🧩 Plan for the Template

We'll structure it in **sections** so you can easily scroll to what you need:

1. **Boilerplate & Fast I/O**
2. **Macros & Type Aliases**
3. **Debug Tools (optional in local mode)**
4. **Math Utilities**
    - GCD, LCM, ModPow, Inverse, nCr, Sieve, etc.
5. **Data Structures**
    - DSU, Segment Tree, Fenwick Tree, Sparse Table
6. **Graph Algorithms**
    - DFS, BFS, Dijkstra, Bellman-Ford, Floyd-Warshall, Toposort
7. **Trees**
    - LCA, Binary Lifting, Euler Tour
8. **Strings**
    - KMP, Z-Algorithm, Rolling Hash
9. **Miscellaneous**
    - Coordinate compression, custom comparators, etc.

```
#include <bits/stdc++.h>
using namespace std;
// ---------- FAST I/O ----------
#define FAST_IO ios::sync_with_stdio(false); cin.tie(nullptr)
// ---------- MACROS ----------
#define int long long
#define pb push_back
#define all(x) (x).begin(), (x).end()
#define rall(x) (x).rbegin(), (x).rend()
#define sz(x) (int)(x).size()
#define fi first
#define se second
// ---------- CONSTANTS ----------
const int MOD = 1e9 + 7;
const int INF = 1e18;
const double EPS = 1e-9;

// ---------- UTILITY ----------
template<class T> bool ckmin(T &a, const T &b){ if(b<a){a=b;return true;} return false; }
template<class T> bool ckmax(T &a, const T &b){ if(b>a){a=b;return true;} return false; }
int addmod(int a,int b,int m=MOD){ a%=m; b%=m; int r=a+b; if(r>=m) r-=m; return r; }
int submod(int a,int b,int m=MOD){ a%=m; b%=m; int r=a-b; if(r<0) r+=m; return r; }
int mulmod(long long a,long long b,int m=MOD){ return (int)((a*b)%m); }


// ---------- MATH UTILITIES ----------
int gcd(int a, int b) { return b == 0 ? a : gcd(b, a % b); }
int lcm(int a, int b) { return a / gcd(a, b) * b; }
int modpow(int a, int b, int mod = MOD) {
    int res = 1;
    while (b) {
        if (b & 1) res = res * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return res;
}
int modinv(int a, int mod = MOD) { return modpow(a, mod - 2, mod); }
vector<int> fact(1, 1);
void init_fact(int n) {
    fact.resize(n+1);
    for (int i = 1; i <= n; i++) fact[i] = fact[i-1] * i % MOD;
}

int nCr(int n, int r) {
    if (r < 0 || r > n) return 0;
    return fact[n] * modinv(fact[r]) % MOD * modinv(fact[n - r]) % MOD;
}
```

```
// ---------- SIEVE ----------
vector<int> primes;
vector<bool> is_prime;
void sieve(int n) {
    is_prime.assign(n + 1, true);
    is_prime[0] = is_prime[1] = false;
    for (int i = 2; i * i <= n; i++)
        if (is_prime[i])
            for (int j = i * i; j <= n; j += i)
                is_prime[j] = false;
    for (int i = 2; i <= n; i++)
        if (is_prime[i]) primes.pb(i);
}

// ---------- DSU ----------
struct DSU {
    vector<int> p, sz;
    DSU(int n) { p.resize(n); sz.assign(n, 1); iota(all(p), 0); }
    int find(int x) { return p[x] == x ? x : p[x] = find(p[x]); }
    bool unite(int a, int b) {
        a = find(a), b = find(b);
        if (a == b) return false;
        if (sz[a] < sz[b]) swap(a, b);
        p[b] = a; sz[a] += sz[b];
        return true;
    }
};

// ---------- FENWICK TREE ----------
struct Fenwick {
    int n; vector<int> bit;
    Fenwick(int n) : n(n), bit(n+1) {}
    void update(int idx, int val) {
        for (; idx <= n; idx += idx & -idx)
            bit[idx] += val;
    }
    int query(int idx) {
        int res = 0;
        for (; idx > 0; idx -= idx & -idx)
            res += bit[idx];
        return res;
    }
};
```

```
// ---------- SEGMENT TREE ----------
struct SegTree {
    int n; vector<int> t;
    SegTree(int n) : n(n), t(4*n, 0) {}
    void build(vector<int> &a, int v, int tl, int tr) {
        if (tl == tr) t[v] = a[tl];
        else {
            int tm = (tl + tr) / 2;
            build(a, v*2, tl, tm);
            build(a, v*2+1, tm+1, tr);
            t[v] = t[v*2] + t[v*2+1];
        }
    }
    int query(int v, int tl, int tr, int l, int r) {
        if (l > r) return 0;
        if (l == tl && r == tr) return t[v];
        int tm = (tl + tr) / 2;
        return query(v*2, tl, tm, l, min(r, tm)) +
               query(v*2+1, tm+1, tr, max(l, tm+1), r);
    }
    void update(int v, int tl, int tr, int pos, int val) {
        if (tl == tr) t[v] = val;
        else {
            int tm = (tl + tr) / 2;
            if (pos <= tm) update(v*2, tl, tm, pos, val);
            else update(v*2+1, tm+1, tr, pos, val);
            t[v] = t[v*2] + t[v*2+1];
        }
    }
};
```

```
// ----------------- LCA (binary lifting) -----------------
struct LCA {
    int n, LOG;
    vector<int> depth;
    vector<vector<int>> up;
    vector<vector<int>> adj;
    LCA(int n=0){ init(n); }
    void init(int n_){
        n=n_;
        LOG = 1; while((1<<LOG) <= n) LOG++;
        depth.assign(n,0);
        up.assign(LOG, vector<int>(n, -1));
        adj.assign(n, {});
    }
    void addEdge(int u,int v){ adj[u].pb(v); adj[v].pb(u); }
    void dfs(int v,int p){
        up[0][v] = p==-1 ? v : p;
        for(int i=1;i<LOG;i++) up[i][v] = up[i-1][ up[i-1][v] ];
        for(int to: adj[v]){
            if(to==p) continue;
            depth[to] = depth[v]+1;
            dfs(to,v);
        }
    }
    void build(int root=0){
        dfs(root, -1);
    }
    int kthAncestor(int v, int k){
        for(int i=0;i<LOG;i++) if(k&(1<<i)) v = up[i][v];
        return v;
    }
    int lca(int a,int b){
        if(depth[a] < depth[b]) swap(a,b);
        int diff = depth[a]-depth[b];
        a = kthAncestor(a, diff);
        if(a==b) return a;
        for(int i=LOG-1;i>=0;i--) if(up[i][a] != up[i][b]) { a = up[i][a]; b = up[i][b]; }
        return up[0][a];
    }
};
```

```
// ---------- GRAPH ----------
vector<vector<int>> adj;
vector<int> dist, vis;

void bfs(int s) {
    queue<int> q; q.push(s);
    dist.assign(sz(adj), INF);
    dist[s] = 0;
    while (!q.empty()) {
        int v = q.front(); q.pop();
        for (int u : adj[v]) {
            if (dist[u] > dist[v] + 1) {
                dist[u] = dist[v] + 1;
                q.push(u);
            }
        }
    }
}

void dfs(int v) {
    vis[v] = 1;
    for (int u : adj[v])
        if (!vis[u]) dfs(u);
}

// ---------- DIJKSTRA ----------
vector<int> dijkstra(int n, int s, vector<vector<pair<int,int>>> &g) {
    vector<int> d(n, INF);
    d[s] = 0;
    priority_queue<pair<int,int>, vector<pair<int,int>>, greater<>> pq;
    pq.push({0, s});
    while (!pq.empty()) {
        auto [dist_u, u] = pq.top(); pq.pop();
        if (dist_u != d[u]) continue;
        for (auto [v, w] : g[u]) {
            if (d[v] > d[u] + w) {
                d[v] = d[u] + w;
                pq.push({d[v], v});
            }
        }
    }
    return d;
}
```

```
// ---------- STRINGS ----------
vector<int> prefix_function(string s) {
    int n = sz(s);
    vector<int> pi(n);
    for (int i = 1; i < n; i++) {
        int j = pi[i-1];
        while (j > 0 && s[i] != s[j]) j = pi[j-1];
        if (s[i] == s[j]) j++;
        pi[i] = j;
    }
    return pi;
}

vector<int> z_function(string s) {
    int n = sz(s);
    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;
}




// ----------------- MISC HELPERS -----------------
int bitcount(long long x){ return __builtin_popcountll((unsigned long long)x); }
int highest_bit(long long x){ return 63 - __builtin_clzll((unsigned long long)x); }



// ---------- MAIN ----------
void solve() {
    // your code here
}

int32_t main() {
    FAST_IO;
    int T = 1;
    // cin >> T;
    while (T--) solve();
}
```

```
import sys, math
from collections import deque, defaultdict, Counter
from bisect import bisect_left, bisect_right
from heapq import heappush, heappop
sys.setrecursionlimit(3 * 10**5)
input = sys.stdin.readline

# ---------------- FAST I/O ----------------
def inp(): return sys.stdin.readline().strip()
def inpi(): return int(inp())
def inpl(): return list(map(int, inp().split()))
def inps(): return inp().split()

# ---------------- CONSTANTS ----------------
INF = 10**18
MOD = 10**9 + 7

# ========================================================
# 🧱 MATH UTILITIES
# ========================================================
def gcd(a, b):
    """Greatest common divisor. O(log min(a,b))."""
    while b: a, b = b, a % b
    return a

def lcm(a, b):
    """Least common multiple using gcd."""
    return a * b // gcd(a, b)

def modpow(a, b, mod=MOD):
    """Fast exponentiation (a^b % mod). O(log b)."""
    res = 1
    a %= mod
    while b:
        if b & 1: res = res * a % mod
        a = a * a % mod
        b >>= 1
    return res

def modinv(a, mod=MOD):
    """Modular inverse using Fermat's theorem (mod must be prime)."""
    return modpow(a, mod - 2, mod)

# Example:
# print(gcd(10, 15)) → 5
# print(lcm(3, 5)) → 15
# print(modpow(2, 10)) → 1024
# print(modinv(3)) → 333333336 (since 3 * 333333336 % MOD = 1)
```

```
# ============================================================
# 🔢 COMBINATORICS
# ============================================================
MAXN = 10**6
fact = [1]*(MAXN+1)
invfact = [1]*(MAXN+1)
for i in range(1, MAXN+1): fact[i] = fact[i-1]*i%MOD
invfact[MAXN] = modinv(fact[MAXN])
for i in range(MAXN, 0, -1): invfact[i-1] = invfact[i]*i%MOD

