Number Theory

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

ll gcd(ll a, ll b) { return b ? gcd(b, a % b) : a;}

ll inverse_mod(ll a, ll b) {
 return 1 < a ? b - inverse_mod(b % a, a) * b / a : 1;
}

ll inv[N];  // inverse modulo pre calcaulate
void imod() {
 inv[1] = 1;
 for (ll i = 2; i < N; i++) inv[i] = (mod - (mod / i) * inv[mod
% i]) % mod;
}
// another way to find inverse modulo of n is bigmod(n,
mod - 2, mod);

// inclusion-exclusion returns number of multiples of divs
in [l, r]
ll iep(int l, int r) {
 if (l > r) return 0;

 ll sum = 0, sz = divs.size();
 for (ll j = 1; j < (1LL << sz); j++) {
   ll gun = 1, one = 0;
   for (ll i = 0; i < sz; i++)
     if (j & (1LL << i)) {
       one++;
       gun *= divs[i];
     }

   ll mult = (r / gun) - ((l - 1) / gun);
   if (one % 2 == 1)
     sum += mult;
   else
     sum -= mult;
 }
 return sum;
}
```

```
// bitset <N+1> mark;
int phi[N];
void phi_sieve() {
 for (int i = 0; i < N; i++) phi[i] = i;

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

int Phi(int n) {
 int ph = n;
 for (int i = 2; i * i <= n; i++) {
   if (n % i) continue;
   while (n % i == 0) n /= i;
   ph -= ph / i;  // ph * (1 - 1/p)
 }
 if (n > 1) ph -= ph / n;
 return ph;
}
```

Chinese Remainder Theorem

```
ll CRT_weak(vector<ll>A, vector<ll>B) {
    ll X=0;
    ll N=1;
    ll y,z;
    for(ll i=0; i<B.size(); i++)
        N*=B[i];
    for(ll i=0; i<A.size(); i++) {
        y=N/B[i];
        z=modInv(y,B[i]);
        X+=(A[i]*y*z);
        X%=N;
    }
    return (X+N)%N;
}
```

Miller Rabin Primality Test

```
/* Miller Rabin Primality Test for <= 10^18 */
#define ll long long
```

```
ll mulmod(ll a, ll b, ll c) {
 ll x = 0, y = a % c;
 while (b)  {
       if (b & 1) x = (x + y) % c;
       y = (y << 1) % c;
       b >>= 1;
   }
   return x % c;
}
ll fastPow(ll x, ll n, ll MOD) {
   ll ret = 1;
   while (n) {
       if (n & 1) ret = mulmod(ret, x, MOD);
       x = mulmod(x, x, MOD);
       n >>= 1;
   }
   return ret % MOD;
}
bool isPrime(ll n) {
   if(n == 2 || n == 3) return true;
   if(n == 1 || !(n & 1)) return false;
   ll d = n - 1;
   int s = 0;
   while (d % 2 == 0) {
       s++;
       d /= 2;
   }

   int a[9] = { 2, 3, 5, 7, 11, 13, 17, 19, 23 };
   for(int i = 0; i < 9; i++)  {
       if(n == a[i]) return true;
       bool comp = fastPow(a[i], d, n) != 1;
       if(comp) for(int j = 0; j < s; j++)  {
              ll fp = fastPow(a[i], (1LL << (ll)j)*d, n);
              if (fp == n - 1) {
                  comp = false;
                  break;
              }
          }
       if(comp) return false;
   }
   return true;
}
```

FFT

```
/***
* Multiply (7x^2 + 8x^1 + 9x^0) with (6x^1 + 5x^0)
* ans = 42x^3 + 83x^2 + 94x^1 + 45x^0
* A = {9, 8, 7}
* B = {5, 6}
* V = multiply(A,B)
* V = {45, 94, 83, 42}
***/
/*** Tricks
* Use vector < bool > if you need to check only the status of the sum
* Use bigmod if the power is over same polynomial && power is big
* Use long double if you need more precision
* Use long long for overflow
***/
typedef vector<int> vi;
const double PI = 2.0 * acos(0.0);
using cd = complex<double>;
void fft(vector<cd> &a, bool invert = 0) {
 int n = a.size();
 for (int i = 1, j = 0; i < n; i++) {
   int bit = n >> 1;
   for (; j & bit; bit >>= 1)
     j ^= bit;
   j ^= bit;

   if (i < j)
     swap(a[i], a[j]);
 }
 for (int len = 2; len <= n; len <<= 1) {
   double ang = 2 * PI / len * (invert ? -1 : 1);
   cd wlen(cos(ang), sin(ang));
   for (int i = 0; i < n; i += len) {
     cd w(1);
     for (int j = 0; j < len / 2; j++) {
       cd u = a[i + j], v = a[i + j + len / 2] * w;
       a[i + j] = u + v;
       a[i + j + len / 2] = u - v;
       w *= wlen;
     }
   }
 }
 if (invert) {
   for (cd &x : a)
     x /= n;
 }
}

void ifft(vector<cd> &p) { fft(p, 1); }

vi multiply(vi const &a, vi const &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] *= fb[i];
 ifft(fa);

 vi result(n);
 for (int i = 0; i < n; i++)
   result[i] = round(fa[i].real());
 return result;
}
```

Combinatorics

Matrix Exponential

```
const ll N = 2, mod = 1000000007;
struct matrix {
 ll mat[N][N];

 matrix(int a, int b, int c, int d) {
   mat[0][0] = a;
   mat[0][1] = b;
   mat[1][0] = c;
   mat[1][1] = d;
 }

 matrix operator*(const matrix &another) {
   matrix res(0, 0, 0, 0);

   for (int i = 0; i < N; i++)
     for (int j = 0; j < N; j++) {
       for (int k = 0; k < N; k++) {
         res.mat[i][j] = (res.mat[i][j] + mat[i][k] *
another.mat[k][j]);
         if (res.mat[i][j] > 8 * mod * mod)
           res.mat[i][j] -=
               8 * mod * mod;  // to reduce mod operation,
8*mod*mod and mod should be <= 1e9+7
       }

       res.mat[i][j] %= mod;
     }
   // res.mat[i][j] = (res.mat[i][j] + mat[i][k] *
another.mat[k][j]) % mod;
   return res;
 }
};
matrix expo(matrix a, ll n) {
    if(n == 1) return a;

    matrix ret = expo(a, n / 2);
    ret = ret * ret;
    if(n & 1) ret = ret * a;

    return ret;
}
```

Data Structure

BIT

```
void edit(int pos) {
    while (pos <= n) {
        bit[pos] += 1;
        pos += pos & -pos;
    }
}

int sum(int pos) {
    int s = 0;
    while (pos) {
        s += bit[pos];
        pos -= pos & -pos;
    }
    return s;
}
```

Graph

Bellman Ford

```
//bellman ford with negative cycle print
struct edge {
 ll v, w;
};
const ll N = 3e3 + 6, inf = 1LL << 60;
ll n, m, dis[N], par[N];
vector<edge> g[N];

int bellman_ford() {
 lop(n + 1) dis[i] = inf;
 dis[1] = 0;
 int cy;
```

```
 lop(n + 1) {
   cy = -1;
   for (int u = 1; u <= n; u++) {
     for (auto z : g[u]) {
       ll v = z.v, w = z.w;
       if (dis[u] + w < dis[v]) {
         dis[v] = dis[u] + w;
         par[v] = u;
         cy = v; // if(u == n) negative cycle;
       } } } }
 return cy; //cy is a adjacent node or a node of
negative cycle
}

int main() {
 cin >> n >> m;
 lop(m) {
   ll u, v, w;
   cin >> u >> v >> w;
   g[u].pb({v, w});
 }

 int x = bellman_ford();
 if (x == -1) {
   //no negative cycle
   return 0;
 }

//x can be not a part of cycle, so if we go through
//path sometimes, x will be a node of cycle
 lop(n) x = par[x];
 vector<int> cycle;
 int i = x;
 while (i != x or cycle.size() <= 1) {
   cycle.pb(i);//retrieving cycle
   i = par[i];
 }
 cycle.pb(i);
 reverse(all(cycle));
 for (int z : cycle)
   cout << z << ' ';
 return 0;
}
```

Kruskal

```
struct edge {
 int u, v, w;
 bool operator<(const edge &b) const { return w > b.w; }
};

const int N = 2e5;
int n, m, par[N];
vector<edge> eg;

int findpar(int x) { return par[x] = par[x] == x ? x :
findpar(par[x]); }

void Union(int u, int v) { par[findpar(u)] = findpar(v); }

int kruskal() {
 sort(eg.begin(), eg.end());
 iota(par, par + n, 0);

 int cost = 0, connected = 0;
 while (connected != n - 1) {
   edge z = eg.back();
   eg.pop_back();

   int x = findpar(z.u), y = findpar(z.v);
   if (x != y) {
     connected++;
     cost += z.w;
     Union(x, y);
   }
 }

 return cost;
}

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

 cout << kruskal() << '\n';

 return 0;
}
```

DP

LIS (NlogN)

