) if(prime[i]==false ) v.pb(i);
}
void pre() {
    for(int i=1;i<mx;i++) {
        if(i%2==0) pr[i]=2;
        else pr[i]=i;
    }
    for(int i=3;i<mx;i+=2) {
        if(pr[i]==i) {
            for(int j=i;j<mx;j+=i) pr[j]=i;
        }
    }
}
```

2.2 Extended Euclid

```
int extended(int a, int b,int &x, int &y) {
    if(b==0) {
        x=1; y=0; return a;
    }
    int x1,y1;
    int d=extended(b,a%b,x1,y1);
    x=y1; y=x1-(a/b)*y1;
    return d;
}
int  main() {
    int a,b,x,y;
    cin>>a>>b;
    int ans=extended(a,b,x,y);
    cout<<x<<' '<<y<<endl;
}
```

2.3 Catalan

```
ll cat[1120];
ll supcat[1120];
formula 1: cn=(1/(n+1)) * (2n c n)
formula 2: cn=(2n)! / ((n+1)! * n! )
void pre() {
    cat[0]=1;
    for(ll i=1;i<=50;i++) {
        for(ll j=0;j<i;j++) {
            cat[i]+=cat[j]*cat[i-1-j];
        }
    }
    supcat[0]=1;
    for(ll i=1;i<=50;i++) {
        supcat[i]=supcat[i-1];
        for(ll j=1;j<i;j++) {
            supcat[i]+=supcat[j]*supcat[i-j];
        }
    }
    supcat[1]=2;
    for(ll i=1;i<=50;i++) supcat[i]/=2;
}
Application:
1. Cn is the number of Dyck words[2] of length 2n. A Dyck
     word is a string
consisting of n X's and n Y's such that no initial segment
     of the string has
more Y's than X's. For example, the following are the Dyck
     words of length 6:
XXXYYY    XYXXYY    XYXYXY    XXYYXY    XXYXYY.
2. Re-interpreting the symbol X as an open parenthesis and
     Y as a close
parenthesis, Cn counts the number of expressions containing
     n pairs of
parentheses which are correctly matched:
((()))    ()(())    ()()()    (())()    (()())
3. Cn is the number of different ways n + 1 factors can be
     completely parenthesized .
For n = 3,for example, we have the following five different
     parenthesizations of four factors:
((ab)c)d    (a(bc))d    (ab)(cd)    a((bc)d)    a(b(cd))
4. Cn is the number of non-isomorphic ordered (or plane)
     trees with n + 1 vertices
5. Cn is the number of monotonic lattice paths along the
     edges of a grid with n  n square
cells, which do not pass above the diagonal.
6. A convex polygon with n + 2 sides can be cut into
     triangles by connecting
vertices with non-crossing line segments in cn ways.
```

2.4 Chinese Remainder Theorem

```
vector<ll>r,m;
ll extended(ll a, ll b,ll &x, ll &y) {
    //ekhane template dekhe likhe dibo
}
ll crt() //result is unique when mod by lcm(all m)
{
    ll r1=r[0],m1=m[0];
    for(ll i=1;i<(int)r.size();i++) {
        ll r2=r[i]; ll m2=m[i];
        ll gcd=__gcd(m1,m2);
        if(r1%gcd != r2%gcd) return -1;
        ll lcm=m1/gcd*m2;
        m1/=gcd; m2/=gcd;
        ll p,q;
        extended(m1,m2,p,q);
        r1=((__int128)(r1*m2)%lcm*q%lcm+(__int128)(r2*m1)%lcm*p%lcm)
        r1%=lcm;
        m1=lcm;
        if(r1<0) r1+=lcm;
    }
    return r1; //all solution x=r1+lcm*k
}
int main() {
    ll n;
    cin>>n;
    for(ll i=0;i<n;i++) {
        ll a,b;
        cin>>a>>b;
        m.pb(a); r.pb(b);
    }
    ll ans=crt();
    if(ans==-1) cout<<"Impossible"<<endl;
    else cout<<ans<<endl;
}
```

2.5 Derangement Number

```
formula 1: dp[n]=(n-1)*(dp[n-1)+dp[n-2)) where
     dp[1]=0,dp[2]=1

formula 2:
     dp[n]=n!*(1-1/1!+1/2!-1/3!+1/4!-1/5!+1/6!-.....................
```

2.6 Mobius

```
int mob[10009];
int mobius() {
    /// 1 when n==1
    /// 0 when there is a prime more than once
    /// (-1)^p here p is the total distinct prime factor
    mob[1]=1;
```

```
    for(int i=1;i<=10008;i++) {
        for(int j=2*i;j<=10003;j+=i) {
            mob[j]-=mob[i];
        }
    }
}
```

2.7 Mod Inverse

```
ll facto[mx],invfacto[mx];
void pre() {
    facto[0]=1;
    for(ll i=1;i<mx;i++) facto[i]=(facto[i-1]*i)%mod;
    invfacto[0]=1; invfacto[1]=1;
    for(ll i=2;i<mx;i++)
         invfacto[i]=mod-(mod/i)*invfacto[mod%i]%mod; ///pre
         calculate mod inverse i
    for(int i=1;i<mx;i++)
         invfacto[i]=(invfacto[i-1]*invfacto[i])%mod;
         ///precalculate inverse factorial
}
ll ncr(ll n, ll r) {
    if(r>n) return 0;
    ll ans=facto[n]*invfacto[n-r]; ans%=mod;
    ans=(ans*invfacto[r])%mod;
    return ans;
}
```

2.8 Phi

```
int toti[mx];
//sum(n)=phi[1]+phi[2]+phi[3]+....phi[n]
//sum(n)=n*(n+1)/2-for(2 to n)sum(floor(n/i))
//sum of coprime number of n = n*phi(n)/2
void phi() {
    for(int i=1;i<mx;i++) toti[i]=i ;
    for( int i=2;i<mx;i++) {
        if(toti[i]==i) {
            for(int j=i;j<mx;j+=i) {
                toti[j]-=toti[j]/i;
            }
        }
    }
}
```

2.9 Lagrange interpolation

```
ll add(ll a, ll b) {
    if(a+b<mod) return a+b;
    return a+b-mod;
}
ll sub(ll a,ll b) {
    if(a-b<0) return a-b+mod;
    return a-b;
}
ll mul(ll a, ll b) {
    return (a*b)%mod;
}
ll bigmod(ll a, ll b, ll m) {
    ll ans=1;
    a%=m;
    while(b) {
        if(b&1) ans=(ans*a)%m;
        b>>=1;
        a=(a*a)%m;
    }
    return ans;
}
ll facto[mx],invfacto[mx];
ll pre[mx],suf[mx];
void pre1() {
    facto[0]=1;
    for(ll i=1;i<mx;i++) facto[i]=(facto[i-1]*i)%mod;
    invfacto[0]=1;
    invfacto[1]=1;
    for(ll i=2;i<mx;i++)
         invfacto[i]=mod-(mod/i)*invfacto[mod%i]%mod; ///pre
         calculate mod inverse i
    for(ll i=1;i<mx;i++)
         invfacto[i]=(invfacto[i-1]*invfacto[i])%mod;
         ///precalculate inverse factorial
}
ll x,k;
vector<ll> f;
ll get() {
    if(x<=k+1) return f[x];
    else {
        x%=mod;
        ll sum=0;
        ll n=(ll)f.size()-1;
        pre[0]=1;
        suf[k+1]=1;
        for(ll i=0;i<=k+1;i++) pre[i+1]=mul(pre[i],(x-i));
        for(ll i=k+1;i>=1;i--) suf[i-1]=mul(suf[i],(x-i));
        for(ll i=0;i<=n;i++) {
            ll lobb=mul(pre[i],suf[i]);
            lobb=mul(lobb,invfacto[i]);
            lobb=mul(lobb,invfacto[n-i]);
            ll anss=mul(f[i],lobb);
            if((n-i)%2) sum=sub(sum,anss);
            else sum=add(sum,anss);
        }
        return sum;
    }
}
int main() {
    //find 1^k + 2^k + 3^k + .... + n^k
    //sum is a k+1 degree polynomial, so need (k+2) terms to
        get the actual result
    pre1();
    cin>>x>>k;
    ll sum=0;
    f.pb(0);
    for(ll i=1;i<=k+1;i++) {
        sum=add(sum,bigmod(i,k,mod));
        f.pb(sum);
    }
    cout<<get()<<endl;
}
```

2.10 Random Generator

```
mt19937
    rng(chrono::steady_clock::now().time_since_epoch().count());
int main() {
       auto dist=uniform_int_distribution<int>(l,r);
            //random number between l and r
       int val=dist(rng);
       cout<<val<<' ';
}
```

2.11 Matrix

```
ll dm;
struct matrix {
    ll  mat[20][20];
    matrix() {
        memset(mat,0,sizeof mat);
    }
    matrix operator *(const matrix &a) const {
        matrix d;
        for(ll i=0;i<dm;i++) {
            for(ll j=0;j<dm;j++) {
                for(ll k=0;k<dm;k++) {
                    d.mat[i][j]+=(mat[i][k]*a.mat[k][j])%mod;
                    d.mat[i][j]%=mod;
                }
            }
        }
        return d;
    }
    void identity() {
        for(ll i=0;i<dm;i++) {
            mat[i][i]=1;
        }
```

```
    }
};
matrix power(matrix c,ll b) {
    matrix d;
    d.identity();
    while(b) {
        if(b&1) d=d*c;
        b>>=1; c=c*c;
    }
    return d;
}
int main() {
    ///dm->dimension
    ///nth fibonacci
    ///complexity O(d^3 log(n)) where d is dimension of
        matrix
    matrix a;
    int n;
    while(cin>>n){
        dm=2;
        a.mat[0][0]=1; a.mat[0][1]=1; a.mat[1][0]=1;
            a.mat[1][1]=0;
        if(n==1 or n==2) cout<<1<<endl;
        else {
            a=power(a,n-2);
            ll ans=a.mat[0][0]+a.mat[0][1];
            cout<<ans<<endl;
        }
    }
}
```

2.12 Gauss(mod 2)

```
ll basis[30]; ll len=0;
void add(ll mask) {
    for(ll i=0;i<30;i++) {
        if((mask & 1LL<<i)==0) continue;
        if(basis[i]==0) {
            basis[i]=mask; len++;
            return;
        }
        mask^=basis[i];
    }
}
bool check(ll mask) //array element niye particular xor
    banano jay kina
{
    for(ll i=0;i<30;i++) {
        if((mask & 1LL<<i)==0) continue;
        if(basis[i]==0) return 0;
        mask^=basis[i];
    }
    return 1;
}
```

2.13 FFT

```
struct CD {
    double x, y;
    CD(double x=0, double y=0) :x(x), y(y) {}
    CD operator+(const CD& o) { return {x+o.x, y+o.y};}
    CD operator-(const CD& o) { return {x-o.x, y-o.y};}
    CD operator*(const CD& o) { return {x*o.x-y*o.y,
        x*o.y+o.x*y};}
    CD operator/(double d) { return {x/d, y/d};}
};
namespace FFT {
    int N;
    vector<int> perm;
    void precalculate() {
        perm.resize(N);
        perm[0] = 0;
        for (int k=1; k<N; k<<=1) {
            for (int i=0; i<k; i++) {
                perm[i] <<= 1;
                perm[i+k] = 1 + perm[i];
            }
        }
    }
    void fft(vector<CD> &v, bool invert = false) {
        if (v.size() != perm.size()) {
            N = v.size();
            assert(N && (N&(N-1)) == 0);
            precalculate();
        }
        for (int i=0; i<N; i++)
            if (i < perm[i])
                swap(v[i], v[perm[i]]);
        for (int len = 2; len <= N; len <<= 1) {
            double angle = 2 * PI / len;
            if (invert) angle = -angle;
            CD factor = {cos(angle), sin(angle)};
            for (int i=0; i<N; i+=len) {
                CD w(1);
                for (int j=0; j<len/2; j++) {
                    CD x = v[i+j], y = w * v[i+j+len/2];
                    v[i+j] = x+y;
                    v[i+j+len/2] = x-y;
                    w = w*factor;
                }
            }
        }
        if (invert)
            for (CD &x : v) x=x/N;
    }
    vector<ll> multiply(const vector<ll> &a, const
        vector<ll> &b) {
        vector<CD> fa(a.begin(), a.end()), fb(b.begin(),
            b.end());
        int n = 1;
        while (n < a.size()+ b.size()) n<<=1;
        fa.resize(n);
        fb.resize(n);
        fft(fa);
        fft(fb);
        for (int i=0; i<n; i++) fa[i] = fa[i] * fb[i];
        fft(fa, true);
        vector<ll> ans(n);
        for (int i=0; i<n; i++)
            ans[i] = round(fa[i].x);
        return ans;
    }
}
using namespace FFT;
```

2.14 NTT (mod by 998244353)

```
namespace ntt {
    const int mod = 998244353;
    const int root=5;
    int power(int a, int b, int m ,int ans=1) {
        for (; b; b>>=1, a=1LL*a*a%m) if (b&1)
            ans=1LL*ans*a%m;
        return ans;
    }
    const int root_1=power(root, mod-2, mod);
    using VI = vector<int> ;
    void ntt(VI& a, bool invert) {
        int n=a.size();
        int PW=power(invert?root_1:root, (mod-1)/n,
            mod);
        for (int m=n, h; h=m/2, m>=2;
            PW=1LL*PW*PW%mod, m=h) {
            for (int i=0, w=1; i<h; ++i,
                w=1LL*w*PW%mod)
            for (int j=i; j<n; j+=m) {
                int k=j+h,
                    x=(a[j]-a[k]+mod)%mod;
                a[j]+=a[k]; a[j] %= mod;
                a[k]=1LL*w*x%mod;
            }
        }
        for (int i=0, j=1; j<n-1; ++j) {
            for (int k=n/2; k>(i^=k); k /= 2);
            if (j<i) swap(a[i], a[j]);
        }
        if (invert) {
            int rev=power(n, mod-2, mod);
            for (int i=0; i<n; ++i)
                a[i]=1LL*a[i]*rev%mod;
        }
```

```
    }
    VI multiply(const VI& a, const VI& b, int ok=0) {
        int n=1, mx=a.size()+b.size()-1;
        while (n<mx) n<<=1;
        if (mx<256) {
            VI c(mx);
            for (int i=0; i<a.size(); i++) for
                (int j=0; j<b.size(); j++) {
                  c[i+j]=(c[i+j]+1ll*a[i]*b[j])%mod;
            }
            return c;
        }
        VI na=a, nb=b;
        na.resize(n); nb.resize(n);
        ntt(na, false);
        if (ok) nb=na;
        else ntt(nb, false);
        for (int i=0; i<n; ++i)
            na[i]=1LL*na[i]*nb[i]%mod;
        ntt(na, true);
        na.resize(mx);
        return na;
    }
};
using namespace ntt;
note: stirling number of second kind (for fixed n) can be
    generated by multiplyig two
polynomials A and B where
Ai=(-1)^i / (i!)
Bi=i^n / (i!)
```

2.15 NTT (mod by any)