def nCr(n, r):
    """n choose r mod MOD. O(1) after precompute."""
    if r<0 or r>n: return 0
    return fact[n]*invfact[r]%MOD*invfact[n-r]%MOD

# Example:
# print(nCr(5, 2)) → 10

# ============================================================
# 🔍 SIEVE OF ERATOSTHENES
# ============================================================
def sieve(n):
    """Returns all primes <= n. O(n log log n)."""
    isprime = [True]*(n+1)
    isprime[0]=isprime[1]=False
    for i in range(2, int(n**0.5)+1):
        if isprime[i]:
            for j in range(i*i, n+1, i):
                isprime[j]=False
    return [i for i in range(2,n+1) if isprime[i]]

# Example:
# print(sieve(10)) → [2,3,5,7]
```

```
# ========================================================
# 🌆 GRAPH ALGORITHMS
# ========================================================
def bfs(start, adj):
    """Shortest path in unweighted graph. O(V+E)."""
    dist=[-1]*len(adj)
    q=deque([start]); dist[start]=0
    while q:
        v=q.popleft()
        for u in adj[v]:
            if dist[u]==-1:
                dist[u]=dist[v]+1
                q.append(u)
    return dist

# Example:
# adj=[[1,2],[0,3],[0,3],[1,2]]
# print(bfs(0,adj)) → [0,1,1,2]

def dfs(v, adj, vis):
    """Depth-first search (connectivity, tree traversal)."""
    vis[v]=1
    for u in adj[v]:
        if not vis[u]:
            dfs(u,adj,vis)

# Example:
# vis=[0]*4; dfs(0,[[1,2],[0,3],[0,3],[1,2]],vis)
# print(vis) → [1,1,1,1]

def dijkstra(start, graph):
    """Shortest path with positive weights. O(E log V)."""
    n=len(graph)
    dist=[INF]*n; dist[start]=0
    pq=[(0,start)]
    while pq:
        d,u=heappop(pq)
        if d!=dist[u]: continue
        for v,w in graph[u]:
            if dist[v]>d+w:
                dist[v]=d+w
                heappush(pq,(dist[v],v))
    return dist

# Example:
# g=[[(1,2),(2,5)],[(0,2),(2,1)],[(0,5),(1,1)]]
# print(dijkstra(0,g)) → [0,2,3]
```

```
def bellman_ford(n, edges, src):
    """
    Shortest path allowing negative edges (no negative cycle).
    edges: list of (u,v,w)
    O(VE)
    """
    dist=[INF]*n
    dist[src]=0
    for _ in range(n-1):
        for u,v,w in edges:
            if dist[u]!=INF and dist[u]+w<dist[v]:
                dist[v]=dist[u]+w
    # detect negative cycle (optional)
    for u,v,w in edges:
        if dist[u]!=INF and dist[u]+w<dist[v]:
            return None  # negative cycle detected
    return dist

# Example:
# edges=[(0,1,1),(1,2,-1),(2,3,-1),(3,0,2)]
# print(bellman_ford(4,edges,0)) → [0,1,0,-1]

def topo_sort(n, adj):
    """Topological order (DAG). O(V+E)."""
    indeg=[0]*n
    for u in range(n):
        for v in adj[u]: indeg[v]+=1
    q=deque([i for i in range(n) if indeg[i]==0])
    topo=[]
    while q:
        u=q.popleft(); topo.append(u)
        for v in adj[u]:
            indeg[v]-=1
            if indeg[v]==0: q.append(v)
    return topo

# Example:
# adj=[[1,2],[],[1],[]]
# print(topo_sort(4,adj)) → [0,2,1,3] (one valid topo order)
```

```
# ======================================================
# ⚔️ DISJOINT SET UNION (DSU)
# ======================================================
class DSU:
    """Union-Find. O(α(N))."""
    def __init__(self,n):
        self.p=list(range(n))
        self.sz=[1]*n
    def find(self,x):
        if self.p[x]!=x:
            self.p[x]=self.find(self.p[x])
        return self.p[x]
    def unite(self,a,b):
        a=self.find(a); b=self.find(b)
        if a==b: return False
        if self.sz[a]<self.sz[b]: a,b=b,a
        self.p[b]=a; self.sz[a]+=self.sz[b]
        return True

# Example:
# d=DSU(4); d.unite(0,1); print(d.find(1)==d.find(0)) → True
```

```
# ========================================================
# 🌳 SEGMENT TREE
# ========================================================
class SegTree:
    """Range query (sum/min/max). O(log N)."""
    def __init__(self,arr,func=sum,default=0):
        self.n=len(arr)
        self.func=func
        self.default=default
        self.size=1
        while self.size<self.n: self.size<<=1
        self.data=[default]*(2*self.size)
        for i in range(self.n): self.data[self.size+i]=arr[i]
        for i in range(self.size-1,0,-1):
            self.data[i]=func([self.data[i*2],self.data[i*2+1]])
    def update(self,idx,val):
        idx+=self.size; self.data[idx]=val
        while idx>1:
            idx>>=1
            self.data[idx]=self.func([self.data[idx*2],self.data[idx*2+1]])
    def query(self,l,r):
        l+=self.size; r+=self.size
        resl,resr=self.default,self.default
        while l<=r:
            if l&1:
                resl=self.func([resl,self.data[l]]); l+=1
            if not (r&1):
                resr=self.func([self.data[r],resr]); r-=1
            l>>=1; r>>=1
        return self.func([resl,resr])

# Example:
# seg=SegTree([1,2,3,4])
# print(seg.query(1,3)) → 9
# seg.update(2,10); print(seg.query(0,2)) → 13
```

```
# ============================================================
# 🔤 STRING ALGORITHMS
# ============================================================
def prefix_function(s):
    """KMP preprocessing. Find longest prefix-suffix for each pos."""
    n=len(s); pi=[0]*n
    for i in range(1,n):
        j=pi[i-1]
        while j and s[i]!=s[j]: j=pi[j-1]
        if s[i]==s[j]: j+=1
        pi[i]=j
    return pi

# Example:
# print(prefix_function("ababc")) → [0,0,1,2,0]

def z_function(s):
    """Z[i] = length of longest prefix starting from i. O(N)."""
    n=len(s); z=[0]*n; l=r=0
    for i in range(1,n):
        if i<=r: z[i]=min(r-i+1,z[i-l])
        while i+z[i]<n and s[z[i]]==s[i+z[i]]: z[i]+=1
        if i+z[i]-1>r: l,r=i,i+z[i]-1
    return z

# Example:
# print(z_function("aaaa")) → [0,3,2,1]

# ============================================================
# 🔎 BINARY SEARCH
# ============================================================
def binary_search_first(lo,hi,cond):
    """Find smallest x s.t cond(x)=True."""
    ans=hi+1
    while lo<=hi:
        mid=(lo+hi)//2
        if cond(mid):
            ans=mid; hi=mid-1
        else: lo=mid+1
    return ans

# Example:
# print(binary_search_first(0,10,lambda x:x>=5)) → 5
```

```
# ============================================================
# 🧠 SOLVE
# ============================================================
def solve():
    """Example usage: Compute shortest path with negative edges."""
    n,m=4,5
    edges=[(0,1,4),(0,2,5),(1,2,-3),(2,3,4),(3,1,-10)]
    res=bellman_ford(n,edges,0)
    if res is None:
        print("Negative cycle detected")
    else:
        print("Shortest distances:",res)

# ============================================================
# 🚀 MAIN
# ============================================================
if __name__=="__main__":
    T=1
    # T=inpi()
    for _ in range(T):
        solve()
```

```
/* ---------- BITWISE OPERATIONS TEMPLATE ----------


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

/* ---------- BASIC OPERATIONS ----------
   a & b  → bitwise AND
   a | b  → bitwise OR
   a ^ b  → bitwise XOR
   ~a     → bitwise NOT (flips bits)
   a << k → left shift (multiply by 2^k)
   a >> k → right shift (divide by 2^k)
*/
// Example values
void basicExamples() {
    int a = 13, b = 11;
    cout << "a & b = " << (a & b) << "\n";  // 9
    cout << "a | b = " << (a | b) << "\n";  // 15
    cout << "a ^ b = " << (a ^ b) << "\n";  // 6
    cout << "~a = " << (~a) << "\n";        // -14 (two's complement)
    cout << "a << 2 = " << (a << 2) << "\n"; // 52
    cout << "a >> 1 = " << (a >> 1) << "\n"; // 6
}
/* ---------- BITWISE UTILITIES ---------- */

bool isBitSet(int x, int k) {
    return (x >> k) & 1;         // Check if k-th bit is 1
}

int setBit(int x, int k) {
    return x | (1 << k);         // Set k-th bit
}

int clearBit(int x, int k) {
    return x & ~(1 << k);        // Clear k-th bit
}

int toggleBit(int x, int k) {
    return x ^ (1 << k);         // Flip k-th bit
}

int countBits(int x) {
    return __builtin_popcount(x); // Built-in function for counting 1s
}
bool isPowerOfTwo(int x) {
    return x > 0 && (x & (x - 1)) == 0;
}
```

```
/* ---------------- ADVANCED TRICKS ---------------- */

int rightmostSetBit(int x) {
    return x & -x;  // isolate lowest set bit
}

int turnOffRightmostBit(int x) {
    return x & (x - 1); // remove lowest set bit
}

/* ---------------- SUBSET ENUMERATION ----------------
   Commonly used in bitmask DP problems.
*/
void iterateSubsets(int n) {
    for (int mask = 0; mask < (1 << n); mask++) {
        cout << "Subset: ";
        for (int i = 0; i < n; i++) {
            if (mask & (1 << i))
                cout << i << " ";
        }
        cout << "\n";
    }
}

/* ---------------- XOR USE CASES ----------------
   1. Find unique element when every other element appears twice.
   2. Swap two numbers without temp variable.
*/
int findUnique(const vector<int> &a) {
    int x = 0;
    for (int v : a) x ^= v;
    return x;
}

void swapUsingXor(int &a, int &b) {
    if (a != b) {
        a ^= b;
        b ^= a;
        a ^= b;
    }
}
```

