UIT.Meow Notebook - ICPC Asia Hanoi 2021

Contents

1 Vjp-pro

1.1 PTN vjp

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/trie_policy.hpp>
#include <ext/rope>

using namespace std;
using namespace __gnu_pbds;
using namespace __gnu_cxx;


//----------define--------------//
#define FILE_IN      "B.inp"
#define FILE_OUT     "B.out"
#define ofile        freopen(FILE_IN,"r",stdin);freopen(FILE_OUT,"w",stdout)
#define fio          ios::sync_with_stdio(0);cin.tie(0)
#define MOD          (ll(998244353))
#define MAXN         100010
#define INF          (ll(1000000007))
#define x            first
#define y            second
#define pii          pair<int,int>
#define pll          pair<long long,long long>
#define pli          pair<long long, int>
#define piii         pair<int,pii>
#define pb           push_back
#define endl         "\n"
#define vt           vector
typedef tree<long long,null_type,less<long long>,rb_tree_tag,
    tree_order_statistics_node_update> order_set;
typedef trie<string,null_type,trie_string_access_traits<>,pat_trie_tag,
    trie_prefix_search_node_update> order_trie;
typedef long long ll;

//---------functions-------------//
class DisjointSet{ public:
    vector<int> parent, size;
    DisjointSet(int n): parent(n), size(n) { for(int i=0; i<n; i++) {parent[i
        ] = i; size[i] = 0;}; }
    int sz(int a){return size[find(a)];}
    void join(int a, int b) { if (!check(a, b)) size[find(a)] += size[find(b)
        ]; parent[find(b)] = find(a); }
    int find(int a){ return a == parent[a] ? a : parent[a] = find(parent[a]);
         }
    bool check(int a, int b){ return find(a) == find(b); }
};
ll powmod(ll a,ll b) {ll res=1;a%=MOD; assert(b>=0); for(;b;b>>=1){if(b&1)res
    =res*a%MOD;a=a*a%MOD;}return res;}

//---------END-------------------//

int main(){
}
```

2 Flow

2.1 Dinic's Algo for Sparse max-flow

```cpp
#include <bits/stdc++.h>

using namespace std;

const int MAXN = 1007;

struct cung
{
    int to, cap, flow;
};

vector <cung> E;
int n, m, s, t, D[MAXN];
vector <int> adj[MAXN];
```

```
int incFlow(int u, int mi)
{
    if (u == t) return mi;
    for (int i: adj[u])
    {
        int v = E[i].to, c = E[i].cap, f = E[i].flow;
        if (D[v] == D[u] + 1 && c > f)
        {
            int df = incFlow(v, min(mi, c - f));
            if (df > 0)
            {
                E[i].flow += df;
                E[i ^ 1].flow -= df;
                return df;
            }
        }
    }
    return 0;
}

bool check()
{
    for (int i = 1; i <= n; ++i) D[i] = INT_MAX;
    D[s] = 0;
    queue <int> Q;
    Q.push(s);
    while (Q.size())
    {
        int u = Q.front(); Q.pop();
        if (u == t) return 1;
        for (int i: adj[u])
        {
            int v = E[i].to, f = E[i].flow, c = E[i].cap;
            if (D[v] > D[u] + 1 && c > f)
            {
                D[v] = D[u] + 1;
                Q.push(v);
            }
        }
    }
    return 0;
}

int main()
{
//    freopen("TEST.INP", "r", stdin);
    ios_base::sync_with_stdio(0);
    cin.tie(0), cout.tie(0);
    cin >> n >> m >> s >> t;
    while (m--)
    {
        int u, v, c;
        /// u-->v: kha nang thong qua la c
        cin >> u >> v >> c;
        adj[u].push_back(E.size());
        E.push_back({v, c, 0});
        adj[v].push_back(E.size());
        E.push_back({u, 0, 0});
    }

    int ans = 0;
    while (check())
    {
        int df;
        do
        {
            ans += df = incFlow(s, INT_MAX);
        } while (df > 0);
    }

    cout << ans;
    return 0;
}
```

2.2 Min cost Max Flow

```
#include <bits/stdc++.h>

#define For(i, a, b) for(ll i=a; i<=b; i++)
#define Fod(i, a, b) for(ll i=a; i>=b; i--)
#define Sc(x) scanf("%lld", &x)
#define p_b(x) push_back(x)
#define ha "mincost"

using namespace std;

typedef long long ll;

const ll N = 109,
INF = 1e10;

ll n, m, k, s, t, f[N][N], d[N][N], c[N][N], u, v, luong, w, doi[N*N*N];
vector <vector <ll> > vertex(N);

ll cost[N], Truy[N];
bool kt[N];

bool Timluong() {
        For(i, 1, n) {
                kt[i] = 0;
                cost[i] = INF;
        }
        ll i = 1, j = 1;
        doi[1] = s;
        cost[s] = 0;
        Truy[s] = 0;

        while (i <= j) {
                u = doi[i];
                i++;
                kt[u] = 0;

                For(z, 0, vertex[u].size() - 1) {
                        v = vertex[u][z];
                        if (c[u][v] > f[u][v] && cost[v] > cost[u] + (f[u][v]
                                >= 0? d[u][v] : -d[u][v])) {
                                cost[v] = cost[u] + (f[u][v] >= 0? d[u][v] :
                                        -d[u][v]);
                                Truy[v] = u;
                                if (!kt[v]) {
                                        kt[v] = 1;
                                        doi[++j] = v;
                                }
                        }
                }
        }

        return (cost[t] != INF);
}

ll dayhang(ll x) {
        v = t; u = Truy[v];
        ll tam = x;
        while (v != s) {
                if (f[u][v] < 0) tam = min(tam, -f[u][v]);
                else tam = min(tam, c[u][v] - f[u][v]);
                v = u; u = Truy[v];
        }

        v = t; u = Truy[v];
        while (v != s) {
                f[u][v] += tam;
                f[v][u] -= tam;
```

```
                v = u; u = Truy[v];
        }

        return tam;
}

int main() {
        Sc(n); Sc(m); Sc(k); Sc(s); Sc(t);
        ///n: so dinh, m: so canh, k: so hang hoa, s->t
        For(i, 1, m) {
            ///u -> v & v -> u: w la chi phi, luong la kha nang thong qua
                Sc(u); Sc(v); Sc(w); Sc(luong);
                vertex[u].push_back(v);
                vertex[v].push_back(u);
                c[v][u] = c[u][v] = luong;
                d[u][v] = d[v][u] = w;
        }

        while (Timluong()) {
                k -= dayhang(k);
                if (!k) break;
        }

        if (k > 0) {
                cout <<-1;
                return 0;
        }

        ll kq = 0;
        For(i, 1, n)
                For(j, 1, n) if (f[i][j] > 0) kq += (f[i][j] * d[i][j]);
        cout <<kq <<endl;
        For(i, 1, n)
                For(j, 1, n) if (f[i][j] > 0) cout <<i <<" " <<j <<" " <<f[i
                    ][j] <<endl;
                cout <<"0 0 0";
}
```

3 Geometry

3.1 Convex hull

```
struct pt { // Kieu diem
  double x, y;
};

bool cmp (pt a, pt b) { // So sanh theo toa do x, trong truong hop bang nhau
    so sanh theo y
  return a.x < b.x || a.x == b.x && a.y < b.y;
}

bool cw (pt a, pt b, pt c) { // a -> b -> c di theo thu tu xuoi chieu kim
    dong ho
  return a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y) < 0;
}

bool ccw (pt a, pt b, pt c) { // a -> b -> c di theo thu tu nguoc chieu kim
    dong ho
  return a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y) > 0;
}

void convex_hull (vector<pt> & a) {
  if (a.size() == 1) { // chi co 1 diem
    return;
  }