```
const int N = 1e5 + 9, mod = 998244353; //can change mod
    value
struct base {
  double x, y;
  base() { x = y = 0; }
  base(double x, double y): x(x), y(y) { }
};
inline base operator + (base a, base b) { return base(a.x +
    b.x, a.y + b.y); }
inline base operator - (base a, base b) { return base(a.x -
    b.x, a.y - b.y); }
inline base operator * (base a, base b) { return base(a.x *
    b.x - a.y * b.y, a.x * b.y + a.y * b.x); }
inline base conj(base a) { return base(a.x, -a.y); }
int lim = 1;
vector<base> roots = {{0, 0}, {1, 0}};
vector<int> rev = {0, 1};
const double PI = acosl(- 1.0);
void ensure_base(int p) {
  if(p <= lim) return;
  rev.resize(1 << p);
  for(int i = 0; i < (1 << p); i++) rev[i] = (rev[i >> 1]
      >> 1) + ((i & 1) << (p - 1));
  roots.resize(1 << p);
  while(lim < p) {
    double angle = 2 * PI / (1 << (lim + 1));
    for(int i = 1 << (lim - 1); i < (1 << lim); i++) {
      roots[i << 1] = roots[i];
      double angle_i = angle * (2 * i + 1 - (1 << lim));
      roots[(i << 1) + 1] = base(cos(angle_i), sin(angle_i));
    }
    lim++;
  }
}
void fft(vector<base> &a, int n = -1) {
  if(n == -1) n = a.size();
  assert((n & (n - 1)) == 0);
  int zeros = __builtin_ctz(n);
  ensure_base(zeros);
  int shift = lim - zeros;
  for(int i = 0; i < n; i++) if(i < (rev[i] >> shift))
      swap(a[i], a[rev[i] >> shift]);
  for(int k = 1; k < n; k <<= 1) {
    for(int i = 0; i < n; i += 2 * k) {
      for(int j = 0; j < k; j++) {
        base z = a[i + j + k] * roots[j + k];
        a[i + j + k] = a[i + j] - z;
        a[i + j] = a[i + j] + z;
      }
    }
  }
}
//eq = 0: 4 FFTs in total
//eq = 1: 3 FFTs in total
vector<int> multiply(vector<int> &a, vector<int> &b, int eq
    = 0) {
  int need = a.size() + b.size() - 1;
  int p = 0;
  while((1 << p) < need) p++;
  ensure_base(p);
  int sz = 1 << p;
  vector<base> A, B;
  if(sz > (int)A.size()) A.resize(sz);
  for(int i = 0; i < (int)a.size(); i++) {
    int x = (a[i] % mod + mod) % mod;
    A[i] = base(x & ((1 << 15) - 1), x >> 15);
  }
  fill(A.begin() + a.size(), A.begin() + sz, base{0, 0});
  fft(A, sz);
  if(sz > (int)B.size()) B.resize(sz);
  if(eq) copy(A.begin(), A.begin() + sz, B.begin());
  else {
    for(int i = 0; i < (int)b.size(); i++) {
      int x = (b[i] % mod + mod) % mod;
      B[i] = base(x & ((1 << 15) - 1), x >> 15);
    }
    fill(B.begin() + b.size(), B.begin() + sz, base{0, 0});
    fft(B, sz);
  }
  double ratio = 0.25 / sz;
  base r2(0, - 1), r3(ratio, 0), r4(0, - ratio), r5(0, 1);
  for(int i = 0; i <= (sz >> 1); i++) {
    int j = (sz - i) & (sz - 1);
    base a1 = (A[i] + conj(A[j])), a2 = (A[i] - conj(A[j]))
        * r2;
    base b1 = (B[i] + conj(B[j])) * r3, b2 = (B[i] -
        conj(B[j])) * r4;
    if(i != j) {
      base c1 = (A[j] + conj(A[i])), c2 = (A[j] -
          conj(A[i])) * r2;
      base d1 = (B[j] + conj(B[i])) * r3, d2 = (B[j] -
          conj(B[i])) * r4;
      A[i] = c1 * d1 + c2 * d2 * r5;
      B[i] = c1 * d2 + c2 * d1;
    }
    A[j] = a1 * b1 + a2 * b2 * r5;
    B[j] = a1 * b2 + a2 * b1;
  }
  fft(A, sz); fft(B, sz);
  vector<int> res(need);
  for(int i = 0; i < need; i++) {
    long long aa = A[i].x + 0.5;
    long long bb = B[i].x + 0.5;
    long long cc = A[i].y + 0.5;
    res[i] = (aa + ((bb % mod) << 15) + ((cc % mod) <<
        30))%mod;
  }
  return res;
}
```

2.16 Stirling number of first kind

```
const int N = 1 << 18;
const int mod = 998244353;
const int root = 3;
int lim, rev[N], w[N], wn[N], inv_lim;
void reduce(int &x) { x = (x + mod) % mod; }
int POW(int x, int y, int ans = 1) {
  for (; y; y >>= 1, x = (long long) x * x % mod) if (y &
      1) ans = (long long) ans * x % mod;
  return ans;
}
void precompute(int len) {
  lim = wn[0] = 1; int s = -1;
  while (lim < len) lim <<= 1, ++s;
  for (int i = 0; i < lim; ++i) rev[i] = rev[i >> 1] >> 1 |
      (i & 1) << s;
  const int g = POW(root, (mod - 1) / lim);
  inv_lim = POW(lim, mod - 2);
  for (int i = 1; i < lim; ++i) wn[i] = (long long) wn[i -
      1] * g % mod;
```

```
}
void ntt(vector<int> &a, int typ) {
  for (int i = 0; i < lim; ++i) if (i < rev[i]) swap(a[i],
       a[rev[i]]);
  for (int i = 1; i < lim; i <<= 1) {
    for (int j = 0, t = lim / i / 2; j < i; ++j) w[j] = wn[j
         * t];
    for (int j = 0; j < lim; j += i << 1)
      for (int k = 0; k < i; ++k) {
        const int x = a[k + j], y = (long long) a[k + j + i]
             * w[k] % mod;
        reduce(a[k + j] += y - mod), reduce(a[k + j + i] = x
             - y);
      }
  }
  if (!typ) {
    reverse(a.begin() + 1, a.begin() + lim);
    for (int i = 0; i < lim; ++i) a[i] = (long long) a[i] *
         inv_lim % mod;
  }
}
vector<int> multiply(vector<int> &f, vector<int> &g) {
  int n=(int)f.size() + (int)g.size() - 1;
  precompute(n);
  vector<int> a = f, b = g;
  a.resize(lim); b.resize(lim);
  ntt(a, 1), ntt(b, 1);
  for (int i = 0; i < lim; ++i) a[i] = (long long) a[i] *
       b[i] % mod;
  ntt(a, 0);
  //while((int)a.size() && a.back() == 0) a.pop_back();
  return a;
}
int fact[N], ifact[N];
vector<int> shift(vector<int> &f, int c) { //f(x + c)
  int n=(int)f.size();
  precompute(n + n - 1);
  vector<int> a = f; a.resize(lim);
  for (int i = 0; i < n; ++i) a[i] = (long long) a[i] *
       fact[i] % mod;
  reverse(a.begin(), a.begin()+n);
  vector<int> b; b.resize(lim); b[0] = 1;
  for (int i = 1; i < n; ++i) b[i] = (long long) b[i - 1] *
       c % mod;
  for (int i = 0; i < n; ++i) b[i] = (long long) b[i] *
       ifact[i] % mod;
  ntt(a, 1), ntt(b, 1);
  for (int i = 0; i < lim; ++i) a[i] = (long long) a[i] *
       b[i] % mod;
  ntt(a, 0), reverse(a.begin(), a.begin() + n);
  vector<int> g; g.resize(n);
  for (int i = 0; i < n; ++i) g[i] = (long long) a[i] *
       ifact[i] % mod;
  return g;
}
// (x+1)*(x+2)*(x+3)...(x+n)
```

```
// O(n log n) only for ntt friendly primes
// otherwise use divide and conquer in O(n log^2 n)
vector<int> range_mul(int n) {
  if (n == 0) return vector<int>({1});
  if (n & 1) {
    vector<int> f = range_mul(n - 1);
    f.push_back(0);
    for (int i = (int)f.size()-1; i; --i) f[i] = (f[i - 1] +
         (long long) n * f[i]) % mod;
    f[0] = (long long) f[0] * n % mod;
    return f;
  }
  else {
    int n_ = n >> 1;
    vector<int> f = range_mul(n_);
    vector<int> tmp = shift(f, n_);
    f.resize(n_ + 1);
    tmp.resize(n_ + 1);
    return multiply(f, tmp);
  }
}
// returns stirling1st(n, i) for 0 <= i <= n
vector<int> stirling(int n) {
  if (n == 0) return {1};
  vector<int> ans = range_mul(n - 1);
  ans.resize(n + 1);
  for (int i = n - 1; i >= 0; i--) {
    ans[i + 1] = ans[i];
  }
  ans[0] = 0;
  return ans;
}
int32_t main() {
  fact[0] = 1;
  for (int i = 1; i < N; ++i) fact[i] = (long long) fact[i
       - 1] * i % mod;
  ifact[N - 1] = POW(fact[N - 1], mod - 2);
  for (int i = N - 1; i; --i) ifact[i - 1] = (long long)
       ifact[i] * i % mod;
  int n; cin >> n;
  auto ans = stirling(n);
  for (int i = 0; i <= n; i++) {
    cout << ans[i] << ' ';
  }

}
```

2.17 Number series

```
Some interesting facts:
1. there are N boxes and K types of balls (each type has
     infinity amount), K<=N
 how many ways (mod p) are there to fill all the N boxes
      where every type should be used?
sol: S(N,K) * k! -----> using stirling number of second kind

combinatorics:
nc1+(n+1)c2+(n+2)c3+.....+(n+r-1)cr = (n+r)cr -1

Catalan number series:
1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786,
     208012

Stirling number of second kind:
   k:   0      1      2      3      4      5      6
-------------------------------------------------------------
n
0 :    1
1 :    0      1
2 :    0      1      1
3 :    0      1      3      1
4 :    0      1      7      6      1
5 :    0      1      15     25     10     1
6 :    0      1      31     90     65     15     1
7 :    0      1      63     301    350    140    21
8 :    0      1      127    966    1701   1050   266
9 :    0      1      255    3025   7770   6951   2646
10:    0      1      511    9330   34105  42525  22827

Stirling number of first kind:

  k:   0      1      2      3      4      5      6
-------------------------------------------------------------
n
0 :    1
1 :    0      1
2 :    0      1      1
3 :    0      2      3      1
4 :    0      6      11     6      1
5 :    0      24     50     35     10     1
6 :    0      120    274    225    85     15     1
7 :    0      720    1764   1624   735    175    21
8 :    0      5040   13068  13132  6769   1960   322
9 :    0      40320  109584 118124 67284  22449  4536
10:    0      362880 1026576 1172700 723680 269325 63273
```

3 Geometry

```
const int maxn = 1000001;
const double pi = acos(-1.0);
const double eps = 1e-9;
double ang;
typedef double T;
struct pt {
    T x, y;
    pt() {}
    pt(T _x, T _y) : x(_x), y(_y) {}
```

```
    pt operator+(const pt &p) const {
        return pt(x + p.x, y + p.y);
    }
    pt operator-(const pt &p) const {
        return pt(x - p.x, y - p.y);
    }
    pt operator*(const T &d) const {
        return pt(x * d, y * d);
    }
    pt operator/(const T &d) const {
        return pt(x / d, y / d);
    }
    bool operator==(const pt &p) const {
        return (x == p.x and y == p.y);
    }
    bool operator!=(const pt &p) const {
        return !(x == p.x and y == p.y);
    }
    bool operator<(const pt &p) const {
        if (x != p.x)
            return x < p.x;
        return y < p.y;
    }
};
T sq(pt p) {
    return p.x * p.x + p.y * p.y;
}
double abs(pt p) {
    return sqrt(sq(p));
}
pt translate(pt v, pt p) {
    return p + v;
}
pt scale(pt c, double factor, pt p) {
    return c + (p - c) * factor;
}
pt rot(pt p, double a) {
    return pt(p.x * cos(a) - p.y * sin(a), p.x * sin(a) +
        p.y * cos(a));
}
pt perp(pt p) {
    return pt(-p.y, p.x);
}
T dot(pt v, pt w) {
    return v.x * w.x + v.y * w.y;
}
bool isPerp(pt v, pt w) {
    return dot(v, w) == 0;
}
double smallAngle(pt v, pt w) {
    double cosTheta = dot(v, w) / abs(v) / abs(w);
    if (cosTheta < -1)
        cosTheta = -1;
    if (cosTheta > 1)
        cosTheta = 1;
    return acos(cosTheta);
}
T cross(pt v, pt w) {
    return v.x * w.y - v.y * w.x;
}
T orient(pt a, pt b, pt c) {
    return cross(b - a, c - a);
}
bool inAngle(pt a, pt b, pt c, pt x) {
    assert(orient(a, b, c) != 0);
    if (orient(a, b, c) < 0)
        swap(b, c);
    return orient(a, b, x) >= 0 and orient(a, c, x) <= 0;
}
//Line
struct line {
    pt v;
    T c;
    line() {}
    //From points P and Q
    line(pt p, pt q) {
        v = (q - p);
        c = cross(v, p);
    }
    //From equation ax + by = c
    line(T a, T b, T c) {
        v = pt(b, -a);
        c = c;
    }
    //From direction vector v and offset c
    line(pt v, T c) {
        v = v;
        c = c;
    }
    //These work with T = int / double
    T side(pt p);
    double dist(pt p);
    double sqDist(pt p);
    line perpThrough(pt p);
    bool cmpProj(pt p, pt q);
    line translate(pt t);
    //These require T = double
    line shiftLeft(double dist);
    pt proj(pt p);
    pt refl(pt p);
};
T line ::side(pt p) {
    return cross(v, p) - c;
}
double line ::dist(pt p) {
    return abs(side(p)) / abs(v);
}
double line ::sqDist(pt p) {
    return side(p) * side(p) / (double)sq(v);
}
line line ::perpThrough(pt p) {
    return line(p, p + perp(v));
}
bool line ::cmpProj(pt p, pt q) {
    return dot(v, p) < dot(v, q);
}
line line ::translate(pt t) {
    return line(v, c + cross(v, t));
}
line line ::shiftLeft(double dist) {
    return line(v, c + dist * abs(v));
}
bool areParallel(line l1, line l2) {
    return (l1.v.x * l2.v.y == l1.v.y * l2.v.x);
}
bool areSame(line l1, line l2) {
    return areParallel(l1, l2) and (l1.v.x * l2.c == l2.v.x
        * l1.c) and (l1.v.y * l2.c == l2.v.y * l1.c);
}
bool inter(line l1, line l2, pt &out) {
    T d = cross(l1.v, l2.v);
    if (d == 0)
        return false;
    out = (l2.v * l1.c - l1.v * l2.c) / d;
    return true;
}
pt line ::proj(pt p) {
    return p - perp(v) * side(p) / sq(v);
}
pt line ::refl(pt p) {
    return p - perp(v) * 2 * side(p) / sq(v);
}
line intBisector(line l1, line l2, bool interior) {
    assert(cross(l1.v, l2.v) != 0);
    double sign = interior ? 1 : -1;
    return line(l2.v / abs(l2.v) + l1.v * sign / abs(l1.v),
                l2.c / abs(l2.v) + l1.c * sign / abs(l1.v));
}
//Segment
bool inDisk(pt a, pt b, pt p) {
    return dot(a - p, b - p) <= 0;
}
bool onSegment(pt a, pt b, pt p) {
    return orient(a, b, p) == 0 and inDisk(a, b, p);
}
bool properInter(pt a, pt b, pt c, pt d, pt &i) {
    double oa = orient(c, d, a),
           ob = orient(c, d, b),
           oc = orient(a, b, c),
           od = orient(a, b, d);

    //Proper intersection exists iff opposite signs
    if (oa * ob < 0 and oc * od < 0)
    {
        i = (a * ob - b * oa) / (ob - oa);
        return true;
    }
    return false;
```