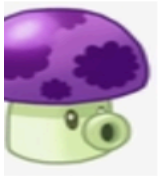
```
/* ---------------- DEMO ---------------- */
int main() {
    basicExamples();

    int x = 42; // 101010 (binary)
    cout << "Bit 1 set? " << isBitSet(x, 1) << "\n";
    cout << "Set bit 0: " << setBit(x, 0) << "\n";
    cout << "Clear bit 1: " << clearBit(x, 1) << "\n";
    cout << "Toggle bit 3: " << toggleBit(x, 3) << "\n";
    cout << "Count bits: " << countBits(x) << "\n";
    cout << "Rightmost set bit: " << rightmostSetBit(x) << "\n";
    cout << "After turning off rightmost bit: " << turnOffRightmostBit(x) << "\n";

    vector<int> nums = {5, 3, 5, 7, 3};
    cout << "Unique element: " << findUnique(nums) << "\n";

    int a = 10, b = 5;
    swapUsingXor(a, b);
    cout << "Swapped (a, b): " << a << " " << b << "\n";

    cout << "\nAll subsets of n=3:\n";
    iterateSubsets(3);
}
```

7 months ago, hide # | ☆

$1 \oplus 2 \oplus 3 \oplus \cdots \oplus (4n-1) = 0$ for all $n \in \mathbb{Z}^+$.

⟶ Reply

56



ew_Pew.

7 months ago, hide # ^ | ☆

A corollary:

If $n = 0 \ mod \ 4$ then XOR from 1 to n would be n

If $n = 1 \ mod \ 4$ then XOR from 1 to n would be 1

If $n = 2 \ mod \ 4$ then XOR from 1 to n would be $n + 1$

If $n = 3 \ mod \ 4$ then XOR from 1 to n would be 0

⟶ Reply

```
# ---------- Expected Value & Probability Template ----------
# Author: Krish
# Optimized for ICPC (PyPy3)
# Covers: modular arithmetic, rational probability, expectation DP
# -----------------------------------------------------------------

import sys, math
from fractions import Fraction
sys.setrecursionlimit(3 * 10**5)
input = sys.stdin.readline

MOD = 10**9 + 7
INF = float('inf')

# ---------- MODULAR MATH UTILS ----------
def modadd(a, b, m=MOD): return (a + b) % m
def modsub(a, b, m=MOD): return (a - b + m) % m
def modmul(a, b, m=MOD): return (a * b) % m

def modpow(a, b, m=MOD):
    """Fast modular exponentiation (a^b mod m)"""
    res = 1
    a %= m
    while b:
        if b & 1:
            res = (res * a) % m
        a = (a * a) % m
        b >>= 1
    return res

def modinv(a, m=MOD):
    """Modular inverse using Fermat's theorem (mod must be prime)."""
    return modpow(a, m - 2, m)

# Example:
# p = modmul(1, modinv(6)) → represents 1/6 mod 1e9+7

# ======================================================
# 🎲 Expected Value under Modulo (Discrete Random Events)
# ======================================================
def expected_mod(values, probs):
    """
    Expected value mod MOD.
    values: list of outcomes (int)
    probs: list of probabilities (each as modular fraction mod MOD)
    """
    assert len(values) == len(probs)
    E = 0
```

```
    for v, p in zip(values, probs):
        E = modadd(E, modmul(v, p))
    return E

# Example:
# Toss a biased coin: P(H)=1/3, P(T)=2/3, X=1 for head, 0 for tail
# values = [1, 0]
# probs = [modmul(1, modinv(3)), modmul(2, modinv(3))]
# print(expected_mod(values, probs)) → modular expectation = 333333336

# ======================================================
# 🎲 Expected Value with Fractions (Exact arithmetic)
# ======================================================
def expected_fraction(values, probs):
    """
    Expected value using Python Fractions for exact precision.
    values: [v1, v2, ...], probs: [p1, p2, ...] (Fractions)
    """
    E = Fraction(0, 1)
    for v, p in zip(values, probs):
        E += Fraction(v) * Fraction(p)
    return E

# Example:
# Expected outcome of dice roll (1..6)
# values = [1,2,3,4,5,6]
# probs = [Fraction(1,6)]*6
# print(expected_fraction(values, probs)) → 7/2

# ======================================================
# 🧩 DP on Expected Value (Common ICPC Pattern)
# ======================================================
"""
E[i] often represents expected steps/costs from state i.
Recurrence: E[i] = 1 + Σ(p(j) * E[next_state])
You solve by substitution or reverse iteration if transitions are acyclic.
"""

def expected_steps_dice(n):
    """
    Example: Expected rolls to reach sum >= n with dice rolls 1..6
    E[i] = 1 + (1/6)*Σ(E[min(i+k, n)]) for k=1..6
    """
    E = [0]*(n+7)
    for i in range(n-1, -1, -1):
        s = 0
        for k in range(1, 7):
            s += E[min(i+k, n)]
```

```
        E[i] = 1 + s/6
    return E[0]

# Example:
# print(expected_steps_dice(10)) → Expected rolls to reach 10

# ========================================================
# 📊 Conditional Probability Template
# ========================================================
def conditional_probability(p_a, p_b_given_a):
    """P(B) = P(B|A) * P(A)"""
    return p_b_given_a * p_a

def bayes_theorem(p_a, p_b_given_a, p_b):
    """P(A|B) = (P(B|A)*P(A)) / P(B)"""
    return (p_b_given_a * p_a) / p_b

# Example:
# P(A)=0.3, P(B|A)=0.7, P(B)=0.5
# print(bayes_theorem(0.3, 0.7, 0.5)) → 0.42

# ========================================================
# 🎯 Linearity of Expectation (Trick for ICPC)
# ========================================================
def linearity_of_expectation(expected_list):
    """Sum of expected values = expected sum (no independence needed)."""
    return sum(expected_list)

# Example:
# Two dice: E(X1)=3.5, E(X2)=3.5 → total E(X1+X2)=7
# print(linearity_of_expectation([3.5, 3.5])) → 7.0

# ========================================================
# 🧱 Expected Value for Continuous Distribution (approximation)
# ========================================================
def expected_continuous(func, a, b, steps=10000):
    """Numerical integration ∫ f(x)*x dx over [a,b]"""
    dx = (b - a) / steps
    s = 0
    for i in range(steps):
        x = a + i * dx
        s += func(x) * x * dx
    return s

# Example:
# Expectation of X when f(x)=2x for x in [0,1] (triangular distribution)
# print(round(expected_continuous(lambda x: 2*x, 0, 1), 3)) → 0.667
```

```
# ==========================================================
# 🧠 SOLVE (Sample usage combining everything)
# ==========================================================
def solve():
    # Example 1: Modular expected value (biased coin)
    values = [1, 0]
    probs = [modmul(1, modinv(3)), modmul(2, modinv(3))]
    print("Expected value mod:", expected_mod(values, probs))

    # Example 2: Fraction expected value (fair dice)
    vals = [1,2,3,4,5,6]
    probs = [Fraction(1,6)]*6
    print("Expected value fraction:", expected_fraction(vals, probs))

    # Example 3: Expected steps to reach n with dice rolls
    print("Expected rolls to reach 10:", round(expected_steps_dice(10), 3))

    # Example 4: Bayes theorem
    print("Bayes theorem example:", round(bayes_theorem(0.3, 0.7, 0.5), 3))

if __name__ == "__main__":
    solve()
```

