Template of IUH.Ocean

Team

IUH.OCEAN

**ICPC Quốc gia năm 2022**

ICPC Vietnam National Programming Contest 2022

**Binary search**

ll l = 0, r = n;

ll find = 0;

while(r - l > 1)

{

ll mid = (l + r) / 2;

if(check(a[mid]))

{

find = 1;

break;

}

else if(a[mid] > q)

{

r = mid;

}

else l = mid;

}

if(check(a[l]) || check(a[r] == q)) find = 1;

int lower\_bound(int arr[], int N, int X)

{

ll mid;

ll l = 0;

ll r = N;

while (l < r) {

mid = l + (r - l) / 2;

if (X <= arr[mid]) {

r = mid;

}

else {

l = mid + 1;

}

}

if(l < N && arr[l] < X) {

l++;

}

return l;

}

int upper\_bound(int arr[], int N, int X)

{

int mid;

int l = 0;

int r = N;

while (l < r) {

mid = l + (r - l) / 2;

if (X >= arr[mid]) {

l = mid + 1;

}

else {

r = mid;

}

}

if(low < N && arr[low] <= X) {

low++;

}

return low;

}

int binary\_search() // Low

{

int l = 1, r = n;

while (l < r){

int mid = (l + r + 1) >> 1;

if (a[mid] > x){

r = mid - 1;

}

else{

l = mid;

}

}

if (a[l] == x) cout << "YES";

else cout << "NO";

}

// Dung if else de toi uu Binary search

int binarySearch(int arr[], int l, int r, int x) // de qui

{

if (r >= l) {

int mid = l + (r - l) / 2;

if (arr[mid] == x)

return mid;

if (arr[mid] > x)

return binarySearch(arr, l, mid - 1, x);

return binarySearch(arr, mid + 1, r, x);

}

return -1;

}

Tim x nho nhat thoa man

while(l <= r)

{

int mid = (l + r)/2;

if(a[mid] == find)

{

ans = mid;

break;

}

else if(a[mid] < find)

{

l = mid + 1;

}

else r = mid - 1;

cout << l << " " << r << endl;

}

**Sắp Xếp**

void CountingSort(vector<int>& arr)

{

int max = \*max\_element(arr.begin(), arr.end());

int min = \*min\_element(arr.begin(), arr.end());

int range = max - min + 1;

vector<int> count(range), out(arr.size());

for(int i=0; i<arr.size(); i++) count[arr[i] - min]++;

for(int i=1; i<count.size(); i++) count[i] += count[i-1];

for(int i=arr.size()-1; i>-1; i--)

{

out[count[arr[i] - min] - 1] = arr[i];

count[arr[i]-min]--;

}

for(int i=0; i<arr.size(); i++) arr[i] = out[i];

}

**Binull**

long long MOD = 1e9 + 7;

long long binmul(long long a, long long b)

{

long long res = 0;

while(b > 0)

{

if(b & 1) res = (res % MOD + a % MOD) % MOD;

a = (a % MOD + a % MOD) % MOD;

b /= 2;

}

return res % MOD;

}

**Binpow**

long long binpow(long long a, long long b)

{

long long res = 1;

while (b > 0)

{

if (b & 1)

res = res \* a;

a = a \* a;

b >>= 1;

}

return res;

}

//O(logn)

**Lũy thừa tránh tràn số**

int poww(int a, int b)

{

int ans = 1;

for(int i=1; i<=b; i++)

{

if(INF / a < ans) return INF;

ans \*= a;

}

return ans;

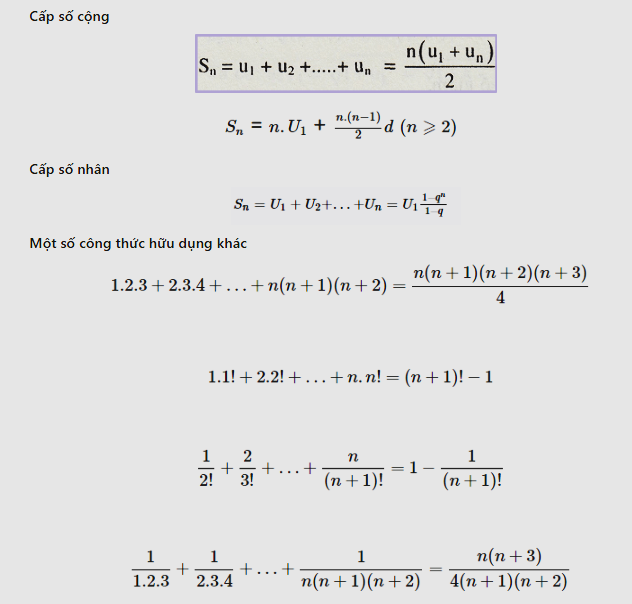
}

**Công thức tính tổng**

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Công thức CSC/CSN**



**Tính UCLN/BCNN**

// De quy

int gcd (int a, int b)