```
}
bool inters(pt a, pt b, pt c, pt d) {
    pt out;
    if (properInter(a, b, c, d, out))
        return true;
    if (onSegment(c, d, a))
        return true;
    if (onSegment(c, d, b))
        return true;
    if (onSegment(a, b, c))
        return true;
    if (onSegment(a, b, d))
        return true;
    return false;
}
double segPoint(pt a, pt b, pt p) {
    if (a != b) {
        line l(a, b);
        if (l.cmpProj(a, p) and l.cmpProj(p, b))
            return l.dist(p);
    }
    return min(abs(p - a), abs(p - b));
}
double segSeg(pt a, pt b, pt c, pt d) {
    pt dummy;
    if (properInter(a, b, c, d, dummy))
        return 0;
    return min(min(min(segPoint(a, b, c), segPoint(a, b,
        d)), segPoint(c, d, a)), segPoint(c, d, b));
}

int latticePoints (pt a, pt b){
    //requires int representation
    return __gcd (abs (a.x - b.x), abs (a.y - b.y)) + 1;
}
bool isConvex(vector<pt> &p) {
    bool hasPos = false, hasNeg = false;
    for (int i = 0, n = p.size(); i < n; i++) {
        int o = orient(p[i], p[(i + 1) % n], p[(i + 2) % n]);
        if (o > 0)
            hasPos = true;
        if (o < 0)
            hasNeg = true;
    }
    return !(hasPos and hasNeg);
}
double areaTriangle(pt a, pt b, pt c) {
    return abs(cross(b - a, c - a)) / 2.0;
}
double areaPolygon(const vector<pt> &p) {
    double area = 0.0;
    for (int i = 0, n = p.size(); i < n; i++) {
        area += cross(p[i], p[(i + 1) % n]);
    }
    return fabs(area) / 2.0;
}
```

```
ll is_point_in_convex(vector<pt>& p, pt &x) // O(log n)
{
    ll n = p.size();
    if (n < 3)
        return 1;
    ll a =orient(p[0], p[1], x), b = orient(p[0], p[n - 1],
        x);
    if (a < 0 || b > 0)
        return 1;
    ll l = 1, r = n - 1;
    while (l + 1 < r)
    {
        int mid = l + r >> 1;
        if (orient(p[0], p[mid], x) >= 0)
            l = mid;
        else
            r = mid;
    }
    ll k = orient(p[l], p[r], x);
    if (k <= 0)
        return -k;
    if (l == 1 && a == 0)
        return 0;
    if (r == n - 1 && b == 0)
        return 0;
    return -1;
}
bool pointInPolygon(const vector<pt> &p, pt q) //rezaul
    vai's
{
    bool c = false;
    for (int i = 0, n = p.size(); i < n; i++) {
        int j = (i + 1) % p.size();
        if ((p[i].y <= q.y and q.y < p[j].y or p[j].y <= q.y
            and q.y < p[i].y) and
            q.x < p[i].x + (p[j].x - p[i].x) * (q.y -
                p[i].y) / (p[j].y - p[i].y))
            c = !c;
    }
    return c;
}
pt centroidPolygon(const vector<pt> &p) {
    pt c(0, 0);
    double scale = 6.0 * areaPolygon(p);
    //    if (scale < eps) return c;
    for (int i = 0, n = p.size(); i < n; i++) {
        int j = (i + 1) % n;
        c = c + (p[i] + p[j]) * cross(p[i], p[j]);
    }
    return c / scale;
}
//Circle
pt circumCenter(pt a, pt b, pt c) {
    b = b - a;
    c = c - a;
    assert(cross(b, c) != 0);
    return a + perp(b * sq(c) - c * sq(b)) / cross(b, c) / 2;
}
bool circle2PtsRad(pt p1, pt p2, double r, pt &c) {
    double d2 = sq(p1 - p2);
    double det = r * r / d2 - 0.25;
    if (det < 0.0)
        return false;
    double h = sqrt(det);
    c.x = (p1.x + p2.x) * 0.5 + (p1.y - p2.y) * h;
    c.y = (p1.y + p2.y) * 0.5 + (p2.x - p1.x) * h;
    return true;
}
int circleLine(pt c, double r, line l, pair<pt, pt> &out) {
    double h2 = r * r - l.sqDist(c);
    if (h2 < 0)
        return 0; // the line doesn't touch the circle;
    pt p = l.proj(c);
    pt h = l.v * sqrt(h2) / abs(l.v);
    out = make_pair(p - h, p + h);
    return 1 + (h2 > 0);
}
int circleCircle(pt c1, double r1, pt c2, double r2,
    pair<pt, pt> &out) {
    pt d = c2 - c1;
    double d2 = sq(d);
    if (d2 == 0) { //concentric circles
        assert(r1 != r2);
        return 0;
    }
    double pd = (d2 + r1 * r1 - r2 * r2) / 2;
    double h2 = r1 * r1 - pd * pd / d2; // h ^ 2
    if (h2 < 0)
        return 0;
    pt p = c1 + d * pd / d2, h = perp(d) * sqrt(h2 / d2);
    out = make_pair(p - h, p + h);
    return 1 + h2 > 0;
}
int tangents(pt c1, double r1, pt c2, double r2, bool
    inner, vector<pair<pt, pt>> &out) {
    if (inner)
        r2 = -r2;
    pt d = c2 - c1;
    double dr = r1 - r2, d2 = sq(d), h2 = d2 - dr * dr;
    if (d2 == 0 or h2 < 0) {
        assert(h2 != 0);
        return 0;
    }
    for (int sign : {-1, 1}) {
        pt v = pt(d * dr + perp(d) * sqrt(h2) * sign) / d2;
        out.push_back(make_pair(c1 + v * r1, c2 + v * r2));
    }
    return 1 + (h2 > 0);
}
//Convex Hull - Monotone Chain
int sz; //returned polygon's size;
pt H[100000 + 5];
```

```
vector<pt> monotoneChain(vector<pt> &v) /// from
     you_know_who
{
    if(v.size()==1) return v;
        sort(v.begin(), v.end());
        vector<pt> up(2*v.size()+2), down(2*v.size()+2);
    int szup=0, szdw=0;
        for(int i=0;i<v.size();i++) {
        while(szup>1 && orient(up[szup-2], up[szup-1],
             v[i])>=0)
            szup--;
        while(szdw>1 && orient(down[szdw-2], down[szdw-1],
             v[i])<=0)
            szdw--;
        up[szup++]=v[i];
        down[szdw++]=v[i];
        }
    if(szdw>1) szdw--;
    reverse(up.begin(), up.begin()+szup);
        for(int i=0;i<szup-1;i++) down[szdw++]= up[i];
    if(szdw==2 && down[0].x==down[1].x &&
         down[0].y==down[1].y) szdw--;
    sz = szdw;
        return down;
}
pt toPoint(pt p1, pt p2) {
    return pt(p2.x-p1.x,p2.y - p1.y);
}
double angle(pt a, pt o, pt b) {
    //double result = atan2(P3.y - P1.y, P3.x - P1.x) -
         atan2(P2.y - P1.y, P2.x - P1.x);
    double result;
    pt oa = toPoint(o, a);
    pt ob = toPoint(o, b);
    if(sq(oa)==0 || sq(ob)==0){
        return 0.0;
    }
    result = acos(dot(oa, ob)/sqrt(sq(oa)*sq(ob)));
    result*=180.0;
    result/=Pi;
    return result;
}
struct circle {
    pt c;
    double r;
     circle() {}
    //From points P , radius rq
    circle(pt p, double rq)
    {
        c = p;
        r = rq;
    }
};
bool in_circle(circle C, pt p) {
    double dist = sq(C.c - p);
    dist= sqrt(dist);
    if(dist<=C.r) return 1;
    else return 0;
}
circle spanning_circle(vector<pt>& T) {
    int n = T.size();
   // random_shuffle(all(T));
    circle C(pt(), -100000000.0);
    for (int i = 0; i < n; i++) if (!in_circle(C, T[i])) {
        C = circle(T[i], 0);
        for (int j = 0; j < i; j++) if (!in_circle(C, T[j]))
             {
            C = circle((T[i] + T[j]) / 2, abs(T[i] - T[j]) /
                 2);
            for (int k = 0; k < j; k++) if (!in_circle(C,
                 T[k])) {
                pt o = circumCenter(T[i], T[j], T[k]);
                C = circle(o, abs(o - T[k]));
            }
        }
    }
    return C;
}
```

4 Data Structure

4.1 SQRT Decomposition

```
int n; int ara[10000]; int block[100]; int bs;
void update(int ind, int val) {
    block[ind/bs]+=val-ara[ind];
    ara[ind]=val;
}
int query(int a,int b) {
    int sum=0;
    for(int i=a;i<=b;){
        if(i%bs==0 and i+bs-1<=b) sum+=block[i/bs], i+=bs;
        else sum+=ara[i], i++;
    }
    return sum;
}
void pre() {
     bs=sqrtl(n);
     for(int i=0;i<n;i++) {
         block[i/bs]+=ara[i];
     }
}
int main() {
    cin>>n;
    for(int i=0;i<n;i++) cin>>ara[i];
    pre();
    update(0,0);
    cout<<query(0,3);
}
```

4.2 Mo's Algorithm

```
int n,m; int ara[mx]; int block;
ll ans[mx]; int cnt[mx]; int anss=0;
struct query {
    int l,r,id;
}q[mx];
bool comp(query a, query b) {
    if(a.l/block!=b.l/block) return a.l/block<b.l/block;
    return a.r<b.r;
}
void add(int ind) {
   if(cnt[ara[ind]]==0) anss++;
   cnt[ara[ind]]++;
}
void _remove(int ind) {
   if(cnt[ara[ind]]==1) anss--;
   cnt[ara[ind]]--;
}
void MO() {
    anss=0;
    int l=0,r=-1;
    for(int i=0;i<m;i++) {
        int left=q[i].l;
        int right=q[i].r;
        left--; right--;
        while(l<left) _remove(l++);
        while(l>left) add(--l);
        while(r<right) add(++r);
        while(r>right) _remove(r--);
        ans[q[i].id]=anss;
    }
}
int  main() {
    cin>>n>>m;
    for(int i=0;i<n;i++) cin>>ara[i];
    for(int i=0;i<m;i++) {
        cin>>q[i].l>>q[i].r;
        q[i].id=i;
    }
    block=sqrtl(n);
    sort(q,q+m,comp);
    MO();
    for( int i=0;i<m;i++) {
        cout<<ans[i]<<endl;
    }
}
```

4.3 Segment Tree(Lazy)

```
int ara[mx]; int tree[mx*4]; int lazy[mx*4];
void init(int node,int b, int e) {
    if(b==e) {
```

```
        tree[node]=ara[b];
        return ;
    }
    int mid=(b+e)>>1;
    init(2*node, b,mid);
    init(2*node+1,mid+1,e);
    tree[node]=tree[2*node]+tree[2*node+1];
}
void pushdown(int node,int b,int e) {
    if(lazy[node]) {
        tree[node]+=(e-b+1)*lazy[node];
        if(b!=e) {
            lazy[node*2]+=lazy[node];
            lazy[node*2+1]+=lazy[node];
        }
        lazy[node]=0;
    }
}
void update(int node,int b, int e, int i, int j,int value) {
    pushdown(node,b,e);
    if(e<i or b>j) return ;
    if(b>=i and e<=j) {
        lazy[node]+=value;
        pushdown(node,b,e);
        return ;
    }
    int mid=(b+e)>>1;
    update(2*node, b,mid,i,j,value);
    update(2*node+1,mid+1,e,i,j,value);
    tree[node]=tree[2*node]+tree[2*node+1];
}
int  query(int node,int b, int e, int i,int j) {
    pushdown(node,b,e);
    if(e<i or b>j) return 0;
    if(b>=i and e<=j) {
        return tree[node];
    }
    int mid=(b+e)>>1;
    int x=query(2*node, b,mid,i,j);
    int y=query(2*node+1,mid+1,e,i,j);
    return x+y;
}
```

4.4 Persistent Segment Tree

```
int ara[mx]; int cnt=1;
struct node {
    int l,r,val;
}tr[4*mx+mx*20];
void init(int node, int b, int e) {
    if(b==e) {
        tr[node].val=0;
        return ;
    }
    int mid=(b+e)>>1;
    tr[node].l=cnt++;
    tr[node].r=cnt++;
    init(tr[node].l,b,mid);
    init(tr[node].r,mid+1,e);
    tr[node].val=tr[tr[node].l].val+tr[tr[node].r].val;
}
int update(int pre, int b, int e, int i, int j,int val) {
    if(b>j or e<i) return pre;
    int curr=cnt++;
    if(b>=i and e<=j) {
        tr[curr].val=tr[pre].val+val;
        return curr;
    }
    int mid=(b+e)>>1;
    tr[curr].l=update(tr[pre].l,b,mid,i,j,val);
    tr[curr].r=update(tr[pre].r,mid+1,e,i,j,val);
    tr[curr].val=tr[tr[curr].l].val+tr[tr[curr].r].val;
    return curr;
}
ll query(ll pre, ll cur, ll b, ll e, ll k) {
    ll  mid=(b+e)>>1;
    if(b==e) return b;
    ll  val=tr[tr[cur].l].val-tr[tr[pre].l].val;
    if(val>=k) return query(tr[pre].l,tr[cur].l,b,mid,k);
    else return query(tr[pre].r,tr[cur].r,mid+1,e,k-val);
}
int root[mx];
int main() {
    //kth smallest number
    int n,q;
    cin>>n>>q;
    root[0]=0; init(root[0],0,n);
    vector<int> v;
    for(int i=0;i<n;i++) {
       cin>>ara[i];
       v.pb(ara[i]);
    }
    sort(v.begin(),v.end());
    map<int,int> mm;
    for(int i=0;i<n;i++) {
        int
            d=lower_bound(v.begin(),v.end(),ara[i])-v.begin();
        mm[d]=ara[i];
        ara[i]=d;
    }
    for(int i=1;i<=n;i++) {
       root[i]=update(root[i-1],0,n,ara[i-1],ara[i-1],1);
    }
    while(q--) {
       int l,r,k;
       cin>>l>>r>>k;
       cout<<mm[query(root[l-1],root[r],0,n,k)]<<endl;
    }
}
```

4.5 Persistent Segment Tree(Lazy)

```
struct node {
    int val,lazy;
    node *l,*r;
    node() {
        val=lazy=0; l=r=nullptr;
    }
    node(const node* p) {
        if(p==nullptr) {
            val=lazy=0; l=r=nullptr;
        }
        else {
            val=p->val; lazy=p->lazy;
            l=p->l; r=p->r;
        }
    }
};
void pushdown(node *&cur, int b, int e) {
    cur->l=new node(cur->l);
    cur->r=new node(cur->r);
    if(cur->lazy) {
        cur->val+=(e-b+1)*cur->lazy;
        if(b!=e) {
            cur->l->lazy+=cur->lazy;
            cur->r->lazy+=cur->lazy;
        }
        cur->lazy=0;
    }
}
void update(node *&cur, int b, int e, int i, int j, int
     val) {
    pushdown(cur,b,e);
    if(b>j or e<i) return ;
    if(b>=i and e<=j) {
        cur->val+=(e-b+1)*val;
        cur->l->lazy+=val;
        cur->r->lazy+=val;
        return ;
    }
    int mid=(b+e)>>1;
    update(cur->l,b,mid,i,j,val);
    update(cur->r,mid+1,e,i,j,val);
    cur->val=cur->l->val+cur->r->val;
}
int query(node *&cur, int b, int e, int i, int j) {
    pushdown(cur,b,e);
    if(b>j or e<i) return 0;
    if(b>=i and e<=j) return cur->val;
    int mid=(b+e)>>1;
    int x=query(cur->l,b,mid,i,j);
    int y=query(cur->r,mid+1,e,i,j);
    return x+y;
}
```

