University Of Asia Pacific Notebook

1. CP Setup

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

#define ll long long
#define vi vector<int>
#define pi pair<int, int>
#define f first
#define sc second
#define all(a) a.begin(), a.end()

int main() {
ios_base::sync_with_stdio(0);
cin.tie(0);
//code here
}
```

2.Big O notation and Complexity Analysis

n	Possible complexities
$n \leq 10$	$\mathcal{O}(n!)$, $\mathcal{O}(n^7)$, $\mathcal{O}(n^6)$
$n \leq 20$	$\mathcal{O}(2^n \cdot n)$, $\mathcal{O}(n^5)$
$n \leq 80$	$\mathcal{O}(n^4)$
$n \leq 400$	$\mathcal{O}(n^3)$
$n \leq 7500$	$\mathcal{O}(n^2)$
$n \leq 7 \cdot 10^4$	$\mathcal{O}(n\sqrt{n})$
$n \leq 5 \cdot 10^5$	$\mathcal{O}(n \log n)$
$n \leq 5 \cdot 10^6$	$\mathcal{O}(n)$
$n \leq 10^{18}$	$\mathcal{O}(\log^2 n)$, $\mathcal{O}(\log n)$, $\mathcal{O}(1)$

3.Standard Template Library and other C++ stuffs

```
auto [a, b] = p;
pair<int, int> p;
pair<int, pair<int, int>> p3;
deque<int> dq; dq.push_front(1); dq.push_back(3); pop_front(); pop_back();
stack<int> st; st.push(2); st.pop(); st.top();st.size();
queue<int> q; q.push(5); q.pop(); q.front();q.back();size();empty();
set < int > s; insert(); erase(); begin(); end(); size(); count(); empty();
priority_queue<int> pq; pq.push(1); top(); pop(); size();
multiset < int > m; insert(); erase(); begin(); end(); size(); count(); empty();
map<int, int> mp1;
map<int, pair<int, int>> mp2; mp2[0].first; mp2[0].second;
int index = lower_bound(v.begin(), v.end(), val) - v.begin()
int index = upper_bound(v.begin(), v.end(), val) - v.begin();
auto it = s.lower_bound(6); *it; it--;
auto it = s.upper_bound(6); *it; it--;
```

- Policy-based data structures

```
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;

template <typename T> using o_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  o_set<int> se;
  se.insert(4);
  se.insert(2);
  se.insert(5);
  // sorted set se = [2, 4, 5]
  cout << se.order_of_key(5) << '\n'; // number of elements < 5
  cout << se.order_of_key(6) << '\n'; // number of elements < 6
  cout << (*se.find_by_order(1)) << '\n'; // if you imagine this as a 0-indexed vector, what is se[1]?
  return 0;
}
```

4. Brute Forces

- Generating Permutaions

•permutaion {0, 1, 2, 3, 4} এর next_permutaion {0, 1, 2, 4, 3}
• permutaion {0, 1, 2, 4, 3} এর next_permutaion {0, 1, 3, 2, 4}
• permutaion {0, 2, 1, 4, 3} এর next_permutaion {0, 2, 3, 1, 4}
• permutaion {4, 3, 2, 1, 0} এর next_permutaion নাই।

```
do{
for(int x: v) cout << x << " "; cout << "\n";
// do something else with v
}while(next_permutation(v.begin(), v.end()));
```

- Generate All possible Subset With BitMasking

```
int n = 4;
for(int i = 0; i < (1<<n); i++){
vector<int> ss;
for(int j = 0; j < n; j++) if((i&(1<<j)) != 0) ss.push_back(j);
for(int x: ss) cout << x << " "; cout << "\n";
}
```

- Generating All possible Subsets

```
void bf(int k){
  if(k == n) {
    for(int x: ss) cout << x << " ";
    cout << "\n";
    return;
  }
    ss.pb(k);
    bf(k+1);
    ss.pop_back();
    bf(k+1);
}
int main() {
n = 3;
bf(0);
/*output:
0 1 2
0 1
0 2
0
1 2
1
2
//this one is empty subset
*/
}
```

5. Bit masking

- Decimal to other base

```
string convert_base(int n, int base) {
  if(n == 0) return "0";
  int power = 1;
  while(power * base <= n) power *= base;
  string result;
  while(n > 0) {
    int k = n / power;
    result += k + '0';
    n -= power * k;
    power /= base;
  }
  return result;
}
```

- Other base to decimal

```
int convert_to_decimal(string s , int base) {
  int n = 0 , power = 1;
  for(int i = (int) s.size() - 1; i >= 0 ; i--) {
    n += power * (s[i] - '0');
    power *=  base;
  }
  return n;
}
```

- Bitwise operations

```
int and_value = (a & b);
int or_value = (a | b);
int xor_value = (a ^ b);
int right_shift = (a >> b);
int left_shift = (a << b);
int bitwise_not = (~a);
int ones = __builtin_popcount(a); // __buitlin_popcountll(a) in case of long long
int lz = __builtin_clz(a); // __builtin_clzll(a) in case of long long
int tz = __builtin_ctz(a); // __builtin_ctzll(a) in case of long lon
```

6. Binary search

```
long long l = 0 ,r = 1e18, ans = 1e18;
while(l <= r) {
long long m = (l + r) / 2;
  if(cal(m) < x){
    l = m + 1;
  }
else{
    ans = m;
    r = m - 1;
  }
}
cout << ans << '\n';
return;
```

7. Number Theory

Lemma / Observations.

1. Upper Bound of Number of Divisors 2sqrt(n) :(.
2. Euclid's Lemma: If a prime p divides the product a·b of two integers a and b, then p must divide at least one of those integers a and b.
3. The smallest number greater than 1 that divides *n* is also the smallest prime factor of *n!*
4. *A number which has exactly 3 divisors is always a square of a prime!*
5. *A number which has exactly 4 divisors is either of the form p3 or p∗q where p and q are prime.*
6. *A prime gap is the difference between two successive prime numbers. The gaps are too small in real life!*
7. *Two integers a and b are coprime, relatively prime or mutually prime if the only positive integer that is a divisor of both of them is 1.*
8. *Every two consecutive numbers are coprime!*
9. *The common divisors of a and b are exactly the divisors of gcd(a,b).*

- **Optimized Way to find divisor to O(√n**

```
#include<bits/stdc++.h>
using namespace std;
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n; cin >> n;
  vector<int> divs;
  for (int i = 1; i * i <= n; i++) {
    if (n % i == 0) {
      divs.push_back(i);
      if (i != n / i) divs.push_back(n / i);
    }
  }
  sort(divs.begin(), divs.end());
  for (auto x: divs) cout << x << ' ';
  return 0;
}
```

- **Checking if a number is prime or not in *O* (sqrt *n*):**

```
#include<bits/stdc++.h>
using namespace std;
bool is_prime(int n) {
  for (int i = 2; i * i <= n; i++) {
```

```
    if (n % i == 0) {
      return false;
    }
  }
  return true;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cout << is_prime(7) << '\n';
  return 0;
}
```

- **How to find all the prime factors of *n* in *O* (sqrt *n*):**