{

return b ? gcd (b, a % b) : a;

}

//Khu de quy

int gcd (int a, int b)

{

while (b) {

a %= b;

swap(a, b);

}

return a;

}

// O(log(min(a, b))

int lcm (int a, int b)

{

return a / gcd(a, b) \* b;

}

**Sàng nguyên tố & Số nguyên tố**

Số các số nguyên tố từ 1 đến n là n / ln(n)

**Check số nguyên tố**

bool isPrime[N+5];

void Sieve()

{

for(int i = 0; i <= N;++i)

{

isPrime[i] = true;

}

isPrime[0] = false;

isPrime[1] = false;

for(int i = 2; i \* i <= N; ++i)

{

if(isPrime[i] == true)

{

for(int j = i \* i; j <= N; j += i)

isPrime[j] = false;

}

}

}

**Phân tích một số ra ước nhỏ nhất**

long long isprime[N+5];

void Sieve\_gcd(){

isprime[0] = 0;

isprime[1] = 1;

for(int i = 2; i\*i<N; i++)

{

if(isprime[i] == 0)

{

for(int j = i\*i; j <=N; j+=i)

{

if(isprime[j] == 0)

{

isprime[j] = i;

}

}

}

}

for (int i = 2; i < N; ++i)

{

if (isprime[i] == 0)

{

isprime[i] = i;

}

}

}

**Sàng để phân tích thừa số nguyên tố**

long long primeDiv[N+5];

void Sieve\_Fac()

{

for(int i=2; i \* i <= N; i++)

{

if(primeDiv[i] == 0)

{

for(int j=i\*i; j<=N; j+=i)

{

primeDiv[j] = i;

}

}

}

for(int i=2; i<=N; i++)

{

if(primeDiv[i] == 0) primeDiv[i] = i;

}

}

**Phân tích một số ra thừa số nguyên tố**

//map<int, int> Fac;

//vector<int> Fac(N);

while(x > 1)

{

Fac[primeDiv[x]]++;

x /= primeDiv[x];

}

**Kiểm tra số nguyên tố cực lớn**

vector<long long> snt;

// Chua cac so nguyen to sau khi chay sang nguyen to truoc do

bool Ok\_prime(long long x)

{

for(auto it : snt)

{

if(1ll \* it \* it > x) return true;

if(x % it == 0) return false;

}

assert(false);

}

**Sàng nguyên tố trên đoạn**

long long L, R; cin >> L >> R;

long long cnt = 0;

vector<bool> isPrime(R - L + 1, true);

// x là số nguyên tố khi và chỉ khi isPrime[x - l] == true

for (long long i = 2; i \* i <= R; ++i) {

for (long long j = max(i \* i, (L + i - 1) / i \* i); j <= R; j += i) {

isPrime[j - L] = false;

}

}

if (1 >= L) { // Xét riêng trường hợp số 1

isPrime[1 - L] = false;

}

**Ước số**

**Đếm ước của một số**

long long countDiv(long long n)

{

long long cnt = 0;

for(int i=1; 1ll\*i\*i <= n; i++)

{

if(n % i == 0)

{

if(n / i == i) cnt++;

else cnt += 2;

}

}

return cnt;

}

**Lấy các ước của một số**

vector<int> Div;

void countDiv(long long n)

{

for(int i=1; 1ll\*i\*i <= n; i++)

{

if(n % i == 0)

{

if(n / i == i) Div.pb(i);

else

{

Div.pb(i);

Div.pb(n/i);

}

}

}

}

**Đếm ước nâng cao cho nhiều truy vấn**

long long cntDiv[N+5];

for(int i = 1; i\*i <= N; i++)

for(int j = i\*i; j <= N; j+=i)

cntDiv[j]++;

for(int i = 1; i <= N; i++) cntDiv[i]\*=2;

for(int i = 1; i\*i <= N; i++) cntDiv[i\*i]--;