  // Sap xep cac diem theo toa do x, neu bang nhau sap xep theo y
  sort (a.begin(), a.end(), &cmp);

  pt p1 = a[0],  p2 = a.back();
  vector<pt> up, down; // chuoi tren va chuoi duoi
  up.push_back (p1);
  down.push_back (p1);

  for (size_t i=1; i<a.size(); ++i) { // xet lan luot cac diem
    // Them vao chuoi tren
    if (i==a.size()-1 || cw (p1, a[i], p2)) {
      while (up.size()>=2 && !cw (up[up.size()-2], up[up.size()-1], a[i]))
        up.pop_back();
      up.push_back (a[i]);
    }

    // Them vao chuoi duoi
    if (i==a.size()-1 || ccw (p1, a[i], p2)) {
      while (down.size()>=2 && !ccw (down[down.size()-2], down[down.size()
          -1], a[i]))
        down.pop_back();
      down.push_back (a[i]);
    }
  }

  // Gap 2 chuoi tren va duoi de lay bao loi
  a.clear();
  for (size_t i=0; i<up.size(); ++i)
    a.push_back (up[i]);
  for (size_t i=down.size()-2; i>0; --i)
    a.push_back (down[i]);
}
```

3.2 ccw

```
//<0: clockwise, >0: counter-clockwise, 0: collinear
//x: first, y: second
int ccw (Point P0, Point P1, Point P2) {
    return (P1.x - P0.x)*(P2.y - P0.y) - (P1.y - P0.y)*(P2.x - P0.x);
}

//start: first, end: second
//kiem tra 2 doan thang cat nhau
bool intersect (Vector2D v1, Vector2D v2) {
    if ((((ccw(v1.start, v1.end, v2.start) *
    ccw(v1.start, v1.end, v2.end)) == 0) &&
    ((ccw(v2.start, v2.end, v1.start) *
    ccw(v2.start, v2.end, v1.end)) == 0)))
        return 0;

    return (((ccw(v1.start, v1.end, v2.start) *
    ccw(v1.start, v1.end, v2.end)) <= 0) &&
    ((ccw(v2.start, v2.end, v1.start) *
    ccw(v2.start, v2.end, v1.end)) <= 0));
}
```

4 Numerical algorithms

4.1 Basis vector

```
int basis[d]; // basis[i] keeps the mask of the vector whose f value is i

int sz; // Current size of the basis

void insertVector(int mask) {
        for (int i = 0; i < d; i++) {
                if ((mask & 1 << i) == 0) continue; // continue if i != f(
                    mask)
```

```
                if (!basis[i]) { // If there is no basis vector with the i'th
                    bit set, then insert this vector into the basis
                        basis[i] = mask;
                        ++sz;

                        return;
                }

                mask ^= basis[i]; // Otherwise subtract the basis vector from
                    this vector
        }
}

// query for k-th largest xor sum among subsets
// insert vector need to be from i := d - 1 -> 0
int query(int k) {
        int mask = 0;

        int tot = 1 << sz;
        for (int i = LOG_K - 1; i >= 0; i--)
                if (basis[i]) {
                        int low = tot / 2;

                        if ((low < k && (mask & 1 << i) == 0) ||
                                (low >= k && (mask & 1 << i) > 0)) mask ^=
                                    basis[i];

                        if (low < k) k -= low;

                        tot /= 2;
                }

        return mask;
}
```

4.2 Generate prime numbers 1-¿1e9

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

// credit: min_25
// takes 0.5s for n = 1e9
vector<int> sieve(const int N, const int Q = 17, const int L = 1 << 15) {
  static const int rs[] = {1, 7, 11, 13, 17, 19, 23, 29};
  struct P {
    P(int p) : p(p) {}
    int p; int pos[8];
  };
  auto approx_prime_count = [] (const int N) -> int {
    return N > 60184 ? N / (log(N) - 1.1)
                     : max(1., N / (log(N) - 1.11)) + 1;
  };

  const int v = sqrt(N), vv = sqrt(v);
  vector<bool> isp(v + 1, true);
  for (int i = 2; i <= vv; ++i) if (isp[i]) {
    for (int j = i * i; j <= v; j += i) isp[j] = false;
  }

  const int rsize = approx_prime_count(N + 30);
  vector<int> primes = {2, 3, 5}; int psize = 3;
  primes.resize(rsize);

  vector<P> sprimes; size_t pbeg = 0;
  int prod = 1;
  for (int p = 7; p <= v; ++p) {
    if (!isp[p]) continue;
    if (p <= Q) prod *= p, ++pbeg, primes[psize++] = p;
    auto pp = P(p);
    for (int t = 0; t < 8; ++t) {
      int j = (p <= Q) ? p : p * p;
      while (j % 30 != rs[t]) j += p << 1;
      pp.pos[t] = j / 30;
    }
    sprimes.push_back(pp);
  }

  vector<unsigned char> pre(prod, 0xFF);
  for (size_t pi = 0; pi < pbeg; ++pi) {
    auto pp = sprimes[pi]; const int p = pp.p;
    for (int t = 0; t < 8; ++t) {
      const unsigned char m = ~(1 << t);
      for (int i = pp.pos[t]; i < prod; i += p) pre[i] &= m;
    }
  }

  const int block_size = (L + prod - 1) / prod * prod;
  vector<unsigned char> block(block_size); unsigned char* pblock = block.data
      ();
  const int M = (N + 29) / 30;

  for (int beg = 0; beg < M; beg += block_size, pblock -= block_size) {
    int end = min(M, beg + block_size);
    for (int i = beg; i < end; i += prod) {
      copy(pre.begin(), pre.end(), pblock + i);
    }
    if (beg == 0) pblock[0] &= 0xFE;
    for (size_t pi = pbeg; pi < sprimes.size(); ++pi) {
      auto& pp = sprimes[pi];
      const int p = pp.p;
      for (int t = 0; t < 8; ++t) {
        int i = pp.pos[t]; const unsigned char m = ~(1 << t);
        for (; i < end; i += p) pblock[i] &= m;
        pp.pos[t] = i;
      }
    }
    for (int i = beg; i < end; ++i) {
      for (int m = pblock[i]; m > 0; m &= m - 1) {
        primes[psize++] = i * 30 + rs[__builtin_ctz(m)];
      }
    }
  }
  assert(psize <= rsize);
  while (psize > 0 && primes[psize - 1] > N) --psize;
  primes.resize(psize);
  return primes;
}

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, a, b; cin >> n >> a >> b;
  auto primes = sieve(n);
  vector<int> ans;
  for (int i = b; i < primes.size() && primes[i] <= n; i += a) ans.push_back(
      primes[i]);
  cout << primes.size() << ' ' << ans.size() << '\n';
  for (auto x: ans) cout << x << ' '; cout << '\n';
  return 0;
}
// https://judge.yosupo.jp/problem/enumerate_primes
```

4.3 Print int128

```
ostream& operator<<(ostream& dest, __int128_t value)
{
    ostream::sentry s(dest);
    if (s) {
        __uint128_t tmp = value < 0 ? -value : value;
        char buffer[128];
```

```
        char* d = end( buffer );
        do{
            -- d;
            *d = "0123456789"[tmp % 10];
            tmp /= 10;
        } while (tmp != 0);
        if (value < 0){
            --d;
            *d = '-';
        }
        int len = end(buffer) - d;
        if (dest.rdbuf()->sputn(d, len) != len)
            dest.setstate(ios_base::badbit);
    }
    return dest;
}
int main(){
    __int128 a = __int128(100000000000000) * __int128(10007827712000000);
    cout << a;
}
```

4.4 Sum of n/1 + n/2 +...+ n/n

```
/**
 * Description: Sum of floor(n/1) + floor(n/2) + floor(n/3) + ... + floor(n/n
    )
 * Usage: See below. O(sqrt(n))
 * Note: This method can be extended to any function that takes only O(sqrt(n
    )) distinct values.
 * Source: https://github.com/dragonslayerx
 */

#include <iostream>
#include <cstdio>
#include <vector>
using namespace std;

const int MOD = 1e9 + 7 ;

int main() {
    long long n;
    scanf("%lld", &n);
    long long answer = 0;
    int i;
    for (i = 1; (long long)i*i <= n; i++) {
        answer += n/i;
        answer %= MOD;
    }
    i--;
    for (int j = 1; n/(j+1) >= i; j++) {
        answer += (((n/j - n/(j+1)) % MOD) * j) % MOD;
        answer %= MOD;
    }
    printf("%lld\n", answer);
}
```