```
int lis(vector<int> const& a) {
 int n = a.size();
 const int INF = 1e9; //INF must be > max(a)
 vector<int> d(n + 1, INF);
 d[0] = -INF;

 for (int i = 0; i < n; i++) {
   int j = upper_bound(d.begin(), d.end(), a[i]) - d.begin();
   if (d[j - 1] <= a[i] && a[i] <= d[j]) d[j] = a[i];
 }

 int ans = 0;
 for (int i = 0; i <= n; i++) {
   if (d[i] < INF) ans = i;
 }
 return ans;
}
```

String

Hash

```
const ll N = 1e6 + 10, mod = 2e9 + 63, base1 = 1e9 + 21,
base2 = 1e9 + 181;
ll pw1[N], pw2[N], hp1[N], hp2[N], hs1[N], hs2[N], n, q;
string s;

void pw_cal() {
 pw1[0] = pw2[0] = 1;
 for (int i = 1; i < N; i++) {
   pw1[i] = (pw1[i - 1] * base1) % mod;
   pw2[i] = (pw2[i - 1] * base2) % mod;
 }
}
void init() {
 hp1[0] = hp2[0] = hs1[n + 1] = hs2[n + 1] = 0;
 for (int i = 1; i <= n; i++) {
   hp1[i] = (hp1[i - 1] * base1 + s[i - 1]) % mod;
   hp2[i] = (hp2[i - 1] * base2 + s[i - 1]) % mod;
 }
 for (int i = n; i > 0; i--) {
   hs1[i] = (hs1[i + 1] * base1 + s[i - 1]) % mod;
   hs2[i] = (hs2[i + 1] * base2 + s[i - 1]) % mod;
 }
}

ll hashp(int l, int r) {
 ll hash1 = ( hp1[r] - hp1[l - 1] * pw1[r - l + 1]) % mod;
 if (hash1 < 0) hash1 += mod;
 ll hash2 = ( hp2[r] - hp2[l - 1] * pw2[r - l + 1]) % mod;
 if (hash2 < 0) hash2 += mod;
 return (hash1 << 32) | hash2;
}
ll hashs(int l, int r) {
 ll hash1 = (hs1[l] - hs1[r + 1] * pw1[r - l + 1]) % mod;
 if (hash1 < 0) hash1 += mod;
 ll hash2 = (hs2[l] - hs2[r + 1] * pw2[r - l + 1]) % mod;
 if (hash2 < 0) hash2 += mod;
 return (hash1 << 32) | hash2;
}

bool ispal(int l, int r) {
 int mid = (r + l) / 2;
 ll x = hashp(l, mid), y = hashs(mid, r);
 return x == y;
}
```

Suffix Array

```
#define MAX_N 1000020
int n, t;
// char s[500099];
string s;
int SA[MAX_N], LCP[MAX_N];
int RA[MAX_N], tempRA[MAX_N];
int tempSA[MAX_N];
int c[MAX_N];
int Phi[MAX_N], PLCP[MAX_N];
// second approach: O(n log n)
// the input string, up to 100K characters
// the length of input string
// rank array and temporary rank array
// suffix array and temporary suffix array
// for counting/radix sort
void countingSort(int k) {    // O(n)
    int i, sum, maxi = max(300, n);
    // up to 255 ASCII chars or length of n
    memset(c, 0, sizeof c);
    // clear frequency table
    for (i = 0; i < n; i++)
        // count the frequency of each integer rank
        c[i + k < n ? RA[i + k] : 0]++;
    for (i = sum = 0; i < maxi; i++) {
        int t = c[i]; c[i] = sum; sum += t;
    }
    for (i = 0; i < n; i++)
        // shuffle the suffix array if necessary
        tempSA[c[SA[i] + k < n ? RA[SA[i] + k] : 0]++] =
SA[i];
```

```
    for (i = 0; i < n; i++)
        // update the suffix array SA
        SA[i] = tempSA[i];
}

void buildSA() {
    int i, k, r;
    for (i = 0; i < n; i++) RA[i] = s[i];
    // initial rankings
    for (i = 0; i < n; i++) SA[i] = i;
    // initial SA: {0, 1, 2, ..., n-1}
    for (k = 1; k < n; k <<= 1) {
        // repeat sorting process log n times
        countingSort(k); // actually radix sort: sort based on
the second item
        countingSort(0);
        // then (stable) sort based on the first item
        tempRA[SA[0]] = r = 0;
        // re-ranking; start from rank r = 0
        for (i = 1; i < n; i++)
            // compare adjacent suffixes
            tempRA[SA[i]] = // if same pair => same rank r;
otherwise, increase r
                (RA[SA[i]] == RA[SA[i - 1]] && RA[SA[i] + k] ==
RA[SA[i - 1] + k]) ? r : ++r;
        for (i = 0; i < n; i++)
            // update the rank array RA
            RA[i] = tempRA[i];

        if (RA[SA[n - 1]] == n - 1) break;
        // nice optimization trick
    }
}

void buildLCP() {
    int i, L;
    Phi[SA[0]] = -1;
    // default value
    for (i = 1; i < n; i++)
        // compute Phi in O(n)
        Phi[SA[i]] = SA[i - 1];
    // remember which suffix is behind this suffix
    for (i = L = 0; i < n; i++) {
        // compute Permuted LCP in O(n)
        if (Phi[i] == -1) { PLCP[i] = 0; continue; }
        // special case
        while (s[i + L] == s[Phi[i] + L]) L++;
        // L increased max n times
        PLCP[i] = L;
        L = max(L - 1, 0);
        // L decreased max n times
    }
    for (i = 0; i < n; i++)
        // compute LCP in O(n)
        LCP[i] = PLCP[SA[i]];
    // put the permuted LCP to the correct position
}
// n = string length + 1
// s = the string
// memset(LCP, 0, sizeof(LCP));  setting all index of LCP
to zero
// buildSA(); for building suffix array
// buildLCP(); for building LCP array
// LCP is the longest common prefix with the previous
suffix here
// SA[0] holds the empty suffix "\0".

int main()
{
    s = "banana";
    s += "$";
    n = s.size();

    memset(LCP, 0, sizeof(LCP));
    buildSA();
    buildLCP();

    for (int i = 0; i < n; i++) cout << SA[i] << ' ' <<
s.substr(SA[i], n - SA[i]) << endl;;
    printf("\n");
    for (int i = 0; i < n; i++) printf("%d ", LCP[i]);
    printf("\n");

    return 0;
}
```

Trie

```
const int N = 1e5+10, M = 26;
int trie[N][M], nnode;
bool isword[N];

void reset(int k) {
        for (int i = 0; i < M; i++)
                trie[k][i] = -1;
}
```

```
void Insert(string &s) {
	int n = s.size(), node = 0;
	for (int i = 0; i < n; i++) {
		if (trie[node][s[i] - 'a'] == -1) {
			trie[node][s[i] - 'a'] = ++nnode;
			reset(nnode);
		}
		node = trie[node][s[i] - 'a'];
	}
	isword[node] = 1;
}

bool Search(string &s) {
	int n = s.size(), node = 0;
	for (int i = 0; i < s.size(); i++) {
		if (trie[node][s[i] - 'a'] == -1) return 0;
		node = trie[node][s[i] - 'a'];
	}
	return isword[node];
}
//find maximum subarray xor sum
int doxor(int s) {
  int nw = 0, t = 0;
  for (int i = 31; i >= 0; i--) {
    bool p = (1 << i) & s;
    if (node[nw][p ^ 1] != -1) {
      t |= 1 << i;
      nw = node[nw][p ^ 1];
    } else
      nw = node[nw][p];
  }
  return t;
}
//minimum subarray xor sum
int doxor2(int s) {
  int nw = 0, t = 0;
  for (int i = 31; i >= 0; i--) {
    bool p = (1 << i) & s;
    if (node[nw][p] != -1) nw = node[nw][p];
    else {
      t |= 1 << i;
      nw = node[nw][p ^ 1];
    }
  }
  return t;
}
//at first insert(0), then calculate xor before inserting each element of the array
//calculate number of subarray having xor>=k
int doxor(int s) {
  int nw = 0, t = 0;
  for (int i = 31; i >= 0; i--) {
    bool p = (1 << i) & s;
    bool q = (1 << i) & k;
    if (!q) {
      t += (node[nw][p ^ 1] != -1 ? word[node[nw][p ^ 1]] : 0);
      nw = node[nw][p];
    } else
      nw = node[nw][p ^ 1];
    if (nw == -1)
      break;
  }
  if (nw != -1)
    t += word[nw];
  return t;
}
//insert(0), sum returned value, insert prefix xor

int main() {
	reset(0);
	int n; cin >> n;
	for (int i = 0; i < n; i++) {
		string s;
		cin >> s;
		Insert(s);
	}
	int q; cin >> q;
	while (q--) {
		string s;
		cin >> s;
		cout << Search(s) << endl;
	}
}
```

Z-Algo