```
#include<bits/stdc++.h>
using namespace std;
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n; cin >> n;
  vector<int> v;
  for (int i = 2; i * i <= n; i++) {
    if (n % i == 0) {
      while (n % i == 0) {
        v.push_back(i);
        n /= i;
      }
    }
  }
  if (n > 1) v.push_back(n);
  for (auto x: v) cout << x << ' ';
  return 0;
}
```

- N**umber of divisors using the prime factorization of *n.***

```
long long numberOfDivisors(long long num) {
    long long total = 1;
    for (int i = 2; (long long)i * i <= num; i++) {
        if (num % i == 0) {
            int e = 0;
            do {
                e++;
                num /= i;
            } while (num % i == 0);
            total *= e + 1;
        }
    }
    if (num > 1) {
        total *= 2;
    }
    return total;
}
```

- **Sum of divisors using the prime factorization of *n.***

```
long long SumOfDivisors(long long num) {
    long long total = 1;
    for (int i = 2; (long long)i * i <= num; i++) {
        if (num % i == 0) {
            int e = 0;
            do {
                e++;
                num /= i;
            } while (num % i == 0);

            long long sum = 0, pow = 1;
            do {
                sum += pow;
                pow *= i;
            } while (e-- > 0);
            total *= sum;
        }
    }
    if (num > 1) {
        total *= (1 + num);
    }
    return total;
}
```

- **Find all primes which are less than *n* in *O* (*n*loglog(*n*)). Seive**

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e7 + 9;
bool f[N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n = N - 9;
```

```
  vector<int> primes;
  f[1] = true;
  for (int i = 2; i <= n; i++) {
    if (!f[i]) {
      primes.push_back(i);
      for (int j = i + i; j <= n; j += i) {
        f[j] = true;
      }
    }
  }
  cout << primes.size() << '\n';
  return 0;
}
```

- **Code (Fast Sieve, Using bit set, Works till 10^8 in less than 1s, Memory Complexity: *O* (*n*/64))**

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e8 + 9;
bitset<N> f;
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n = N - 9;
  vector<int> primes;
  f[1] = true;
  for (int i = 2; i * i <= n; i++) {
    if (!f[i]) {
      for (int j = i * i; j <= n; j += i) {
        f[j] = true;
      }
    }
  }
  for (int i = 2; i <= n; i++) {
    if (!f[i]) {
      primes.push_back(i);
    }
  }
  cout << primes.size() << '\n';
  return 0;
}
```

- **Find the number of divisors for all integers from1 to n in *O*(*n*loglog(*n*)).**

```
#include<bits/stdc++.h>
using namespace std;
int d[104];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n = 100;
  for (int i = 1; i <= n; i++) {
    for (int j = i; j <= n; j += i) {
      d[j]++;
      // d[j] += i // for sum of divisors
    }
  }
  for (int i = 1; i <= n; i++) {
    cout << d[i] << ' ';
  }
  return 0;
}
```

- **Prime Factorization using Sieve**

You are given q =10^6 queries. In each query, you need to find out the prime factorization of n where n ≤106. How to do this in 1s?

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e6 + 9;
int spf[N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  for (int i = 2; i < N; i++) {
    spf[i] = i;
  }
  for (int i = 2; i < N; i++) {
    for (int j = i; j < N; j += i) {
      spf[j] = min(spf[j], i);
    }
  }
  int q; cin >> q; // queries q <= 1e6
  while (q--) {
    int n; cin >> n; // find prime factorization of n <= 1e6
    vector<int> ans;
    while (n > 1) {
      ans.push_back(spf[n]);
      n /= spf[n];
    }
    for (auto x: ans) cout << x << ' '; cout << '\n';
  }
```

```
  return 0;
}
```

- **GCD and LCM**

```
int gcd(int a, int b){
    if(a == 0) return b;
    return gcd(b%a, a);
}
long long lcm(long long a, long long b) {
   return (a / __gcd(a, b)) * b;
}
```

- **GCD of two numbers when one of them can be very large**

```
#include<bits/stdc++.h>
using namespace std;
typedef long long int ll;

ll gcd(ll a, ll b) {
  if (!a)
    return b;
  return gcd(b % a, a);
}
ll reduceB(ll a, char b[]) {
  ll mod = 0;
  for (int i = 0; i < strlen(b); i++)
    mod = (mod * 10 + b[i] - '0') % a;

  return mod; // return modulo
}
ll gcdLarge(ll a, char b[]) {
  ll num = reduceB(a, b);
  return gcd(a, num);
}
int main(){
  ll a = 1221;
  char b[] = "1234567891011121314151617181920212223242526272829";
  if (a == 0)
    cout << b << endl;
  else
    cout << gcdLarge(a, b) << endl;
  return 0;
}
```

- Here is an implementation using factorization in O (sqrt n)

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

- **Euler Totient Function 1 to n in $O(n\log\log(n))$.**

```
void phi_1_to_n(int n) {
    vector<int> phi(n + 1);
    for (int i = 0; i <= n; i++)
        phi[i] = i;
    for (int i = 2; i <= n; i++) {
        if (phi[i] == i) {
            for (int j = i; j <= n; j += i)
                phi[j] -= phi[j] / i;
        }
    }
}
```

- Including the totient from 1 to n using the divisor sum property

```
void phi_1_to_n(int n) {
    vector<int> phi(n + 1);
    phi[0] = 0;
    phi[1] = 1;
    for (int i = 2; i <= n; i++)
        phi[i] = i - 1;

    for (int i = 2; i <= n; i++)
        for (int j = 2 * i; j <= n; j += i)
              phi[j] -= phi[i];
}
```

- **Legendres formula**

n and a prime number *p*, find the largest *x* such that *px* divides *n*! (factorial) in *O*(log*n*).

```
#include<bits/stdc++.h>
using namespace std;
int legendre(long long n, long long p) {
  int ans = 0;
  while (n) {
    ans += n / p;
    n /= p;
  }
  return ans;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);

  return 0;
}
```

- **Count trailing zeroes in factorial of an integer *n* in *O*(log*n*).**

```
#include <iostream>
using namespace std;
int findTrailingZeros(int n){
  if (n < 0) // Negative Number Edge Case
    return -1;
  int count = 0;
  for (int i = 5; n / i >= 1; i *= 5)
    count += n / i;
  return count;
}
int main(){
  int n = 100;
  cout << findTrailingZeros(n);
  return 0;
}
```

- **Counting Digits of (n!) number!**

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

int findDigits(int n){
  if (n < 0)
    return 0;
  if (n <= 1)
    return 1;
  double digits = 0;
  for (int i = 2; i <= n; i++)
    digits += log10(i);

  return floor(digits) + 1;
}
int main() {
  cout << findDigits(120) << endl;
  return 0;
}
```

- Modular Arithmetic

```
int vagsesh = (a % m - b % m + m) % m; // For Substraction
long long res = 1; // For Multiplicaion
for(int i = 1; i <= n; i++) {
res = (res * a) % m;
}
cout << res << endl;
```

- Big Mod

```
long long binpow(long long a, long long b, long long m) {
    a %= m;
    long long res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
}
```

- Binary Multiplication with Mod