4.5 Gcd in O(log)

```
#include <iostream>

using namespace std;


// Ax + By = GCD(A, B)

int d, x, y;
void extendedEuclid(int A, int B, int &x, int &y) {
    if (B == 0) {
        d = A;
        x = 1;
        y = 0;
    }
    else {
        extendedEuclid(B, A%B, x, y);
        int temp = x;
        x = y;
        y = temp - (A/B)*y;
    }
}

// Ax = 1 (mod m) (when GCD(m, A) == 1)
// prove:
// Ax + By = GCD(A, B)
// Let B = m => Ax = 1 (mod m)
// => Solve extendedEuclid(A, m), get x

// Function to find modulo inverse of a
void modInverse(int a, int m)
{
    int x, y;
    extendedEuclid(a, m, x, y);
    // m is added to handle negative x
    int res = (x % m + m) % m;
    cout << "Modular multiplicative inverse is " << res;
}

int main() {
    extendedEuclid(16, 10, x, y);
    cout << "gcd(16, 10) = " << d << endl;
    cout << "x, y: " << x <<  ", " << y << endl;
    modInverse(2, 11);
    return 0;
}
```

4.6 Count divisors

```
// complexity: O(N^1/3)
N = input()
primes = array containing primes till 10^6
ans = 1
for all p in primes :
            if p*p*p > N:
                  break
            count = 1
            while N divisible by p:
                  N = N/p
                  count = count + 1
            ans = ans * count
if N is prime:
            ans = ans * 2
else if N is square of a prime:
            ans = ans * 3
else if N != 1:
            ans = ans * 4
```

4.7 Count prime numbers ¡= n

```
# python is slow as fk, convert this code to C++ instead

from bisect import bisect # bisect == binary search

# just like normal sieve, but faster
def prime_sieve(n):
    """
```

```
    Efficient prime sieve, due to Robert William Hanks.
    Source: http://stackoverflow.com/a/2068548
    """
    sieve = [True] * (n//2)
    for i in range(3,int(n**0.5)+1,2):
        if sieve[i//2]:
            sieve[i*i//2::i] = [False] * ((n-i*i-1)//(2*i)+1)
    return [2] + [2*i+1 for i in range(1,n//2) if sieve[i]]
"""
Limit controls the number of primes that are sieved
to cache small values of pi(x).  Without caching,
runtime will be exponential.
When computing pi(x), limit should be
at least sqrt(x).  A higher value of limit
that caches more values can sometimes improve performance.
"""

limit = 10**6
primes = prime_sieve(limit)
print('done with primes')

phi_cache = {}
def phi(x, a):
    """
    Implementation of the partial sieve function, which
    counts the number of integers <= x with no prime factor less
    than or equal to the ath prime.
    """
    # If value is cached, just return it
    if (x, a) in phi_cache: return phi_cache[(x, a)]

    # Base case: phi(x, a) is the number of odd integers <= x
    if a == 1: return (x + 1) / 2

    result = phi(x, a-1) - phi(x / primes[a-1], a-1)
    phi_cache[(x, a)] = result # Memoize
    return result


pi_cache = {}
def pi(x):
    """
    Computes pi(x), the number of primes <= x, using
    the Meissel-Lehmer algorithm.
    """
    # If value is cached, return it
    if x in pi_cache: return pi_cache[x]

    # If x < limit, calculate pi(x) using a bisection
    # algorithm over the sieved primes.
    if x < limit:
        result = bisect(primes, x)
        pi_cache[x] = result
        return result

    a = pi(int(x ** (1./4)))
    b = pi(int(x ** (1./2)))
    c = pi(int(x ** (1./3)))

    # This quantity must be integral,
    # so we can just use integer division.
    result = phi(x,a) + (b+a-2) * (b-a+1) / 2

    for i in range(a+1, b+1):
        w = x / primes[i-1]
        b_i = pi(w ** (1./2))
        result = result - pi(w)
        if i <= c:
            for j in range(i, b_i+1):
                result = result - pi(w / primes[j-1]) + j - 1
    pi_cache[x] = result
    return result

# Example
result = pi(10**6)
print(prime_sieve(100))
```

5 Graph algorithms

5.1 Topo sort

```
#include <bits/stdc++.h>

using namespace std;

const int maxN = 110;

int n, m;
int visit[maxN], ans[maxN];
vector <int> g[maxN];
stack <int> topo;

void dfs(int u) {
    visit[u] = 1;
    for (auto v : g[u]) {
        if (visit[v] == 1) {
            cout << "Error: graph contains a cycle";
            exit(0);
        }
        if (!visit[v]) dfs(v);
    }
    topo.push(u);
    visit[u] = 2;
}

main() {
    cin >> n >> m;
    while (m--) {
        int u, v;
        cin >> u >> v;
        g[u].push_back(v);
    }
    for (int i = 1; i <= n; ++i)
        if (!visit[i]) dfs(i);

    /* Sau khi xac dinh duoc thu tu To-po cua do thi, ta su dung
       mang ans de danh so lai cac dinh */
    int cnt = 0;
    while (!topo.empty()) {
        ans[topo.top()] = ++cnt;
        topo.pop();
    }

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

5.2 Liet ke thanh phan lien thong manh

```
#include <stdio.h>

#include <algorithm>
#include <iostream>
#include <stack>
#include <vector>
using namespace std;

const int N = 100005;
const int oo = 0x3c3c3c3c;
```

```
int n, m, Num[N], Low[N], cnt = 0;
vector<int> a[N];
stack<int> st;
int Count = 0;

void visit(int u) {
    Low[u] = Num[u] = ++cnt;
    st.push(u);

    for (int v : a[u])
        if (Num[v])
            Low[u] = min(Low[u], Num[v]);
        else {
            visit(v);
            Low[u] = min(Low[u], Low[v]);
        }

    if (Num[u] == Low[u]) {  // found one
        Count++;
        int v;
        do {
            v = st.top();
            st.pop();
            Num[v] = Low[v] = oo;  // remove v from graph
        } while (v != u);
    }
}

int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++) {
        int x, y;
        scanf("%d%d", &x, &y);
        a[x].push_back(y);
    }

    for (int i = 1; i <= n; i++)
        if (!Num[i]) visit(i);

    cout << Count << endl;
}
```

5.3 Khop cau

```
#include <bits/stdc++.h>

using namespace std;

const int MAXN = 10007;

int num[MAXN], low[MAXN], cntDFS = 0, n, m, bridge = 0, child, K[MAXN];
vector <int> adj[MAXN];

void DFS(int u, int par)
{
    num[u] = low[u] = ++cntDFS;
    for (int v: adj[u])
        if (!num[v])
        {
            child += !par;
            DFS(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] >= num[u]) K[u] = 1;
            if (low[v] > num[u]) ++bridge;
        }
        else if (v != par)
            low[u] = min(low[u], num[v]);
}

int main()
{
//    freopen("TEST.INP", "r", stdin);
    ios_base::sync_with_stdio(0);
    cin.tie(0), cout.tie(0);
    cin >> n >> m;
    while (m--)
    {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    for (int i = 1; i <= n; ++i)
        if (!num[i])
        {
            child = 0;
            DFS(i, 0);
            K[i] = child > 1;
        }

    int khop = 0;
    for (int i = 1; i <= n; ++i) khop += K[i];
    cout << khop << ' ' << bridge;
    return 0;
}
```

5.4 Dijkstra nhung trong so canh co the ¡ 0 O(nm)

```
#include <stdio.h>
#include <queue>
using namespace std;