**Xử lý sai số SQRT**

long long getSqrt(unsigned long long x)

{

long long temp = sqrt(x);

for(int i=-3; i<=3; i++)

{

if(temp + i >= 0 && (temp + i)\*(temp + i) == x)

{

return temp + i;

}

}

return -1;

}

**Nhân ma trận**

#define R1 2 // number of rows in Matrix-1

#define C1 2 // number of columns in Matrix-1

#define R2 2 // number of rows in Matrix-2

#define C2 2 // number of columns in Matrix-2

void mulMat(int mat1[][C1], int mat2[][C2])

{

int rslt[R1][C2];

cout << "Multiplication of given two matrices is:\n";

for (int i = 0; i < R1; i++)

{

for (int j = 0; j < C2; j++)

{

rslt[i][j] = 0;

for (int k = 0; k < R2; k++)

{

rslt[i][j] += mat1[i][k] \* mat2[k][j];

}

cout << rslt[i][j] << "\t";

}

cout << endl;

}

}

int main()

{

Hello\_i\_am\_Salmon

// R1 = 4, C1 = 4 and R2 = 4, C2 = 4 (Update these

// values in MACROs)

int mat1[R1][C1] = { { 1, 1 },

{ 2, 2 } };

int mat2[R2][C2] = { { 1, 1 },

{ 2, 2 } };

mulMat(mat1, mat2);

return 0;

}

**Cấu trúc dữ liệu**

**Vector**

// Khai bao

#include <vector>

vector<int> a;

vector<pair<int, int>> b;

vector<int> c = {1, 2, 3, 5};

vector<int> d(10);

vector<int> e(10, 0);

//Thao tac co ban

a.push\_back(1);

a.pop\_back(1);

a.insert(a.begin()+1, 2);

a.insert(a.begin()+1, 10, 3); // chen nhieu

a.erase(a.begin()+2);

a.back();

a.front();

a.clear();

a.empty();

a.size();

a.reverse();

a.resize();

**queue**

//Khai bao

#include <queue>

queue<int> Q;

queue<int> Q(10, 1);

//Thao tac

Q.push(1);

Q.size();

Q.empty();

Q.front();

Q.back();

Q.pop();

**Deque**

// Khai bao

#include <vector>

deque<int> a;

deque<pair<int, int>> b;

deque<int> c = {1, 2, 3, 5};

deque<int> d(10);

deque<int> e(10, 0);

//Thao tac co ban

a.push\_back(1);

a.push\_front();

a.pop\_back(1);

a.pop\_front();

a.insert(a.begin()+1, 2);

a.insert(a.begin()+1, 10, 3); // chen nhieu

a.erase(a.begin()+2);

a.back();

a.front();

a.clear();

a.empty();

a.size();

a.reverse();

a.resize();

**priority\_queue**

#include <queue>

priority\_queue<int> Q;

priority\_queue<int> Q(10, 1);

priority\_queue<int, vector<int>, greater<int>> Q;

//Thao tac

Q.push(1);

Q.size();

Q.empty();

Q.top()

Q.pop();

**Set**

//Khai bao

#include<set>

set<int> st;

//Thao tac

st.insert(1);

st.begin();

st.end();

st.clear();

st.count(1);

st.empty();

st.erase();

st.find();

st.lower\_bound();

st.size();

st.upper\_bound();

**map**

//Khai bao

#include <map>

map<int, int> mp;

//Thao tac

mp.begin();

mp.end();

mp.count();

mp.clear();

mp.empty();

mp.erase();

mp.find();

mp.insert();

mp.lower\_bound();

mp.size();

mp.upper\_bound();

**multiset**

// Khai bao

#include<set>

multiset<int> mst;

//Thao tac

mst.clear();

mst.count();

mst.begin();

mst.end();

mst.empty();

mst.erase();

mst.find();

mst.insert();

mst.lower\_bound();

mst.size();

mst.upper\_bound();

**unordered\_map**

//Khai bao

#include <unordered\_map>

unordered\_map<int, int> mp;

//Thao tac

mp.begin();

mp.end();

mp.count();

mp.clear();

mp.empty();

mp.erase();

mp.find();

mp.insert();

mp.size();

mp.reserve();

**Order set**

#include <iostream>

using namespace std;

#include <ext/pb\_ds/assoc\_container.hpp>

#include <ext/pb\_ds/tree\_policy.hpp>

using namespace \_\_gnu\_pbds;