```
long long binmul(long long a, long long b, long long m) {
    long long res = 0LL;
    a = a % m;
    while (b > 0) {
        if (b & 1) res = (res + a) % m;
        a = (a + a) % m;
        b >>= 1;
    }
    return res;
}
```

8. MATH

Divisibility by 2 or 5

For 2, The number should be even! Well, that's not something exciting :(. For 5 the number should end with 0 or 5.

Divisibility by 3 or 9

The sum of digits should be a multiple of 3 or a multiple of 9.

Divisibility by 4

The basic rule for divisibility by 4 is that if the number formed by the last two digits in a number is divisible by 4, the original number is divisible by 4.

Divisibility by 6

Think using primes! Hint: 6=2·3.

Divisibility by 11

Add and subtract digits in an alternating pattern (add a digit, subtract next digit, add next digit, etc.). Then check if that answer is divisible by 11.

- **Divisibility and Large Numbers**

```
#include<bits/stdc++.h>
using namespace std;
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  string a; int b; cin >> a >> b;
  int ans = 0;
  for (int i = 0; i < a.size(); i++) {
    ans = (ans * 10LL % b + (a[i] - '0')) % b;
  }
  if (ans == 0) {
    cout << "a is divisible by b\n";
  }
  else {
    cout << "sad\n";
  }
  return 0;
}
```

9. Basic Data Structures

- Segment Tree

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

const int N = 1e5 + 9;
int a[N];
long long t[N * 4];
void build(int n, int b, int e) {
  if (b == e) {
    t[n] = a[b];
    return;
  }
  int mid = (b + e) / 2, l = 2 * n, r = 2 * n + 1;
  build(l, b, mid);
  build(r, mid + 1, e);
  t[n] = t[l] + t[r];
}

void upd(int n, int b, int e, int i, int v) {
  if (i < b or e < i) return;
```

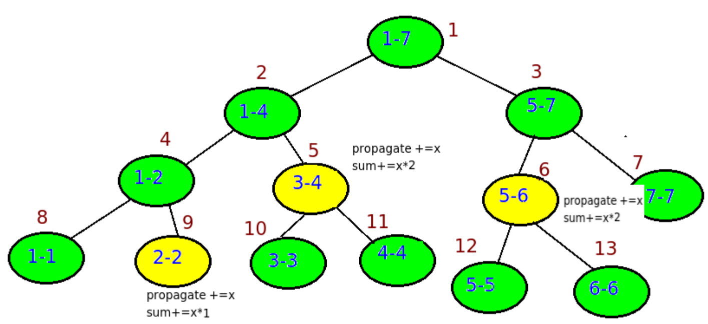
```
  if (b == e) {
    t[n] = v;
    return;
  }
  int mid = (b + e) / 2, l = 2 * n, r = 2 * n + 1;
  upd(l, b, mid, i, v);
  upd(r, mid + 1, e, i, v);
  t[n] = t[l] + t[r];
}

long long query(int n, int b, int e, int i, int j) {
  if (e < i or j < b) return 0;
  if (b >= i and e <= j) {
    return t[n];
  }
  int mid = (b + e) / 2, l = 2 * n, r = 2 * n + 1;
  return query(l, b, mid, i, j) + query(r, mid + 1, e, i, j);
}

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, q; cin >> n >> q;
  for (int i = 1; i <= n; i++) {
    cin >> a[i];
  }
  build(1, 1, n);
  while (q--) {
    int ty; cin >> ty;
    if (ty == 1) {
      int i, v; cin >> i >> v;
      ++i;

      upd(1, 1, n, i, v);
    }
    else {
      int l, r; cin >> l >> r;
      --r; ++l; ++r;
      long long ans = query(1, 1, n, l, r);
      cout << ans << '\n';
    }
  }
  return 0;
}
```

- **Segment Tree Lazy**



Can you do all these

- single update, range sum query
- single update, range max/min/gcd/lcm/OR/AND/XOR query
- range add update, single query
- range add update, range sum query
- range add update, range max/min query
- range assignement update, range sum/max/min query

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

const int N = 1e5 + 9;
int a[N];
long long t[4 * N], lazy[4 * N];

void push(int n, int b, int e) {
  if (lazy[n] == -1) {
    return;
  }
  // if we assign lazy[n] to the elements in this segment, what will happen to our t[n]?
  // remember that t[n] = the sum of elements in this segment
  t[n] = 1LL * (e - b + 1) * lazy[n];

  // push to the childs
  if (b != e) {
    int mid = (b + e) / 2, l = 2 * n, r = 2 * n + 1;
```

```
    lazy[l] = lazy[n];
    lazy[r] = lazy[n];
  }

  lazy[n] = -1;
}
void build(int n, int b, int e) {
  lazy[n] = -1;
  if (b == e) {
    t[n] = a[b];
    return;
  }
  int mid = (b + e) / 2, l = 2 * n, r = 2 * n + 1;
  build(l, b, mid);
  build(r, mid + 1, e);
  t[n] = t[l] + t[r];
}

void upd(int n, int b, int e, int i, int j, int v) {
  push(n, b, e);
  if (e < i or j < b) return;
  if (b >= i and e <= j) {
    // assign v to every element in this segment
    lazy[n] = v;
    push(n, b, e);
    return;
  }
  int mid = (b + e) / 2, l = 2 * n, r = 2 * n + 1;
  upd(l, b, mid, i, j, v);
  upd(r, mid + 1, e, i, j, v);
  t[n] = t[l] + t[r];
}

long long query(int n, int b, int e, int i, int j) {
  push(n, b, e);
  if (e < i or j < b) return 0;
  if (b >= i and e <= j) {
    return t[n];
  }
  int mid = (b + e) / 2, l = 2 * n, r = 2 * n + 1;
  return query(l, b, mid, i, j) + query(r, mid + 1, e, i, j);
}

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, q; cin >> n >> q;