```
node *root[mx];
int ara[mx];
int main() {
    root[0]=new node();
    int n; cin>>n;
    for(int i=1;i<=n;i++) cin>>ara[i];
    for(int i=1;i<=n;i++) {
        root[i]=new node(root[i-1]);
        update(root[i],1,n,1,ara[i],5); //add 5 to the range
             1 to ara[i]
    }
}
```

4.6 RMQ(sparse table)

```
int ara[mx]; int mini[mx][22]; int n;
void sparse() {
    for(int i=0;i<n;i++) mini[i][0]=ara[i];
    for(int j=1;(1LL<<j)<=n;j++) {
        for(int i=0;i+(1LL<<(j))-1<n;i++) {
            int d=1LL<<(j-1);
            mini[i][j]=min(mini[i][j-1],mini[i+d][j-1]);
        }
    }
}
int query(int l, int r) {
    int d=__lg(r-l+1);
    return min(mini[l][d],mini[r-(1LL<<d)+1][d]);
}
int main() {
    cin>>n;
    for(int i=0;i<n;i++) cin>>ara[i];
    sparse();
    int q; cin>>q;
    while(q--) {
        int l,r;
        cin>>l>>r;
        cout<<query(l,r)<<endl;
    }
}
```

4.7 BIT

```
struct bit {
    int n;
    int tree[mx];
    void update(int ind, int value) {
        for(int i=ind;i<=n;i+=(i&-i)) {
            tree[i]+=value;
        }
    }
    int query(int ind)
    {
        int sum=0;
        for(int i=ind;i>0;i-=(i&-i)) {
            sum+=tree[i];
        }
        return sum;
    }
}a,b;
int main() {
    a.n=5; a.update(3,4);
    cout<<a.query(5)<<endl;
}
```

4.8 Implicit Treap

```
struct node {
    ll key,pri,sz,lazy,sum;
    node *l, *r; bool rev;
    node(){};
    node (ll val) {
        lazy=0,sum=val;
        rev=0; sz=0; key=val; pri=rng(); l=r=NULL;
    }
};
node* root=NULL;
int cnt(node *cur) {
    return cur? cur->sz: 0;
}
void push(node *cur) {
    if(cur==NULL) return ;
    if(cur and cur->lazy) {
        cur->key+=cur->lazy;
        cur->sum+=cur->lazy*cnt(cur);
        if(cur->l) cur->l->lazy+=cur->lazy;
        if(cur->r) cur->r->lazy+=cur->lazy;
        cur->lazy=0;
    }
    if(cur and cur->rev) {
        cur->revˆ=1;
        swap(cur->l,cur->r);
        if(cur->l) cur->l->revˆ=1;
        if(cur->r) cur->r->revˆ=1;
    }
}
vector<int> ans;
void path(node *cur) {
    if(cur==NULL) return ;
    push(cur); path(cur->l);
    cout<<cur->key<<' ';
    path(cur->r);
}
int getsum(node *cur) {
   if(cur==NULL) return 0;
   return cur->sum;
}
void update (node *cur) {
   if(cur==NULL) return ;
   push(cur->l); push(cur->r);
   cur->sum=cur->key;
   cur->sz=1+cnt(cur->l)+cnt(cur->r);
   cur->sum+=getsum(cur->l)+getsum(cur->r);
}
void split(node *cur,node *&l, node *&r,int pos, int add=0)
     {
    if(cur==NULL) {
        l=r=NULL;
        return ;
    }
    push(cur);
    int kk=add+cnt(cur->l);
    if(pos<=kk) split(cur->l,l,cur->l,pos,add), r=cur;
         //element at pos goes to node r
    else split(cur->r,cur->r,r,pos,kk+1), l=cur;
    update(cur);
}
void merge(node * &cur, node *l, node *r) {
    push(l); push(r);
    if(l==NULL or r==NULL) cur=l?l:r;
    else if(l->pri > r->pri) merge(l->r,l->r,r),cur=l;
    else merge(r->l,l,r->l),cur=r;
    update(cur);
}
void insert(node *&cur, node *it,int pos) {
    node *l,*r;
    split(cur,l,r,pos);
    merge(l,l,it); merge(cur,l,r);
}
void erase(node *&cur, int pos) {
    if(cur==NULL) return ;
    node *l,*mid,*r;
    split(cur,l,mid,pos);
    split(mid,mid,r,1);
    merge(cur,l,r);
}
void reverse(node *root, int a, int b) {
    node *l,*r,*mid;
    split(root,l,mid,a);
    split(mid,mid,r,b-a+1);
    mid->revˆ=1;
    merge(root,l,mid); merge(root,root,r);
}
void right_shift(node *root,int a, int b) {
    node *a1; //store 0-> b-1;
    node *a2; //store b-> n-1;
    split(root,a1,a2,b);
    node *bb; //store b->b
```

```
    split(a2,bb,a2,1);
    //now a2 will store b+1-> n-1
    node *a3; //store 0->a-1
    split(a1,a3,a1,a);
    //now a1 will store a->b-1
    merge(root,a3,bb); merge(a1,a1,a2);
    merge(root,root,a1);
}
void left_shift(node *root,int a, int b) {
    node *a1; //store 0-> b;
    node *a2; //store b+1-> n-1;
    split(root,a1,a2,b+1);
    node *a3; //store a+1 -> b
    split(a1,a1,a3,a+1);
    //now a1 will store 0 -> a
    node *aa; //store a -> a
    split(a1,a1,aa,a);
    //now a1 will store 0 -> a
    merge(a1,a1,a3); merge(a1,a1,aa);
    merge(root,a1,a2);
}
// 0 indexing treap
//merge(a,b,c) b and c will be merged and their root will
     be stored in a
//split(a,b,pos) a will store array index from (0 to
     pos-1), b will store array ( pos to n-1)
int main() {
   insert(root,new node(4),2);
   path(root);
}
```

5 Graph

5.1 2-SAT

```
vector<int> v[mx*2],rev[mx*2];
int scc[mx*2]; bool visit[mx*2];
stack<int> st;
void dfs(int s) {
    visit[s]=true;
    for(auto x: v[s] ) {
        if(visit[x]==false) {
            dfs(x);
        }
    }
    st.push(s);
}
void dfs1(int s,int no) {
    visit[s]=true; scc[s]=no;
    for(auto x: rev[s]) {
        if(visit[x]==false) {
            dfs1(x,no);
        }
    }
}
int n; int ans[mx];
bool twosat() {
    for(int i=0;i<2*n;i++) {
        if(visit[i]==false) {
            dfs(i);
        }
    }
    int mark=0;
    memset(visit,0,sizeof visit);
    while(st.size()) {
        int a=st.top();
        st.pop();
        if(visit[a]==false) {
            mark++;
            dfs1(a,mark);
        }
    }
    for(int i=0;i<n;i++) {
        if (scc[2*i] == scc[2*i+1]) return false;
        ans[2*i] = scc[2*i] > scc[2*i+1];
    }
    return true;
}
void add(int a, int b) {
    v[a].pb(b);
    rev[b].pb(a);
}
int main() {
    cin>>n;
    for(int i=0;i<n;i++) {
        int a,b;
        cin>>a>>b;
        add(a,b);
    }
    if(twosat()==0) no();
    else yes();
}
```

5.2 Articulation Bridge

```
int n;
vector<int>v[mx]; int dis[mx],low[mx];
bool visit[mx]; int t=0;
set<pair<int,int> > edge;
void dfs( int s, int par) {
    dis[s]=low[s]=t++;
    int child=0; visit[s]=true;
    for(auto x: v[s]) {
        if(x==par) continue;
        if(visit[x]) {
            low[s]=min(low[s],dis[x]);
            continue;
        }
        dfs(x,s); child++;
        low[s]=min(low[s],low[x]);
        if(dis[s]<low[x]) {
            edge.insert(mp(min(s,x),max(s,x)));
        }
    }
}
int  main() {
        int m ;
        cin>>n>>m;
        for(int i=0;i<m;i++) {
            int a,b;
            cin>>a>>b;
            v[a].pb(b);
            v[b].pb(a);
        }
        for(int i=0;i<n;i++) {
            if(visit[i]==false) {
                dfs(i,-1);
            }
        }
        printf("%d critical links\n",edge.size());
        for(auto x: edge){
            printf("%d - %d\n",x.first,x.second);
        }
}
```

5.3 Articulation Point

```
vector<int> v[100];
bool visit[100]; int t ;
int dis[100]; int low[100]; bool ans[100];
void dfs(int s,int par){
    dis[s]=t++; low[s]=dis[s];
    visit[s]=true; int child=0;
    for(auto x: v[s]) {
        if(x==par) continue;
        if(visit[x]) {
            low[s]=min(low[s],dis[x]);
            continue;
        }
        dfs(x,s); low[s]=min(low[s],low[x]);
        if(dis[s]<=low[x] and par!=-1) {
            ans[s]=true; ///one point can be counted more
                 than once
        }
        child++;
        if(child>1 and par==-1) ans[s]=true;
    }
}
int main() {
        v[1].pb(2);
        dfs(1,-1);
```

```
        for(int i=1;i<=7;i++) {
            if(ans[i]) cout<<i<<' ' ;
        }
}
```

5.4 Centroid decomposition

```
vector<ll > v[mx];
ll sub[mx],visit[mx];
ll k,ans,cnt[mx];
void dfs(ll s, ll par) {
    sub[s]=1;
    for(auto x: v[s]) {
        if(x==par or visit[x]) continue;
        dfs(x,s);
        sub[s]+=sub[x];
    }
}
ll centroid(ll s,ll par, ll tot) {
    for(auto x: v[s]) {
        if(visit[x] or x==par) continue;
        if(sub[x]*2>tot) return centroid(x,s,tot);
    }
    return s;
}
void dfs1(ll s, ll par,ll lv,ll c) {
    cnt[lv]+=c ;
    for(auto x: v[s]) {
        if(visit[x] or x==par ) continue;
        dfs1(x,s,lv+1,c);
    }
}
void path(ll s, ll par, ll lv) {
    if(lv<=k) ans+=cnt[k-lv];
    for(auto x: v[s]) {
        if(visit[x] or par==x) continue;
        path(x,s,lv+1);
    }
}
void solve(ll s) {
    cnt[0]=2;
    for(auto x: v[s]) {
        if(visit[x]) continue;
        dfs1(x,s,1,1);
    }
    for(auto x: v[s]) {
        if(visit[x]) continue;
        dfs1(x,s,1,-1);
        path(x,s,1);
        dfs1(x,s,1,1);
    }
    for(auto x: v[s]) {
        if(visit[x]) continue;
        dfs1(x,s,1,-1);
    }
}
void build(ll s) ///O(nlogn)
{
    dfs(s,-1);
    ll  d=centroid(s,-1,sub[s]);
    solve(d);
    visit[d]=true;
    for(auto x: v[d]) {
        if(visit[x]) continue;
        build(x);
    }
}
int main() {
    //pair of nodes with exact k distance
    ll  n;
    cin>>n>>k;
    for(ll i=0;i<n-1;i++) {
        ll a,b;
        cin>>a>>b;
        v[a].pb(b);
        v[b].pb(a);
    }
    build(1);
    cout<<ans/2<<endl;
}
```

5.5 Dijkstra

```
ll n,m;
vector<pair<ll,ll> > v[100008];
ll dis[100009]; ll par[100009];
void dij() {
    for(ll i=0;i<=n;i++) dis[i]=LLONG_MAX;
    priority_queue<pair<ll,ll> > pq;
    pq.push(mp(0,1));
    dis[1]=0; par[1]=-1;
    par[n]=INT_MAX;
    while(pq.size()) {
        ll u=pq.top().second;
        int cost=-pq.top().first;
        pq.pop();
        if(cost>dis[u]) continue;
        for(auto x: v[u]) {
            if(cost+x.second<dis[x.first]) {
                dis[x.first]=cost+x.second;
                pq.push(mp(-dis[x.first],x.first));
                par[x.first]=u;
            }
        }
    }
}
void  path(ll x) {
    if(x==-1) return ;
    path(par[x]);
    cout<<x<<' ';
}
int  main() {
    cin>>n>>m;
    for(ll i=0;i<m;i++) {
        ll a,b,c;
        cin>>a>>b>>c;
        v[a].pb(mp(b,c));
        v[b].pb(mp(a,c));
    }
    dij();
    if(par[n]==INT_MAX) cout<<-1<<endl;
    else path(n);
}
```

5.6 Heavy Light Dec.

```
vector<ll > v[mx];
int level[mx],heavy[mx],head[mx],pos[mx],parent[mx];
int curr; int tree[4*mx];
int ara[mx]; int n;
void init(int node, int b, int e) {
    if(b==e) {
        tree[node]=ara[b];
        return ;
    }
    int mid=(b+e)>>1;
    init(node*2,b,mid);
    init(node*2+1,mid+1,e);
    tree[node]=max(tree[node*2],tree[node*2+1]);
}
void update(int node, int b, int e, int i, int val) {
    if(b>i or e<i) return ;
    if(b==e) {
        tree[node]=val;
        return ;
    }
    int mid=(b+e)>>1;
    update(node*2,b,mid,i,val);
    update(node*2+1,mid+1,e,i,val);
    tree[node]=max(tree[node*2],tree[node*2+1]);
}
int query(int node, int b, int e, int i, int j) {
    if(b>j or e<i) return 0;
    if(b>=i and e<=j) return tree[node];
    int mid=(b+e)>>1;
    return
         max(query(node*2,b,mid,i,j),query(node*2+1,mid+1,e,i,j));
}
int dfs(int s, int par) {
    int ss=1; ///subtree size
    int maxi=0;
    for(auto x: v[s]) {
```

```
        if(x!=par) {
            parent[x]=s; level[x]=level[s]+1;
            int d=dfs(x,s); ///returns the subtree size of x
            ss+=d;
            if(d>maxi) {
                maxi=d; heavy[s]=x;
            }
        }
    }
    return ss;
}
void hld(int s, int h) {
    head[s]=h; ///head of chain in which s lies
    pos[s]=curr; ///position of s in the segment tree
    ara[curr++]=s;
    if(heavy[s]!=-1) {
        hld(heavy[s],h); ///increase chain
    }
    for(auto x: v[s]) {
        if(x!=parent[s] and x!=heavy[s]) {
            hld(x,x); ///start new chain
        }
    }
}
int solve(int a, int b) {
    int ans=0;
    for(; head[a]!=head[b];a=parent[head[a]]) ///maximum
         log(n) times
    {
        if(level[head[a]]<level[head[b]]) swap(a,b);
        int temp=query(1,0,n-1,pos[head[a]],pos[a]);
        ans=max(temp,ans);
    }
    if(level[a]>level[b]) swap(a,b);
    ans=max(ans,query(1,0,n-1,pos[a],pos[b]));
    return ans;
}
int main() {
    cin>>n;
    for(int i=0;i<n-1;i++) {
        int a,b;
        cin>>a>>b;
        v[a].pb(b);
        v[b].pb(a);
    }
    memset(heavy,-1,sizeof heavy);
    dfs(1,-1); hld(1,1);
    init(1,0,n-1);
    cout<<solve(1,6)<<endl; ///maximum value on the path
         from a to b
}
```

5.7 LCA