#define ordered\_set tree<int, null\_type,less<int>, rb\_tree\_tag,tree\_order\_statistics\_node\_update>

o\_set.insert(5);

o\_set.insert(1);

o\_set.insert(2);

cout << \*(o\_set.find\_by\_order(1))

<< endl;

cout << o\_set.order\_of\_key(4)

<< endl;

cout << o\_set.order\_of\_key(5)

<< endl;

if (o\_set.find(2) != o\_set.end())

o\_set.erase(o\_set.find(2));

cout << \*(o\_set.find\_by\_order(1))

<< endl;

cout << o\_set.order\_of\_key(4)

<< endl;

return 0;

}

**Segment tree**

#include <bits/stdc++.h>

using namespace std;

const int inf = 1e9 + 7;

const int maxN = 1e5 + 7;

int n, q;

int a[maxN];

int st[4 \* maxN]; // Lí do sử dụng kích thước mảng là 4 \* maxN sẽ được giải thích ở phần sau

// Thủ tục xây dựng cây phân đoạn

void **build**(int id, int l, int r) {

// Đoạn chỉ gồm 1 phần tử, không có nút con

if (l == r) {

st[id] = a[l];

return;

}

// Gọi đệ quy để xử lý các nút con của nút id

int mid = l + r >> 1; // (l + r) / 2

build(2 \* id, l, mid);

build(2 \* id + 1, mid + 1, r);

// Cập nhật lại giá trị min của đoạn [l, r] theo 2 nút con

st[id] = min(st[2 \* id], st[2 \* id + 1]);

}

// Thủ tục cập nhật

void **update**(int id, int l, int r, int i, int val) {

// i nằm ngoài đoạn [l, r], ta bỏ qua nút id

if (l > i || r < i) return;

// Đoạn chỉ gồm 1 phần tử, không có nút con

if (l == r) {

st[id] = val;

return;

}

// Gọi đệ quy để xử lý các nút con của nút id

int mid = l + r >> 1; // (l + r) / 2

update(2 \* id, l, mid, i, val);

update(2 \* id + 1, mid + 1, r, i, val);

// Cập nhật lại giá trị min của đoạn [l, r] theo 2 nút con

st[id] = min(st[2 \* id], st[2 \* id + 1]);

}

// Hàm lấy giá trị

int **get**(int id, int l, int r, int u, int v) {

// Đoạn [u, v] không giao với đoạn [l, r], ta bỏ qua đoạn này

if (l > v || r < u) return inf;

/\* Đoạn [l, r] nằm hoàn toàn trong đoạn [u, v] mà ta đang truy vấn,

ta trả lại thông tin lưu ở nút id \*/

if (l >= u && r <= v) return st[id];

// Gọi đệ quy với các nút con của nút id

int mid = l + r >> 1; // (l + r) / 2

int get1 = get(2 \* id, l, mid, u, v);

int get2 = get(2 \* id + 1, mid + 1, r, u, v);

// Trả ra giá trị nhỏ nhất theo 2 nút con

return min(get1, get2);

}

int main() {

cin >> n;

for (int i = 1; i <= n; ++i) cin >> a[i];

build(1, 1, n);

cin >> q;

while (q--) {

int type, x, y;

cin >> type >> x >> y;

if (type == 1) update(1, 1, n, x, y); // Gán giá trị y cho phần tử ở vị trí x

else cout << get(1, 1, n, x, y) << '\n'; // In ra giá trị nhỏ nhất trong đoạn [x, y]

}

}

**Segment tree với multiset**

#include <bits/stdc++.h>

using namespace std;

const int inf = 1e9 + 7;

const int maxN = 1e5 + 7;

int n, m;

int a[maxN];

multiset <int> st[4 \* maxN];

void build(int id, int l, int r) {

if (l == r) {

st[id].insert(a[l]);

return;

}

int mid = l + r >> 1;

build(2 \* id, l, mid);

build(2 \* id + 1, mid + 1, r);

st[id] = st[2 \* id + 1];

for (auto x : st[2 \* id]) st[id].insert(x);

}

void update(int id, int l, int r, int i, int old, int val) {

if (l > i || r < i) return;

if (l == r) {

st[id].clear();

st[id].insert(val);

return;

}

int mid = l + r >> 1;

update(2 \* id, l, mid, i, old, val);

update(2 \* id + 1, mid + 1, r, i, old, val);

st[id].erase(st[id].find(old));

st[id].insert(val);