  build(1, 1, n);
  while (q--) {
    int ty; cin >> ty;
    if (ty == 1) {
      int l, r, v; cin >> l >> r >> v;
      --r;
      ++l; ++r;
      upd(1, 1, n, l, r, v);
    }
    else {
      int i; cin >> i;
      ++i;
      cout << query(1, 1, n, i, i) << '\n';
    }
  }
  return 0;
}
```

- Bitwise Or and And With Range update Query

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


const int N = 1e5 + 9, B = 30;

int t[4 * N];
int lazy[4 * N];

void push(int n, int b, int e) {
  if (lazy[n] == 0) {
    return;
  }
  t[n] = t[n] | lazy[n];

  // push to the childs
  if (b != e) {
    int mid = (b + e) / 2, l = 2 * n, r = 2 * n + 1;
    lazy[l] |= lazy[n];
    lazy[r] |= lazy[n];
  }

  lazy[n] = 0;
}

int merge(int l, int r) {
  int ans = l & r;
  return ans;
}
void build(int n, int b, int e) {
  lazy[n] = 0;
  if (b == e) {
```

```
    t[n] = 0;
    return;
  }
  int mid = (b + e) / 2, l = 2 * n, r = 2 * n + 1;
  build(l, b, mid);
  build(r, mid + 1, e);
  t[n] = merge(t[l], t[r]);
}

void upd(int n, int b, int e, int i, int j, int v) {
  push(n, b, e);
  if (e < i or j < b) return;
  if (b >= i and e <= j) {
    lazy[n] = v;
    push(n, b, e);
    return;
  }
  int mid = (b + e) / 2, l = 2 * n, r = 2 * n + 1;
  upd(l, b, mid, i, j, v);
  upd(r, mid + 1, e, i, j, v);
  t[n] = merge(t[l], t[r]);
}

int query(int n, int b, int e, int i, int j) {
  push(n, b, e);
  if (e < i or j < b) {

    return (1 << B) - 1;
  }
  if (b >= i and e <= j) {
    return t[n];
  }
  int mid = (b + e) / 2, l = 2 * n, r = 2 * n + 1;
  return merge(query(l, b, mid, i, j), query(r, mid + 1, e, i, j));
}

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, q; cin >> n >> q;

  build(1, 1, n);
  while (q--) {
    int ty; cin >> ty;
    if (ty == 1) {
      int l, r, v; cin >> l >> r >> v;
      --r;

      ++l; ++r;
      upd(1, 1, n, l, r, v);
    }
    else {
      int l, r; cin >> l >> r;
      --r;
      ++l; ++r;
      int val = query(1, 1, n, l, r);
      cout << val << '\n';
    }
  }
  return 0;
}
```

10. Graph

- **Adjacency List**

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

const int N = 105;
vector<int> g[N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  while (m--) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
    g[v].push_back(u);
  }
  return 0;
}
```

- **DFS**

```
const int N = 1e5 + 9;
vector<int> g[N];
bool vis[N];

void dfs(int u) {
  vis[u] = true;
  for (auto v: g[u]) {
    if (!vis[v]) {
```

```
      dfs(v);
    }
  }
}
```

- **BFS**

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

const int N = 1e5 + 9;
vector<int> g[N];
bool vis[N]; int dis[N], par[N];

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  while (m--) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
    g[v].push_back(u);
  }
  queue<int> q;
  q.push(1); vis[1] = true; dis[1] = 0;
  while (!q.empty()) {
    int u = q.front();
    q.pop();
    for (auto v: g[u]) {
      if (!vis[v]) {
        q.push(v);
        par[v] = u;
        dis[v] = dis[u] + 1;
        vis[v] = true;
      }
    }
  }
  for (int i = 1; i <= n; i++) {
    cout << dis[i] << ' ';
  }
  cout << '\n';
  int v = 4;
  while (v != 1) {
    cout << v << ' ';
    v = par[v];
  }
  cout << 1 << '\n';
}
```

- **Bicoloring and Bipartite Graphs**

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

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

const int N = 1e5 + 9;
vector<int> g[N];
bool vis[N]; int col[N];
bool ok;

void dfs(int u) {
  vis[u] = true;
  for (auto v: g[u]) {
    if (!vis[v]) {
      col[v] = col[u] ^ 1;
      dfs(v);
    }
    else {
      if (col[u] == col[v]) {
        ok = false;
      }
    }
  }
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  while (m--) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
    g[v].push_back(u);
  }
  ok = true;
  for (int i = 1; i <= n; i++) {
    if (!vis[i]) dfs(i);
  }
  if (ok) {
    cout << "YES\n";
  }
  else {
    cout << "NO\n";
  }
}
```

- **Finding A Cycle**

How to check if an undirected graph has a cycle or not?

How to check if a directed graph has a cycle or not?

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

const int N = 1e5 + 9;
vector<int> g[N];
int col[N], par[N];
bool cycle;
void dfs(int u) {
  col[u] = 1;
  for (auto v: g[u]) {
    if (col[v] == 0) {
      par[v] = u;
      dfs(v);
    }
    else if (col[v] == 1) {
      cycle = true;
      // you can track the cycle using par array
    }
  }
  col[u] = 2;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  for (int i = 1; i <= m; i++) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
  }
  cycle = false;
  for (int i = 1; i <= n; i++) {
    if (col[i] == 0) dfs(i);
  }
  cout << (cycle ? "YES\n" : "NO\n") << '\n';
  return 0;
}
```

- **Topological Sorting**

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

const int N = 1e5 + 9;
int indeg[N];
vector<int> g[N];
bool vis[N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  while (m--) {
    int u, v; cin >> u >> v;
    indeg[v]++;
    g[u].push_back(v);
  }
  vector<int> z;
  for (int i = 1; i <= n; i++) {
    if (indeg[i] == 0) {
      z.push_back(i);
      vis[i] = true;
    }
  }
  vector<int> ans;
  while (ans.size() < n) {
    if (z.empty()) {
      cout << "IMPOSSIBLE\n";
      return 0;
    }
    int cur = z.back();
    z.pop_back();
    ans.push_back(cur);
    for (auto v: g[cur]) {
      indeg[v]--;
      if (!vis[v] and indeg[v] == 0) {
        z.push_back(v);
        vis[v] = true;
      }
    }
  }
  for (auto x: ans) cout << x << ' ';
  return 0;
}
```

University Of Asia Pacific Notebook

- **Diameter of a Tree**

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

const int N = 1e5 + 9;
vector<int> g[N];
int dep[N];
int mx, node, ans;
void dfs1(int u, int p) {
  dep[u] = dep[p] + 1;
  if (dep[u] > mx) {
    mx = dep[u];
    node = u;
  }
  for (auto v: g[u]) {
    if (v != p) {
      dfs1(v, u);
    }
  }
}
void dfs2(int u, int p) {
  dep[u] = dep[p] + 1;
  ans = max(ans, dep[u] - 1);
  for (auto v: g[u]) {
    if (v != p) {
      dfs2(v, u);
    }
  }
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n; cin >> n;
  for (int i = 1; i < n; i++) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
    g[v].push_back(u);
  }
  dfs1(1, 0);
  dfs2(node, 0);
  cout << ans << '\n';
  return 0;
}
```

11. DP

- Recursive code for Fibonacci:

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

const int N = 55;
int f[N];
bool is_computed[N];
int fibo(int i) {
  if (i == 0) return 0;
  if (i == 1) return 1;
  if (is_computed[i]) return f[i];
  f[i] = fibo(i - 1) + fibo(i - 2);
  is_computed[i] = true;
  return f[i];
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cout << fibo(50) << '\n';
  return 0;
}
```

- **Minimum Steps to Reach 1**

[Problem 1]Minimum Steps to Reach 1

You are given an integer $n (1 \le n \le 10^5)$. You can perform the following operations on it(as many as times as you want).

1. Subtract 1 from it. (assign $n := n - 1$)
2. If its divisible by 2, divide by 2. (if $n \mod 2 == 0$, then assign $n := n/2$)
3. If its divisible by 3, divide by 3. (if $n \mod 3 == 0$, then assign $n := n/3$).

Find the minimum number of operations to make $n = 1$.

Example: For $n = 7$, output: $3 (7 - 1 = 6/3 = 2/2 = 1)$

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

const int N = 1e5 + 9;
int step[N];
int min_steps(int i) {
  if (i == 1) return 0;
  if (step[i] != -1) return step[i];
  int ans = min_steps(i - 1) + 1;
  if (i % 2 == 0) {
    ans = min(ans, min_steps(i / 2) + 1);
  }
  if (i % 3 == 0) {
    ans = min(ans, min_steps(i / 3) + 1);
  }
  step[i] = ans;
  return step[i];
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  memset(step, -1, sizeof step);
  int n = 10;
  cout << min_steps(n) << '\n';
  return 0;
}
```

- **Number of ways**

[Problem 2]Number of ways

You are given an integer $n(1 \leq n \leq 10^5)$. Find the number of ways to write n as sums of 1 and 3. Output the answer modulo $10^9 + 7$.