#define X first
#define Y second
typedef pair<int, int> ii;
vector<ii> a[230997];
int n, m;

int d[230997];
bool inqueue[230997];

void bellman(int u) {
    queue<int> qu;

    for (int i = 1; i <= n; i++)
        d[i] = 1000111000;
    d[u] = 0;
    qu.push(u);
    inqueue[u] = true;

    while (qu.size()) {
        u = qu.front();
        inqueue[u] = false;
        qu.pop();

        for (int i = 0; i < a[u].size(); i++) {
            int v = a[u][i].Y;
            int uv = a[u][i].X;
            if (d[v] > d[u] + uv) {
                d[v] = d[u] + uv;
                if (!inqueue[v]) {
                    qu.push(v);
                    inqueue[v] = true;
                }
            }
        }
    }
}
```

```
int main() {
    int u, v;
    scanf("%d%d%d%d", &n, &m, &u, &v);
    u++;
    v++;
    while (m--) {
        int p, q, w;
        scanf("%d%d%d", &p, &q, &w);
        p++;
        q++;
        a[p].push_back(ii(w, q));
        a[q].push_back(ii(w, p));
    }
    bellman(u);
    printf("%d\n", d[v]);
}
```

5.5 Liet ke nhom cuc dai cua do thi vo huong

```
int n, m, w[MAXN];
vector<vector<int>> edge(MAXN);

int res = 0;

void BronKerbosch(set<int> R, set<int> P, set<int> X, int gold = 0){
    if (P.empty() && X.empty()) {
        res = max(res, gold);
    }
    set<int> P_copy = P;
    for (int v : P_copy){
        set<int> R1 = R, P1 = {}, X1 = {};
        if (R.find(v) == R.end()) gold += w[v];
        R1.insert(v);
        for (int u : edge[v])
            if (P.find(u) != P.end()) P1.insert(u);
        for (int u : edge[v])
            if (X.find(u) != X.end()) X1.insert(u);
        BronKerbosch(R1, P1, X1, gold);
        gold -= w[v];
        X.insert(v);
        P.erase(v);
    }
}
int main(){
    while (cin >> n >> m){
        set<int> P;
        res = 0;
        for (int i = 1; i <= n; i++)
            P.insert(i);
        for (int i = 1; i <= n; i++){
            cin >> w[i];
            res = max(res, w[i]);
            edge[i].clear();
        }
        for (int i = 1; i <= m; i++){
            int u, v;
            cin >> u >> v;
            edge[u].pb(v);
            edge[v].pb(u);
        }
        BronKerbosch({}, P, {}, 0);
        cout << res << endl;
    }
}
```

5.6 Eulerian Path Algo

```
#include "template.h"

struct Edge;
typedef list<Edge>::iterator iter;

struct Edge {
  int next_vertex;
  iter reverse_edge;

  Edge(int next_vertex) :next_vertex(next_vertex)
    { }
};

const int max_vertices = 10;
// int num_vertices = 6;
list<Edge> adj[max_vertices];   // adjacency list

vector<int> path;

void find_path(int v) {
  while(adj[v].size() > 0) {
    int vn = adj[v].front().next_vertex;
    adj[vn].erase(adj[v].front().reverse_edge);
    adj[v].pop_front();
    find_path(vn);
  }
  path.push_back(v);
}

void add_edge(int a, int b) {
  adj[a].push_front(Edge(b));
  iter ita = adj[a].begin();
  adj[b].push_front(Edge(a));
  iter itb = adj[b].begin();
  ita->reverse_edge = itb;
  itb->reverse_edge = ita;
}

int main() {
  int total=0, start_vertex = 0;
  rep(i, max_vertices)
    if(adj[i].size()&1)                          // if the size is odd then
        increment 'total'
      total++, start_vertex=i;          // put the starting vertex as an odd
          degree vertex
  if(total==0||total==2) {                       // necessary and sufficient
      condition to check the existence of an EC
    find_path(start_vertex);
    rep(i, path.size()) cout << path[i] << " ";
  }
  else
    cout << "No Eulerian Circuit\n";
  return 0;
}
```

5.7 Tim song lien thong

```
// Danh sach ke
typedef vector<vector<int>> dsk;

#define N 30001

// Cau truc disjoint-sets
int root[N];
int find(int u) {
    if (root[u] != u) root[u] = find(root[u]);
    return root[u];
}

bool visited[N];
int active[N];
```

```
vector<int> stack;

void dfs(int u, const dsk &ke) {
    visited[u] = true;
    root[u] = u;
    stack.push_back(u);

    for (int v: ke[u]) if (visited[v]) {
        v = find(active[v]);
        while (stack.back() != v) {
            root[find(stack.back())] = v;
            stack.pop_back();
        }
    }

    for (int v: ke[u]) if (!visited[v]) {
        active[u] = v;
        dfs(v, ke);
    }

    if (stack.back() == u) stack.pop_back();
}
```

5.8 Closest Pair of points in a 2D Plane

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <iomanip>
using namespace std;

const int N = 2e5;
const long long  INF = 3e18;

class point{
public:
    long long  x, y;
};

class cmp_x{
public:
    bool operator()(const point &A, const point &B) {
        return A.x < B.x;
    }
};

class cmp_y{
public:
    bool operator()(const point &A, const point &B) {
        return A.y < B.y;
    }
};

point set_of_points[N];

long long  distance_AB(const point &A, const point &B) {
    return (A.x - B.x)*(A.x - B.x) + (A.y - B.y)*(A.y - B.y);
}

long long  find_min_distance(int l, int r) {
    //Neu doan [l, r] chi co 1 phan tu
    if (r-l == 0) return INF;
    //Neu doan [l, r] co 2 phan tu
    if (r-l == 1) {
        if (set_of_points[l].y > set_of_points[r].y) swap(set_of_points[l],
            set_of_points[r]);
        return distance_AB(set_of_points[l], set_of_points[r]);
    }

    long long  result = INF;

    int mid = (l+r) >>1;
    point mid_point = set_of_points[mid];
    result = min(find_min_distance(l, mid), find_min_distance(mid+1, r));

    static point Tam[N];
    int d = -1;
    //Su dung Merge Sort de sap xep doan [l, r] tang dan theo tung do va cho
        vao mang Tam
    merge(set_of_points+l, set_of_points+mid+1, set_of_points+mid+1,
        set_of_points+r+1, Tam, cmp_y());
    //Sap xep lai doan [l, r] tren set_of_points bang cach dan mang Tam vao
        set_of_points
    copy(Tam, Tam+r-l+1, set_of_points+l);

    for(int i = l; i <= r; i++) if (abs(set_of_points[i].x - mid_point.x) <
        sqrt(result)) {
        for(int j = d; (j >= 0) && set_of_points[i].y < Tam[j].y + sqrt(
            result); j--)
            result = min(result, distance_AB(set_of_points[i], Tam[j]));
        Tam[++d] = set_of_points[i];
    }
    return result;
}

int main() {
    ios::sync_with_stdio(false); cin.tie(0);

    //Input
    int n;
    cin >>n;
    for(int i=0; i<n; i++) cin >>set_of_points[i].x >>set_of_points[i].y;

    //Solution
    sort(set_of_points, set_of_points+n, cmp_x());
    long long  min_distance = find_min_distance(0, n-1);

    //Output
    cout <<fixed <<setprecision(9) <<sqrt((long double) min_distance);
}
```