```
vector<int> z_function(string s) {
    int n = (int) s.length();
    vector<int> z(n);
    for (int i = 1, l = 0, r = 0; i < n; ++i) {
        if (i <= r)
            z[i] = min (r - i + 1, z[i - l]);
```

```
        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
            ++z[i];
        if (i + z[i] - 1 > r)
            l = i, r = i + z[i] - 1;
    }
    return z;
}
```

Miscellaneous

Debug

```
//generator to generate testcase
#include <bits/stdc++.h>
using namespace std;
#define ll long long
mt19937_64
rng(chrono::steady_clock::now().time_since_epoch().cou
nt());
// return a random number in [l, r] range
ll rand(ll l, ll r) {
        return uniform_int_distribution<ll>(l, r)(rng);
}
void tree() {
 int n = rand(1, 10);
 int t = rand(1, 5);

 vector<int> p(n);
 for (int i = 0; i < n; i++)
   if (i > 0) p[i] = rand(i, t);

 printf("%d\n", n);
 vector<int> perm(n);
 for (int i = 0; i < n; i++) perm[i] = i;
 shuffle(perm.begin() + 1, perm.end(), rng);
 vector<pair<int, int> > edges;

 for (int i = 1; i < n; i++)
   if (rand(0, 1))
     edges.push_back(make_pair(perm[i], perm[p[i]]));
   else
     edges.push_back(make_pair(perm[p[i]], perm[i]));

 shuffle(edges.begin(), edges.end(), rng);
 for (int i = 0; i + 1 < n; i++)
   printf("%d %d\n", edges[i].first+1, edges[i].second+1);
}

int main(int argc, char* argv[]) {
        ll t = rand(1, 1);
        cout << t << endl;
        while (t--) {
                ll n = rand(1, 15);
                cout << n << endl;
        }
        return 0;
}

//Bash script to auto check output
for((i = 1; ; ++i)); do
    echo $i
    ./gen $i > in
    # ./a < in > out1
    # ./brute < in > out2
    # diff -w out1 out2 || break
    diff -w <(./sol < in) <(./brute < in) || break
done
echo case
cat in
/* create and build a bruteforce code named brute.cpp,
main solution code sol.cpp and a random test case
generator gen.cpp. To make this script runable, run this
command chmod +x s.sh (s.sh is this bash script name).
Then run the script by ./s.sh or bash s.sh */
```

Make Faster

```
#pragma GCC optimize("Ofast,unroll-loops")
#pragma GCC target("avx,avx2,fma")
ios_base::sync_with_stdio(0); cin.tie(0);
```

C++17 Sublime Build

```
{
"cmd" : ["g++ -std=c++17 -Wshadow -Drifat
-Wmaybe-uninitialized -fsanitize=address $file_name -o
$file_base_name && timeout 20s ./$file_base_name
<input.txt >output.txt"],
"selector" : "source.cpp",
"shell": true,
"working_dir" : "$file_path"
}
```

Extra Notes

Stars and Bars

The number of ways to put n identical objects into k labeled boxes is ((n+k-1)choose(n)).....(n+k-1)C(n)

1. Suppose, there are n objects to be placed into k bins, ways= (n-1)C(k-1)
2. Statement of 1no. and empty bins are valid, ways= (n+k-1)C(k-1)

GCD

1. gcd(a, b) = gcd(a, a - b) [a > b]

2. gcd(F(a), F(b)) = F(gcd(a,b)) [F=fibonacci]

coordinate geometry formula

1. Point Slope Form: $y - y_1 = m(x - x_1)$
2. Slope, m = Δy/Δx = (y2 − y1)/(x2 − x1)
3. Slope from line, m = -(A/B)
4. Angle, tan θ = [(m1 – m2)/ (1 + m1m2)]
5. Distance from a Point to a Line, d = [|Ax0 + By0 + C| / √(A2 + B2)]

circle formula

Area of segment in radian angle : A = (½) × r^2 (θ – Sin θ)

sod nod

1. (a+1)(b+1)(c+1) [Number of divisors, a, b, c are powers of prime number]
2. $\frac{(p^{a+1}-1)}{p-1}.\frac{(q^{b+1}-1)}{q-1}$ Here p,q is a prime numbers [Sum of Divisors]

n-th term

1.সমান্তর ধারা: nতম পদ=$a + (n-1)d$, sum= $\frac{n\{2a+(n-1)d\}}{2}$

2.গুণোত্তর ধারা: nতম পদ=ar^{n-1}, sum=$\frac{a(r^{n}-1)}{r-1}$

3.Catalan Numbers: 1, 1, 2, 5, 14, 42, 132…..
C_n =(2n)!/(n+1)!n! n>=0

Progression

1. Sum of first n positive number = n*(n+1)/2
2. Sum of first n odd number = n^2
3. Sum of first n even number = n*(n+1)

polygon area, diagonal formula

The sum of interior angles of a polygon with “n” sides =180°*(**n-**2). Number of diagonals of a “n-sided” polygon = [n(n-3)]/2. The measure of interior angles of a regular n-sided polygon = [(n-2)180°]/n. The measure of exterior angles of a regular n-sided polygon = 360°/n. Picks theorem: A = I + (B/2) - 1 where A = Area of Polygon, B = Number of integral points on edges of polygon, I = Number of integral points strictly inside the polygon.

modular arithmetic

$a^{\varphi(n)} = 1 \% n$ where φ(n) is Euler Totient Function.

$a^b \% m = a^{b \% \varphi(m)} \% m$ where *a* and *m* are coprime.

~ 4 Direction
```
int dr[] = {1,-1,0,0}; int dc[] = {0,0,1,-1};
```
~ 8 Direction
```
int dr[] = {1,-1,0,0,1,1,-1,-1}; int dc[] = {0,0,1,-1,1,-1,1,-1};
```
~ Knight Direction
```
int dr[] = {1,-1,1,-1,2,2,-2,-2};int dc[] = {2,2,-2,-2,1,-1,1,-1};
```
~ Hexagonal Direction
```
int dr[] = {2,-2,1,1,-1,-1}; int dc[] = {0,0,1,-1,1,-1};
```
~ bitmask operations
```
int Set(int n, int pos) { return n = n | (1 << pos); }
int reset(int n, int pos) { return n = n & ~(1 << pos); }
int toggle(int n, int pos) { return n = (n ^ (1 << pos));}
bool check(int n, int pos) { return (bool)(n & (1 << pos)); }
bool isPower2(int x) { return (x && !(x & (x - 1))); }
ll LargestPower2LessEqualX(ll x) { for(int i = 1; i <= x / 2;
i *= 2) x = x | (x >> i); return (x + 1) / 2;}
```

Template of Ashraful :::

In Out DP

```
int in[100002];
int out[100002];
vector<int> adj[100003];
int n;
void dfs1(int node, int parent) {
  for (int son : adj[node]) {
    if (son == parent) continue;
    dfs1(son, node);
    in[node] = max(in[node], 1 + in[son]);
  }
}
void dfs2(int node, int parent) {
  int mx1 = -1, mx2 = -1;
  for (int son : adj[node]) {
    if (son == parent) continue;
    if (in[son] >= mx1)
      mx2 = mx1, mx1 = in[son];
    else if (in[son] > mx2)
      mx2 = in[son];
  }
  for (int son : adj[node]) {
    if (son == parent) continue;
    int use = mx1;
    if (in[son] == use) use = mx2;
    out[son] = max(1 + out[node], 2 + use);
    dfs2(son, node);
  }
}
int main() {
  /// find the height of the tree from every node.
  cin >> n;  /// how many nodes
  for (int i = 0; i < n - 1; i++) {
    int u, v;
    cin >> u >> v;
    adj[u].pb(v);
    adj[v].pb(u);
  }
  dfs1(1, 0);  /// calculate in values from node 1.
```