}

int get(int id, int l, int r, int u, int v, int k) {

if (l > v || r < u) return inf;

if (l >= u && r <= v) {

auto it = st[id].lower\_bound(k);

if (it == st[id].end()) return inf;

return \*it;

}

int mid = l + r >> 1;

int get1 = get(2 \* id, l, mid, u, v, k);

int get2 = get(2 \* id + 1, mid + 1, r, u, v, k);

return min(get1, get2);

}

int main() {

cin >> n >> m;

for (int i = 1; i <= n; ++i) cin >> a[i];

build(1, 1, n);

while (m--){

int type, l, r, k;

cin >> type;

if (type == 1) {

cin >> l >> k;

update(1, 1, n, l, a[l], k);

a[l] = k;

}

else {

cin >> l >> r >> k;

int ans = get(1, 1, n, l, r, k);

cout << ((ans == inf) ? -1 : ans) << '\n';

}

}

}

**BFS**

ll moveX[] = {0, 1, -1, 0, 1, -1, -1, 1};

ll moveY[] = {1, 0, 0, -1, 1, -1, 1, -1};

bool vis[Nn][Nn];

char a[Nn][Nn];

ll n, m;

bool check\_id(ll x, ll y)

{

return (x >= 0 && x < n && y >= 0 && y < m);

}

vector<pair<ll, ll>> lo;

void bfs(ll sx, ll sy)

{

queue<pair<ll, ll>> Q;

vis[sx][sy] = true;

Q.push({sx, sy});

while(!Q.empty())

{

ll x = Q.front().fi;

ll y = Q.front().se;

lo.pb({x, y});

// fix(sz(lo));

Q.pop();

for(int i=0; i<8; i++)

{

ll u = x + moveX[i];

ll v = y + moveY[i];

if(!check\_id(u, v))

{

// show2

continue;

}

if(a[u][v] == '\*' && !vis[u][v])

{

// show1

vis[u][v] = true;

Q.push({u, v});

// a[u][v] = '.';

}

}

}

}

int main()

{

Hello\_i\_am\_Salmon

int t; cin >> t;

while(t--)

{

cin >> n >> m;

for(int i=0; i<n; i++)

{

for(int j=0; j<m; j++)

{

cin >> a[i][j];

}

}

ll endgame = 1;

for(int i=0; i<n; i++)

{

for(int j=0; j<m; j++)

{

if(a[i][j] == '\*' && !vis[i][j])

{

bfs(i, j);

if(sz(lo) != 3) endgame = 0;

else

{

sort(all(lo));

ll x1 = lo[0].fi;

ll y1 = lo[0].se;

ll x2 = lo[1].fi;

ll y2 = lo[1].se;

ll x3 = lo[2].fi;

ll y3 = lo[2].se;

ll f = 0;

if(x2 == x1+1 && y2 == y1 && x3 == x1+1 && y3 == y1+1) f = 1;

if(x2 == x1+1 && y2 == y1-1 && x3 == x1+1 && y3 == y1) f = 1;

if(x2 == x1 && y2 == y1+1 && x3 == x1+1 && y3 == y1+1) f = 1;

if(x2 == x1 && y2 == y1+1 && x3 == x1+1 && y3 == y1) f = 1;

if(!f) endgame = 0;

}

lo.clear();

}

}

}

cout << (endgame ? "YES" : "NO") << ed;

for(int i=0; i<n; i++)

{

for(int j=0; j<m; j++)

{

vis[i][j] = 0;

}

}

}

}

**DFS**

int timeDfs = 0; // Thứ tự duyệt DFS

void dfs(int u, int pre) {

num[u] = low[u] = ++timeDfs;

for (int v : g[u]){

if (v == pre) continue;

if (!num[v]) {

dfs(v, u);

low[u] = min(low[u], low[v]);

}

else low[u] = min(low[u], num[v]);

}

tail[u] = timeDfs;

}

**Dijkstra**

vector<pair<ll, ll>> adj[N];

ll pre[N];

ll n, m;

void dijkstra(ll s)

{

vector<ll> d(n+1, INF); // mang luu khoang cach duong di

d[s] = 0;

priority\_queue<pair<ll, ll>, vector<pair<ll, ll>>, greater<pair<ll, ll>>> Q;

// {khoang cach, dinh}

Q.push({0, s});

while(!Q.empty())