Example: For $n = 4$, output: $3(1+1+1+1, 1+3, 3+1)$

Can you solve it in $O(n)$?

Recursive Code:

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

const int N = 1e5 + 9;
int step[N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n = 100000;
  step[1] = 0;
  for (int i = 2; i <= n; i++) {
    int ans = step[i - 1] + 1;
    if (i % 2 == 0) {
      ans = min(ans, step[i / 2] + 1);
    }
    if (i % 3 == 0) {
      ans = min(ans, step[i / 3] + 1);
    }
    step[i] = ans;
  }
  cout << step[n] << '\n';
  return 0;
}
```

- **Number of Ways 2**

[Problem 3]Number of Ways 2

You are given an integer $n(1 \leq n \leq 1000)$. Find the number of ways to write n as sums of positive integers. Output the answer modulo $10^9 + 7$.

Example: For $n = 3$, output: $4(1+1+1, 1+2, 2+1, 3)$

Can you solve it in $O(n^2)$?

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

const int N = 1e3 + 9, mod = 1e9 + 7;
int ways[N];
int count(int n) {
  if (n == 0) return 1;
  if (ways[n] != -1) return ways[n];
  int ans = 0;
  for (int i = 1; i <= n; i++) {
    ans += count(n - i);
    ans %= mod;
  }
  return ways[n] = ans;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
```

```
  cin.tie(0);
  memset(ways, -1, sizeof ways);
  cout << count(4) << '\n';
  return 0;
}
```

- **0/1 Knapsack**

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

const int N = 105;
#define int long long
int n, W, w[N], v[N], dp[N][100005];
int rec(int i, int weight) {
  if (i == n + 1) return 0;
  if (dp[i][weight] != -1) return dp[i][weight];
  int ans = rec(i + 1, weight);
  if (weight + w[i] <= W) ans = max(ans, rec(i + 1, weight + w[i]) + v[i]);
  return dp[i][weight] = ans;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cin >> n >> W;
  for (int i = 1; i <= n; i++) {
    cin >> w[i] >> v[i];
  }
  memset(dp, -1, sizeof dp);
  cout << rec(1, 0) << '\n';
  return 0;
}
```

- **DP on Matrix**

[Problem 5]DP on Matrix

Problem: Given a cost matrix Cost[][] of size $n \times m$ where Cost[i][j] denotes the Cost of visiting cell with coordinates (i,j), find a min-cost path to reach the cell (n,m) from the cell (1,1) under the condition that you can only travel one step right or one step down. (We assume that all costs are positive integers)

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

int n, m, a[10][10], inf = 1e9 + 7;
int dp[10][10];
int min_cost(int i, int j) {
  if (j > m or i > n) return inf;
  if (i == n and j == m) return a[i][j];
  if (dp[i][j] != -1) return dp[i][j];
  return dp[i][j] = a[i][j] + min(min_cost(i + 1, j), min_cost(i, j + 1));
}
void path(int i, int j) {
  cout << "(" << i << ", " << j << ") -> ";
  if (i == n and j == m) return;
  int right = min_cost(i, j + 1);
  int down = min_cost(i + 1, j);
  if (right <= down) {
    path(i, j + 1);
  }
  else {
    path(i + 1, j);
  }
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cin >> n >> m;
  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= m; j++) {
      cin >> a[i][j];
    }
  }
  memset(dp, -1, sizeof dp);
  cout << min_cost(1, 1) << '\n';
  path(1, 1);
  return 0;
}
```

- **Longest Common Subsequence(**

You are given strings s and t. Find one longest string that is a subsequence of both s and t.

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

```
const int N = 3030;
string a, b;
int dp[N][N];
int lcs(int i, int j) {
  if (i >= a.size() or j >= b.size()) return 0;
  if (dp[i][j] != -1) return dp[i][j];
  int ans = lcs(i + 1, j);
  ans = max(ans, lcs(i, j + 1));
  if (a[i] == b[j]) {
    ans = max(ans, lcs(i + 1, j + 1) + 1);
  }
  return dp[i][j] = ans;
}
void print(int i, int j) {
  if (i >= a.size() or j >= b.size()) return;
  if (a[i] == b[j]) {
    cout << a[i];
    print(i + 1, j + 1);
    return;
  }
  int x = lcs(i + 1, j);
  int y = lcs(i, j + 1);
  if (x >= y) {
    print(i + 1, j);
  }
  else {
    print(i, j + 1);
  }
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cin >> a >> b;
  memset(dp, -1, sizeof dp);
  // cout << lcs(0, 0) << '\n';
  print(0, 0);
  return 0;
}
```

- We are given an array with n numbers [0,1….n - 1] , the task is to find the longest, strictly increasing subsequence in a.

$$i_1 < i_2 < \cdots < i_k, \quad a[i_1] < a[i_2] < \cdots < a[i_k]$$

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

const int N = 10010;
int a[N], dp[N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n; cin >> n;
  for (int i = 1; i <= n; i++) {
    cin >> a[i];
  }
  for (int i = 1; i <= n; i++) {
    dp[i] = 1;
    for (int j = 1; j < i; j++) {
      if (a[j] < a[i]) {
        dp[i] = max(dp[i], dp[j] + 1);
      }
    }
  }
  int ans = 0;
  for (int i = 1; i <= n; i++) {
    ans = max(ans, dp[i]);
  }
  cout << ans << '\n';
  return 0;
}
```

12. Basic Strings

- **Find if two strings are equal or not using Hashing**

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

const int p = 137, mod = 1e9 + 7;

const int N = 1e5 + 9;

int pw[N];
void prec() {
  pw[0] = 1;
  for (int i = 1; i < N; i++) {
```

```
    pw[i] = 1LL * pw[i - 1] * p % mod;
  }
}
int get_hash(string s) {
  int n = s.size();
  int hs = 0;
  for (int i = 0; i < n; i++) {
    hs += 1LL * s[i] * pw[i] % mod;
    hs %= mod;
  }
  return hs;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  prec();
  string a, b; cin >> a >> b;
  cout << (get_hash(a) == get_hash(b)) << '\n';
  return 0;
}
```