```
  dfs2(1, 0);  /// calculate out values from node  1.
  for (int i = 1; i <= n; i++) cout << max(in[i], out[i]) <<
endl;
}
```

PBDS

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
    using namespace __gnu_pbds;
template <typename T>
using orderedSet =
    tree<T, null_type, less<T>, rb_tree_tag,
tree_order_statistics_node_update>;
/// use less_equal in pbds template to work as multiset.
int main() {
  /// for erasing a single value use
st.erase(st.upper_bound(val));
  orderedSet<int> st;
  st.insert(10);
  st.insert(30);
  st.insert(33);
  st.insert(3);
  st.insert(5);
  int x = st.order_of_key(7);  /// find number of element
strictly less than 7.....
  cout << x << endl;
  /// Finding the element in specific index.
  cout << *(st.find_by_order(3)) << endl;
}
```

Topsort Kahn's Algo

```
vector<vector<int>>  adj(500);
vector<int> res;
queue<int> q;
int indegree[500];
void topsort() {
  for (int i = 1; i <= n; i++) {
    if (indegree[i] == 0) q.push(i);
  }
  while (!q.empty()) {
    int x = q.front();
    q.pop();
    res.pb(x);
    for (auto i : adj[x]) {
      --indegree[i];
      if (indegree[i] == 0) q.push(i);
    }
  }
}
```

Ternary Search

```
int ternary_search(int lo, int hi) {
  int res = 0;
  while (lo + 2 < hi)  /// cause,if len<=3 and we are
working with only integer
                       /// range then ternary will not work.
  {
    int m1 = lo + (hi - lo) / 3;
    int m2 = hi - (hi - lo) / 3;
    double op1 = check(m1);  /// always try to check the
answer in
                             /// double..cause,two different middle
point
    double op2 = check(m2);  /// can give the same answer
in integer
    if (op1 < op2) {
      res = op2;
      lo = m1 + 1;
    } else {
      res = op1;
      hi = m2 - 1;
    }
  }
  for (int i = lo; i <= hi; i++) res = max(res, (int)check(i));
  return res;
}
```

SCC_Tarjan's Algo

```
/* SCC means the largest subset of nodes where we can
go from any node to other nodes. SCC may contain
multiple loops. Complexity : O(V+E)*/
vector<int> adj[100004];
int n, m;
int vis[100005];
stack<int> st;
int on_stack[100004];
int in[100005];
int lo[100005];
int tme, cnt;
int scc_num[10004];
void dfs(int node) {
  in[node] = lo[node] = ++tme;
  vis[node] = 1;
  on_stack[node] = 1;
  st.push(node);
  for (auto son : adj[node]) {
    if (on_stack[son] && vis[son]) {
      lo[node] = min(lo[node], in[son]);
    } else if (!vis[son]) {
      dfs(son);
      if (on_stack[son]) lo[node] = min(lo[node], lo[son]);
    }
```

```
  }
  if (in[node] == lo[node])  /// From Where the SCC
started.
  {
    cout << "SCC num " << ++cnt << ":" << endl;
    while (1) {
      int x = st.top();
      st.pop();
      cout << x << " ";
      scc_num[x] = cnt;  /// Marked the scc num for graph
condensation....
      on_stack[x] = 0;
      if (x == node) break;
    }
    cout << endl;
  }
}
void find_SCC() {
  cin >> n >> m;
  for (int i = 0; i < m; i++) {
    int x, y;
    cin >> x >> y;
    adj[x].pb(y);
  }
  for (int i = 1; i <= n; i++) {
    if (vis[i]) continue;
    dfs(i);
  }
  /// Graph condensation
  vector<int> v[cnt + 3];
  for (int i = 1; i <= n; i++) {
    for (auto j : adj[i]) {
      if (scc_num[i] == scc_num[j]) continue;
      v[scc_num[i]].pb(scc_num[j]);  /// scc_num will be the
node numbers
    }
  }
}
```

Segmented Sieve

```
ll l, r;
vector<int> prime;
bool mark[1000000];
void simpleSieve(int n) {
  for (int i = 3; i <= sqrt(n); i += 2) {
    if (!mark[i]) {
      for (int j = i * i; j <= n; j += i * 2) mark[j] = etrue;
    }
  }
  prime.pb(2);
  for (int i = 3; i <= n; i += 2)
    if (!mark[i]) prime.pb(i);
}
void segmentedSieve(ll l, ll r) {
  simpleSieve(sqrt(r));
  bool mark2[r - l + 1];
  memset(mark2, false, sizeof mark2);
  for (int i = 0; i < prime.size(); i++) {
    ll x = prime[i] * 1LL;
    prime[i] = prime[i] * 1LL;
    x = (l / x) * x;
    if (x < l) x += prime[i];
    if (x == prime[i]) x += prime[i];
    for (ll j = x; j <= r; j += prime[i]) {
      int ind = j - l;
      mark2[ind] = true;
    }
  }
  for (ll i = l; i <= r; i++) {
    int ind = i - l;
    if (!mark2[ind]) cout << i << " ";
  }
  cout << endl;
}
int main() {
  cin >> l >> r;
  segmentedSieve(l, r);
}
```

Euler Tour

```
vector<int> v[1000];
int enter_time[1000];
int out_time[1000];
int save[2000];
int n;
int timer = 1;
int dfs(int node, int par) {
  enter_time[node] = timer;
  save[timer] = node;
  ++timer;
  for (auto son : v[node]) {
    if (son == par) continue;
    dfs(son, node);
  }
  out_time[node] = timer;
  save[timer] = node;
  ++timer;
}
int main() {
  /// needed for tree queries.....................
```

```
  /// nodes in a subtree is
(out_time[node]-enter_time[node])/2.
  cin >> n;
  for (int i = 0; i < n - 1; i++) {
    int x, y;
    cin >> x >> y;
    v[x].pb(y);
    v[y].pb(x);
  }
  dfs(1, 0);
  for (int i = 1; i <= 2 * n; i++) cout << save[i] << " ";
  cout << endl;
}
```

EGCD

```
int egcd(int a, int b, int *x, int *y) {
  if (b == 0) {
    *x = 1;
    *y = 0;
    return a;
  }
  int x1, y1;
  int g = egcd(b, a % b, &x1, &y1);
  *x = y1;
  *y = x1 - (a / b) * y1;
  return g;
}
int32_t main() {
  /// This works  for all mods.....
  /// a and mod must be co-prime.....
  /*
     1/12(mod 25)=??
     1/12(mod 25)=x;
     12x=1(mod 25)
     12x+25y=1(mod 25)...Diophantine
equation///(ax+by=gcd(a,b))...Bezouts
     identity..... now ,by solving this equation with
extended eucledean algo
     our ans=value of x..... by egcd ax+by=gcd(a,b)
     gcd(a,b)=gcd(b,a%b)..///always keep small .....
  */
  int a, b, x, y;
  cin >> a >> b;
  int gcd = egcd(a, b, &x, &y);
  cout << gcd << endl;
  cout << x << " " << y << endl;
}
```

KMP Algo

```
   /// prefix function/LPS function(longest prefix suffix
function)/Failure
   /// function find longest length of prefix=suffix...of
substr(0,idx) in
   /// lps[idx].....
vector<int> lps(100005, 0);
/// Efficient approach.
void lps_func() {
  int n = s.size();
  for (int i = 1; i < n; i++) {
    int j = lps[i - 1];
    while (j > 0 && s[i] != s[j]) j = lps[j - 1];
    if (s[i] == s[j]) ++j;
    lps[i] = j;
  }
}
int32_t main() {
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cin >> s;
  lps_func();
  for (int i = 0; i < s.size(); i++) cout << lps[i] << " ";
  cout << endl;
  /// Aplications.....
  /// 1-Find if a string s appear in string t.
  /// Just find the lps of :(s+#+t)...if lps[idx]==s.size() then
yes...time
  /// complexity.O(n+m).
}
```

RMQ using sparse table

```
int dp[10000][30];
int query(int l, int r) {
  int d = (r - l + 1);
  int x = log2(d);
  int y = r - (1 << x) + 1;
  return min(dp[l][x], dp[y][x]);
}
int32_t main() {
  /// find minimum in a range using sparse table.
  int n;
  cin >> n;
  int a[n + 5];
  for (int i = 0; i < n; i++) {
    cin >> a[i];
    dp[i][0] = a[i];  /// contains minimum value of 2^0 length
subarray from i.
  }
  for (int i = 1; (1 << i) <= n; i++)  /// 2^i len subarray
  {
    for (int j = 0; (j + (1 << i) - 1) < n; j++)  /// from idx j
    {
```

```
      dp[j][i] = min(dp[j][i - 1], dp[j + (1 << (i - 1))][i - 1]);
    }
  }
  int q;
  cin >> q;
  while (q--) {
    int l, r;
    cin >> l >> r;
    cout << query(l, r) << endl;
  }
}
```