6 Data structures

6.1 HLD

```
void cal(ll u, ll p) {
    nChild[u] = 1;
    for(ll v : vertex[u]) if (v != p) {
        parent[v] = u;
        cal(v, u);
        nChild[u] += nChild[v];
    }
}

// nChain chuoi hien tai. Sau khi ket thuc viec phan tach thi day se la tong
    so chuoi.
// chainHead[c] dinh dau cua chuoi c
// chainInd[u] chuoi ma dinh u nam trong.

void hld(int u) {

    // Neu chuoi hien tai chua co dinh dau (dinh gan goc nhat) thi dat u lam
        dinh dau cua no.
        if (chainHead[nChain] == 0) chainHead[nChain] = u;

    // Gan chuoi hien tai cho u
        chainInd[u] = nChain;

    // danh so thu tu khi su dung trong IT/BIT
        posInBase[u] = ++nBase;
        // Anh xa tu chi so trong cay IT den chi so cua nut tren cay
        BaseInpos[nBase] = u;
```

```
    // Bien luu dinh con dac biet cua u
        int mxVtx = -1;

    // Tim dinh con dac biet trong so nhung dinh con cua u
        for (int i = 0; i < vertex[u].size(); i++) {
                int v = vertex[u][i];
                if (v != parent[u]) {
                        if (mxVtx == -1 || nChild[v] > nChild[mxVtx]) {
                                mxVtx = v;
                        }
                }
        }

    // Neu tim ra dinh con dac biet (u khong phai la dinh leaf) thi di chuyen
         den dinh do
        if (mxVtx > -1)
                hld(mxVtx);

    // Sau khi di het mot chuoi thi tang nChain len va bat dau mot chuoi moi
        for (int i = 0; i < vertex[u].size(); i++) {
                int v = vertex[u][i];
                if (v != parent[u] && v != mxVtx) {
                        nChain++;
                        hld(v);
                }
        }
}

void update(int u, int a) {
    // uchain chuoi hien tai cua u
    // achain chuoi cua a
     int uchain = chainInd[u], achain = chainInd[a];

     while (1) {
        // Neu u va a cung nam tren mot chuoi thi update doan tu u den a va
            ket thuc.
          if (uchain == achain) {
               IT.update_tree(1, 1, n, posInBase[a], posInBase[u]);
               break;
          }
        // Neu u va a khong nam tren cung mot chuoi thi update doan tu u den
            dinh dau cua chuoi hien tai.
          IT.update_tree(1, 1, n, posInBase[chainHead[uchain]], posInBase[u])
              ;

        // Nhay len dinh cha cua dinh dau hien tai.
          u = parent[chainHead[uchain]];
          uchain = chainInd[u];
     }
}
```

7 String Manipulation

7.1 KMP

```
void KMP(string text,string pattern){
      int n = text.length(), m = pattern.length(), F[m+2],i,j;
      F[0] = F[1] = 0;
      for(int i = 2;i<=m;i++){
           j = F[i-1];
           while(true){
                if(pattern[j] == pattern[i-1]) { F[i] = j+1; break;}
                else if(j==0) {F[i] = 0; break;}
                else j = F[j];
           }
      }

      i = j = 0;
      while(j<n){
           if(text[j] == pattern[i]){
                i++; j++;
                if(i==m) printf("%d ",j-i+1);
           }
           else if(i>0) i = F[i];
           else j++;
      }
}

int main(){
     string a,b;
     cin>>a>>b;
     KMP(a,b);
    // getch();
}
```

7.2 Suffix array

```
#include <bits/stdc++.h>

using namespace std;

const int MAXN = 4e5 + 7;

int main()
{
//    freopen("test.inp", "r", stdin);
//    freopen("test.out", "w", stdout);
    ios_base::sync_with_stdio(0);
    cin.tie(0), cout.tie(0);
    string s;
    cin >> s;
    s += "$";
    int n = s.size();
    vector <int> p(n), c(n);
    {
        vector <pair <char, int> > a(n);
        for (int i = 0; i < n; ++i) a[i] = {s[i], i};
        sort(a.begin(), a.end());
        c[p[0] = a[0].second] = 0;
        for (int i = 1; i < n; ++i)
            c[p[i] = a[i].second] = c[p[i - 1]] + (a[i].first != a[i - 1].
                first);
    }

    for (int k = 0; (1 << k) < n; ++k)
    {
        for (int i = 0; i < n; ++i)
            p[i] = (p[i] - (1 << k) + n) % n;

        {
            vector <int> new_p(n), cnt(n, 0), pos(n, 0);
            for (int x: c) ++cnt[x];
            for (int i = 1; i < n; ++i) pos[i] = pos[i - 1] + cnt[i - 1];
            for (int x: p) new_p[pos[c[x]]++] = x;
            p = new_p;
        }

        {
            vector <int> new_c(n, 0);
            for (int i = 1; i < n; ++i)
            {
                pair <int, int> cur = {c[p[i]], c[(p[i] + (1 << k)) % n]};
                pair <int, int> prev = {c[p[i - 1]], c[(p[i - 1] + (1 << k))
                    % n]};
                new_c[p[i]] = new_c[p[i - 1]] + (cur != prev);
            }
            c = new_c;
        }
```

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

    vector <int> lcp(n);
    int k = 0;
    for (int i = 0; i < n - 1; ++i)
    {
        int pi = c[i], j = p[pi - 1];
        while (s[i + k] == s[j + k]) ++k;
        lcp[pi] = k;
        k = max(0, k - 1);
    }
    for (int i = 1; i < n; ++i) cout << lcp[i] << ' ';
        return 0;
}
```

7.3 Z's Algorithm, KMP's Bro

```
int L = 0, R = 0;
Z[0] = n;
for (int i = 1; i < n; i++)
   if (i > R)
   {
      L = R = i;
      while (R < n && S[R] == S[R - L]) R++;
      Z[i] = R - L; R--;
   }
   else
   {
      int k = i - L;
      if (Z[k] < R - i + 1) Z[i] = Z[k];
      else
      {
          L = i;
          while (R < n && S[R] == S[R - L]) R++;
          Z[i] = R - L; R--;
      }
   }
```

8 Extra

8.1 Playing with dates

```
// Routines for performing computations on dates.  In these routines,
// months are expressed as integers from 1 to 12, days are expressed
// as integers from 1 to 31, and years are expressed as 4-digit
// integers.
#include "template.h"
string dayOfWeek[] = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};

// converts Gregorian date to integer (Julian day number)
int dateToInt (int m, int d, int y){
  return
    1461 * (y + 4800 + (m - 14) / 12) / 4 +
    367 * (m - 2 - (m - 14) / 12 * 12) / 12 -
    3 * ((y + 4900 + (m - 14) / 12) / 100) / 4 +
    d - 32075;
}

// converts integer (Julian day number) to Gregorian date: month/day/year
void intToDate (int jd, int &m, int &d, int &y){
  int x, n, i, j;
  x = jd + 68569;
  n = 4 * x / 146097;
  x -= (146097 * n + 3) / 4;
  i = (4000 * (x + 1)) / 1461001;
  x -= 1461 * i / 4 - 31;
  j = 80 * x / 2447;
  d = x - 2447 * j / 80;
  x = j / 11;
  m = j + 2 - 12 * x;
  y = 100 * (n - 49) + i + x;
}

// converts integer (Julian day number) to day of week
string intToDay (int jd){ return dayOfWeek[jd % 7]; }
```

8.2 Mobius function

```
void MOB(int n){
  vector<int> mob(n);
  for(int i = 1; i < n ; ++i)mob[i] = 1;
  for(int i = 2; i < N ; i++){
    if(pf[i] == i){
      if(1LL * i * i < N){
        for(int j = i*i ; j < N ; j += (i*i) ){
          mob[j] = 0;
        }
      }
      for(int j = i ; j < N ; j+=i){
        mob[j] *= -1;
      }
    }
  }
}
```

8.3 Mo's Algorithm O((N+Q)*sqrt(N))