🧱 C++ STL (Standard Template Library) – Complete ICPC Reference

Author: Krish
Prepared For: ICPC / Competitive Programming
Language: C++17 / C++20
Content: Detailed STL Documentation + Compact STL Cheat Sheet

---

📘 Part 1 – Detailed STL Documentation

🧩 Overview

Category	Description
Containers	Store and organize data efficiently
Iterators	Provide a way to traverse container elements
Algorithms	Reusable functions for sorting, searching, counting, etc.
Utilities	Helper data structures like pairs, tuples, and bitsets
Strings	For handling text
Math & Numeric	Basic numerical operations and accumulation

1️⃣ Containers

🟦 `vector<T>`

Description:
Dynamic array with fast random access and amortized O(1) insertion at the end.
Header: `<vector>`

Common Operations:

Function	Description	Complexity
`push_back(x)`	Add to end	O(1) amortized

`pop_back()`	Remove last element	O(1)
`insert(pos, val)`	Insert at position	O(n)
`erase(pos)`	Remove at position	O(n)
`sort(all(v))`	Sort ascending	O(n log n)

Example:

```
vector<int> v = {1, 2, 3};
v.push_back(4);
v.erase(v.begin()+1);
sort(v.begin(), v.end());
for (int x : v) cout << x << " ";
```

🟩 `array<T, N>`

Description:
Fixed-size array with STL functions.
Header: `<array>`

```
array<int, 5> a = {1, 2, 3, 4, 5};
cout << a.size() << " " << a.front() << " " << a.back();
```

🟧 `deque<T>`

Description:
Double-ended queue. Fast insertion/removal from both ends.
Header: `<deque>`

```
deque<int> dq = {1, 2, 3};
dq.push_front(0);
dq.push_back(4);
dq.pop_front();
dq.pop_back();
```

🟨 list<T>

Description:
Doubly linked list allowing O(1) insert/erase given iterator.
Header: `<list>`

```
list<int> l = {1, 2, 3};
l.push_front(0);
l.push_back(4);
l.remove(2);
l.reverse();
```

🟪 stack<T>

Description:
LIFO (Last In First Out).
Header: `<stack>`

```
stack<int> s;
s.push(10);
s.push(20);
cout << s.top(); // 20
s.pop();
```

🟫 queue<T>

Description:
FIFO (First In First Out).
Header: `<queue>`

```
queue<int> q;
q.push(10);
q.push(20);
cout << q.front(); // 10
q.pop();
```

🟥 priority_queue<T>

Description:
Heap-based container.
Header: `<queue>`

```
priority_queue<int> pq; // Max-heap
pq.push(10);
pq.push(5);
cout << pq.top(); // 10

priority_queue<int, vector<int>, greater<int>> minpq; // Min-heap
```

🟩 set<T>

Description:
Stores **unique, sorted** elements.
Header: `<set>`

Operation	Time
insert/erase/find	O(log n)

```
set<int> s = {1, 3, 2};
s.insert(4);
s.erase(2);
for (int x : s) cout << x << " ";
```

🟧 multiset<T>

Description:
Sorted, allows duplicates.
Header: `<set>`

```
multiset<int> ms = {1, 1, 2, 3};
ms.insert(2);
ms.erase(ms.find(1)); // removes one instance
```

🟦 unordered_set<T>

Description:
Hash-based set. Fast O(1) average operations.
Header: `<unordered_set>`

```
unordered_set<int> us = {1, 2, 3};
us.insert(4);
us.erase(2);
```

🟥 map<K,V>

Description:
Sorted key-value pairs.
Header: `<map>`

```
map<string, int> mp;
mp["apple"] = 2;
mp["banana"] = 5;
for (auto &[k, v] : mp)
    cout << k << " " << v << "\n";
```

🟩 unordered_map<K,V>

Description:
Hash-based key-value pairs.
Header: `<unordered_map>`

```
unordered_map<string, int> ump;
ump["one"] = 1;
ump["two"] = 2;
ump.erase("two");
```

🟪 pair<T1, T2>

Description:
Stores two related values.
Header: `<utility>`

```
pair<int, string> p = {1, "apple"};
cout << p.first << " " << p.second;
```

🟨 `tuple<T1, T2, T3,...>`

Description:
Stores multiple related values.
Header: `<tuple>`

```
tuple<int, string, double> t = {1, "hi", 3.5};
cout << get<0>(t);
```

🟩 `bitset<N>`

Description:
Fixed-size binary array.
Header: `<bitset>`

```
bitset<8> b("1011");
b.count(); // number of 1s
b.flip(2);
b[0] = 0;
```

🟦 `string`

Description:
Dynamic character array.
Header: `<string>`

```
string s = "Krish";
s += " rocks";
cout << s.substr(1, 3);
reverse(s.begin(), s.end());
```

2️⃣ Iterators & Algorithms (`<algorithm>`)

Function	Description	Complexity
`sort(a,b)`	Sort ascending	O(n log n)

`reverse(a,b)`	Reverse range	O(n)
`count(a,b,x)`	Count occurrences	O(n)
`find(a,b,x)`	Iterator to element	O(n)
`min_element(a,b)`	Smallest element	O(n)
`max_element(a,b)`	Largest element	O(n)
`accumulate(a,b,init)`	Sum of range	O(n)
`unique(a,b)`	Remove duplicates	O(n)
`next_permutation(a,b)`	Next lexicographic order	O(n)
`lower_bound(a,b,x)`	First ≥ x	O(log n)
`upper_bound(a,b,x)`	First > x	O(log n)
`binary_search(a,b,x)`	Check existence	O(log n)

---

3 Numeric Utilities (`<numeric>`)

```
accumulate(v.begin(), v.end(), 0);              // sum
inner_product(a.begin(), a.end(), b.begin(), 0); // dot product
partial_sum(v.begin(), v.end(), pref.begin());
iota(v.begin(), v.end(), 1);                    // fill v = [1, 2, ...]
```

---

4 String Functions (`<string>`)

Function	Description
`stoi(s)`	Convert string to int
`stoll(s)`	Convert string to long long
`to_string(x)`	Convert number to string

`substr(l, len)`	Get substring
`find(sub)`	Find substring position
`erase(pos, len)`	Remove characters
`replace(l, len, s2)`	Replace substring

5️⃣ Math Utilities (`<cmath>`)

Function	Description
`abs(x)`	Absolute value
`pow(x,y)`	Power
`sqrt(x)`	Square root
`floor(x), ceil(x)`	Rounding
`log(x), log2(x), log10(x)`	Logarithms
`sin(x), cos(x), tan(x)`	Trig functions
`hypot(x,y)`	$\sqrt{(x^2+y^2)}$ safely
`__gcd(a,b)`	Built-in gcd

6️⃣ Example Program

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

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

    vector<int> v = {3, 1, 4, 1, 5, 9};
    sort(v.begin(), v.end());
    v.erase(unique(v.begin(), v.end()), v.end());
```

```
    cout << "Sorted Unique Elements: ";
    for (int x : v) cout << x << " ";
    cout << "\nMax: " << *max_element(v.begin(), v.end());
    cout << "\nSum: " << accumulate(v.begin(), v.end(), 0);
}
```

For checking how many coprimes pairs are there

```
n = int(input())
a = list(map(int, input().split()))
MAX = 10**6

cnt = [0]*(MAX+1)
for x in a:
    cnt[x] += 1

f = [0]*(MAX+1)
for i in range(1, MAX+1):
    for j in range(i, MAX+1, i):
        f[i] += cnt[j]

mu = [1]*(MAX+1)
isprime = [True]*(MAX+1)
for i in range(2, MAX+1):
    if isprime[i]:
        for j in range(i, MAX+1, i):
            isprime[j] = False
            mu[j] *= -1
        j = i*i
        while j <= MAX:
            mu[j] = 0
            j += i*i

ans = 0
for i in range(1, MAX+1):
    if f[i] >= 2:
        ans += mu[i] * f[i]*(f[i]-1)//2
print(ans)
```

4️⃣ Solve ax + by = c (Extended Euclid)

```
def egcd(a,b):
    if b==0:return a,1,0
    g,x1,y1=egcd(b,a%b)
    return g,y1,x1-(a//b)*y1
def solve(a,b,c):
    g,x,y=egcd(a,b)
    if c%g:return None
    return x*(c//g),y*(c//g)
```

7️⃣ Count Coprime Pairs in Range