Lowest Common Ancestor

```
void dfs(int node, int par) {
  LCA[node][0] = par;
  if (node > 1) level[node] = 1 + level[par];

  for (auto son : v[node]) {
    if (son == par) continue;
    dfs(son, node);
  }
}
void init() {
  dfs(1, 0);
  for (int i = 1; i <= 20; i++) {
    for (int j = 1; j <= n; j++) {
      if (level[j] < (1LL << i)) continue;
      int ancestor = LCA[j][i - 1];
      LCA[j][i] = LCA[ancestor][i - 1];
    }
  }
}
int lowest_common_ancestor(int a, int b) {
  if (level[a] > level[b]) swap(a, b);
  int d = level[b] - level[a];
  while (d) {
    int x = log2(d);
    d -= (1 << x);
    b = LCA[b][x];
  }
  if (a == b) return a;
  for (int i = 20; i >= 0; i--) {
    if (LCA[a][i] <= 0 || LCA[a][i] == LCA[b][i]) continue;
    a = LCA[a][i], b = LCA[b][i];
  }
  return LCA[a][0];
}
```

MCM

```
int solve(int i, int j) {
  if (i >= j) return 0;
  if (dp[i][j] != -1) return dp[i][j];
  int ans = INT_MAX;
  for (int k = i; k <= j - 1; k++) {
    int temp = solve(i, k) + solve(k + 1, j) + a[i - 1] * a[k] *
a[j];
    ans = min(ans, temp);
  }
  return dp[i][j] = ans;
}
```

Dinic's Algo

```
/// Complexity O(V*V*E).
#define ll long long
    const ll maxnodes = 10005;
ll nodes = maxnodes, src, dest;
ll dist[maxnodes], q[maxnodes], work[maxnodes];
struct Edge {
  ll to, rev;
  ll f, cap;
};
vector<Edge> g[maxnodes];
void addEdge(ll s, ll t, ll cap) {
  Edge a = {t, g[t].size(), 0, cap};
  Edge b = {s, g[s].size(), 0, 0};
  g[s].push_back(a);
  g[t].push_back(b);
}
bool dinic_bfs() {
  fill(dist, dist + nodes, -1);
  dist[src] = 0;
  ll index = 0;
  q[index++] = src;
  for (ll i = 0; i < index; i++) {
    ll u = q[i];
    for (ll j = 0; j < (ll)g[u].size(); j++) {
      Edge &e = g[u][j];
      if (dist[e.to] < 0 && e.f < e.cap) {
        dist[e.to] = dist[u] + 1;
        q[index++] = e.to;
      }
    }
  }
  return dist[dest] >= 0;
}
ll dinic_dfs(ll u, ll f) {
  if (u == dest) return f;
  for (ll &i = work[u]; i < (ll)g[u].size(); i++) {
    Edge &e = g[u][i];
    if (e.cap <= e.f) continue;
```

```
    if (dist[e.to] == dist[u] + 1) {
      ll flow = dinic_dfs(e.to, min(f, e.cap - e.f));
      if (flow > 0) {
        e.f += flow;
        g[e.to][e.rev].f -= flow;
        return flow;
      }
    }
  }
  return 0;
}
ll maxFlow(ll _src, ll _dest) {
  src = _src;
  dest = _dest;
  ll result = 0;
  while (dinic_bfs()) {
    fill(work, work + nodes, 0);
    while (ll delta = dinic_dfs(src, inf)) result += delta;
  }
  return result;
}
/// addEdge(u, v, C);    edge from u to v. Capacity is C
/// maxFlow(s, t);  max flow from s to t
```

Finding Bridges

```
const int mx = 10005;
vector<int> adj[mx];
int in[mx], lo[mx];  /// For storing entering time and
back-edge time.
int vis[mx];
int n, m;
int timer;
void dfs(int node, int par) {
  in[node] = lo[node] = timer++;
  vis[node] = 1;
  for (auto son : adj[node]) {
    if (son == par) continue;
    if (vis[son]) {
      lo[node] = min(lo[node], in[son]);  /// save the time of
back-edge
    } else {
      dfs(son, node);
      if (in[node] < lo[son]) {
        cout << node << "-" << son << " is a bridge." <<
endl;
      }
      lo[node] = min(lo[node], lo[son]);
    }
  }
}
int32_t main() {
  cin >> n >> m;
  for (int i = 0; i < m; i++) {
    int x, y;
    cin >> x >> y;
    adj[x].pb(y);
    adj[y].pb(x);
  }
  dfs(1, 0);
}
```

Kuhn's Algo

```
/// For Maximum Bipartite Matching
/// Complexity: O(min(n*m, n^3))
ll l_siz, r_siz; // l_siz = left part size, r_siz = right part size;
vector <ll> g[1500], lft, rgt;
vector <bool> used;
bool try_kuhn(ll v)
{
    for (ll &to : g[v]) {
        if(used[to]) continue;
        used[to] = 1;

        if(rgt[to]==-1 || try_kuhn(rgt[to])) {
            lft[v] = to, rgt[to] = v;
            return true;
        }
    }
    return false;
}
ll kuhn()
{
    ll max_match = 0;
    lft.assign(l_siz+1, -1), rgt.assign(r_siz+1, -1);
    for(ll v=1; v<=l_siz; ++v) {
        used.assign(r_siz+1, false);
        max_match += try_kuhn(v);
    }
    return max_match;
}
/* Optimized Kuhn's Algorithm. Blog:
https://codeforces.com/blog/entry/17023 */
ll kuhn2()
{
    ll max_match = 0;
    lft.assign(l_siz+1, -1), rgt.assign(r_siz+1, -1);
    // Shuffle the left part randomly to traverse them
randomly
```

```
    mt19937_64
rng(chrono::steady_clock::now().time_since_epoch().cou
nt()); // Random Seed
    vector <ll> lft_part;
    for(ll v=1; v<=l_siz; ++v) lft_part.push_back(v);
    shuffle(lft_part.begin(), lft_part.end(), rng);

    // Greedy matching with adjacent nodes at first
    for(auto &v : lft_part) {
        // Shuffle the adjacent nodes to match them
randomly
        shuffle(g[v].begin(), g[v].end(), rng);

        for(auto &to : g[v]) {
            if(rgt[to] == -1) {
                lft[v] = to, rgt[to] = v;
                max_match++;
                break;
            }
        }
    }
    // Main Kuhn's Algorithm Part
    bool new_mat = 1;
    while(new_mat) {
        // used is cleared one time in each iteration so that
we can find several
        // matchings in O(E). This makes the whole
algorithm significantly faster.
        used.assign(r_siz+1, false);
        // If no new match is found, the loop will break
        new_mat = 0;
        for(auto &v : lft_part) {
            if(lft[v] != -1)
                continue;

            bool got = try_kuhn(v);
            max_match += got, new_mat |= got;
        }
    }
    return max_match;
}
```

Floyd Warshall

```
void floyd(){
    for(int k=1;k<=n;k++)
        for(int i=1;i<=n;i++)
            for(int j=1;j<=n;j++)
                g[i][j]=min(g[i][j],g[i][k]+g[k][j]);
}
```

Centroid decomposition

```
/// decompose(1, -1)   //For 1 rooted tree
#define ll long long
#define pb push_back
const ll MAX = 1e5;
vector <ll> g[MAX + 9];
ll del[MAX + 9], sz[MAX + 9], par[MAX + 9], curSize;
void dfs(ll u, ll p){
    sz[u] = 1;
    for(ll i = 0; i < g[u].size(); i++) {
        ll nd = g[u][i];
        if(nd == p || del[nd])
            continue;
        dfs(nd, u);
        sz[u] += sz[nd];
    }
}
ll findCentroid(ll u, ll p){
    for(ll i = 0; i < g[u].size(); i++) {
        ll nd = g[u][i];
        if(nd == p || del[nd] || sz[nd] <= curSize / 2)
            continue;

        return findCentroid(nd, u);
    }
    return u;
}
void decompose(ll u, ll p){
    dfs(u, -1);
    curSize = sz[u];
    ll cen = findCentroid(u, -1);
    if(p == -1) p = cen;
    par[cen] = p, del[cen] = 1;
    for(ll i = 0; i < g[cen].size(); i++) {
        ll nd = g[cen][i];
        if(!del[nd])
            decompose(nd, cen);
    }
}
```

SOS DP

```
void SOS_DP()
{
    ///An Array A[] is given find sum for each masks over
all of it's submasks.
    ///iterative version
    for(int mask = 0; mask < (1<<N); ++mask){
```

```
    dp[mask][-1] = A[mask];  ///handle base case
separately (leaf states)
    for(int i = 0; i < N; ++i) {
        if(mask & (1<<i))
            dp[mask][i] = dp[mask][i-1] +
dp[mask^(1<<i)][i-1];
        else
            dp[mask][i] = dp[mask][i-1];
    }
    F[mask] = dp[mask][N-1];
  }
///memory optimized, super easy to code.
  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)];}
}
```