```
S = sqrt(N);
bool cmp(Query A, Query B) // so sanh 2 truy van
{
  if (A.l / S != B.l / S) {
    return A.l / S < B.l / S;
  }
  return A.r < B.r;
}
```

9 Theory

Combinatorics

Lagrange $P(x) = \sum_{i=1}^{n+1} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$

Sums

$\sum_{k=0}^{n} k = n(n+1)/2$
$\sum_{k=a}^{b} k = (a+b)(b-a+1)/2$
$\sum_{k=0}^{n} k^2 = n(n+1)(2n+1)/6$
$\sum_{k=0}^{n} k^3 = n^2(n+1)^2/4$
$\sum_{k=0}^{n} k^4 = (6n^5 + 15n^4 + 10n^3 - n)/30$
$\sum_{k=0}^{n} k^5 = (2n^6 + 6n^5 + 5n^4 - n^2)/12$
$\sum_{k=0}^{n} x^k = (x^{n+1} - 1)/(x - 1)$
$\sum_{k=0}^{n} kx^k = (x - (n+1)x^{n+1} + nx^{n+2})/(x-1)^2$
$1 + x + x^2 + \cdots = 1/(1-x)$

$\binom{n}{k} = \frac{n!}{(n-k)!k!}$
$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$
$\binom{n+1}{k} = \frac{n+1}{n-k+1}\binom{n}{k}$
$\binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}$
$\binom{n}{k} = \frac{n}{n-k}\binom{n-1}{k}$
$\binom{n}{k} = \frac{n-k+1}{k}\binom{n}{k-1}$
$12! \approx 2^{28.8}$
$20! \approx 2^{61.1}$

Binomial coefficients

Number of ways to pick a multiset of size k from n elements: $\binom{n+k-1}{k}$
Number of n-tuples of non-negative integers with sum s: $\binom{s+n-1}{n-1}$, at most s: $\binom{s+n}{n}$
Number of n-tuples of positive integers with sum s: $\binom{s-1}{n-1}$
Number of lattice paths from $(0,0)$ to (a,b), restricted to east and north steps: $\binom{a+b}{a}$

Multinomial theorem. $(a_1 + \cdots + a_k)^n = \sum \binom{n}{n_1, \ldots, n_k} a_1^{n_1} \ldots a_k^{n_k}$, where $n_i \geq 0$ and $\sum n_i = n$.

$$\binom{n}{n_1, \ldots, n_k} = M(n_1, \ldots, n_k) = \frac{n!}{n_1! \ldots n_k!}$$

$$M(a, \ldots, b, c, \ldots) = M(a + \cdots + b, c, \ldots) M(a, \ldots, b)$$

Catalan numbers. $C_n = \frac{1}{n+1}\binom{2n}{n}$. $C_0 = 1$, $C_n = \sum_{i=0}^{n-1} C_i C_{n-1-i}$. $C_{n+1} = C_n \frac{4n+2}{n+2}$.
$C_0, C_1, \ldots = 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, \ldots$
C_n is the number of: properly nested sequences of n pairs of parentheses; rooted ordered binary trees with $n+1$ leaves; triangulations of a convex $(n+2)$-gon.

Derangements. Number of permutations of $n = 0, 1, 2, \ldots$ elements without fixed points is $1, 0, 1, 2, 9, 44, 265, 1854, 14833, \ldots$ Recurrence: $D_n = (n-1)(D_{n-1} + D_{n-2}) = nD_{n-1} + (-1)^n$. Corollary: number of permutations with exactly k fixed points is $\binom{n}{k} D_{n-k}$.

Stirling numbers of 1^{st} kind. $s_{n,k}$ is $(-1)^{n-k}$ times the number of permutations of n elements with exactly k permutation cycles. $|s_{n,k}| = |s_{n-1,k-1}| + (n-1)|s_{n-1,k}|$. $\sum_{k=0}^{n} s_{n,k} x^k = x^{\underline{n}}$

Stirling numbers of 2^{nd} kind. $S_{n,k}$ is the number of ways to partition a set of n elements into exactly k non-empty subsets. $S_{n,k} = S_{n-1,k-1} + kS_{n-1,k}$. $S_{n,1} = S_{n,n} = 1$. $x^n = \sum_{k=0}^{n} S_{n,k} x^{\underline{k}}$

Bell numbers. B_n is the number of partitions of n elements. $B_0, \ldots = 1, 1, 2, 5, 15, 52, 203, \ldots$
$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k = \sum_{k=1}^{n} S_{n,k}$. Bell triangle: $B_r = a_{r,1} = a_{r-1,r-1}$, $a_{r,c} = a_{r-1,c-1} + a_{r,c-1}$.

Bernoulli numbers. $\sum_{k=0}^{m-1} k^n = \frac{1}{n+1} \sum_{k=0}^{n} \binom{n+1}{k} B_k m^{n+1-k}$.
$\sum_{j=0}^{m} \binom{m+1}{j} B_j = 0$. $B_0 = 1$, $B_1 = -\frac{1}{2}$. $B_n = 0$, for all odd $n \neq 1$.

Eulerian numbers. $E(n,k)$ is the number of permutations with exactly k descents $(i : \pi_i < \pi_{i+1})$ / ascents $(\pi_i > \pi_{i+1})$ / excedances $(\pi_i > i)$ / $k+1$ weak excedances $(\pi_i \geq i)$.
Formula: $E(n,k) = (k+1)E(n-1,k) + (n-k)E(n-1,k-1)$. $x^n = \sum_{k=0}^{n-1} E(n,k)\binom{x+k}{n}$.

Burnside's lemma. The number of orbits under group G's action on set X: $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X_g|$, where $X_g = \{x \in X : g(x) = x\}$. ("Average number of fixed points.")
Let $w(x)$ be weight of x's orbit. Sum of all orbits' weights: $\sum_{o \in X/G} w(o) = \frac{1}{|G|} \sum_{g \in G} \sum_{x \in X_g} w(x)$.

Number Theory

Linear diophantine equation. $ax + by = c$. Let $d = \gcd(a,b)$. A solution exists iff $d|c$. If (x_0, y_0) is any solution, then all solutions are given by $(x, y) = (x_0 + \frac{b}{d}t, y_0 - \frac{a}{d}t)$, $t \in \mathbb{Z}$. To find some solution (x_0, y_0), use extended GCD to solve $ax_0 + by_0 = d = \gcd(a,b)$, and multiply its solutions by $\frac{c}{d}$.

Linear diophantine equation in n variables: $a_1x_1 + \cdots + a_nx_n = c$ has solutions iff $\gcd(a_1, \ldots, a_n)|c$. To find some solution, let $b = \gcd(a_2, \ldots, a_n)$, solve $a_1x_1 + by = c$, and iterate with $a_2x_2 + \cdots = y$.

Extended GCD