```
def phi_sieve(n):
    phi=list(range(n+1))
    for i in range(2,n+1):
        if phi[i]==i:
            for j in range(i,n+1,i):
                phi[j]-=phi[j]//i
    for i in range(2,n+1):phi[i]+=phi[i-1]
    return phi
```

`phi_sieve(N) gives prefix φ → number of coprime pairs ≤ n ≈ φ_prefix[n].`

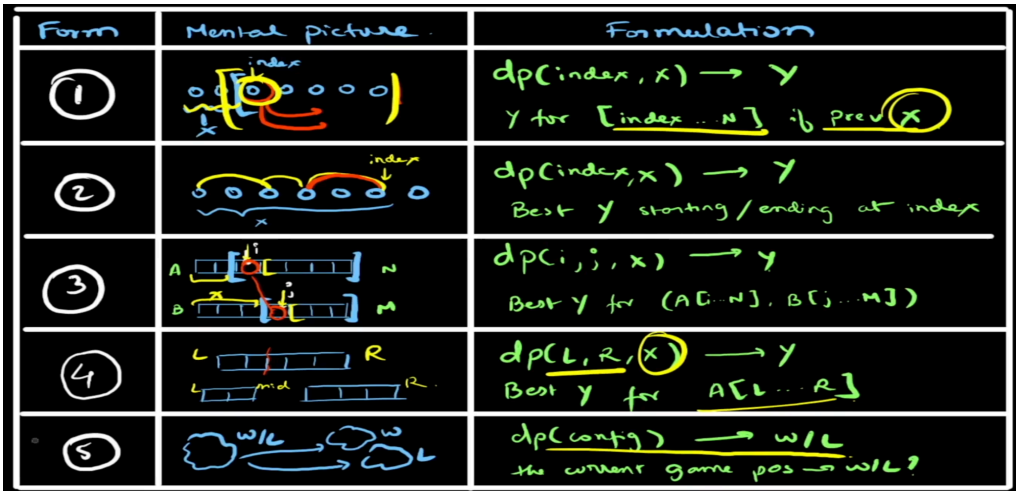
```
def sum_over_divisors(n):

    s=0

    i=1

    while i<=n:

        j=n//(n//i)

        s+=(n//i)*(j-i+1)

        i=j+1

    return s #  here s=summation of n//k for k in range 1 to n
```

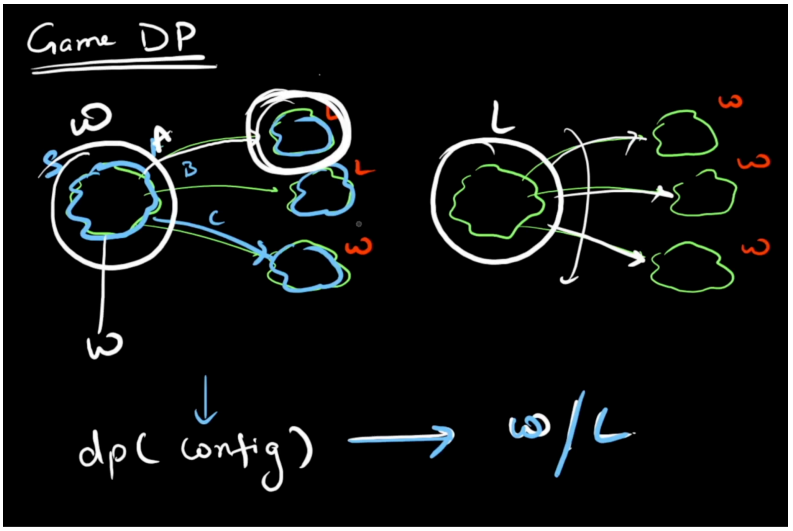
1. If s is a string of just a and b then if 1st and last element are equal then substring ‘ab’==‘ba’
2. If a number has odd divisors it means it is not a power of 2 .
3. If a number is prime it means in array if I want to find minimum effort to make product of an array divisible by number then I should only focus on making one no from number dib by that num.
4. For a length of len if we want to find no of subarrays of at least length k then ans=(len-k+1)*(len-k+2)//2
5. If I want to find two no a and b where a+b=n such that there gcd is maximum is n/2 and n/2 if n is even else n-x,x were x is highest multiple.
6. Always write n-m+1 not -m+1 in code it may give wrong solution.
7. int xor_1_to_n(int n) {
8. if (n % 4 == 0) return n;
9. if (n % 4 == 1) return 1;
10. if (n % 4 == 2) return n + 1;
11. return 0; // when n % 4 == 3
12. }
13. This is xor of n natural numbers.
14. If we want to check if a no is div by 2 do not use math.log(n,2) then int(x)==x it will give wa instead use n&(n-1)==0.

18.Once a Number is Not Divisible by 2^x, It's Never Divisible by Larger Powers Again.
19.a%m=b%m it means (a-b)is divisible by some m a>b.
If we want to find m such that a1%m==a2%m , a4%m==a3%m so on then we have to find gcd for all ai’s.

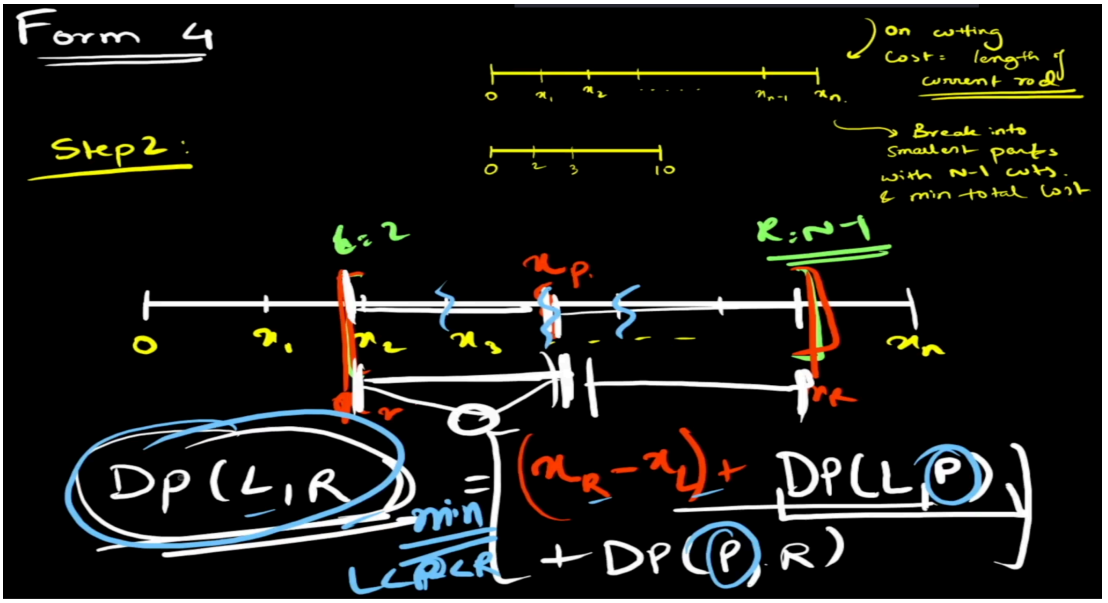
1. When in a question there is matrix then be careful of odd n.
2. Taking a[i] % 2^j is equivalent to: “look at the **last j bits** of a[I]”..

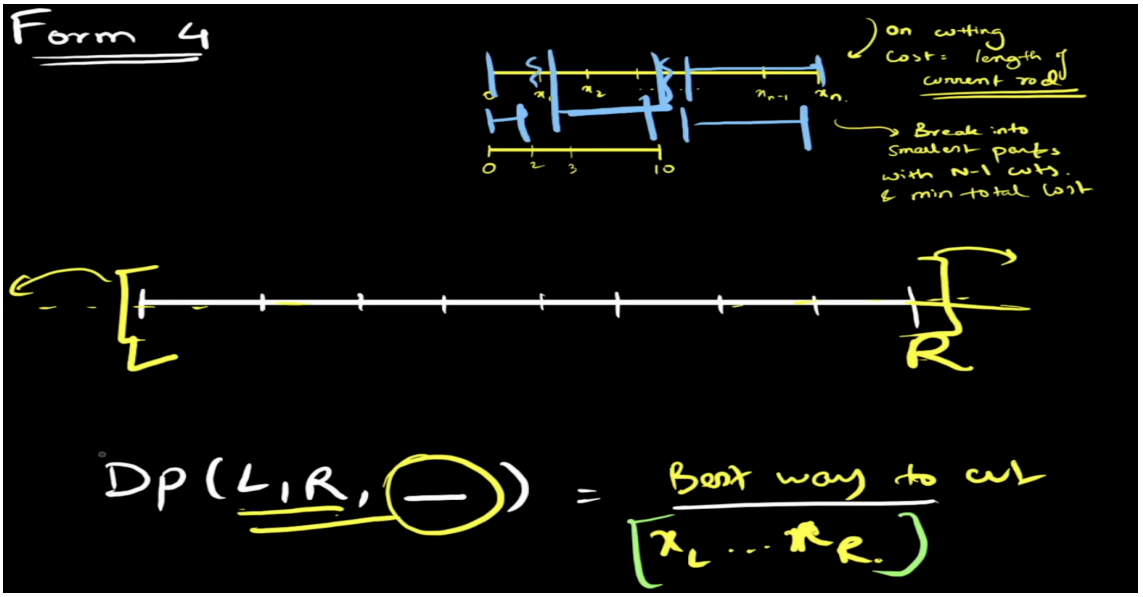
- 22. If I have to find k such that after taking module of every element of array the array should have 2 or 3 or some elements I have to check powers of two .



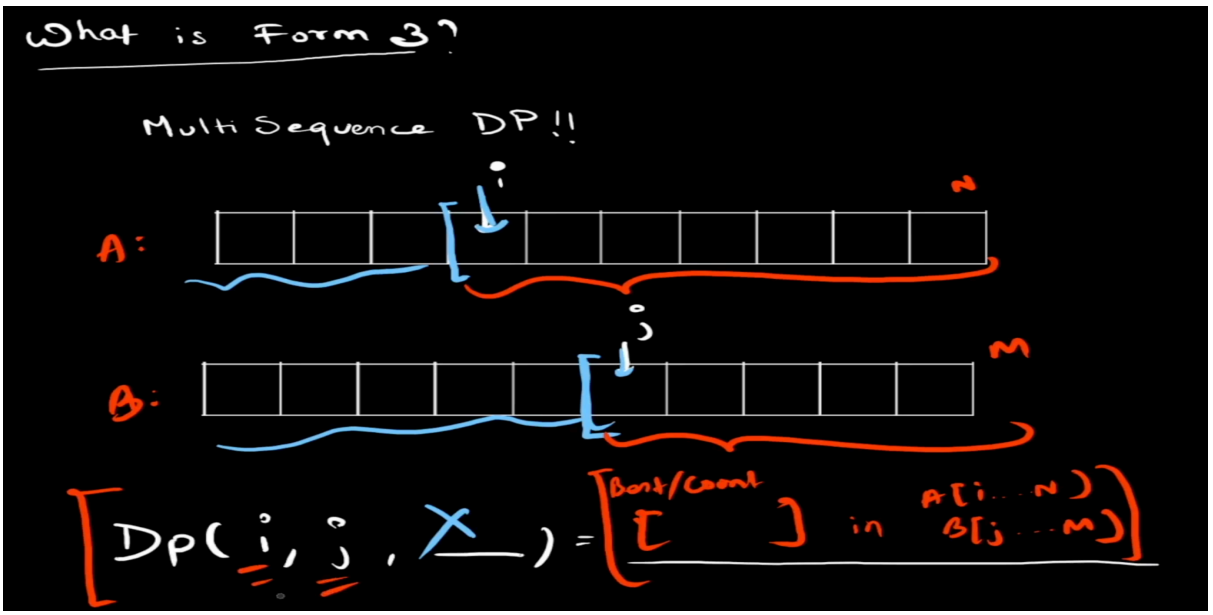