- **Double Hashing**

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

const int p1 = 137, mod1 = 127657753, p2 = 277, mod2 = 987654319;

const int N = 1e5 + 9;

int pw1[N], pw2[N];
void prec() {
  pw1[0] = 1;
  for (int i = 1; i < N; i++) {
    pw1[i] = 1LL * pw1[i - 1] * p1 % mod1;
  }
  pw2[0] = 1;
  for (int i = 1; i < N; i++) {
    pw2[i] = 1LL * pw2[i - 1] * p2 % mod2;
  }
}
pair<int, int> get_hash(string s) {
  int n = s.size();
  int hs1 = 0;
  for (int i = 0; i < n; i++) {
    hs1 += 1LL * s[i] * pw1[i] % mod1;
    hs1 %= mod1;
  }
  int hs2 = 0;
  for (int i = 0; i < n; i++) {
    hs2 += 1LL * s[i] * pw2[i] % mod2;
    hs2 %= mod2;
  }
  return {hs1, hs2};
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  prec();
  string a, b; cin >> a >> b;
  cout << (get_hash(a) == get_hash(b)) << '\n';
  return 0;
}
```

- **Pattern Matching**

Given a string and a pattern, your task is to count the number of positions where the pattern occurs in the string.

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

const int N = 1e6 + 9;
const int p1 = 137, mod1 = 127657753, p2 = 277, mod2 = 987654319;


int power(long long n, long long k, int mod) {
  int ans = 1 % mod; n %= mod; if (n < 0) n += mod;
  while (k) {
    if (k & 1) ans = (long long) ans * n % mod;
    n = (long long) n * n % mod;
    k >>= 1;
  }
  return ans;
}

int ip1, ip2;
pair<int, int> pw[N], ipw[N];
void prec() {
  pw[0] =  {1, 1};
  for (int i = 1; i < N; i++) {
    pw[i].first = 1LL * pw[i - 1].first * p1 % mod1;
    pw[i].second = 1LL * pw[i - 1].second * p2 % mod2;
```

```
  }
  ip1 = power(p1, mod1 - 2, mod1);
  ip2 = power(p2, mod2 - 2, mod2);
  ipw[0] =  {1, 1};
  for (int i = 1; i < N; i++) {
    ipw[i].first = 1LL * ipw[i - 1].first * ip1 % mod1;
    ipw[i].second = 1LL * ipw[i - 1].second * ip2 % mod2;
  }
}

pair<int, int> string_hash(string s) {
  int n = s.size();
  pair<int, int> hs({0, 0});
  for (int i = 0; i < n; i++) {
    hs.first += 1LL * s[i] * pw[i].first % mod1;
    hs.first %= mod1;
    hs.second += 1LL * s[i] * pw[i].second % mod2;
    hs.second %= mod2;
  }
  return hs;
}
pair<int, int> pref[N];
void build(string s) {
  int n = s.size();
  for (int i = 0; i < n; i++) {
    pref[i].first = 1LL * s[i] * pw[i].first % mod1;
    if (i) pref[i].first = (pref[i].first + pref[i - 1].first) % mod1;
    pref[i].second = 1LL * s[i] * pw[i].second % mod2;
    if (i) pref[i].second = (pref[i].second + pref[i - 1].second) % mod2;
  }
}
pair<int, int> get_hash(int i, int j) {
  assert(i <= j);
  pair<int, int> hs({0, 0});
  hs.first = pref[j].first;
  if (i) hs.first = (hs.first - pref[i - 1].first + mod1) % mod1;
  hs.first = 1LL * hs.first * ipw[i].first % mod1;
  hs.second = pref[j].second;
  if (i) hs.second = (hs.second - pref[i - 1].second + mod2) % mod2;
  hs.second = 1LL * hs.second * ipw[i].second % mod2;
  return hs;
}

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  prec();
  string a, b; cin >> a >> b;
  build(a);
  int ans = 0, n = a.size(), m = b.size();
  auto hash_b = string_hash(b);
  for (int i = 0; i + m - 1 < n; i++) {
    ans += get_hash(i, i + m - 1) == hash_b;
  }
  cout << ans << '\n';
  return 0;
}
```

- **Number of Divisors of a String**

Find the number of divisors of *s*. A string *b* is a divisor of *s* if it is possible to glue *b* zero or more times to get the string *s*. For example, the divisors of `abababab` are `ab, abab` and `abababab`. solve it in 2s?

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

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  string s; cin >> s;
  int ans = 0;
  int n = s.size();
  for (int len = 1; len <= n / 2; len++) {
    bool ok = true;
    for (int i = 0; i + len - 1 < n; i += len) {
      ok &= get_hash(i, i + len - 1) == get_hash(0, len - 1);
    }
    ans += ok;
  }
  return 0;
}
```

- **Queries to find the longest common prefix of two substrings.**

Given a string s of size n and q queries of type i, j, x, y. Find the LCP of substrings $s[i \dots j]$ and $s[x \dots y]$. $1 \le n, q \le 10^5$.

```
int lcp(int i, int j, int x, int y) { // O(log n)
  int l = 1, r = min(j - i + 1, y - x + 1), ans = 0;
  while (l <= r) {
    int mid = l + r >> 1;
    if (get_hash(i, i + mid - 1) == get_hash(x, x + mid - 1)) {
      ans = mid;
      l = mid + 1;
    }
```

```
    else {
      r = mid - 1;
    }
  }
  return ans;
}
```

- **Find the largest substring that occurs more than k times**