Monotonous queue

```
void Monotonous_queue()
{
  ///find largest number with smallest index of every
element of an array.....
  ///decreasing dq
  /*
  5
30 40 20 30 50
40 50 30 50 -1
  */
  deque<pair<int,int>>dq;
  int n;
  cin>>n;
  int a[n+6];
  int ans[n+4];

  for(int i=0; i<n; i++)
    cin>>a[i];
  dq.push_back({a[0],0});
  for(int i=1; i<n; i++)
  {
    while(!dq.empty() && dq.back().first<a[i])
    {
      ans[dq.back().second]=a[i];
      dq.pop_back();
    }
    dq.push_back({a[i],i});
  }
  ans[n-1]=-1;
  for(int i=0; i<n; i++)
  {
    cout<<ans[i]<<" ";
  }
  cout<<endl;
}
```

Memory optimized Knapsack

```
 for(int i=0; i<n; i++) {
      int w,p;
      cin>>w>>p;
      for(int j=cap; j>=w; j--)
      {
        dp[j]=max(dp[j],dp[j-w]+p);
      }

  }
```

Knapsack with huge capacity

```
 for(int i=0; i<n; i++)
  {
    cin>>w[i]>>p[i];
    sum+=p[i];
  }
  for(int i=0; i<n; i++)
  {
    for(int j=sum; j>=p[i]; j--)
    {
      dp[j]=min(dp[j],w[i]+dp[j-p[i]]);///max profit with
min cap..
    }
  }
  int ans=0;
  for(int i=0; i<=sum; i++)
    if(dp[i]<=cap) ans=max(ans,i);
  cout<<ans<<endl;
```

Digit DP

```
/**
  Find how many numbers are there less than N such
that,
  "count of distinct digits of the number is equal to the
maximum digit of the number".
*/
int dp[20][2][2][10][1LL<<10];
vector<int>num;
int n;
int solve(int pos,int is_small,int is_start,int mx,int msk)
{
  if(pos==-1)
```

```
    {
        if(!is_start) return 0;
        return  (mx==__builtin_popcount(msk));
    }
    int &ans=dp[pos][is_small][is_start][mx][msk];
    if(ans!=-1 && is_small) return ans;///return all possible
answer 99999999.....................
    ans=0;


    int till=(is_small?9:num[pos]);
    if(is_start)
    {
        for(int i=0; i<=till; i++)

ans+=solve(pos-1,is_small|(i<num[pos]),1,max(mx,i),msk
|(1LL<<i));
    }
    else
    {
        ans+=solve(pos-1,1,0,0,0);
        for(int i=1; i<=till; i++)
            {

ans+=solve(pos-1,is_small|(i<num[pos]),1,max(mx,i),msk
|(1LL<<i));
            }
    }
    return ans;
}
void pre_calc(int x)
{
    num.clear();
    while(x)
    {
        num.pb(x%10);
        x/=10;
    }
    n=num.size();
}
int32_t main()
{
    memset(dp,-1,sizeof dp);
    int t;
    cin>>t;
    while(t--)
    {
        int x;
        cin>>x;
        pre_calc(x);
        cout<<solve(n-1,0,0,0,0)<<endl;
    }
}
```