```
    // base case
    if(x==0){
        return 0;
    }
    //cache check
    if(dp[x]!=-1){
        return dp[x];
    }
    // compute
    int ans = 0;
    for(int m=0;(1<<m)<=x;m++){
        if(rec(x-(1<<m))==0){
            ans = 1;
            break;
        }
    }
    // save and return

    return dp[x] = ans;
}
```



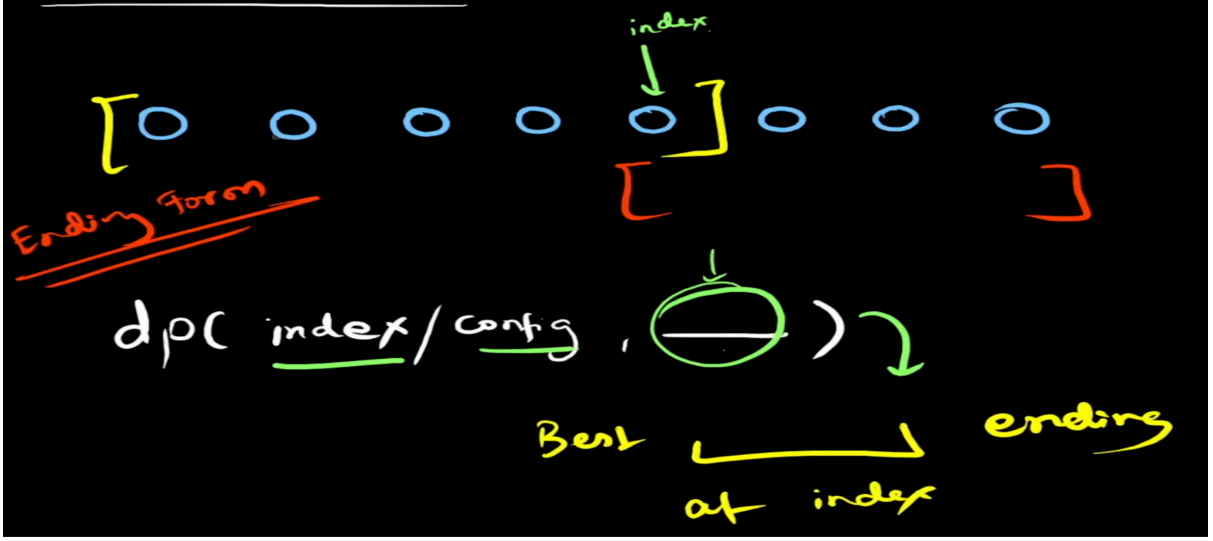



```cpp
int dp[1001][1001];

int rec(int l,int r){
    // pruning

    // basecase
    if(l+1==r){
        return 0;
    }
    // cache check
    if(dp[l][r]!=-1){
        return dp[l][r];
    }
    // compute
    int ans = 1e9;
    for(int p=l+1;p<=r-1;p++){
        ans = min(ans,x[r]-x[l]+rec(l,p)+rec(p,r));
    }
    // save and return
    return dp[l][r] = ans;
}
```

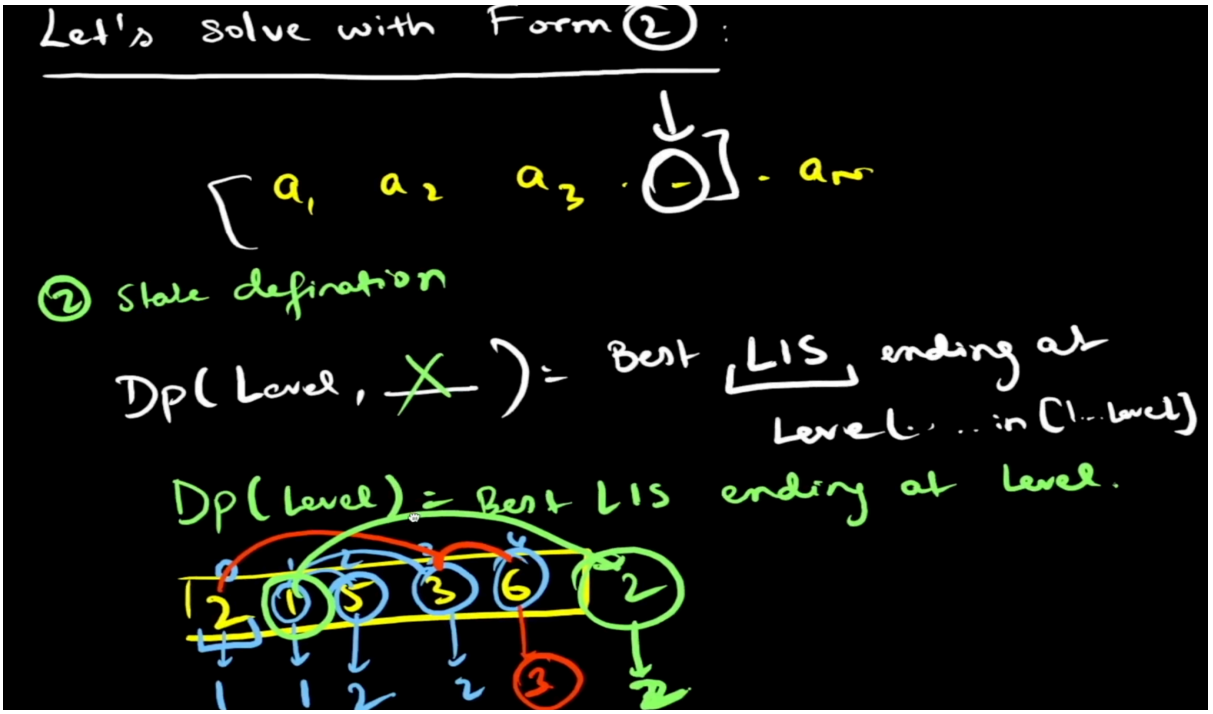


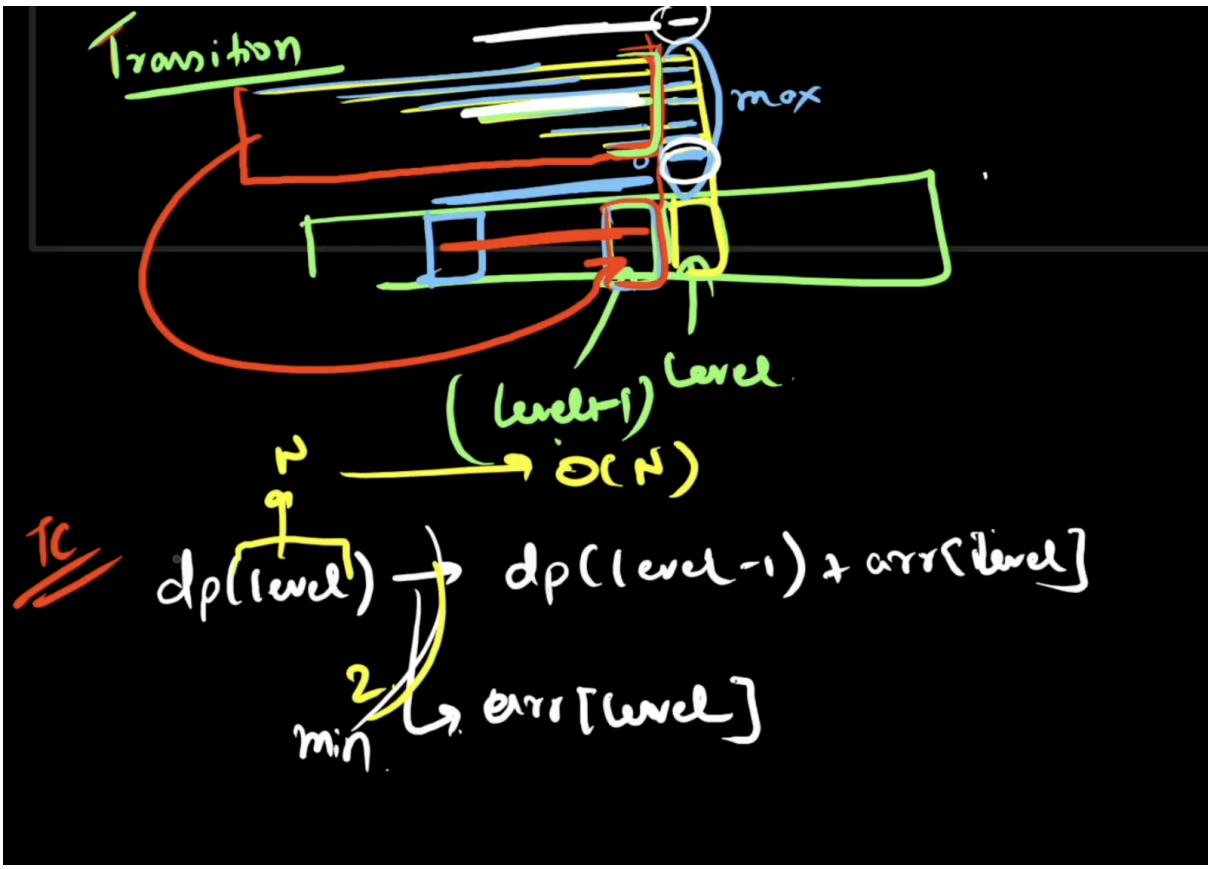