```
int n; int level[mx];
int table[mx][22];
vector<int>v[mx];
void dfs(int curr,int pre) {
    if(pre==-1) level[curr]=0;
    else level [curr]=level[pre]+1;
    table[curr][0]=pre;
    for(auto x: v[curr]) {
        if(x==pre) continue;
        dfs(x,curr);
    }
}
int lca(int p, int q) {
     if(level[p]<level[q]) swap(p,q);
     int log=__lg(level[p]);
     for( int i=log;i>=0;i--) {
         if(level[p]-(1<<i)>=level[q]) {
             p=table[p][i];
         }
     }
     if(p==q) return p ;
     for( int i=log;i>=0;i--) {
         if(table[p][i]!=-1 and table[p][i]!=table[q][i]) {
             p=table[p][i],q=table[q][i];
         }
     }
     return table[p][0];
}
void sparse() {
    ///node number starts from 0
    for(int j=1;1<<j<=n;j++) {
         for(int i=0;i<n;i++) {
             if(table [i][j-1]!=-1) {
                 table [i][j]=table [table[i][j-1]][j-1];
             }
         }
     }
}
int main() {
    n=6;
        v[0].pb(2);
        memset(table,-1,sizeof table);
        dfs(0,-1);
        sparse();
        printf( "%d\n",lca(6,5) );
}
```

5.8 Finding bridges (online)

```
vector<int> par, dsu_2ecc, dsu_cc, dsu_cc_size;
int bridges;
int lca_iteration;
vector<int> last_visit;

void init(int n) {
    par.resize(n);
    dsu_2ecc.resize(n);
    dsu_cc.resize(n);
    dsu_cc_size.resize(n);
    lca_iteration = 0;
    last_visit.assign(n, 0);
    for (int i=0; i<n; ++i) {
        dsu_2ecc[i] = i;
        dsu_cc[i] = i;
        dsu_cc_size[i] = 1;
        par[i] = -1;
    }
    bridges = 0;
}
int find_2ecc(int v) {
    if (v == -1)
        return -1;
    return dsu_2ecc[v] == v ? v : dsu_2ecc[v] =
         find_2ecc(dsu_2ecc[v]);
}
int find_cc(int v) {
    v = find_2ecc(v);
    return dsu_cc[v] == v ? v : dsu_cc[v] =
         find_cc(dsu_cc[v]);
}
void make_root(int v) {
    v = find_2ecc(v);
    int root = v;
    int child = -1;
    while (v != -1) {
        int p = find_2ecc(par[v]);
        par[v] = child;
        dsu_cc[v] = root;
        child = v;
        v = p;
    }
    dsu_cc_size[root] = dsu_cc_size[child];
}
void merge_path (int a, int b) {
    ++lca_iteration;
    vector<int> path_a, path_b;
    int lca = -1;
    while (lca == -1) {
        if (a != -1) {
            a = find_2ecc(a);
            path_a.push_back(a);
            if (last_visit[a] == lca_iteration){
                lca = a;
                break;
                }
            last_visit[a] = lca_iteration;
            a = par[a];
        }
        if (b != -1) {
            b = find_2ecc(b);
```

```
            path_b.push_back(b);
            if (last_visit[b] == lca_iteration){
                lca = b;
                break;
                }
            last_visit[b] = lca_iteration;
            b = par[b];
        }

    }
    for (int v : path_a) {
        dsu_2ecc[v] = lca;
        if (v == lca)
            break;
        --bridges;
    }
    for (int v : path_b) {
        dsu_2ecc[v] = lca;
        if (v == lca)
            break;
        --bridges;
    }
}
void add_edge(int a, int b) {
    a = find_2ecc(a);
    b = find_2ecc(b);
    if (a == b)
        return;

    int ca = find_cc(a);
    int cb = find_cc(b);

    if (ca != cb) {
        ++bridges;
        if (dsu_cc_size[ca] > dsu_cc_size[cb]) {
            swap(a, b);
            swap(ca, cb);
        }
        make_root(a);
        par[a] = dsu_cc[a] = b;
        dsu_cc_size[cb] += dsu_cc_size[a];
    } else {
        merge_path(a, b);
    }
}
int main() {
    int n,m;
    cin>>n>>m;
    init(n+5);
    for(int i=0;i<m;i++) {
        int a,b;
        cin>>a>>b;
        add_edge(a,b);
        cout<<bridges<<endl;
    }
}
```

5.9 SCC

```
vector<int> v[10009],rev[10009],scc[10009];
bool visit[10009];
stack<int> st;
void dfs(int s) {
    visit[s]=true;
    for(auto x: v[s] ) {
        if(visit[x]==false) {
            dfs(x);
        }
    }
    st.push(s);
}
void dfs1(int s,int no) {
    visit[s]=true;
    scc[no].pb(s);
    for(auto x: rev[s]) {
        if(visit[x]==false) {
            dfs1(x,no);
        }
    }
}
int  main() {
       ///only for directed graph
     int n;
     scanf("%d",&n);
     set<int> s;
     for(int i=0;i<n;i++) {
         int  a,b;
         scanf("%d%d",&a,&b);
         v[a].pb(b);
         rev[b].pb(a);
         s.insert(a);
         s.insert(b);
     }
     for(auto x: s) {
         if(visit[x]==false) {
             dfs(x);
         }
     }
     int mark=0;
     memset(visit,0,sizeof visit);
     while(st.size()) {
         int a=st.top();
         st.pop();
         if(visit[a]==false)
         {
             mark++;
             dfs1(a,mark);
         }
     }
     for(int i=1;i<=mark;i++) {
         cout<<i<<": ";
         for(auto x:scc[i]) {
             cout<<x<<' ';
         }
         cout<<endl;
     }
}
```

5.10 Kruskal

```
struct edge {
    ll u,v,w;
};
ll n,m; vector<edge> v;
ll par[1000000];
ll findp(ll x) {
    if(par[x]==x) return x;
    return par[x]=findp(par[x]);
}
bool cmp(edge a, edge b) {
    return a.w<b.w;
}
void mst() {
    sort(v.begin(),v.end(),cmp);
    ll ans=0;
    for(ll i=1;i<=n;i++) par[i]=i;
    for(ll i=0;i<v.size();i++) {
        ll a=v[i].u; ll b=v[i].v;
        ll cost=v[i].w;
        ll m1=findp(a); ll m2=findp(b);
        if(m1!=m2) {
            par[m1]=m2;
            ans+=cost;
        }
    }
    int cnt=0;
    for(ll i=1;i<=n;i++) {
        ll x=findp(i);
        if(x==i) cnt++;
    }
    if(cnt>1) cout<<"IMPOSSIBLE"<<endl;
    else cout<<ans<<endl;
}
```

5.11 Bellman Ford

```
struct edge {
    int a,b,c;
};
vector<edge> v;
ll dis[1009];
int n,m;
void bford() {
    for(int i=0;i<n;i++) dis[i]=INT_MAX;
```

```
    dis[0]=0;
    for(int i=0;i<n-1;i++) {
        for(int j=0;j<v.size();j++) {
            int a=v[j].a;
            int b=v[j].b;
            int c=v[j].c;
            if(dis[a]==INT_MAX) continue ;
            if(dis[a]+c<dis[b]) {
                dis[b]=dis[a]+c;
            }
        }
    }
    int k=0;
    for(int j=0;j<v.size();j++) {
        int a=v[j].a;
        int b=v[j].b;
        int c=v[j].c;
        if(dis[a]+c<dis[b]) {
            k=1;
            break;
        }
    }
    if(k) cout<<"find negative cycle"<<endl;
    else cout<<"no negative cycle"<<endl;
}
int main() {
    ///check negative cycle
    cin>>n>>m;
    for(int i=0;i<m;i++) {
        edge d;
        cin>>d.a>>d.b>>d.c;
        v.pb(d);
    }
    bford();
}
```

5.12 Dinic

```
const int mx = 500+5 ;
const int INF = 1000000000 ;
struct edge {
    int a, b, cap, flow ;
};
int src,snk,level[mx],ptr[mx] ;
vector<edge>e;
vector<int>g[mx];
void add(int a,int b,int cap) {
    edge e1 = { a, b, cap, 0 } ;
    edge e2 = { b, a, 0 , 0 } ;
    g[a].push_back((int)e.size());
    e.push_back(e1);
    g[b].push_back((int)e.size());
    e.push_back(e2);
}
bool bfs() {
    int s = src , t = snk ;
    queue < int > q ;
    q.push(s);
    memset(level,-1,sizeof(level));
    level[s]=0 ;
    while (!q.empty()) {
        int u=q.front() ;
        q.pop() ;
        for(int i=0; i<(int)g[u].size(); i++) {
            int id=g[u][i];
            int v=e[id].b;
            if(level[v]==-1&&e[id].flow<e[id].cap) {
                q.push(v) ;
                level[v]=level[u]+1 ;
            }
        }
    }
    return level[t]!=-1 ;
}
int dfs(int u,int flow) {
    if (!flow) return 0 ;
    if ( u==snk ) return flow ;
    for(; ptr[u]<(int)g[u].size(); ptr[u]++) {
        int id = g[ u][ptr[u]];
        int v = e[id].b ;
        if ( level[v] != level[u]+1 ) continue ;
        int tempflow = dfs (v,min
             (flow,e[id].cap-e[id].flow)) ;
        if (tempflow) {
            e [id].flow+=tempflow ;
            e [id^1].flow-=tempflow ;
            return tempflow ;
        }
    }
    return 0 ;
}
int dinic() {
    int flow = 0 ;
    while(bfs()) {
        memset(ptr,0,sizeof(ptr) ) ;
        while ( int tempflow= dfs(src,INF ) )
            flow += tempflow ;
    }
    return flow ;
}
int main() {
   // complexity O(v*v*e)
    int n,m;
    cin>>n>>m;
    cin>>src>>snk;
    for(int i=0;i<m;i++) {
        int a,b,c;
        cin>>a>>b>>c;
        add(a,b,c);
        add(b,a,c);
    }
    cout<<dinic()<<endl;
}
```

5.13 Hopcroft Karp

```
const ll mx=1009;
const ll inf=(ll)1e15;
vector<ll > v[mx];
ll  n, match[mx], dis[mx];
void add(ll a, ll b) {
    v[a].pb(b);
}
bool  bfs() {
    queue<ll > q;
    for(ll  i=1;i<=n;i++) {
        if(match[i]==0) {
            dis[i]=0; q.push(i);
        }
        else dis[i]=inf;
    }
    dis[0]=inf;
    while(q.size()) {
        ll  a=q.front(); q.pop();
        for(auto x: v[a]) {
            if(dis[match[x]]==inf) {
                dis[match[x]]=dis[a]+1;
                q.push(match[x]);
            }
        }
    }
    return dis[0]!=inf;
}
bool dfs(ll u) {
    if(u) {
        for(auto x: v[u]) {
            if(dis[match[x]]==dis[u]+1) {
                if(dfs(match[x])) {
                    match[x]=u;
                    match[u]=x;
                    return true;
                }
            }
        }
        dis[u]=inf;
        return false;
    }
    return true;
}
ll  hopcroft() {
    ll  ans=0;
    while(bfs()) {
        for(ll  i=1;i<=n;i++) {
            if(match[i]==false and dfs(i)) {
```

```
                ans++;
            }
        }
    }
    return ans;
}
int main() {
    ///complexity O(sqrt(v)*e)
    ///find maximum matching in unweighted graph
    n=4; ///elements of the left side, start from 1
    add(1,6); add(1,7); add(2,5); add(3,6);
    add(4,6); add(4,8);
    cout<<hopcroft()<<endl; ///ans 4
}
minimum vertex cover (cover all edges)      : bpm
minimum edge cover (cover all vertices)     : n-bpm
size of maximum independent set             : n-bpm
```

5.14 Biconnected Comp.

```
vector<int> v[100];
bool visit[100]; int t ;
int dis[100]; int low[100];
bool ans[100];
stack<pair<int,int> > st;
void dfs(int s,int par) {
    dis[s]=t++;
    low[s]=dis[s];
    visit[s]=true;
    int child=0;
    for(auto x: v[s]) {
        if(x==par) continue;
        if(visit[x]) {
            low[s]=min(low[s],dis[x]);
            if(dis[x]<dis[s]) st.push(mp(s,x));
            continue;
        }
        st.push(mp(s,x));
        dfs(x,s);
        low[s]=min(low[s],low[x]);
        if(dis[s]<=low[x] and par!=-1) {
            while(st.top().first!=s or st.top().second!=x) {
                cout<<st.top().first<<'
                    '<<st.top().second<<endl;
                st.pop();
            }
            cout<<s<<' '<<x<<endl;
            st.pop();
        }
        child++;
        if(child>1 and par==-1) {
            while(st.top().first!=s or st.top().second!=x) {
                cout<<st.top().first<<'
                    '<<st.top().second<<endl;
                st.pop();
            }
            cout<<s<<' '<<x<<endl;
            st.pop();
        }
    }
}
int main() {
    ///print biconnected component
    v[1].pb(5);
    dfs(1,-1);
    while(st.size()) {
        cout<<st.top().first<<' '<<st.top().second<<endl;
        st.pop();
    }
}
```

6 DP

6.1 Digit Dp

```
vector<ll >v;
ll dp[16][2][180];
ll call(ll ind, bool check,ll sum) {
    if(ind==-1) return sum;
    ll cnt=0;
    ll limit=check? (v[ind]):9;
    ll &ret=dp[ind][check][sum];
    if(ret!=-1 and !check) return ret;
    for(ll i=0;i<=limit;i++) {
        bool ck=(i==v[ind])? check:0;
        cnt+=call(ind-1,ck,sum+i);
    }
    if(!check) ret=cnt; ///is used so that we can call this
         function without memset for another query
    return cnt;
}
ll get(ll n) {
    v.clear();
    while(n) {
        v.pb(n%10); n/=10;
    }
    return call((int)v.size()-1,1,0);
}
int main() {
    ll  t;
    cin>>t;
    memset(dp,-1,sizeof dp);
    while(t--){
    ll a,b; ///take string for large number
    cin>>a>>b;
    cout<<get(b)-get(a-1)<<endl;
    }
}
```

6.2 Convex Hull(dynamic)

```
const ll is_query = -(1LL<<62);
struct Line {
    ll m, c;
    mutable function<const Line*()> succ;
    bool operator<(const Line& rhs) const {
        if (rhs.c != is_query) return m < rhs.m;
        const Line* s = succ();
        if (!s) return 0;
        ll x = rhs.m;
        return c - s->c < (s->m - m) * x;
    }
};
struct HullDynamic : public multiset<Line> // will maintain
     upper hull for maximum
{
    bool bad(iterator y) {
        auto z = next(y);
        if (y == begin()) {
            if (z == end()) return 0;
            return y->m == z->m && y->c <= z->c;
        }
        auto x = prev(y);
        if (z == end()) return y->m == x->m && y->c <= x->c;
        return 1.0*(x->c - y->c)*(z->m - y->m) >= 1.0*(y->c
             - z->c)*(y->m - x->m);
    }
    void addline(ll m, ll c) {
        auto y = insert({ m, c });
        y->succ = [=] { return next(y) == end() ? 0 :
             &*next(y); };
        if (bad(y)) { erase(y); return; }
        while (next(y) != end() && bad(next(y)))
             erase(next(y));
        while (y != begin() && bad(prev(y))) erase(prev(y));
    }
    ll query(ll x) {
        auto l = *lower_bound((Line) { x, is_query });
        return l.m * x + l.c;
    }
}a;
int main() {
    //complexity nlog^(2)n (proti line add er jonne O(logn))
    rules:
    Keeps upper hull for maximum.
    add lines with -m and -c and return -ans to
    make this code working for minimum.
}
```

6.3 Convex Hull(semi offline)