Arnab Baishnab's template ...

```
/*
#include <ext/pb_ds/assoc_container.hpp> // Common
file
#include <ext/pb_ds/tree_policy.hpp>
#include <functional> // for less
#include <iostream>
using namespace __gnu_pbds;
/**/
using namespace std;
#define   MP            make_pair
#define   PB            push_back
#define   nn            '\n'
#define   endl          '\n'
#define   IOS           ios::sync_with_stdio(0); cin.tie(0);
cout.tie(0);
#define   UNIQUE(vec)
vec.resize(distance(vec.begin(),unique(vec.begin(),vec.e
nd())));
#define   ClearVec(vec)  while(vec.size())vec.pop_back()
#define   ALL(vec)       vec.begin(),vec.end()
#define   int           long long
#define   pii           pair<int,int>

typedef long long LL ;

const int MOD=1e9+7,Base=998244353 ;
const int N=6e5+7 ;
const int
INF=1LL*1000*1000*1000*1000*1000*1000+7LL ,
INF2=(1LL<<62) ;
const double pie=acos(-1.0) ;
const double EPS=1e-9 ;
/**
pdds used to  solve
https://codeforces.com/problemset/problem/1311/F
typedef tree<int , null_type ,  less_equal<int> ,
rb_tree_tag , tree_order_statistics_node_update>
ordered_multiset;
```

```
ordered_multiset omset ; int
i=omset.order_of_key(obj[i].v) ;
/**/
/**
maxflow YouKnowWho 's template . not checked yet ..
https://codeforces.com/contest/498/submission/45850199

struct edge{
	int to, rev, flow, w;
};
struct dinic {
	int d[N], done[N], s, t;
	vector<edge> g[N];
	/// N equals to node_number

	void addedge(int u, int v, int w){
		edge a={v,(int)g[v].size(),0,w};
		edge b={u,(int)g[u].size(),0,0};

		/// If the graph has bidirectional edges
		/// Capacity for the edge b will equal to w
		/// For directed, it is 0

		g[u].emplace_back(a);
		g[v].emplace_back(b);
	}
	bool bfs(){
		memset(d,-1,sizeof(d));
		d[s]=0;
		queue<int>q;
		q.push(s);
		while(!q.empty()){
			int u=q.front();
			q.pop();
			for(auto &e: g[u])
			{
				int v=e.to;
				if(d[v]==-1 && e.flow<e.w)
				{
					d[v]=d[u]+1;
					q.push(v);
				}
			}
		}
		return d[t]!=-1;
	}

	int dfs(int u, int flow){
		if(u==t) return flow;
		for(int &i=done[u]; i<(int)g[u].size(); i++){
			edge &e=g[u][i];
			if(e.w<=e.flow) continue;
			int v=e.to;
			if(d[v]==d[u]+1){
			  int nw=dfs(v,min(flow,e.w-e.flow));
				if(nw>0){
					e.flow+=nw;
					g[v][e.rev].flow-=nw;
					return nw;
				}
			}
		}
		return 0;
	}

	int max_flow(int _s, int _t){
		s=_s;
		t=_t;
		int flow=0;
		while(bfs())
		{
			memset(done,0,sizeof(done));
			while(int nw=dfs(s,INF)) flow+=nw;
		}
		return flow;
	}
} ;

dinic flow ;
 /***/
/**
subset of all subset enumeration ... O(3^n)
for(int msk=0;msk<(1<<n);++msk){
   for(int i=msk;i>0;i=(i-1)&msk){
      if(msk^i)
dp[pa][msk]=min({dp[pa][msk],dp[nd][i]+dp[pa][msk^i]}) ;
   }
}
/**/
/**
trie used to solve
https://codeforces.com/contest/842/problem/D ;
https://codeforces.com/problemset/problem/282/E ;

struct node{
   node *ch[2] ;
   int cnt ;
```

```
    node() {
        ch[0]=ch[1]=NULL, cnt=0 ;
    }
}*root ;

void insert(int x){
    node *curr=root ;
    for(int i=20,bit; i>=0; --i){
        bit=(x>>i)&1 ;
        if(curr->ch[bit]==NULL)
            curr->ch[bit]=new node() ;
        curr=curr->ch[bit], curr->cnt++ ;
    }
}

int mex(int x){
    node *curr=root ; int num=0 ;
    for(int i=20,bit; i>=0; --i){
        bit=(x>>i)&1 ;
        if(curr->ch[bit]!=NULL and
curr->ch[bit]->cnt==(1<<i))
            curr=curr->ch[bit^1] , num^=(1<<i) ;
        else
            curr=curr->ch[bit] ;
        if(curr==NULL)
            return num ;
    }
    return num ;
}
/**/
/** random generator used to  solve
https://codeforces.com/problemset/problem/1114/E
void findD() {
        vector<int> List; int RandomRange = n;
        while (queryRemaining > 0 && RandomRange >
0) {
                int demandedIndex = rng32() %
RandomRange;
                cout << "? " << id[demandedIndex] <<
endl; fflush(stdout);
                int z; cin >> z; List.push_back(z);
                RandomRange--; queryRemaining--;
                swap(id[demandedIndex],
id[RandomRange]);
        }
        sort(List.begin(), List.end());
        if (List.back() != Max) List.push_back(Max);
        for (int i=1; i<List.size(); i++) {
                d = __gcd(d, List[i] - List[i-1]);
        }
}
/**/
/** kmp used to solve
https://codeforces.com/problemset/problem/494/B /**/ /**
void precal(){
    int n=t.size(), i=1, j=0 ;
    while(i<n){
        if(t[i]==t[j])
            ++j, p[i]=j, ++i ;
        else if(j)
            j=p[j-1] ;
        else
            ++i ;
    }
}
void kmp(){
    precal() ;
    int ns=(int)s.size(), nt=(int)t.size() ;
    for(int i=0,j=0; i<ns; ++i){
        while(j and s[i]!=t[j])
            j=p[j-1] ;
        if(s[i]==t[j])
            ++j ;
        if(j==nt)
            a[i-nt+1]=1, j=p[j-1] ;
    }
}
/***/
/*****//// Hash function used to solve
https://codeforces.com/problemset/problem/633/C

const int MOD=1e9+7,Base=998244353 ;

void Calc_Power(){
    p[0]=1 ;
    for(int i=1;i<N;++i){
        p[i]=p[i-1]*Base%MOD ;
    }
}
int Get_Hash(string s,int rev=1){
    int h=0 ;
    if(rev)
    for(int i=s.size()-1,j=0;i>=0;--i,++j){
        s[i]=tolower(s[i]) ;
        h+=p[j]*(int)(s[i]-'a'+1)%MOD ;
    }
    return h%MOD ;
}
```

```
/**/
/** longest increasing subsequence and decreasing subsequence
used to solve https://codeforces.com/problemset/problem/582/B

int lis(int n){ /// actually non decreasing ...
vector<int>v(n+2,INF) ; v[0]=-INF ;
for (int i = 1; i <= n; ++i) {
    int j = upper_bound(v.begin(), v.end(), a[i]) - v.begin();
      pdp[i]=j ; v[j]=a[i] ;
   }
}

int lds(int n){ /// actually non increasing ...
vector<int>v(n+2,-INF) ; v[n+1]=INF ;
for(int i=n;i>=1;--i){
    int j=upper_bound(ALL(v),a[i]-1)-v.begin() ;
       --j ; v[j]=a[i] ;
       j=v.end()-upper_bound(ALL(v),a[i]-1)-1 ;
       sdp[i]=j ;
    }
}
/**/
/**  segtree used to solve https://codeforces.com/problemset/problem/1482/E ...  **/
/**
void build(int nd,int be,int en) ;
void update(int nd,int be,int en,int l,int r,int v) ;
int query(int nd,int be,int en,int l,int r) ;
void lazy(int nd,int be,int en){
    if(prop[nd]==0)return ; tree[nd]+=prop[nd] ;
    if((nd<<1)<4*N)prop[nd<<1]+=prop[nd] ;
if((nd<<1|1)<4*N)prop[nd<<1|1]+=prop[nd] ; prop[nd]=0 ;
}
void build(int nd,int be,int en){
    prop[nd]=0 ;
    if(be==en){tree[nd]=a[be];return;}
    int md=(be+en)/2 ; build(nd<<1,be,md) ;
build(nd<<1|1,md+1,en) ;
    tree[nd]=tree[nd<<1]+tree[nd<<1|1] ;
}
void update(int nd,int be,int en,int l,int r,int v){
    lazy(nd,be,en) ;  if(be>r or en<l)return ;
    if(be>=l and en<=r){
        prop[nd]=v ; lazy(nd,be,en) ; return ;
    }
    int md=(be+en)>>1 ; update(nd<<1,be,md,l,r,v) ;
update(nd<<1|1,md+1,en,l,r,v) ;
    tree[nd]=min(tree[nd<<1],tree[nd<<1|1]) ;
}
int query(int nd,int be,int en,int l,int r){
    lazy(nd,be,en) ;  if(be>r or en<l)return INF ; if(be>=l and en<=r)return tree[nd] ;
    int md=(be+en)>>1 , p ,q ;
    p=query(nd<<1,be,md,l,r) ;
q=query(nd<<1|1,md+1,en,l,r) ;
    tree[nd]=min(tree[nd<<1],tree[nd<<1|1]) ; return min(p,q) ;
}
/***/
/******/ ///bit
void Update(int idx,int v){
    while(idx<=n)
        f[d1][d2][d3][idx]+=v  , idx+=idx&-idx ;
}
int Query(int idx){
    int sum=0 ;
    while(idx>0)
        sum+=f[d1][d2][d3][idx] , idx-=idx&-idx ;
    return sum ;
}
int Range(int l,int r){
    return Query(r)-Query(l-1) ;
}
/***
template of imAnik .

struct Matrix{
    int row, col , m[103][103];
    Matrix() {memset(m,0,sizeof(m));}
    void Set(int r,int c) {row = r; col = c;}
    Matrix(int r,int c) {memset(m,0,sizeof(m)); Set(r,c);}
};
Matrix Multiply(Matrix a, Matrix b){
    Matrix c(a.row,b.col) ;
    for(int i=1;i<=a.row;++i){
        for(int j=1;j<=b.col;++j){
            c.m[i][j]=0 ;
            for(int k=1;k<=a.col;++k){
                c.m[i][j]+=a.m[i][k]*b.m[k][j]%MOD ;
            }
            c.m[i][j]%=MOD ;
        }
    }
    return c ;
}
Matrix Power(Matrix a , int p ){
```

```
    if(p==1)
       return a ;
    Matrix x=Power(a , p/2) ;
    x=Multiply(x,x) ;
    if(p&1)
       x=Multiply(x,a) ;
    return x ;
}
/***/
/** **////DSU (checked well)
int Find_Parent(int node){
    if(p[node]==node)
       return node ;
    return p[node]=Find_Parent(p[node]) ;
}

void Union(int u,int v){
    p[Find_Parent(u)]=Find_Parent(v) ;
}
/****

90 degree rotation when origin given ...

used to solve
https://codeforces.com/problemset/problem/474/C

pii rot(pii pp,pii oo){
    pii np ;
    np.first=oo.first-(pp.second-oo.second) ;
    np.second=oo.second+pp.first-oo.first ;
    return np ;
}
int dist(pii a,pii b){
    return (a.first-b.first)*(a.first-b.first) +