```
int n;
int max_oc(int len) {
  map<pair<int, int>, int> mp;
  for (int i = 0; i + len - 1 < n; i++) {
    mp[get_hash(i, i + len - 1)]++;
  }
  int ans = 0;
  for (auto [x, y]: mp) {
    ans = max(ans, y);
  }
  return ans;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  prec();
  string s; cin >> s;
  build(s);
  int k; cin >> k;
  n = s.size();
  int l = 1, r = s.size(), ans = -1;
  while (l <= r) {
    int mid = (l + r) >> 1;
    if (max_oc(mid) >= k) {
      ans = mid;
      l = mid + 1;
    }
    else {
      r = mid - 1;
    }
  }
  cout << ans << '\n';
  return 0;
}
```

- **Find the Longest Common Substring of Two Strings**

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

const int N = 1e5 + 9;
const int p1 = 137, mod1 = 127657753, p2 = 277, mod2 = 987654319;

int power(long long n, long long k, int mod) {
  int ans = 1 % mod; n %= mod; if (n < 0) n += mod;
  while (k) {
    if (k & 1) ans = (long long) ans * n % mod;
    n = (long long) n * n % mod;
    k >>= 1;
  }
  return ans;
}

int ip1, ip2;
pair<int, int> pw[N], ipw[N];
void prec() { // O(n)
  pw[0] =  {1, 1};
  for (int i = 1; i < N; i++) {
    pw[i].first = 1LL * pw[i - 1].first * p1 % mod1;
    pw[i].second = 1LL * pw[i - 1].second * p2 % mod2;
  }
  ip1 = power(p1, mod1 - 2, mod1);
  ip2 = power(p2, mod2 - 2, mod2);
  ipw[0] =  {1, 1};
  for (int i = 1; i < N; i++) {
    ipw[i].first = 1LL * ipw[i - 1].first * ip1 % mod1;
    ipw[i].second = 1LL * ipw[i - 1].second * ip2 % mod2;
  }
}

pair<int, int> string_hash(string s) { // O(n)
  int n = s.size();
  pair<int, int> hs({0, 0});
  for (int i = 0; i < n; i++) {
    hs.first += 1LL * s[i] * pw[i].first % mod1;
    hs.first %= mod1;
    hs.second += 1LL * s[i] * pw[i].second % mod2;
    hs.second %= mod2;
  }
  return hs;
}
struct Hashing {
  pair<int, int> pref[N];
  void build(string s) { // O(n)
    int n = s.size();
    for (int i = 0; i < n; i++) {
      pref[i].first = 1LL * s[i] * pw[i].first % mod1;
      if (i) pref[i].first = (pref[i].first + pref[i - 1].first) % mod1;
```

```
      pref[i].second = 1LL * s[i] * pw[i].second % mod2;
      if (i) pref[i].second = (pref[i].second + pref[i - 1].second) % mod2;
    }
  }
  pair<int, int> get_hash(int i, int j) { // O(1)
    // assert(i <= j);
    pair<int, int> hs({0, 0});
    hs.first = pref[j].first;
    if (i) hs.first = (hs.first - pref[i - 1].first + mod1) % mod1;
    hs.first = 1LL * hs.first * ipw[i].first % mod1;
    hs.second = pref[j].second;
    if (i) hs.second = (hs.second - pref[i - 1].second + mod2) % mod2;
    hs.second = 1LL * hs.second * ipw[i].second % mod2;
    return hs;
  }
}A, B;
int n;
string a, b;
string res;
bool ok(int k) { // is there a k length substring that occurs in both a and b
  set<pair<int, int>> substring_hashes_in_a;
  for (int i = 0; i + k - 1 < n; i++) {
    substring_hashes_in_a.insert(A.get_hash(i, i + k - 1));
  }

  for (int i = 0; i + k - 1 < n; i++) {
    auto substring_hash_in_b = B.get_hash(i, i + k - 1);
    if (substring_hashes_in_a.find(substring_hash_in_b) != substring_hashes_in_a.end()) {
      res = b.substr(i, k);
      return true;
    }
  }
  return false;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  prec();
  cin >> n;
  cin >> a >> b;
  A.build(a);
  B.build(b);
  int l = 1, r = n, ans = 0;
  while (l <= r) {
    int mid = (l + r) / 2;
    if (ok(mid)) {
      ans = mid;
      l = mid + 1;
    }
    else {
      r = mid - 1;
    }
  }
  // cout << ans << '\n';
  ok(ans);
  cout << res << '\n';
  // O(n log^2 n)
  return 0;
}
```

- **Disjoint Set Union (Data Structure)**.

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5 + 69;
int p[N];

int find(int x) {
  return p[x] == x ? x : p[x] = find(p[x]);
}
void join (int u , int v) {
  u = find(u);
  v = find(v);
  if(u != v) {
    p[u] = v;
  }
}
int main()
{
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n , q;
  cin >> n >> q;
  map<string , int > mp;
  for(int i = 0; i < n; i++) {
    string s;
    cin >> s;
    mp[s] = i; p[i]  = i;
  }
  for(int i = 0; i < q; i++) {
    int t;
    cin >> t;
    string a , b;
    cin >> a >> b;
    if(t == 1) {
      int u = mp[a];
      int v = mp[b];
```