```
ll s[mx],p[mx];
ll ara[mx];
struct cht {
    vector<ll> m,c;
    bool bad(ll f1, ll f2, ll f3) {
        ///change sign depends on slope and upper/lower
             envelop
        return 1.0*(c[f3]-c[f1])*(m[f1]-m[f2])>=
        1.0*(c[f2]-c[f1])*(m[f1]-m[f3]);
    }
    void addline(ll m1,ll c1) {
        m.pb(m1);
        c.pb(c1); ll sz=m.size();
        while(sz>=3 and bad(sz-3,sz-2,sz-1)) {
            m.erase(m.end()-2);
            c.erase(c.end()-2);
            sz--;
        }
    }
    ll value(ll ind, ll x) {
        return m[ind]*x+c[ind];
    }
    ll query(ll x) {
        ll low=0,high=(int)m.size()-1;
        ll ans=-LLONG_MAX;
        while(low<=high) {
            int mid1 = low + (high-low)/3;
            int mid2 = high - (high-low)/3;
            if(value(mid1,x)>=value(mid2,x)) {
                ans=value(mid1,x);
                high=mid2-1;
            }
            else {
                ans=value(mid2,x);
                low=mid1+1;
            }
        }
        return ans;
    }
}a;
int main() {
    //complexity nlogn
    ll n;
    cin>>n;
    for(ll i=1;i<=n;i++) {
        cin>>ara[i];
        s[i]=s[i-1]+ara[i];
        p[i]=p[i-1]+i*ara[i];
    }
    //a.addline(0,0); if query occurs before adding line
    ll ans=0;
    for(int i=1;i<=n;i++) {
        a.addline(-i,i*s[i-1]-p[i-1]-s[i-1]);
        ans=max(ans,a.query(s[i])+p[i]+s[i]);
    }
    cout<<ans<<endl;
}
rules:
bool bad part:
-------------
slope decrease and find minimum: <=
slope decrease and find maximum: >=
-----------------------------------
//slope increase hole sign ulta hobe
-----------------------------------
slope increase and find minimum: >=
slope increase and find maximum: <=
query part:
find minimum:
>>>>>>>>>>>>>>>>>>ans<<<<<<<<<<<<<<<<<<
find maximum:
<<<<<<<<<<<<<<<<<<ans>>>>>>>>>>>>>>>>>>
```

6.4 Subset DP

```
for(int i = 0; i<(1<<N); ++i)
       F[i] = A[i];
for(int i = 0;i < N; ++i) for(int mask = 0; mask < (1<<N);
     ++mask){
        if(mask & (1<<i))
               F[mask] += F[mask^(1<<i)];
}
```

7 String

7.1 Aho Corasick

```
ll node=1;
ll nxt[mx][68]; ll link[mx];
vector<int> mark[mx];
ll ans[mx]; ll id;
void add(string &pat) {
    ll now=0;
    for(auto x: pat) {
        ll d=x-'A';
        if(nxt[now][d]==-1) {
            link[node]=0;
            mark[node].clear();
            nxt[now][d]=node++;
        }
        now=nxt[now][d];
    }
    mark[now].pb(id);
}
void aho() {
    queue<ll> q;
    ll root=0;
    q.push(root);
    while(q.size()) {
        ll a=q.front();
        q.pop();
        for(ll i=0;i<60;i++) {
            ll d=nxt[a][i];
            if(d!=-1) {
                if(a==root) link[d]=root;
                else {
                    link[d]=nxt[link[a]][i];
                    ll e=nxt[link[a]][i];
                    for(auto x: mark[e]) mark[d].pb(x);
                    /// add all pattern no to node d which
                         ends to its link node
                }
                q.push(d);
            }
            else {
                if(a==root) nxt[a][i]=root;
                else nxt[a][i]=nxt[link[a]][i];
            }
        }
    }
}
void match(string &txt) {
   ll now=0;
   memset(ans,0,sizeof ans);
   for(ll i=0;i<txt.size();i++) {
     ll d=txt[i]-'A';
     now=nxt[now][d];
     for(auto x: mark[now]) ans[x]=true;
    }
}
int main() {
    ///find whether the pattern is in the text of not
    ///time complexity(total length of (pattern +text) + no
         of occurrences)
    node=1;
    memset(nxt,-1,sizeof nxt);
    int n; string pat,txt;
    cin>>txt>>n;
    for(ll i=0;i<n;i++) {
        cin>>pat;
        id=i+1; add(pat);
    }
    aho(); match(txt);
    for(int i=1;i<=n;i++) {
        if(ans[i]) cout<<"y"<<endl;
        else cout<<"n"<<endl ;
    }
}
```

7.2 Hashing

```
#define mod 1000000007
#define mod1 1000000009
ll p[mx],p1[mx];
ll base=13331LL;
ll base1=23333LL;
void pre() {
    p[0]=1;
    p1[0]=1;
    for(ll i=1;i<mx;i++) {
        p[i]=(p[i-1]*base)%mod;
        p1[i]=(p1[i-1]*base1)%mod1;
    }
}
struct Hash {
    ll h[mx],h1[mx];
    ll n;
    void init(string &str) {
        n=str.size();
        h[0]=0; h1[0]=0;
        for(ll i=1;i<=n;i++) {
            h[i]=h[i-1]*base+str[i-1];
            h[i]%=mod;
            h1[i]=h1[i-1]*base1+str[i-1];
            h1[i]%=mod1;
        }
    }
    ll range(ll l,ll r) {
        ll a=h[r]-h[l-1]*p[r-l+1];
        a%=mod; a=(a+mod)%mod;
        ll b=h1[r]-h1[l-1]*p1[r-l+1];
        b%=mod1; b=(b+mod1)%mod1;
        return (a<<18)^b;
    }
}a,b;
//1 indexing
```

7.3 KMP

```
string str,pat;
int ara[2*50009]; ///don't need to memset before every case
int pre() {
    int now=0;
    ara[0]=now;
    for(int i=1;i<pat.size();i++) {
        while(now and pat[now]!=pat[i]) now=ara[now-1];
        if(pat[now]==pat[i]) now++;
        ara[i]=now;
    }
}
bool match() {
    int now=0;
    for(int i=0;i<str.size();i++) {
        while(now and str[i]!=pat[now]) now=ara[now-1];
        if(str[i]==pat[now]) now++;
        else now=0;
        if(now==pat.size()) return 1;
    }
    return 0;
}
int main() {
    cin>>str>>pat;
    pre();
    for(int i=0;i<pat.size();i++) cout<<ara[i]<<' ';
}
```

7.4 Manacher

```
int d[2][mx];
string str;
//d[0][i]=total number of odd length palindromes whose
    centre is i
//d[1][i]=total number of even length palindromes whose
    centre is i
void pre() {
    int n=(int)str.size();
    for(int j=0;j<2;j++) {
        for(int i=0,l=0,r=-1;i<n;i++) {
            int k=(i>r)?(!j):min(d[j][l+r-i+j],r-i+1);
            while(i-k-j>=0 and i+k<n and
                str[i-k-j]==str[i+k]) k++;
            d[j][i]=k--;
            if(i+k>r) l=i-k-j,r=i+k;
        }
    }
}
int main() {
    cin>>str;
    pre();
}
```

7.5 Suffix Array

```
struct SA {
    vector<int> sa, lcp;
    SA(){};
    void build(string& s, int lim = 256) {
        int n = s.size() + 1, k = 0, a, b;
        vector<int> x(s.begin(), s.end() + 1), y(n),
            ws(max(n, lim)), rank(n);
        sa = lcp = y;
        iota(sa.begin(), sa.end(), 0);
        for (int j = 0, p = 0; p < n; j = max(1, j * 2), lim
            = p) {
            p = j, iota(y.begin(), y.end(), n - j);
            for (int i = 0; i <= n - 1; i++) {
                if (sa[i] >= j) y[p++] = sa[i] - j;
            }
            fill(ws.begin(), ws.end(), 0);
            for (int i = 0; i <= n - 1; i++) ws[x[i]]++;
            for (int i = 1; i <= lim - 1; i++) ws[i] += ws[i
                - 1];
            for (int i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
            swap(x, y), p = 1, x[sa[0]] = 0;
            for (int i = 1; i <= n - 1; i++) {
                a = sa[i - 1], b = sa[i];
                x[b] = (y[a] == y[b] && y[a + j] == y[b + j])
                    ? p - 1 : p++;
            }
        }
        for (int i = 1; i <= n - 1; i++) rank[sa[i]] = i;
        for (int i = 0, j; i < n - 1; lcp[rank[i++]] = k) {
            for (k && k--, j = sa[rank[i] - 1]; s[i + k] ==
                s[j + k]; k++);
        }
        for(int i=1;i<n-1;i++) lcp[i]=lcp[i+1];
        lcp[n-1]=0;
    }
}sa;
int main() {
    fast;
    string str; cin>>str;
    sa.build(str);
    ll n=(int)str.size();
    long long ans=n*(n+1)/2;
    for(auto x: sa.lcp) ans-=x;
    cout<<ans<<endl;
}
```

7.6 Suffix Array (sort substring range)

```
///no need to clear any ara before every test case
const int mx=4e5+9;
int  n, m;
int ra[mx], tempra[mx];
int sa[mx], tempsa[mx];
int c[mx]; string str;
void countSort(int k) {
    int maxi = max(300, n);
    int sum=0;
    memset(c, 0, sizeof c);
    for (int i = 0; i < n; i++) c[i + k < n ? ra[i + k] :
        0]++;
    for (int i = sum = 0; i < maxi; i++) {
        int t = c[i];
        c[i] = sum;
```

```
        sum += t;
    }
    for (int i = 0; i < n; i++) tempsa[c[sa[i] + k < n ?
        ra[sa[i] + k] : 0]++] = sa[i];
    for (int i = 0; i < n; i++) sa[i] = tempsa[i];
}
void makesa() {
    for (int i = 0; i < n; i++) {
        ra[i] = str[i]; sa[i] = i;
    }
    for (int k = 1; k < n; k <<= 1) {
        countSort(k); countSort(0);
        tempra[sa[0]]=0;
        int r=0;
        for (int i = 1; i < n; i++)
            tempra[sa[i]] =(ra[sa[i]] == ra[sa[i - 1]] &&
                ra[sa[i] + k] == ra[sa[i - 1] + k]) ? r :
                ++r;
        for (int i = 0; i < n; i++) ra[i] = tempra[i];
    }
}
int temp[mx];
int lcp[mx]; ///lcp[i] is the length of longest common
    prefix of sa[i] and sa[i+1]
void buildlcp() {
    for(int i=0; i<n; i++) temp[sa[i]]=i;
    int k=0;
    for(int i=0; i<n; i++) {
        if(temp[i]==n-1) {
            lcp[temp[i]]=0;
            k=0;
            continue;
        }
        int j=sa[temp[i]+1];
        while(i+k<n and j+k<n and str[i+k]==str[j+k]) k++;
        lcp[temp[i]]=k;
        if(k) k--;
    }
}
struct edge {
   int rank, l, r;
   bool operator < (edge &p) const {
      if(rank==p.rank) {
         int len1=r-l+1;
         int len2=p.r-p.l+1;
         if(len1==len2) {
            if(l==p.l) return l;
            return r<p.r;
         }
         return len1<len2;
      }
      return rank<p.rank;
   }
};
int mini[mx][22];
void sparse() {
    //template dekhe lcp er minimum sparse table
}
int query(int l, int r) {
   //sparse table er query function
}
bool check(int mid,int ses,int l, int r) {
   int soto=query(mid,ses-1);
   int len=r-l+1;
   if(soto<len) return false;
   return true;
}
int main() {
    // sort the substring range
   cin>>str; str+=' ';
   n=str.size();
   makesa(); buildlcp(); sparse();
   vector<int> dp(n+4);
   vector<pair<int,int> > vv;
   for(int i=0;i<n;i++) {
      dp[sa[i]]=i;
   }
   vector<edge> v;
   int q; cin>>q;
   while(q--)
   {
      int l,r; cin>>l>>r;
      l--; r--;
      int low=1,high=dp[l]-1;
      int suru=-1;
      while(low<=high) {
         int mid=(low+high)/2;
         if(check(mid,dp[l],l,r)) {
            suru=mid;
            high=mid-1;
         }
         else low=mid+1;
      }
      if(suru==-1) v.pb({dp[l],l,r});
      else v.pb({suru,l,r});
   }
   sort(v.begin(),v.end());
   for(auto x: v) cout<<x.l+1<<' '<<x.r+1<<endl;
}
```

7.7 Suffix Automata

```
int nxt[mx<<1][26],link[mx<<1],len[mx<<1];
int last=1,sz=1;
void add(int c) {
    int cur=++sz;
    int p=last; last=cur;
    len[cur]=len[p]+1;
    while(p and nxt[p][c]==0) nxt[p][c]=cur,p=link[p]; //add
        character c to the suffix of p
    if(p==0) link[cur]=1;
    else {
        int q=nxt[p][c];
        if(len[q]==len[p]+1) link[cur]=q; //continuous
            transition , so don't change anything
        else {
            int cl=++sz; ///non continuous transition , so
                make clone of q
            len[cl]=len[p]+1;
            link[cl]=link[q];
            for(int i=0;i<26;i++) nxt[cl][i]=nxt[q][i];
            while(p and nxt[p][c]==q) nxt[p][c]=cl,p=link[p];
            link[cur]=link[q]=cl;
        }
    }
}
int main() {
    string str;
    cin>>str;
    for(auto x: str) add(x-'a');
    cout<<sz<<endl;
}
1. length of distinct substrings ends at node i =
    (len[link[i]]+1 to len[i])
2. when a character is added to the string, number of new
    distinct substring
   that is created is = index of the character(1 base) -
       len[link[last]
```

7.8 Trie

```
int nxt[100009*26][26];
bool mark[100009*26];
int node=1;
void add(string str) {
    int now=0;
    for(int i=0;i<str.size();i++) {
        int d=str[i]-'a';
        if(nxt[now][d]==0) nxt[now][d]=node++;
        now=nxt[now][d];
    }
    mark[now]=true;
}
bool check(string str) {
    int now=0;
    for(int i=0;i<str.size();i++) {
        int d=str[i]-'a';
        if(nxt[now][d]==0) return false;
        now=nxt[now][d];
    }
    return mark[now] ;
}
int main() {
    string str;
```

```
    cin>>str; add(str);
    string str1; cin>>str1;
    if(check(str1)) printf("found");
    else printf("not found");
}
```

7.9 Persistent Trie

```
struct edge {
    ll l,r;
    edge() {
        l=0,r=0;
    }
}nxt[mx*50];
int root[mx*50];
ll cnt=1;
void init(ll k) {
    ll now=1;
    for(ll i=31;i>=0;i--) {
        bool d=k&(1<<i);
        if(d==0) nxt[now].l=++cnt;
        else nxt[now].r=++cnt;
        now=cnt;
    }
}
void add(ll k, ll pre) {
    ll now=++cnt;
    for(ll i=31;i>=0;i--) {
        bool d=k&(1<<i);
        if(d==0) {
            nxt[now].l=++cnt;
            nxt[now].r=nxt[pre].r;
            pre=nxt[pre].l;
        }
        else {
            nxt[now].r=++cnt;
            nxt[now].l=nxt[pre].l;
            pre=nxt[pre].r;
        }
        now=cnt;
    }
}
int main() {
    int n; cin>>n;
    int r,val;
    cin>>r>>val;
    init(val); root[r]=1;
    for(int i=0;i<n;i++) {
        int a,b,val;
        cin>>a>>b>>val; //add edge a to b, value of b is val
        root[b]=cnt+1;
        add(val,root[a]);
    }
}
```

7.10 Palindromic Tree