(a.second-b.second)*(a.second-b.second) ;
}

square check

pii m2(pii a){
return {a.first*2,a.second*2} ;
}
// used to solve
https://codeforces.com/problemset/problem/474/C

bool square(pii a , pii b , pii c , pii d){

vector<pii>v ; pii md ; v.PB(a) ; v.PB(b) ; v.PB(c) ; v.PB(d)
;

for(int i=0;i<4;++i)v[i]=m2(v[i]) , md.second+=v[i].second ,
md.first+=v[i].first ;

if(md.second%4 or md.first%4)return false ; set<int>dset
; md.first/=4 ; md.second/=4 ;

for(int i=0;i<4;++i)dset.insert(dist(md,v[i])) ;
if(dset.size()>1)return false ; dset.clear() ;

for(int i=0;i<4;++i){
    for(int j=i+1;j<4;++j){
        dset.insert(dist(v[i],v[j])) ;
    }
}

if(dset.size()!=2 or *dset.begin()==0)return false ; return
true ;

}

/// used to solve
https://codeforces.com/problemset/problem/598/C

long double Angle(pdd a){
return atan2l(a.second,a.first)*180.0/pie ;
}

long double AngleBetween(pdd a,pdd b){
return
min(abs(360.0-abs(Angle(a)-Angle(b))),abs(Angle(a)-Ang
le(b))) ;
}

bool cmp(pdd a,pdd b){
return Angle(a)<Angle(b) ;
}
/**/

/**
closest pair ...

well checked ...

int Closest_Pair(int low,int high){
    if(high-low<=5){
        int Min=INF ;
        for(int i=low;i<=high;++i)
        for(int j=i+1;j<=high;++j)
```

```
        Min=min(Min,dist(i,j)) ;
        return Min ;
    }

    int mid=(high+low)>>1 , d ;


d=min(Closest_Pair(low,mid),Closest_Pair(mid+1,high)) ;

    vector<pair<int,int>>vec ;

    for(int i=low;i<=high;++i){
        if(sq(p[i].first-p[mid].first)<=d){
            vec.PB(p[i]) ;
        }
    }
    sort(ALL(vec),cmpy) ;

    for(int i=0;i<vec.size();++i){
        for(int j=i+1;j<=i+7 and j<vec.size();++j){
           d=min(Dist(vec[i],vec[j]),d) ;
        }
    }
    return d ;
}
/**/
/**
///(convex hull trick) Li chao tree used to solve
https://codeforces.com/problemset/problem/1083/E

int pnt[N] ;

struct Line
{
    int m, c ;
} tree[4*N] ;

bool exist[4*N] ;

int f(Line line,int x){
    return line.m*x+line.c ;
}

void AddLine(int node, int low, int high, Line line){
    exist[node]=true ;

    if(low==high){
        if(f(line,pnt[low])>f(tree[node],pnt[low]))
            tree[node]=line ;
        return ;
    }

    int mid=low+high>>1 ;

    bool l = f(line,pnt[low])>f(tree[node],pnt[low]) ;

    bool m = f(line,pnt[mid])>f(tree[node],pnt[mid]) ;

    if(m)
        swap(tree[node],line) ;

    if(l!=m)
        AddLine(node<<1,low,mid,line) ;
    else
        AddLine(node<<1|1,mid+1,high,line) ;
}

int Query(int node,int low,int high,int idx)
{
    if(low==high){
        return f(tree[node],pnt[idx]) ;
    }

    int mid=low+high>>1, Max=f(tree[node],pnt[idx]) ;

    if(idx<=mid and exist[node<<1])
        Max=max(Max,Query(node<<1,low,mid,idx)) ;
    else if(idx>mid and exist[node<<1|1])
        Max=max(Max,Query(node<<1|1,mid+1,high,idx)) ;

    return Max ;
}
/**/
/**
line equation a , b , c ; used to solve
https://codeforces.com/contest/1163/problem/C2
set<pair< pair<int,int> , int > >lines ;
struct line{
int a , b , c ;
}obj;
void store(int i,int j){
int dy=p[i].second-p[j].second , dx=p[i].first-p[j].first ,
g=__gcd(abs(dy),abs(dx)) , c=-p[i].second*dx+dy*p[i].first
, a=-dy , b=dx ;
if(a<0 or( a==0 and b<0))a*=-1 , b*=-1 , c*=-1 ; a/=g ,
b/=g , c/=g ;  lines.insert({{a,b},c}) ;
}
/***/
```

```
double Angle(int x,int y)
{
   if(x==0){
      if(y>0)
         return (double)90 ;
      else
         return (double)270 ;
   }

   if(y==0){
      if(x<0)
         return 180.0 ;
      else
         return 0.0 ;
   }

   double
ang=atan((double)abs(y)/(double)abs(x))*180.0/pie ;

   if(x>=0 and y>=0)
      return ang ;
   else if(x>=0 and y<0)
      return (double)360-ang ;
   else if(x<0 and y>=0)
      return (double)180-ang ;
   else if(x<0 and y<0)
      return (double)180+ang ;
}
/**
not well checked ...

vector<double>GetLine(pair<double,double>P ,
pair<double,double>Q){
   vector<double>v ; v.PB(0.0) , v.PB(0.0) , v.PB(0.0) ;
   if(P.first==Q.first){
      v[0]=1 , v[2]=-P.first ; return v ;
   }
   double d=(P.second-Q.second)/(P.first-Q.first) ;
   v[0]=-d , v[1]=1.0 , v[2]=d*P.first-Q.first ;
   return v ;
}
/**/
/***
this portion is checked well . I got ac using this portion  ...

double Radian(double x){
   return x*pie/180.0 ;
}

pdd Rotate(pdd p,double ang){
      ang=Radian(ang) ;
  /** ( x + yi )*( cos(ang) + isin(ang) )
      ( x*cos(ang) - y*sin(ang) ) + i*( x*sin(ang) +
y*cos(ang) ) /**/
     return { p.x*cos(ang) - p.y*sin(ang) , p.x*sin(ang) +
p.y*cos(ang) } ;
}

pdd LineLineIntersection(pdd A, pdd B, pdd C, pdd D){
   // Line AB represented as a1x + b1y = c1
   double a1 = B.second - A.second;
   double b1 = A.first - B.first;
   double c1 = a1*(A.first) + b1*(A.second);

   // Line CD represented as a2x + b2y = c2
   double a2 = D.second - C.second;
   double b2 = C.first - D.first;
   double c2 = a2*(C.first)+ b2*(C.second);

   double determinant = a1*b2 - a2*b1;

   if (determinant == 0){
      // The lines are parallel. This is simplified
      // by returning a pair of FLT_MAX
      return make_pair(INF, INF);
   }
   else{
      double x = (b2*c1 - b1*c2)/determinant;
      double y = (a1*c2 - a2*c1)/determinant;
      return make_pair(x, y);
   }
}

double Dist(pdd a, pdd b){
   double y=(a.second-b.second), x=(a.first-b.first) ;
   return sqrt(x*x+y*y) ;
}
/**/
/**
void centroid_decomposition(int nd,int pa){     /// O(n)
complexity ...
   if(pa!=-1)p[nd]=pa ; int wanted=-1 ,  siz=-1 ;

   for(auto ch:adj[nd]){                    /// size calculation ...
      if(ch==pa)continue ;
      centroid_decomposition(ch,nd) ; sz[nd]+=sz[ch] ;
      if(sz[ch]>siz)              /// take the heaviest child ...
         siz=sz[ch] , wanted=ch ;
```

```
    }
    sz[nd]++ ;

    if(siz*2<=sz[nd])
        centroid[nd]=nd ;
    else {
        wanted=centroid[wanted] ;
        while(sz[wanted]*2<sz[nd])
            wanted=p[wanted] ;
         centroid[nd]=wanted ;
    }
}
/**/
/**
dijkstra ...

int dijkstra(int s,int t){
    int dist[n+5] , node , d ;
    for(int i=0;i<=n;++i)
        dist[i]=INF ;
    dist[s]=0 , obj2.d=0 , obj2.node=s ;
    priority_queue<data>pq ; pq.push(obj2) ;
    while(pq.size()){
        obj2=pq.top() , pq.pop() ;
        node=obj2.node , d=obj2.d ;
        for(int i=0,child;i<adj[node].size();++i){
            child=adj[node][i] ;
            if(dist[child]>dist[node]+cost[node][i]){
                dist[child]=dist[node]+cost[node][i] ;
                obj2.d=dist[child] , obj2.node=child ;
                pq.push(obj2) ;
            }
        }
    }
    return dist[t] ;
}
/**/
/**
/// least common ancestor used to solve
https://codeforces.com/problemset/problem/832/D

void dfs(int nd,int pa=-1,int c=0){

d[nd]=c ;

if(pa!=-1){anc[nd][0]=pa ;
for(int j=1,md;j<20;++j)
   md=anc[nd][j-1] , anc[nd][j]=anc[md][j-1] ;
}

for(int j=0,ch;j<adj[nd].size();++j){
    ch=adj[nd][j] ;
    if(ch==pa)continue ;
    dfs(ch,nd,c+1) ;
}
}

int lca(int u,int v){

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

for(int j=19;j>=0 and d[u]>d[v];--j){   /// u niche ...
    if(d[anc[u][j]]>=d[v])
        u=anc[u][j] ;
}
for(int j=19;j>=0 and u!=v;--j){
    if(anc[u][j]!=anc[v][j])
        u=anc[u][j] , v=anc[v][j] ;
    u=anc[u][0] , v=anc[v][0] ;
}
return u ;
}

int dist(int u,int v){
return abs(d[u]-d[v]) ;
}

// well checked
int Inclusion_Exclusion(vector<int>Primes){
    int v, Lim=Primes.size(), ans=0 ;
    for(int x=1; x<(1<<Lim); ++x){
        v=1 ; int Bits=0 ;
        for(int j=0; j<Lim; ++j)
            if(x&(1<<j))
                ++Bits, v*=Primes[j] ;
        if(Bits&1)
            ans+=Occ[v] ;
        else
            ans-=Occ[v] ;
    }
    return ans ;
}

/// WELL CHECKED ...
int extended_euclid(LL a,LL b,LL &x,LL &y){
    if(b==0)
    {
        x=1;
```

```
        y=0;
        return a ;
    }
    /**
    gcd = x*b + y*(a%b) ;
    gcd = x*b + y*(a-(a/b)*b) ;
    gcd = x*b + y*a - y*(a/b)*b ;
    gcd = y*a + (x-(y*(a/b)))*b ;
    /**/
    LL temp , g ;
    g=extended_euclid(b,a%b,x,y);
    temp=x-y*(a/b) , x=y , y=temp ;
    return g ;
}

LL inverse_mod(LL C){
    LL x,y;
    extended_euclid(MOD,C,x,y);
    return ((y%MOD)+MOD)%MOD;
}

int mod_expo(int a,int b){
    if(b==0)
        return 1 ;
    int x=mod_expo(a,b>>1) ;
    x=x*x%MOD ; if(b&1)x=x*a%MOD ; return x ;
}

LL ncr(LL n,LL r){
    if(r>n)
        return 0 ;
    r=Fact[r]*Fact[n-r]%MOD;
    return Fact[n]*inverse_mod(r)%MOD;
    /// ncr calculate using triangle formula nCr = (n-1)Cr +
(n-1)C(r-1) ;
}

int count(int l, int r,int j) {
    return upper_bound(b+j+1, b+n+1, r) -
lower_bound(b+j+1, b+n+1, l);
}
```