```
// return the LCS of a[i...n-1] and b[j...m-1].. c[k...x-
// pruning

// basecase
if(i>=n||j>=m||k>=x){
    return 0;
}
// save and compute
if(dp[i][j][k]!=-1){
    return dp[i][j][k];
}
// compute
int ans = 0;
ans = max(ans,rec(i+1,j,k));
ans = max(ans,rec(i,j+1,k));
ans = max(ans,rec(i,j,k+1));
if(a[i]==b[j]&&b[j]==c[k]){
    ans = max(ans,1+rec(i+1,j+1,k+1));
}
// save and return
return dp[i][j] = ans;
```

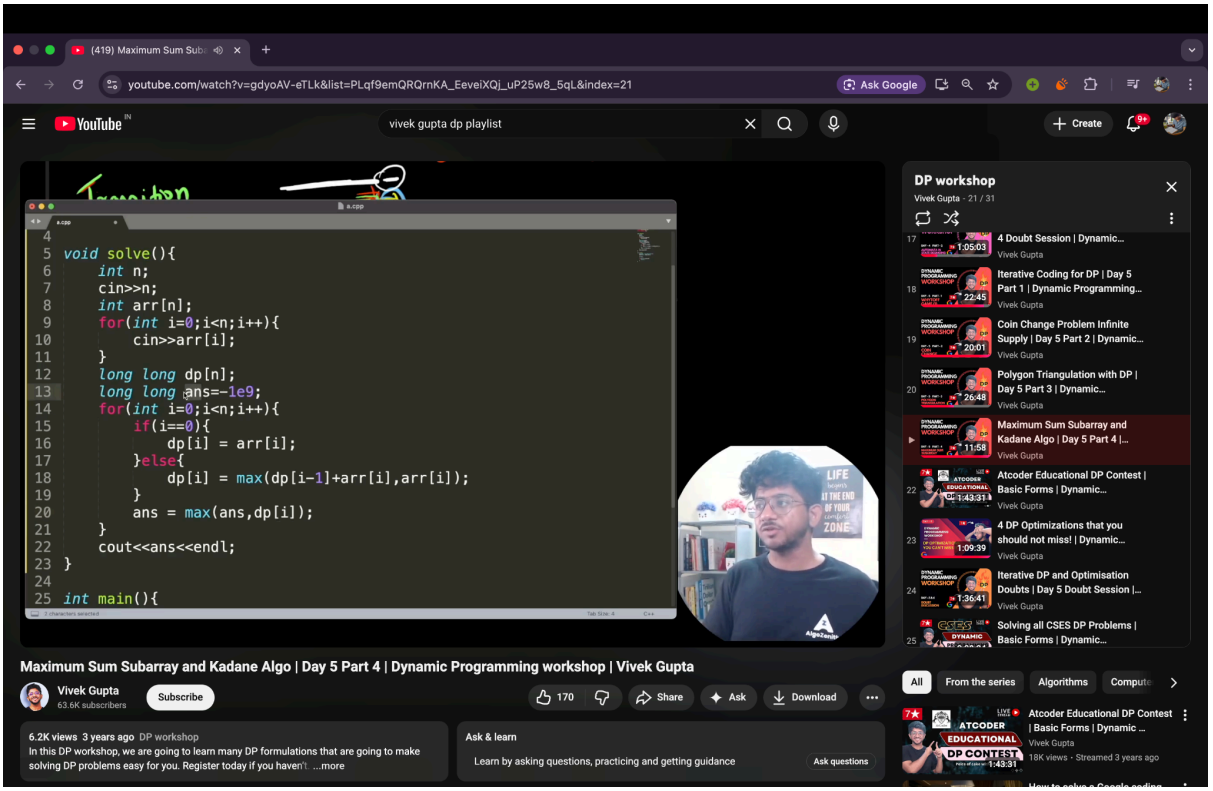
What is Form 2

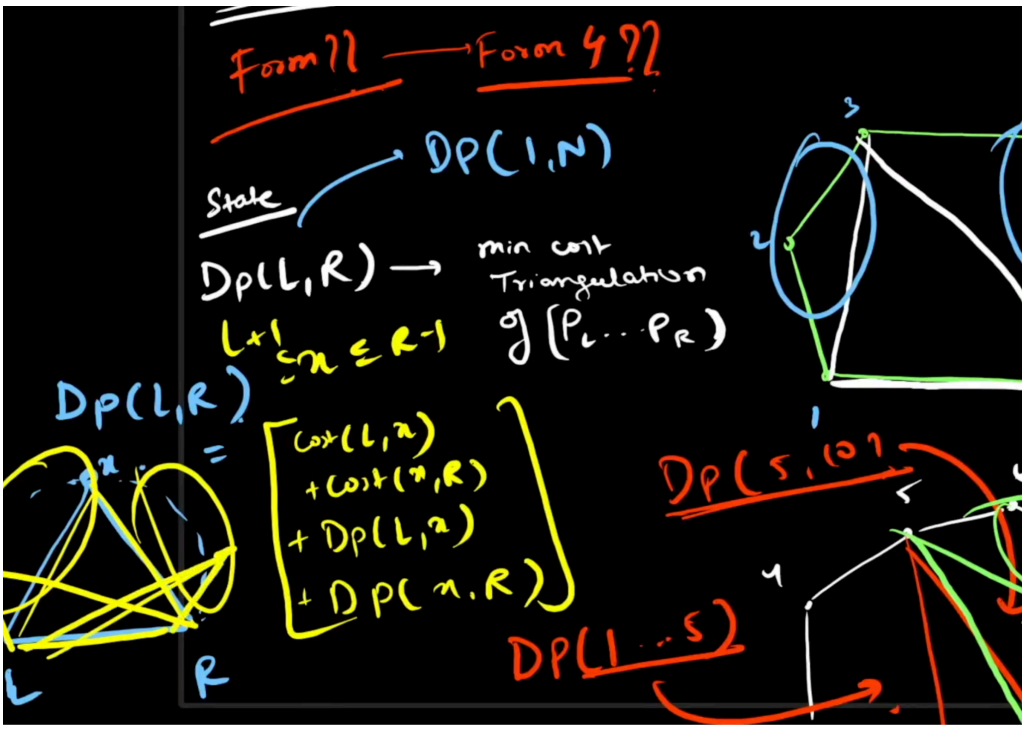


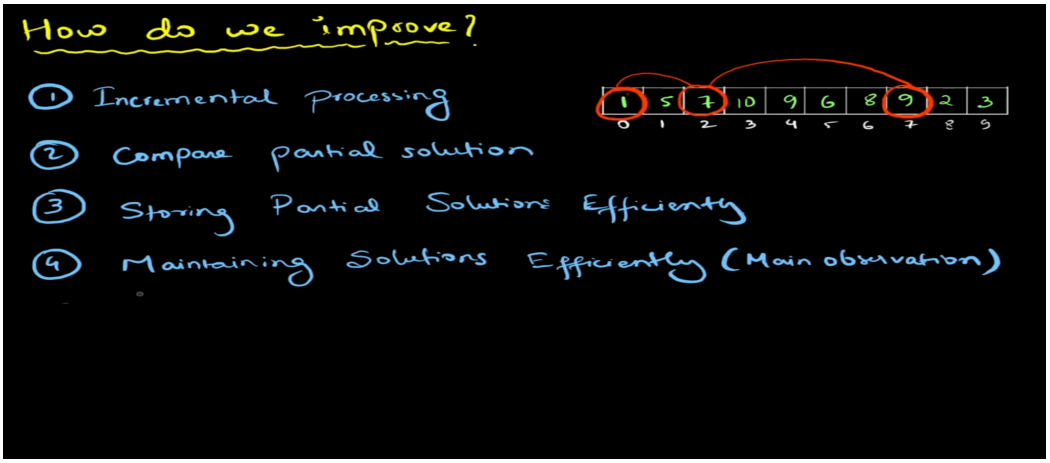
Let's solve with Form (2):




Transition
max
(level-1)
level
N
O(N)
TC
dp(level)
dp(level-1) + arr[level]
2
min
arr[level]

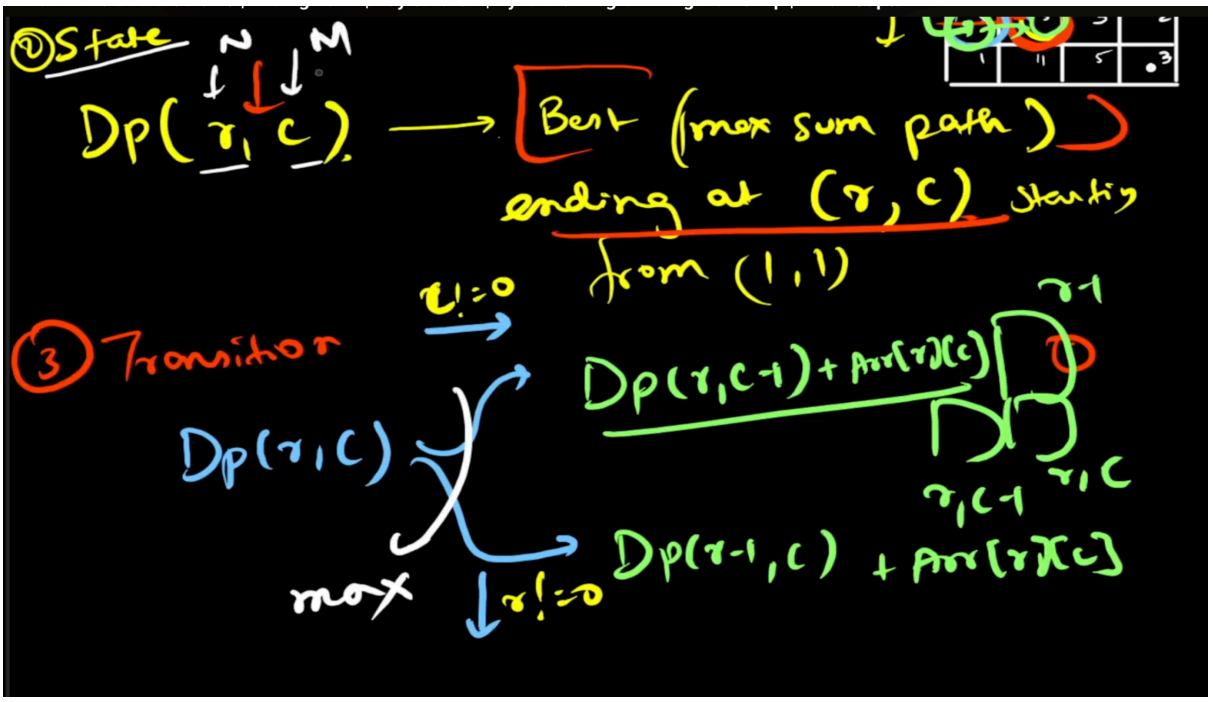

(419) Maximum Sum Sub
youtube.com/watch?v=gdyoAV-eTLk&list=PLqf9emQRQrnKA_EeveiXQj_uP25w8_5qL&index=21
Ask Google
YouTube
vivek gupta dp playlist
Create
Transition
a.cpp
void solve(){
int n;
cin>>n;
int arr[n];
for(int i=0;i<n;i++){
cin>>arr[i];
}
long long dp[n];
long long ans=-1e9;
for(int i=0;i<n;i++){
if(i==0){
dp[i] = arr[i];
}else{
dp[i] = max(dp[i-1]+arr[i],arr[i]);
}
ans = max(ans,dp[i]);
}
cout<<ans<<endl;
}
int main(){
DP workshop
Vivek Gupta - 21 / 31
4 Doubt Session | Dynamic...
Iterative Coding for DP | Day 5 Part 1 | Dynamic Programming...
Coin Change Problem Infinite Supply | Day 5 Part 2 | Dynamic...
Polygon Triangulation with DP | Day 5 Part 3 | Dynamic...
Maximum Sum Subarray and Kadane Algo | Day 5 Part 4 |...
Atcoder Educational DP Contest | Basic Forms | Dynamic...
4 DP Optimizations that you should not miss! | Dynamic...