```
ll node,t;
ll nxt[mx][26] ,link[mx],len[mx];
ll cnt[mx],cnt1[mx];
string str;
void pre() {
    memset(link,0,sizeof link);
    memset(nxt,0,sizeof nxt);
    link[1]=link[2]=1;
    len[1]=-1; len[2]=0;
    node=t=2;
}
void add(ll p) {
    while(str[p-len[t]-1]!=str[p]) t=link[t];
    ll x=link[t];
    while(str[p-len[x]-1]!=str[p]) x=link[x];
    ll c=str[p]-'a';
    if(!nxt[t][c]) {
        nxt[t][c]=++node;
        len[node]=len[t]+2;
        link[node]=len[node]==1? 2: nxt[x][c];
    }
    t=nxt[t][c];
}
int main() {
    //number of common palindromic substring of two strings
   cin>>str;
   pre();
   for(ll i=0;i<str.size();i++) add(i),cnt[t]++;
   cin>>str;
   t=2 ;
   for(ll i=0;i<str.size();i++) add(i),cnt1[t]++;
   ll ans=0;
   for(ll i=node;i>=3;i--) {
       cnt[link[i]]+=cnt[i];
       cnt1[link[i]]+=cnt1[i];
       ans+=cnt[i]*cnt1[i];
       cnt[i]=cnt1[i]=0;
   }
   cout<<ans<<endl;
}
```

7.11 Z Algorithm

```
string str,str1 ;
ll ara[200009];///2*sizeof string, need to memset for every
     case
void findz() {
    ll left=0,right=0;
    for(ll i=1;i<str.size();i++) {
        if(right>=i) ara[i]=min(ara[i-left],right-i+1);
        while(i+ara[i]<str.size() and
             str[ara[i]]==str[i+ara[i]]) ara[i]++;
        if(i+ara[i]-1>right) left=i,right=i+ara[i]-1;
    }
}
int  main() {
    ///minimum length palindrome by adding character to the
         end
    cin>>str1; str=str1;
    reverse(str.begin(),str.end());
    str=str+'#'+str1;
    findz(); ll d=0 ;
    for( ll i=0;i<str.size();i++) {
        if(ara[i]+i==str.size()) {
            d=ara[i];
            break;
        }
    }
    cout<<str1;
    for(ll i=d;str[i]!='#';i++ ) {
        cout<<str[i];
    }
}
```

8 Direct Solution

8.1 way to select k equal substrings

```
int nxt[mx<<1][26],link[mx<<1],len[mx<<1];
int last=1,sz=1;
int cnt[mx<<1];
void add(int c) {
    int cur=++sz;
    cnt[cur]=1;
    int p=last; last=cur;
    len[cur]=len[p]+1;
    while(p and nxt[p][c]==0) nxt[p][c]=cur,p=link[p]; //add
         character c to the suffix of p
    if(p==0) link[cur]=1;
    else {
        int q=nxt[p][c];
        if(len[q]==len[p]+1) link[cur]=q; //continuous
             transition , so don't change anything
        else {
            int cl=++sz; ///non continuous transition , so
                 make clone of q
            len[cl]=len[p]+1;
            link[cl]=link[q];
            for(int i=0;i<26;i++) nxt[cl][i]=nxt[q][i];
            while(p and nxt[p][c]==q) nxt[p][c]=cl,p=link[p];
```

```
            link[cur]=link[q]=cl;
        }
    }
}
ll facto[mx],invfacto[mx];
void pre() {
    facto[0]=1;
    for(ll i=1;i<mx;i++) facto[i]=(facto[i-1]*i)%mod;
    invfacto[0]=1;
    invfacto[1]=1;
    for(ll i=2;i<mx;i++) {
        invfacto[i]=mod-(mod/i)*invfacto[mod%i]%mod; ///pre
            calculate mod inverse i
    }
    for(int i=1;i<mx;i++)
        invfacto[i]=(invfacto[i-1]*invfacto[i])%mod;
        ///precalculate inverse factorial
}
ll ncr(ll n, ll r) {
    if(r>n) return 0;
    ll ans=facto[n]*invfacto[n-r];
    ans%=mod;
    ans=(ans*invfacto[r])%mod;
    return ans;
}
ll ans[mx];
vector<int>szz[mx];
int main() {
    pre();
    int n,q;
    cin>>n>>q;
    string str; cin>>str;
    for(auto x: str) add(x-'a');
    for(int i=1;i<=sz;i++) szz[len[i]].pb(i);
    for(int i=n;i>0;i--) {
       for(auto x: szz[i]) cnt[link[x]]+=cnt[x];
    }
    for(int i=2;i<=sz;i++) {
        int suru=len[link[i]]+1;
        int ses=len[i];
        int len=ses-suru+1;
        for(int j=1;j<=cnt[i];j++) {
            ll d=ncr(cnt[i],j);
            d=(d*len)%mod;
            ans[j]=(ans[j]+d)%mod;
        }
    }
    while(q--)
    {
        int k; cin>>k;
        if(k>(int)str.size()) cout<<0<<endl;
        else cout<<ans[k]<<endl;
    }
}
```

8.2 longest common substring

```
int nxt[mx<<1][28],link[mx<<1],len[mx<<1];
int last=1,sz=1;
int cnt[mx<<1][4];
int ind[mx<<1];
void add(int c, int val) {
    int cur=++sz;
    cnt[cur][val]=1;
    int p=last; last=cur;
    len[cur]=len[p]+1;
    ind[cur]=len[cur]-1;
    while(p and nxt[p][c]==0) nxt[p][c]=cur,p=link[p]; //add
        character c to the suffix of p
    if(p==0) link[cur]=1;
    else {
        int q=nxt[p][c];
        if(len[q]==len[p]+1) link[cur]=q; //continuous
            transition , so don't change anything
        else {
            int cl=++sz; ///non continuous transition , so
                make clone of q
            len[cl]=len[p]+1;
            link[cl]=link[q];
            for(int i=0;i<27;i++) nxt[cl][i]=nxt[q][i];
            while(p and nxt[p][c]==q) nxt[p][c]=cl,p=link[p];
            link[cur]=link[q]=cl;
            ind[cl]=ind[q];
        }
    }
}
int c=0;
int main() {
    string str,str1;
    cin>>str>>str1;
    for(auto x: str1) add(x-'a',0);
    add(26,2);
    for(auto x: str) add(x-'a',1);
    vector<pair<int,int> > v;
    for(int i=1;i<=sz;i++) v.pb({len[i],i});
    sort(v.rbegin(),v.rend());
    for(auto x: v) {
        cnt[link[x.second]][0]+=cnt[x.second][0];
        cnt[link[x.second]][1]+=cnt[x.second][1];
    }
    int maxi=0;
    int ii=1e7;
    for(int i=1;i<=sz;i++) {
        int ses=len[i];
        if(cnt[i][0] and cnt[i][1]) {
            if(ses>maxi) {
                maxi=ses;
            }
        }
    }
    for(int i=1;i<=sz;i++) {
        int ses=len[i];
        if(cnt[i][0] and cnt[i][1]) {
            if(ses==maxi) {
                ii=min(ii,ind[i]);
            }
        }
    }
    if(maxi) cout<<str1.substr(ii-maxi+1,maxi)<<endl;
    cout<<maxi<<endl;
}
```

8.3 count number of substring equals to lcp of two given strings

```
struct SA {
    vector<int> sa, lcp;
    SA(){};
    void build(string& s, int lim = 256) {
        int n = s.size() + 1, k = 0, a, b;
        vector<int> x(s.begin(), s.end() + 1), y(n),
            ws(max(n, lim)), rank(n);
        sa = lcp = y;
        iota(sa.begin(), sa.end(), 0);
        for (int j = 0, p = 0; p < n; j = max(1, j * 2), lim
            = p) {
            p = j, iota(y.begin(), y.end(), n - j);
            for (int i = 0; i <= n - 1; i++) {
                if (sa[i] >= j) y[p++] = sa[i] - j;
            }
            fill(ws.begin(), ws.end(), 0);
            for (int i = 0; i <= n - 1; i++) ws[x[i]]++;
            for (int i = 1; i <= lim - 1; i++) ws[i] += ws[i
                - 1];
            for (int i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
            swap(x, y), p = 1, x[sa[0]] = 0;
            for (int i = 1; i <= n - 1; i++) {
                a = sa[i - 1], b = sa[i];
                x[b] = (y[a] == y[b] && y[a + j] == y[b + j])
                    ? p - 1 : p++;
            }
        }
        for (int i = 1; i <= n - 1; i++) rank[sa[i]] = i;
        for (int i = 0, j; i < n - 1; lcp[rank[i++]] = k) {
            for (k && k--, j = sa[rank[i] - 1]; s[i + k] ==
                s[j + k]; k++);
        }
        for(int i=1;i<n-1;i++) lcp[i]=lcp[i+1];
        lcp[n-1]=0;
    }
}sa;
int mini[mx][22];
int n;
void sparse() {
```

```
    for(int i=0;i<n;i++) mini[i][0]=sa.lcp[i];
    for(int j=1;(1LL<<j)<=n;j++) {
        for(int i=0;i+(1LL<<(j))-1<n;i++) {
            int d=1LL<<(j-1);
            mini[i][j]=min(mini[i][j-1],mini[i+d][j-1]);
        }
    }
}
int query(int l, int r) {
    if(l>r) swap(l,r);
    int d=log2(r-l+1);
    return min(mini[l][d],mini[r-(1LL<<d)+1][d]);
}
int pos[mx];
int len;
int sz[mx];
bool check(int ses,int mid) {
    int d=query(ses,mid);
    return d>=len;
}
int dp[mx];
int main() {
    //given n strings, find the lcp of string a and b and
        print number of substring equals to this lcp over
        all given strings
    int n1;
    cin>>n1;
    string str;
    for(int i=0;i<n1;i++) {
        if(i) {
            str+='$';
        }
        string a; cin>>a;
        sz[i]=(int)a.size();
        pos[i]=(int)str.size();
        str+=a;
    }
    sa.build(str);
    n=(int)str.size()+1;
    sparse();
    for(int i=0;i<n;i++) dp[sa.sa[i]]=i;
    int q;
    cin>>q;
    while(q--) {
        int a,b;
        cin>>a>>b;
        a--; b--;
        int l=pos[a],r=pos[b];
        l=dp[l]; r=dp[r];
        if(l>r) swap(l,r);
        if(a==b) len=sz[a];
        else len=query(l,r-1);
        if(len==0) {
            cout<<0<<endl; continue;
        }
        len=min({len,sz[a],sz[b]});
        int suru=-1;
        int low=1,high=l-1;
        while(low<=high) {
            int mid=(low+high)>>1;
            if(check(mid,l-1)) {
                suru=mid;
                high=mid-1;
            }
            else low=mid+1;
        }
        if(suru==-1) suru=l;
        int ses=-1;
        low=l,high=n-1;
        while(low<=high) {
            int mid=(low+high)>>1;
            if(check(l,mid)) {
                ses=mid+1;
                low=mid+1;
            }
            else high=mid-1;
        }
        if(ses==-1) ses=l;
        cout<<ses-suru+1<<endl;
    }
}
```

8.4 longest path for each subtree

```
vector<ll> v[100009];
ll dis[100009],dis1[100009];
ll c;
void dfs(ll s, ll par, ll t) {
    for(auto x: v[s]) {
        if(x==par) continue;
        dfs(x,s,t+1);
        dis[s]=max(dis[s],dis[x]+1);
    }
}
void dfs1(ll s, ll par) {
    for(auto x: v[s]) {
        if(x==par) continue;
        dfs1(x,s);
    }
    dis1[s]=dis[s];
    ll max1=-1,max2=-1;
    for(auto x: v[s]) {
        if(x==par) continue;
        dis1[s]=max(dis1[s],dis1[x]);
        if(dis[x]>max1) {
            max2=max1;
            max1=dis[x];
        }
        else if(dis[x]>max2) max2=dis[x];
    }
    if(max1>=0 and max2>=0) dis1[s]=max(dis1[s],max1+max2+2);
}
int main() {
    //longest path for each subtree , tree is rooted c
    ll n;
    cin>>n;
    for(ll i=0;i<n-1;i++) {
        ll a,b;
        cin>>a>>b;
        v[a].pb(b);
        v[b].pb(a);
    }
    cin>>c;
    dfs(c,-1,0);
    dfs1(c,-1);
    for(int i=1;i<=n;i++) {
        cout<<i<<' '<<dis1[i]<<endl;
    }
}
```

8.5 number of triplets where gcd(a,b,c)=1

```
ll mobi[mx];
ll cnt[mx];
ll divi[mx];
void pre() {
    mobi[1]=1;
    for(ll i=1;i<mx;i++) {
        if(cnt[i]) divi[i]=cnt[i];
        for(ll j=2*i;j<mx;j+=i) {
            mobi[j]-=mobi[i];
            if(cnt[j]) divi[i]+=cnt[j];
        }
    }
}
ll ncr(ll n) {
    if(n<3) return 0;
    ll ans=n*(n-1)*(n-2);
    return ans/6;
}
int main() {
   ll n; cin>>n;
   vector<ll>v(n);
   for(ll i=0;i<n;i++) {
       ll x; cin>>x;
       v[i]=x;
       cnt[x]++;
   }
   pre();
   ll ans=0;
   for(ll i=1;i<=1000000;i++) {
       ans+=mobi[i]*ncr(divi[i]);
   }
   cout<<ans<<endl;
```