```
      join (u , v);
    }
    else {
      int u = mp[a];
      int v = mp[b];
      u = find(u);
      v = find(v);
      if(u == v) {
        cout << "yes" << endl;
      }
      else cout << "no" << endl;
    }
  }
  return 0;
}
```

13. List of Useful Equations

General

1. $\sum_{0 \le k \le n} \binom{n-k}{k} = Fib_{n+1}$
2. $\binom{n}{k} = \binom{n}{n-k}$
3. $\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$
4. $k\binom{n}{k} = n\binom{n-1}{k-1}$
5. $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$
6. $\sum_{i=0}^{n} \binom{n}{i} = 2^n$
7. $\sum_{i \ge 0} \binom{n}{2i} = 2^{n-1}$
8. $\sum_{i \ge 0} \binom{n}{2i+1} = 2^{n-1}$
9. $\sum_{i=0}^{k} (-1)^i \binom{n}{i} = (-1)^k \binom{n-1}{k}$
10. $\sum_{i=0}^{k} \binom{n+i}{i} = \sum_{i=0}^{k} \binom{n+i}{n} = \binom{n+k+1}{k}$
11. $1\binom{n}{1} + 2\binom{n}{2} + 3\binom{n}{3} + \ldots + n\binom{n}{n} = n2^{n-1}$
12. $1^2\binom{n}{1} + 2^2\binom{n}{2} + 3^2\binom{n}{3} + \ldots + n^2\binom{n}{n} = (n+n^2)2^{n-2}$
13. **Vandermonde's Identify:** $\sum_{k=0}^{r} \binom{m}{k}\binom{n}{r-k} = \binom{m+n}{r}$
14. **Hockey-Stick Identify:** $n, r \in N, n > r, \sum_{i=r}^{n} \binom{i}{r} = \binom{n+1}{r+1}$
15. $\sum_{i=0}^{k} \binom{k}{i}^2 = \binom{2k}{k}$
16. $\sum_{k=0}^{n} \binom{n}{k}\binom{n}{n-k} = \binom{2n}{n}$
17. $\sum_{k=q}^{n} \binom{n}{k}\binom{k}{q} = 2^{n-q}\binom{n}{q}$
18. $\sum_{i=0}^{n} k^i \binom{n}{i} = (k+1)^n$
19. $\sum_{i=0}^{n} \binom{2n}{i} = 2^{2n-1} + \frac{1}{2}\binom{2n}{n}$

20. $\sum_{i=1}^{n}\binom{n}{i}\binom{n-1}{i-1}=\binom{2n-1}{n-1}$
21. $\sum_{i=0}^{n}\binom{2n}{i}^2=\frac{1}{2}\left(\binom{4n}{2n}+\binom{2n}{n}^2\right)$
22. **Highest Power of 2 that divides ${}^{2n}C_n$:** Let x be the number of 1s in the binary representation. Then the number of odd terms will be 2^x. Let it form a sequence. The n-th value in the sequence (starting from n = 0) gives the highest power of 2 that divides ${}^{2n}C_n$.
23. **Combination with repetition:** Let's say we choose elements from an element set, the order doesn't matter and each element can be chosen more than once. In that case, the number of different combinations is: $\binom{n+k-1}{k}$
24. **Pascal Triangle**
    a. In a row p where p is a prime number, all the terms in that row except the 1s are multiples of p.
    b. Parity: To count odd terms in row n, convert n to binary. Let x be the number of 1s in the binary representation. Then the number of odd terms will be 2^x.
    c. Every entry in row $2^n-1, n\geq 0,$ is odd.
25. An integer $n\geq 2$ is prime if and only if all the intermediate binomial coefficients $\binom{n}{1},\binom{n}{2},\ldots,\binom{n}{n-1}$ are divisible by n.
26. **Kummer's Theorem:** For given integers $n\geq m\geq 0$ and a prime number p, the largest power of p dividing $\binom{n}{m}$ is equal to the number of carries when m is added to n-m in base p. For implementation take inspiration from lucas theorem.
27. Number of different binary sequences of length n such that no two 0's are adjacent=Fib_{n+1}
28. The number non-negative solution of the equation: $x_1+x_2+x_3+\ldots+x_k=n$ is $\binom{n+k-1}{n}$
29. Number of ways to choose n ids from 1 to b such that every id has distance at least k $=\left(\frac{b-(n-1)(k-1)}{n}\right)$
30. $\sum_{i=1,3,5,\ldots}^{i\leq n}\binom{n}{i}a^{n-i}b^i=\frac{1}{2}((a+b)^n-(a-b)^n)$
31. $\sum_{i=0}^{n}\frac{\binom{k}{i}}{\binom{n}{i}}=\frac{\binom{n+1}{n-k+1}}{\binom{n}{k}}$

Math

General

1. $ab \mod ac = a(b \mod c)$
2. $\sum_{i=0}^{n} i\cdot i! = (n+1)!-1.$
3. $a^k-b^k=(a-b)\cdot(a^{k-1}b^0+a^{k-2}b^1+\ldots+a^0b^{k-1})$
4. $\min(a+b,c)=a+\min(b,c-a)$
5. $|a-b|+|b-c|+|c-a|=2(\max(a,b,c)-\min(a,b,c))$
6. $a\cdot b\leq c\rightarrow a\leq\left\lfloor\frac{c}{b}\right\rfloor$ is correct
7. $a\cdot b< c\rightarrow a<\left\lfloor\frac{c}{b}\right\rfloor$ is incorrect
8. $a\cdot b\geq c\rightarrow a\geq\left\lfloor\frac{c}{b}\right\rfloor$ is correct
9. $a\cdot b> c\rightarrow a>\left\lfloor\frac{c}{b}\right\rfloor$ is correct
10. For positive integer n, and arbitrary real numbers m, x,
    $\left\lfloor\frac{\lfloor x/m\rfloor}{n}\right\rfloor=\left\lfloor\frac{x}{mn}\right\rfloor$
    $\left\lceil\frac{\lceil x/m\rceil}{n}\right\rceil=\left\lceil\frac{x}{mn}\right\rceil$
11. $\sum_{i=1}^{n} ia^i=\frac{a(na^{n+1}-(n+1)a^n+1)}{(a-1)^2}$

12. We are given n numbers $a_1, a_2, \ldots, a_n$ and our task is to find a value x that minimizes the sum, optimal $x =$ median of the array.

    if n is even $x = [leftmedian, rightmedian]$ i.e. every number in this range will work.

    For minimizing optimal $x = \dfrac{(a_1 + a_2 + \ldots + a_n)}{n}$

13. Given an array a of n non-negative integers. The task is to find the sum of the product of elements of all the possible subsets. It is equal to the product of $(a_i + 1)$ for all a_i

14. The number of ways to represent n as the sum of four squares is eight times the sum of all its divisors which are not divisible by 4, i.e.

    $r_4(n) = 8 \sum d|n; 4 \nmid dd$
    $r8(n) = 16 \sum_{d|n} (-1)^{n+d} d^3$

Number Theory

General

0. for $i > j$, $\gcd(i, j) = \gcd(i - j, j) \leq (i - j)$

1. $\sum_{x=1}^{n} [d|x^k] = \left\lfloor \dfrac{n}{\prod_{i=0} p_i^{\left\lceil \frac{e_i}{k} \right\rceil}} \right\rfloor$,

   where $d = \prod_{i=0} p_i^{e_i}$. Here, $[a|b]$ means if a divides b then it is 1, otherwise it is 0.

2. The number of lattice points on segment (x_1, y_1) to (x_2, y_2) is $\gcd(abs(x_1 - x_2), abs(y_1 - y_2)) + 1$

3. $(n-1)! \mod n = n - 1$ if n is prime, 2 if $n = 4$, 0 otherwise.

4. A number has odd number of divisors if it is perfect square

5. The sum of all divisors of a natural number n is odd if and only if $n = 2^r \cdot k^2$ where r is non-negative and k is positive integer.

GCD and LCM

154. $\gcd(a, 0) = a$
155. $\gcd(a, b) = \gcd(b, a \mod b)$
156. Every common divisor of a and b is a divisor of $\gcd(a, b)$.
157. if m is any integer, then $\gcd(a + m{\cdot}b, b) = \gcd(a, b)$
158. The gcd is a multiplicative function in the following sense: if a_1 and a_2 are relatively prime, then $\gcd(a_1 \cdot a_2, b) = \gcd(a_1, b) \cdot \gcd(a_2, b)$.
159. $\gcd(a, b) \cdot \operatorname{lcm}(a, b) = |a{\cdot}b|$
160. $\gcd(a, \operatorname{lcm}(b, c)) = \operatorname{lcm}(\gcd(a, b), \gcd(a, c))$.
161. $\operatorname{lcm}(a, \gcd(b, c)) = \gcd(\operatorname{lcm}(a, b), \operatorname{lcm}(a, c))$.
162. For non-negative integers a and b, where a and b are not both zero, $\gcd(n^a - 1, n^b - 1) = n^{\gcd(a,b)} - 1$
163. $\gcd(a, b) = \sum_{k|a \text{ and } k|b} \phi(k)$
164. $\sum_{i=1}^{n} [\gcd(i, n) = k] = \phi\left(\dfrac{n}{k}\right)$

173. $F(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \operatorname{lcm}(i,j) = \sum_{l=1}^{n} \left(\frac{\left(1 + \lfloor \frac{n}{l} \rfloor\right) \left(\lfloor \frac{n}{l} \rfloor\right)}{2} \right)^2 \sum_{d|l} \mu(d) l d$

174. $\gcd(\operatorname{lcm}(a,b), \operatorname{lcm}(b,c), \operatorname{lcm}(a,c)) = \operatorname{lcm}(\gcd(a,b), \gcd(b,c), \gcd(a,c))$

175. $\gcd(A_L, A_{L+1}, \ldots, A_R) = \gcd(A_L, A_{L+1} - A_L, \ldots, A_R - A_{R-1})$.'

176. Given n, If $SUM = LCM(1,n) + LCM(2,n) + \ldots + LCM(n,n)$

then SUM = $\frac{n}{2}(\sum_{d|n} (\phi(d) \times d) + 1$

Miscellaneous

184. $a + b = a \oplus b + 2(a \& b)$.
185. $a + b = a \mid b + a \& b$
186. $a \oplus b = a \mid b - a \& b$
187. k_{th} bit is set in x iff $x \mod 2^{k-1} \geq 2^k$. It comes handy when you need to look at the bits of the numbers which are pair sums or subset sums etc.
188. k_{th} bit is set in x iff $x \mod 2^{k-1} - x \mod 2^k \neq 0$ ($= 2^k$ to be exact). It comes handy when you need to look at the bits of the numbers which are pair sums or subset sums etc.
189. $n \mod 2^i = n \& (2^i - 1)$
190. $1 \oplus 2 \oplus 3 \oplus \cdots \oplus (4k - 1) = 0$ for any $k \geq 0$