Iterative DP and Optimisation Doubts | Day 5 Doubt Session |...
Solving all CSES DP Problems | Basic Forms | Dynamic...
Maximum Sum Subarray and Kadane Algo | Day 5 Part 4 | Dynamic Programming workshop | Vivek Gupta
Vivek Gupta
63.6K subscribers
Subscribe
170
Share
Ask
Download
6.2K views 3 years ago DP workshop
In this DP workshop, we are going to learn many DP formulations that are going to make solving DP problems easy for you. Register today if you haven't ...more
Ask & learn
Learn by asking questions, practicing and getting guidance
Ask questions
All
From the series
Algorithms
Atcoder Educational DP Contest | Basic Forms | Dynamic ...
Vivek Gupta
18K views • Streamed 3 years ago

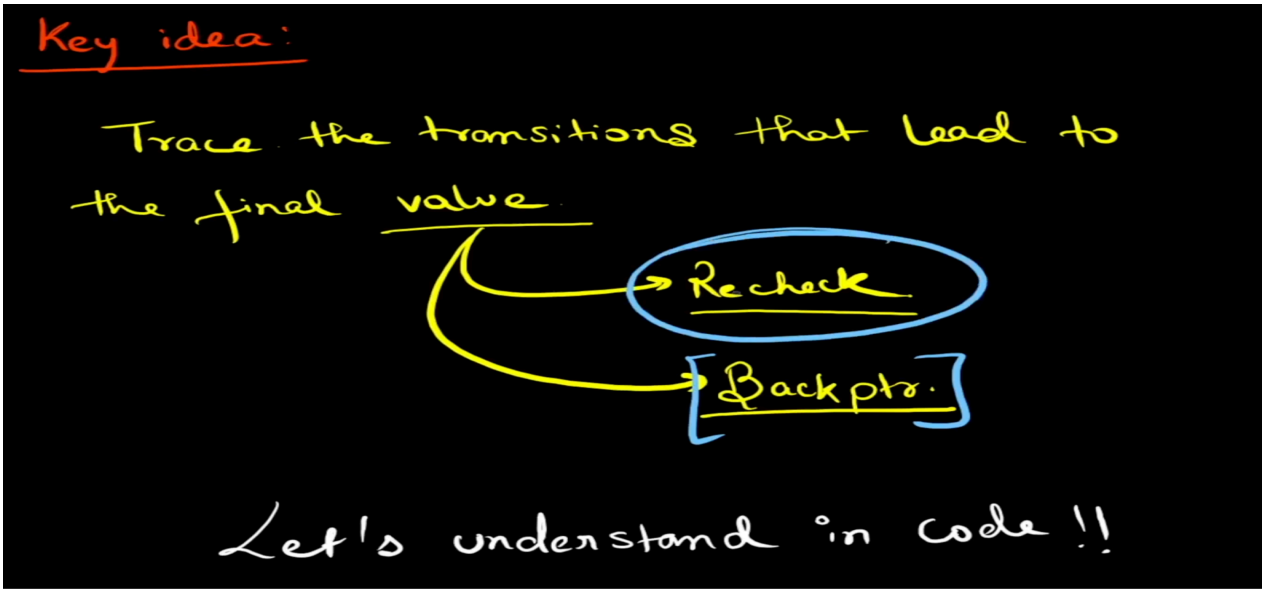




a.cpp

```
26      int ans = 0;
27      if(rec(level+1,left)==1){
28          ans = 1;
29      }else if(rec(rec+1,left-x[level])){
30          ans = 1;
31      }
32      // save and return
33      return dp[level][left]=ans;
34  }
35
36  void printset(int level,int left){
37      // base case
38      if(level==n+1){
39          return;
40      }
41      // find the correct transition
42      if(rec(level+1,left)==1){// don't take
43          printset(level+1,left);
44      }else if(rec(rec+1,left-x[level])){ //take
45          cout<<x[level]<<" ";
46          printset(rec+1,left-x[level])
47      }
48  }
49
```


①State
N
M
Dp(r, c)
Best (max sum path)
ending at (r, c) starting
from (1,1)
③ Transition
c!=0
Dp(r,c-1) + Arr[r][c]
Dp(r,c)
max
r!=0
Dp(r-1,c) + Arr[r][c]
r-1
r,c-1
r,c


Key idea:
Trace the transitions that lead to the final value
Recheck
Backptr.
Let's understand in code !!