```
}
```

8.6 xor update, delete, find kth element

```
struct edge {
    ll l,r;
    edge() {
        l=0,r=0;
    }
}nxt[mx*50];
int mark[mx*32];
int root[mx*32];
ll cnt=1;
void init(ll k) {
    ll now=1;
    for(ll i=31;i>=0;i--) {
        bool d=k&(1<<i);
        if(d==0) nxt[now].l=++cnt;
        else nxt[now].r=++cnt;
        now=cnt;
    }
}
void add(ll k, ll pre) {
    ll now=++cnt;
    for(ll i=31;i>=0;i--) {
        bool d=k&(1<<i);
        if(d==0) {
            nxt[now].l=++cnt;
            nxt[now].r=nxt[pre].r;
            pre=nxt[pre].l;
            mark[cnt]=1+mark[pre];
        }
        else {
            nxt[now].r=++cnt;
            nxt[now].l=nxt[pre].l;
            pre=nxt[pre].r;
            mark[cnt]=1+mark[pre];
        }
        now=cnt;
    }
}
int get(int x,int now, int pre) {
    int ans=0;
    for(int i=31;i>=0;i--) {
        bool d=x&(1LL<<i);
        if(d==0) {
            int temp=mark[nxt[now].r]-mark[nxt[pre].r];
            if(temp) {
                now=nxt[now].r;
                pre=nxt[pre].r;
                ans|=1LL<<i;
            }
            else {
                pre=nxt[pre].l;
                now=nxt[now].l;
            }
        }
        else {
            int temp=mark[nxt[now].l]-mark[nxt[pre].l];
            if(temp) {
                pre=nxt[pre].l;
                now=nxt[now].l;
                ans|=1LL<<i;
            }
            else {
                pre=nxt[pre].r;
                now=nxt[now].r;
            }
        }
    }
    return ans;
}
int soto(int x,int now, int pre) {
    int ans=0;
    for(int i=31;i>=0;i--) {
        bool d=x&(1LL<<i);
        if(d==0) {
            if(nxt[now].l==0) return ans;
            now=nxt[now].l;
            pre=nxt[pre].l;
        }
        else {
            int temp=mark[nxt[now].l]-mark[nxt[pre].l];
            ans+=temp;
            if(nxt[now].r) {
                now=nxt[now].r;
                pre=nxt[pre].r;
            }
            else return ans;
        }
    }
    ans+=mark[now]-mark[pre];
    return ans;
}
int kth(int k,int now, int pre) {
    int ans=0;
    for(int i=31;i>=0;i--) {
        int temp=mark[nxt[now].l]-mark[nxt[pre].l];
        if(temp>=k) {
            now=nxt[now].l;
            pre=nxt[pre].l;
        }
        else {
            ans|=1LL<<i;
            k-=temp;
            now=nxt[now].r;
            pre=nxt[pre].r;
        }
    }
    return ans;
}
int main() {
    int m;
    cin>>m;
    int n=0;
    root[0]=1;
    init(0);
    while(m--) {
        int id; cin>>id;
        if(id==0)
        {
            //add x to the end of array
            n++;
            root[n]=cnt+1;
            int x; cin>>x;
            add(x,root[n-1]);
        }
        else if(id==1) {
            //find x in range (l,r) that maximize (x xor y)
            int l,r,y;
            cin>>l>>r>>y;
            int ans=get(y,root[r],root[l-1])^y;
            cout<<ans<<endl;
        }
        else if(id==2) {
            //delete last k elements in the array
            int k;cin>>k;
            n-=k;
        }
        else if(id==3) {
            //count integer <=x in range (l,r)
            int l,r,x;
            cin>>l>>r>>x;
            int ans=soto(x,root[r],root[l-1]);
            cout<<ans<<endl;
        }
        else {
            //find kth smallest in range (l,r)
            int l,r,k;
            cin>>l>>r>>k;
            int ans=kth(k,root[r],root[l-1]);
            cout<<ans<<endl;
        }
    }
}
```

Theoretical Computer Science Cheat Sheet

Definitions

$f(n) = O(g(n))$	iff $\exists$ positive c, n_0 such that $0 \le f(n) \le cg(n)\ \forall n \ge n_0$.
$f(n) = \Omega(g(n))$	iff $\exists$ positive c, n_0 such that $f(n) \ge cg(n) \ge 0\ \forall n \ge n_0$.
$f(n) = \Theta(g(n))$	iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
$f(n) = o(g(n))$	iff $\lim_{n\to\infty} f(n)/g(n) = 0$.
$\lim_{n\to\infty} a_n = a$	iff $\forall \epsilon > 0$, $\exists n_0$ such that $\lvert a_n - a\rvert < \epsilon, \forall n \ge n_0$.
$\sup S$	least $b \in \mathbb{R}$ such that $b \ge s$, $\forall s \in S$.
$\inf S$	greatest $b \in \mathbb{R}$ such that $b \le s$, $\forall s \in S$.
$\liminf_{n\to\infty} a_n$	$\lim_{n\to\infty} \inf\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\limsup_{n\to\infty} a_n$	$\lim_{n\to\infty} \sup\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\binom{n}{k}$	Combinations: Size k subsets of a size n set.
$\left[{n \atop k}\right]$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.
$\left\{{n \atop k}\right\}$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.
$\left\langle{n \atop k}\right\rangle$	1st order Eulerian numbers: Permutations $\pi_1\pi_2\ldots\pi_n$ on $\{1, 2, \ldots, n\}$ with k ascents.
$\left\langle\left\langle{n \atop k}\right\rangle\right\rangle$	2nd order Eulerian numbers.
C_n	Catalan Numbers: Binary trees with $n+1$ vertices.

Series

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}.$$

In general:

$$\sum_{i=1}^{n} i^m = \frac{1}{m+1}\left[(n+1)^{m+1} - 1 - \sum_{i=1}^{n}\left((i+1)^{m+1} - i^{m+1} - (m+1)i^m\right)\right]$$

$$\sum_{i=1}^{n-1} i^m = \frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k}B_k n^{m+1-k}.$$

Geometric series:

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}, \quad c \ne 1, \quad \sum_{i=0}^{\infty} c^i = \frac{1}{1-c}, \quad \sum_{i=1}^{\infty} c^i = \frac{c}{1-c}, \quad |c| < 1,$$

$$\sum_{i=0}^{n} ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \ne 1, \quad \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad |c| < 1.$$

Harmonic series:

$$H_n = \sum_{i=1}^{n}\frac{1}{i}, \qquad \sum_{i=1}^{n} iH_i = \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4}.$$

$$\sum_{i=1}^{n} H_i = (n+1)H_n - n, \quad \sum_{i=1}^{n}\binom{i}{m}H_i = \binom{n+1}{m+1}\left(H_{n+1} - \frac{1}{m+1}\right).$$

1. $\binom{n}{k} = \frac{n!}{(n-k)!k!}$,
2. $\sum_{k=0}^{n}\binom{n}{k} = 2^n$,
3. $\binom{n}{k} = \binom{n}{n-k}$,
4. $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$,
5. $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$,
6. $\binom{n}{m}\binom{m}{k} = \binom{n}{k}\binom{n-k}{m-k}$,
7. $\sum_{k=0}^{n}\binom{r+k}{k} = \binom{r+n+1}{n}$,
8. $\sum_{k=0}^{n}\binom{k}{m} = \binom{n+1}{m+1}$,
9. $\sum_{k=0}^{n}\binom{r}{k}\binom{s}{n-k} = \binom{r+s}{n}$,
10. $\binom{n}{k} = (-1)^k\binom{k-n-1}{k}$,
11. $\left\{{n \atop 1}\right\} = \left\{{n \atop n}\right\} = 1$,
12. $\left\{{n \atop 2}\right\} = 2^{n-1} - 1$,
13. $\left\{{n \atop k}\right\} = k\left\{{n-1 \atop k}\right\} + \left\{{n-1 \atop k-1}\right\}$,
14. $\left[{n \atop 1}\right] = (n-1)!$,
15. $\left[{n \atop 2}\right] = (n-1)!H_{n-1}$,
16. $\left[{n \atop n}\right] = 1$,
17. $\left[{n \atop k}\right] \ge \left\{{n \atop k}\right\}$,
18. $\left[{n \atop k}\right] = (n-1)\left[{n-1 \atop k}\right] + \left[{n-1 \atop k-1}\right]$,
19. $\left\{{n \atop n-1}\right\} = \left[{n \atop n-1}\right] = \binom{n}{2}$,
20. $\sum_{k=0}^{n}\left[{n \atop k}\right] = n!$,
21. $C_n = \frac{1}{n+1}\binom{2n}{n}$,
22. $\left\langle{n \atop 0}\right\rangle = \left\langle{n \atop n-1}\right\rangle = 1$,
23. $\left\langle{n \atop k}\right\rangle = \left\langle{n \atop n-1-k}\right\rangle$,
24. $\left\langle{n \atop k}\right\rangle = (k+1)\left\langle{n-1 \atop k}\right\rangle + (n-k)\left\langle{n-1 \atop k-1}\right\rangle$,
25. $\left\langle{0 \atop k}\right\rangle = \begin{cases}1 & \text{if } k = 0,\\ 0 & \text{otherwise}\end{cases}$
26. $\left\langle{n \atop 1}\right\rangle = 2^n - n - 1$,
27. $\left\langle{n \atop 2}\right\rangle = 3^n - (n+1)2^n + \binom{n+1}{2}$,
28. $x^n = \sum_{k=0}^{n}\left\langle{n \atop k}\right\rangle\binom{x+k}{n}$,
29. $\left\langle{n \atop m}\right\rangle = \sum_{k=0}^{m}\binom{n+1}{k}(m+1-k)^n(-1)^k$,
30. $m!\left\{{n \atop m}\right\} = \sum_{k=0}^{n}\left\langle{n \atop k}\right\rangle\binom{k}{n-m}$,
31. $\left\langle{n \atop m}\right\rangle = \sum_{k=0}^{n}\left\{{n \atop k}\right\}\binom{n-k}{m}(-1)^{n-k-m}k!$,
32. $\left\langle\left\langle{n \atop 0}\right\rangle\right\rangle = 1$,
33. $\left\langle\left\langle{n \atop n}\right\rangle\right\rangle = 0$ for $n \ne 0$,
34. $\left\langle\left\langle{n \atop k}\right\rangle\right\rangle = (k+1)\left\langle\left\langle{n-1 \atop k}\right\rangle\right\rangle + (2n-1-k)\left\langle\left\langle{n-1 \atop k-1}\right\rangle\right\rangle$,
35. $\sum_{k=0}^{n}\left\langle\left\langle{n \atop k}\right\rangle\right\rangle = \frac{(2n)^n}{2^n}$,
36. $\left\{{x \atop x-n}\right\} = \sum_{k=0}^{n}\left\langle\left\langle{n \atop k}\right\rangle\right\rangle\binom{x+n-1-k}{2n}$,
37. $\left\{{n+1 \atop m+1}\right\} = \sum_{k}\binom{n}{k}\left\{{k \atop m}\right\} = \sum_{k=0}^{n}\left\{{k \atop m}\right\}(m+1)^{n-k}$,

Theoretical Computer Science Cheat Sheet

Identities Cont.

38. $\left[{n+1 \atop m+1}\right] = \sum_{k}\left[{n \atop k}\right]\binom{k}{m} = \sum_{k=0}^{n}\left[{k \atop m}\right]n^{\underline{n-k}} = n!\sum_{k=0}^{n}\frac{1}{k!}\left[{k \atop m}\right]$,
39. $\left[{x \atop x-n}\right] = \sum_{k=0}^{n}\left\langle\left\langle{n \atop k}\right\rangle\right\rangle\binom{x+k}{2n}$,
40. $\left\{{n \atop m}\right\} = \sum_{k}\binom{n}{k}\left\{{k+1 \atop m+1}\right\}(-1)^{n-k}$,
41. $\left[{n \atop m}\right] = \sum_{k}\left[{n+1 \atop k+1}\right]\binom{k}{m}(-1)^{m-k}$,
42. $\left\{{m+n+1 \atop m}\right\} = \sum_{k=0}^{m}k\left\{{n+k \atop k}\right\}$,
43. $\left[{m+n+1 \atop m}\right] = \sum_{k=0}^{m}k(n+k)\left[{n+k \atop k}\right]$,
44. $\binom{n}{m} = \sum_{k}\left\{{n+1 \atop k+1}\right\}\left[{k \atop m}\right](-1)^{m-k}$,
45. $(n-m)!\binom{n}{m} = \sum_{k}\left[{n+1 \atop k+1}\right]\left\{{k \atop m}\right\}(-1)^{m-k}$, for $n \ge m$,
46. $\left\{{n \atop n-m}\right\} = \sum_{k}\binom{m-n}{m+k}\binom{m+n}{n+k}\left[{m+k \atop k}\right]$,
47. $\left[{n \atop n-m}\right] = \sum_{k}\binom{m-n}{m+k}\binom{m+n}{n+k}\left\{{m+k \atop k}\right\}$,
48. $\left\{{n \atop \ell+m}\right\}\binom{\ell+m}{\ell} = \sum_{k}\left\{{k \atop \ell}\right\}\left\{{n-k \atop m}\right\}\binom{n}{k}$,
49. $\left[{n \atop \ell+m}\right]\binom{\ell+m}{\ell} = \sum_{k}\left[{k \atop \ell}\right]\left[{n-k \atop m}\right]\binom{n}{k}$.

Trees

Every tree with n vertices has $n-1$ edges.

Kraft inequality: If the depths of the leaves of a binary tree are $d_1, \ldots, d_n$:

$$\sum_{i=1}^{n} 2^{-d_i} \le 1,$$

and equality holds only if every internal node has 2 sons.

Recurrences

Master method:
$T(n) = aT(n/b) + f(n), \quad a \ge 1, b > 1$

If $\exists \epsilon > 0$ such that $f(n) = O(n^{\log_b a - \epsilon})$ then

$$T(n) = \Theta(n^{\log_b a}).$$

If $f(n) = \Theta(n^{\log_b a})$ then

$$T(n) = \Theta(n^{\log_b a}\log_2 n).$$

If $\exists \epsilon > 0$ such that $f(n) = \Omega(n^{\log_b a + \epsilon})$, and $\exists c < 1$ such that $af(n/b) \le cf(n)$ for large n, then

$$T(n) = \Theta(f(n)).$$

Substitution (example): Consider the following recurrence

$$T_{i+1} = 2^{2^i} \cdot T_i^2, \quad T_1 = 2.$$

Note that T_i is always a power of two. Let $t_i = \log_2 T_i$. Then we have

$$t_{i+1} = 2^i + 2t_i, \quad t_1 = 1.$$

Let $u_i = t_i/2^i$. Dividing both sides of the previous equation by 2^{i+1} we get

$$\frac{t_{i+1}}{2^{i+1}} = \frac{2^i}{2^{i+1}} + \frac{t_i}{2^i}.$$

Substituting we find

$$u_{i+1} = \tfrac{1}{2} + u_i, \qquad u_1 = \tfrac{1}{2},$$

which is simply $u_i = i/2$. So we find that T_i has the closed form $T_i = 2^{i2^{i-1}}$.
Summing factors (example): Consider the following recurrence

$$T(n) = 3T(n/2) + n, \quad T(1) = 1.$$

Rewrite so that all terms involving T are on the left side

$$T(n) - 3T(n/2) = n.$$

Now expand the recurrence, and choose a factor which makes the left side "telescope"

$$\begin{aligned}1\big(T(n) - 3T(n/2) &= n\big)\\ 3\big(T(n/2) - 3T(n/4) &= n/2\big)\\ &\vdots \quad \vdots \quad \vdots\\ 3^{\log_2 n - 1}\big(T(2) - 3T(1) &= 2\big)\end{aligned}$$

Let $m = \log_2 n$. Summing the left side we get $T(n) - 3^m T(1) = T(n) - 3^m = T(n) - n^k$ where $k = \log_2 3 \approx 1.58496$. Summing the right side we get

$$\sum_{i=0}^{m-1}\frac{n}{2^i}3^i = n\sum_{i=0}^{m-1}\left(\tfrac{3}{2}\right)^i.$$

Let $c = \frac{3}{2}$. Then we have

$$\begin{aligned}n\sum_{i=0}^{m-1}c^i &= n\left(\frac{c^m - 1}{c-1}\right)\\ &= 2n(c^{\log_2 n} - 1)\\ &= 2n(c^{(k-1)\log_c n} - 1)\\ &= 2n^k - 2n,\end{aligned}$$

and so $T(n) = 3n^k - 2n$. Full history recurrences can often be changed to limited history ones (example): Consider

$$T_i = 1 + \sum_{j=0}^{i-1} T_j, \quad T_0 = 1.$$

Note that

$$T_{i+1} = 1 + \sum_{j=0}^{i} T_j.$$

Subtracting we find

$$\begin{aligned}T_{i+1} - T_i &= 1 + \sum_{j=0}^{i} T_j - 1 - \sum_{j=0}^{i-1} T_j\\ &= T_i.\end{aligned}$$

And so $T_{i+1} = 2T_i = 2^{i+1}$.

Generating functions:

1. Multiply both sides of the equation by x^i.
2. Sum both sides over all i for which the equation is valid.
3. Choose a generating function $G(x)$. Usually $G(x) = \sum_{i=0}^{\infty} x^i g_i$.
3. Rewrite the equation in terms of the generating function $G(x)$.
4. Solve for $G(x)$.
5. The coefficient of x^i in $G(x)$ is g_i.

Example:

$$g_{i+1} = 2g_i + 1, \quad g_0 = 0.$$

Multiply and sum:

$$\sum_{i\ge0} g_{i+1}x^i = \sum_{i\ge0} 2g_i x^i + \sum_{i\ge0} x^i.$$

We choose $G(x) = \sum_{i\ge0} x^i g_i$. Rewrite in terms of $G(x)$:

$$\frac{G(x) - g_0}{x} = 2G(x) + \sum_{i\ge0} x^i.$$

Simplify:

$$\frac{G(x)}{x} = 2G(x) + \frac{1}{1-x}.$$

Solve for $G(x)$:

$$G(x) = \frac{x}{(1-x)(1-2x)}.$$

Expand this using partial fractions:

$$\begin{aligned}G(x) &= x\left(\frac{2}{1-2x} - \frac{1}{1-x}\right)\\ &= x\left(2\sum_{i\ge0}2^i x^i - \sum_{i\ge0}x^i\right)\\ &= \sum_{i\ge0}(2^{i+1} - 1)x^{i+1}.\end{aligned}$$

So $g_i = 2^i - 1$.