```
// Finds g = gcd(a,b) and x, y such that ax+by=g.
// Bounds: |x|<=b+1, |y|<=a+1.
void gcdext(int &g, int &x, int &y, int a, int b)
{ if (b == 0) { g = a; x = 1; y = 0; }
  else        { gcdext(g, y, x, b, a % b); y = y - (a / b) * x; } }
```

Multiplicative inverse of a modulo m: x in $ax + my = 1$, or $a^{\phi(m)-1} \pmod{m}$.

Chinese Remainder Theorem. System $x \equiv a_i \pmod{m_i}$ for $i = 1, \ldots, n$, with pairwise relatively-prime m_i has a unique solution modulo $M = m_1 m_2 \ldots m_n$: $x = a_1 b_1 \frac{M}{m_1} + \cdots + a_n b_n \frac{M}{m_n} \pmod{M}$, where b_i is modular inverse of $\frac{M}{m_i}$ modulo m_i.

System $x \equiv a \pmod{m}$, $x \equiv b \pmod{n}$ has solutions iff $a \equiv b \pmod{g}$, where $g = \gcd(m,n)$. The solution is unique modulo $L = \frac{mn}{g}$, and equals: $x \equiv a + T(b-a)m/g \equiv b + S(a-b)n/g \pmod{L}$, where S and T are integer solutions of $mT + nS = \gcd(m,n)$.

Prime-counting function. $\pi(n) = |\{p \leq n : p \text{ is prime}\}|$. $n/\ln(n) < \pi(n) < 1.3n/\ln(n)$. $\pi(1000) = 168$, $\pi(10^6) = 78498$, $\pi(10^9) = 50\,847\,534$. n-th prime $\approx n \ln n$.

Miller-Rabin's primality test. Given $n = 2^r s + 1$ with odd s, and a random integer $1 < a < n$.
If $a^s \equiv 1 \pmod{n}$ or $a^{2^j s} \equiv -1 \pmod{n}$ for some $0 \leq j \leq r-1$, then n is a probable prime. With bases 2, 7 and 61, the test indentifies all composites below 2^{32}. Probability of failure for a random a is at most 1/4.

Pollard-ρ. Choose random x_1, and let $x_{i+1} = x_i^2 - 1 \pmod{n}$. Test $\gcd(n, x_{2^k+i} - x_{2^k})$ as possible n's factors for $k = 0, 1, \ldots$ Expected time to find a factor: $O(\sqrt{m})$, where

m is smallest prime power in n's factorization. That's $O(n^{1/4})$ if you check $n = p^k$ as a special case before factorization.

Fermat primes. A Fermat prime is a prime of form $2^{2^n}+1$. The only known Fermat primes are 3, 5, 17, 257, 65537. A number of form 2^n+1 is prime only if it is a Fermat prime.

Perfect numbers. $n > 1$ is called perfect if it equals sum of its proper divisors and 1. Even n is perfect iff $n = 2^{p-1}(2^p-1)$ and 2^p-1 is prime (Mersenne's). No odd perfect numbers are yet found.

Carmichael numbers. A positive composite n is a Carmichael number ($a^{n-1} \equiv 1 \pmod n$ for all a: $\gcd(a,n) = 1$), iff n is square-free, and for all prime divisors p of n, $p-1$ divides $n-1$.

Number/sum of divisors. $\tau(p_1^{a_1} \dots p_k^{a_k}) = \prod_{j=1}^{k}(a_j+1)$. $\sigma(p_1^{a_1} \dots p_k^{a_k}) = \prod_{j=1}^{k} \frac{p_j^{a_j+1}-1}{p_j-1}$.

Euler's phi function. $\phi(n) = |\{m \in \mathbb{N}, m \le n, \gcd(m,n) = 1\}|$.
$\phi(mn) = \frac{\phi(m)\phi(n)\gcd(m,n)}{\phi(\gcd(m,n))}$. $\phi(p^a) = p^{a-1}(p-1)$. $\sum_{d|n}\phi(d) = \sum_{d|n}\phi(\frac{n}{d}) = n$.

Euler's theorem. $a^{\phi(n)} \equiv 1 \pmod n$, if $\gcd(a,n) = 1$.

Wilson's theorem. p is prime iff $(p-1)! \equiv -1 \pmod p$.

Mobius function. $\mu(1) = 1$. $\mu(n) = 0$, if n is not squarefree. $\mu(n) = (-1)^s$, if n is the product of s distinct primes. Let f, F be functions on positive integers. If for all $n \in N$, $F(n) = \sum_{d|n} f(d)$, then $f(n) = \sum_{d|n}\mu(d)F(\frac{n}{d})$, and vice versa. $\phi(n) = \sum_{d|n}\mu(d)\frac{n}{d}$. $\sum_{d|n}\mu(d) = 1$.
If f is multiplicative, then $\sum_{d|n}\mu(d)f(d) = \prod_{p|n}(1-f(p))$, $\sum_{d|n}\mu(d)^2 f(d) = \prod_{p|n}(1+f(p))$. **Common multiplicative function**

+ $\mu * 1 = \varepsilon$ (the Mobius inversion formula)
+ $\varphi * 1 = \mathrm{Id}$
+ $d = 1 * 1$
+ $\sigma = \mathrm{Id} * 1 = \varphi * d$
+ $\sigma_k = \mathrm{Id}_k * 1$
+ $\mathrm{Id} = \varphi * 1 = \sigma * \mu$
+ $\mathrm{Id}_k = \sigma_k * \mu$
+ $\sigma_k(n)$: the divisor function, which is the sum of the k-th powers of all the positive divisors of n.

S da giac loi n dinh $S = \frac{1}{2}\sum_{i=0}^{n-1} x_i y_{i+1} - x_{i+1}y_i =$¿ dien tich: —S—

Legendre symbol. If p is an odd prime, $a \in \mathbb{Z}$, then $\left(\frac{a}{p}\right)$ equals 0, if $p|a$; 1 if a is a quadratic residue modulo p; and -1 otherwise. Euler's criterion: $\left(\frac{a}{p}\right) = a^{\left(\frac{p-1}{2}\right)} \pmod p$.

Jacobi symbol. If $n = p_1^{a_1} \cdots p_k^{a_k}$ is odd, then $\left(\frac{a}{n}\right) = \prod_{i=1}^{k}\left(\frac{a}{p_i}\right)^{k_i}$.

Primitive roots. If the order of g modulo m (min $n > 0$: $g^n \equiv 1 \pmod m$) is $\phi(m)$, then g is called a primitive root. If Z_m has a primitive root, then it has $\phi(\phi(m))$ distinct primitive roots. Z_m has a primitive root iff m is one of 2, 4, p^k, $2p^k$, where p is an odd prime. If Z_m has a primitive root g, then for all a coprime to m, there exists unique integer $i = \mathrm{ind}_g(a)$ modulo $\phi(m)$, such that $g^i \equiv a \pmod m$. $\mathrm{ind}_g(a)$ has logarithm-like properties: $\mathrm{ind}(1) = 0$, $\mathrm{ind}(ab) = \mathrm{ind}(a) + \mathrm{ind}(b)$.

If p is prime and a is not divisible by p, then congruence $x^n \equiv a \pmod p$ has $\gcd(n, p-1)$ solutions if $a^{(p-1)/\gcd(n,p-1)} \equiv 1 \pmod p$, and no solutions otherwise. (Proof sketch: let g be a primitive root, and $g^i \equiv a \pmod p$, $g^u \equiv x \pmod p$. $x^n \equiv a \pmod p$ iff $g^{nu} \equiv g^i \pmod p$ iff $nu \equiv i \pmod p$.)

Discrete logarithm problem. Find x from $a^x \equiv b \pmod m$. Can be solved in $O(\sqrt{m})$ time and space with a meet-in-the-middle trick. Let $n = \lceil\sqrt{m}\rceil$, and $x = ny - z$. Equation becomes $a^{ny} \equiv ba^z \pmod m$. Precompute all values that the RHS can take for $z = 0, 1, \dots, n-1$, and brute force y on the LHS, each time checking whether there's a corresponding value for RHS.

Pythagorean triples. Integer solutions of $x^2 + y^2 = z^2$ All relatively prime triples are given by: $x = 2mn, y = m^2 - n^2, z = m^2 + n^2$ where $m > n, \gcd(m,n) = 1$ and $m \not\equiv n \pmod 2$. All other triples are multiples of these. Equation $x^2 + y^2 = 2z^2$ is equivalent to $(\frac{x+y}{2})^2 + (\frac{x-y}{2})^2 = z^2$.

Postage stamps/McNuggets problem. Let a, b be relatively-prime integers. There are exactly $\frac{1}{2}(a-1)(b-1)$ numbers *not* of form $ax + by$ $(x, y \ge 0)$, and the largest is $(a-1)(b-1) - 1 = ab - a - b$.

Fermat's two-squares theorem. Odd prime p can be represented as a sum of two squares iff $p \equiv 1 \pmod 4$. A product of two sums of two squares is a sum of two squares. Thus, n is a sum of two squares iff every prime of form $p = 4k+3$ occurs an even number of times in n's factorization.

RSA. Let p and q be random distinct large primes, $n = pq$. Choose a small odd integer e, relatively prime to $\phi(n) = (p-1)(q-1)$, and let $d = e^{-1} \pmod{\phi(n)}$. Pairs (e,n) and (d,n) are the public and secret keys, respectively. Encryption is done by raising a message $M \in Z_n$ to the power e or d, modulo n.

String Algorithms

Burrows-Wheeler inverse transform. Let $B[1..n]$ be the input (last column of sorted matrix of string's rotations.) Get the first column, $A[1..n]$, by sorting B. For each k-th occurence of a character c at index i in A, let $next[i]$ be the index of corresponding k-th occurence of c in B. The r-th fow of the matrix is $A[r]$, $A[next[r]]$, $A[next[next[r]]]$, ...

Huffman's algorithm. Start with a forest, consisting of isolated vertices. Repeatedly merge two trees with the lowest weights.

Graph Theory

Euler's theorem. For any planar graph, $V - E + F = 1 + C$, where V is the number of graph's vertices, E is the number of edges, F is the number of faces in graph's planar drawing, and C is the number of connected components. Corollary: $V - E + F = 2$ for a 3D polyhedron.

Vertex covers and independent sets. Let M, C, I be a max matching, a min vertex cover, and a max independent set. Then $|M| \le |C| = N - |I|$, with equality for bipartite graphs. Complement of an MVC is always a MIS, and vice versa. Given a bipartite graph with partitions (A, B), build a network: connect source to A, and B to sink with edges of capacities, equal to the corresponding nodes' weights, or 1 in the unweighted case. Set capacities of the original graph's edges to the infinity. Let (S, T) be a minimum s-t cut. Then a maximum(-weighted) independent set is $I = (A \cap S) \cup (B \cap T)$, and a minimum(-weighted) vertex cover is $C = (A \cap T) \cup (B \cap S)$.

Matrix-tree theorem. Let matrix $T = [t_{ij}]$, where t_{ij} is the number of multiedges between i and j, for $i \ne j$, and $t_{ii} = -\deg_i$. Number of spanning trees of a graph is equal to the determinant of a matrix obtained by deleting any k-th row and k-th column from T.

Euler tours. Euler tour in an undirected graph exists iff the graph is connected and each vertex has an even degree. Euler tour in a directed graph exists iff in-degree of each vertex equals its out-degree, and underlying undirected graph is connected. Construction:

```
doit(u):
  for each edge e = (u, v) in E, do: erase e, doit(v)
  prepend u to the list of vertices in the tour
```

Stable marriages problem. While there is a free man m: let w be the most-preferred woman to whom he has not yet proposed, and propose m to w. If w is free, or is engaged to someone whom she prefers less than m, match m with w, else deny proposal.

Stoer-Wagner's min-cut algorithm. Start from a set A containing an arbitrary vertex. While $A \ne V$, add to A the most tightly connected vertex ($z \notin A$ such that $\sum_{x \in A} w(x, z)$ is maximized.) Store cut-of-the-phase (the cut between the last added vertex and rest of the graph), and merge the two vertices added last. Repeat until the graph is contracted to a single vertex. Minimum cut is one of the cuts-of-the-phase.

Tarjan's offline LCA algorithm. (Based on DFS and union-find structure.)

```
DFS(x):
  ancestor[Find(x)] = x
  for all children y of x:
     DFS(y); Union(x, y); ancestor[Find(x)] = x
  seen[x] = true
  for all queries {x, y}:
    if seen[y] then output "LCA(x, y) is ancestor[Find(y)]"
```

Strongly-connected components. Kosaraju's algorithm.

1. Let G^T be a transpose G (graph with reversed edges.)
1. Call DFS(G^T) to compute finishing times $f[u]$ for each vertex u.
3. For each vertex u, in the order of decreasing $f[u]$, perform DFS(G, u).
4. Each tree in the 3rd step's DFS forest is a separate SCC.

2-SAT. Build an implication graph with 2 vertices for each variable – for the variable and its inverse; for each clause $x \vee y$ add edges $(\overline{x}, y)$ and $(\overline{y}, x)$. The formula is satisfiable iff x and $\overline{x}$ are in distinct SCCs, for all x. To find a satisfiable assignment, consider the graph's SCCs in topological order from sinks to sources (i.e. Kosaraju's last step), assigning 'true' to all variables of the current SCC (if it hasn't been previously assigned 'false'), and 'false' to all inverses.

Randomized algorithm for non-bipartite matching. Let G be a simple undirected graph with even $|V(G)|$. Build a matrix A, which for each edge $(u, v) \in E(G)$ has $A_{i,j} = x_{i,j}$, $A_{j,i} = -x_{i,j}$, and is zero elsewhere. Tutte's theorem: G has a perfect matching iff $\det G$ (a multivariate polynomial) is identically zero. Testing the latter can be done by computing the determinant for a few random values of $x_{i,j}$'s over some field. (e.g. Z_p for a sufficiently large prime p)

Prufer code of a tree. Label vertices with integers 1 to n. Repeatedly remove the leaf with the smallest label, and output its only neighbor's label, until only one edge remains. The sequence has length $n - 2$. Two isomorphic trees have the same sequence, and every sequence of integers from 1 and n corresponds to a tree. Corollary: the number of labelled trees with n vertices is n^{n-2}.

Erdos-Gallai theorem. A sequence of integers $\{d_1, d_2, \ldots, d_n\}$, with $n - 1 \ge d_1 \ge d_2 \ge \cdots \ge d_n \ge 0$ is a degree sequence of some undirected simple graph iff $\sum d_i$ is even and $d_1 + \cdots + d_k \le k(k-1) + \sum_{i=k+1}^{n} \min(k, d_i)$ for all $k = 1, 2, \ldots, n - 1$.

Games

Grundy numbers. For a two-player, normal-play (last to move wins) game on a graph (V, E): $G(x) = \text{mex}(\{G(y) : (x, y) \in E\})$, where $\text{mex}(S) = \min\{n \ge 0 : n \notin S\}$. x is losing iff $G(x) = 0$.

Sums of games.

- *Player chooses a game and makes a move in it.* Grundy number of a position is xor of grundy numbers of positions in summed games.
- *Player chooses a non-empty subset of games (possibly, all) and makes moves in all of them.* A position is losing iff each game is in a losing position.
- *Player chooses a proper subset of games (not empty and not all), and makes moves in all chosen ones.* A position is losing iff grundy numbers of all games are equal.
- *Player must move in all games, and loses if can't move in some game.* A position is losing if any of the games is in a losing position.

Misère Nim. A position with pile sizes $a_1, a_2, \ldots, a_n \ge 1$, not all equal to 1, is losing iff $a_1 \oplus a_2 \oplus \cdots \oplus a_n = 0$ (like in normal nim.) A position with n piles of size 1 is losing iff n is *odd*.

Bit tricks

Clearing the lowest 1 bit: `x & (x - 1)`, all trailing 1's: `x & (x + 1)`
Setting the lowest 0 bit: `x | (x + 1)`
Enumerating subsets of a bitmask m:
`x=0; do { ...; x=(x+1+~m)&m; } while (x!=0);`
`__builtin_ctz`/`__builtin_clz` returns the number of trailing/leading zero bits.

`__builtin_popcount(unsigned x)` counts 1-bits (slower than table lookups). For 64-bit unsigned integer type, use the suffix '`ll`', i.e. `__builtin_popcountll`.

Math

Stirling's approximation $z! = \Gamma(z+1) = \sqrt{2\pi}\, z^{z+1/2}\, e^{-z}(1+\frac{1}{12z}+\frac{1}{288z^2}-\frac{139}{51840z^3}+\dots)$

Taylor series. $f(x) = f(a) + \frac{x-a}{1!}f'(a) + \frac{(x-a)^2}{2!}f^{(2)}(a) + \cdots + \frac{(x-a)^n}{n!}f^{(n)}(a) + \dots$.

$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$

$\ln x = 2(a + \frac{a^3}{3} + \frac{a^5}{5} + \dots)$, where $a = \frac{x-1}{x+1}$. $\ln x^2 = 2\ln x$.

$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots$, $\arctan x = \arctan c + \arctan \frac{x-c}{1+xc}$ (e.g c=.2)

$\pi = 4\arctan 1$, $\pi = 6\arcsin \frac{1}{2}$

List of Primes

1e5	3	19	43	49	57	69	103	109	129	151	153
1e6	33	37	39	81	99	117	121	133	171	183	
1e7	19	79	103	121	139	141	169	189	223	229	
1e8	7	39	49	73	81	123	127	183	213		