{

// show1

pair<ll, ll> temp = Q.top(); Q.pop();

ll u = temp.se, kc = temp.fi;

if(kc > d[u]) continue;

for(auto it : adj[u])

{

ll v = it.fi, w = it.se;

if(d[v] > d[u] + w)

{

d[v] = d[u] + w;

Q.push({d[v], v});

pre[v] = u;

}

}

}

vector<ll> path;

ll endd = n;

if(d[endd] == INF) cout << -1 << ed;

else

{

while(1)

{

path.pb(endd);

if(endd == 1) break;

endd = pre[endd];

}

reverse(all(path));

for(auto it : path) cout << it << \_;

cout << ed;

}

}

int main()

{

Hello\_i\_am\_Salmon

cin >> n >> m;

for(int i=0; i<m; i++)

{

ll x, y, z; cin >> x >> y >> z;

adj[x].pb({y, z});

adj[y].pb({x, z});

}

dijkstra(1);

}

**DSU**

ll parent[maxN];

ll sz[maxN];

void make\_set(ll v)

{

parent[v] = v;

sz[v] = 1;

}

ll find\_set(ll v)

{

if(v == parent[v]) return v;

ll p = find\_set(parent[v]);

parent[v] = p;

return p;

}

void union\_sets(ll a, ll b)

{

a = find\_set(a);

b = find\_set(b);

if(a != b)

{

if(sz[a] < sz[b]) swap(a, b);

parent[b] = a;

sz[a] += sz[b];

}

}

int main()

{

Hello\_i\_am\_Salmon

int t; cin >> t;

while(t--)

{

ll n; cin >> n;

ll a[n+1];

for(int i=1; i<=n; i++) cin >> a[i];

for(int i=1; i<=n; i++)

{

make\_set(i);

}

for(int i=1; i<=n; i++)

{

union\_sets(i, a[i]);

}

for(int i=1; i<=n; i++)

{

find\_set(i);

}

cout << endl;

for(int i=1; i<=n; i++)

{

cout << sz[parent[i]] << " ";

}

cout << endl;

}

}

**DSU lưu sum min max**

void make\_set(int v) {

parent[v] = v;

sz[v] = 1;

mn[v] = value[v];

sum[v] = value[v];

// value[v] là giá trị của phần tử thứ v

}

int find\_set(int v) {

return v == parent[v] ? v : parent[v] = find\_set(parent[v]);

}

void union\_sets(int a, int b) {

a = find\_set(a);

b = find\_set(b);

if (a != b) {

if (sz[a] < sz[b]) swap(a, b);

parent[b] = a;

sz[a] += sz[b];

sum[a] += sum[b];

mn[a] = min(mn[a], mn[b]);

}

}

int find\_sum(int v) { // Trả về tổng của các phần tử trong tập hợp chứa v

v = find\_set(v);

return sum[v];

}

int find\_min(int v) { // Trả về giá trị bé nhất của các phần tử trong tập hợp chứa v

v = find\_set(v);

return mn[v];

}

**Quy hoạch động**

**Dãy con có tổng lớn nhất**

max\_so\_far = a[0];

curr\_max = a[0];

for (int i = 1; i < n-1; i++)

{

curr\_max = max(a[i], curr\_max+a[i]);

max\_so\_far = max(max\_so\_far, curr\_max);

}

**LIS**

long long n; cin >> n;

long long a[n];

for(int i=0; i<n; i++) cin >> a[i];

long long check[n+1];

// for(int i=0; i<n; i++)

// {

// check[i] = INF;

// }

memset(check, 0x3f, sizeof(check));

for(int i=0; i<n; i++)

{

int id = lower\_bound(check, check+n, a[i]) - check;

check[id] = min(check[id], a[i]);

}

int sz = 0;

for(int i=0; i<n; i++)

{

if(check[i] < INF) sz++;

else break;

}

cout << sz << ed;

**Tính tổng các mảng con**

const int MOD = 1e9 + 7;

int SubArraySum(int arr[] , int n)

{

int result = 0;

for (int i=0; i<n; i++)

result += (arr[i]%MOD \* (i+1)%MOD \* (n-i)%MOD) %MOD;

return result % MOD ;

}

**Cái túi dởm**

ll dp[1010][100010];

int main()

{

Hello\_i\_am\_Salmon

ll n, x; cin >> n >> x;

ll coin[n+1];

ll book[n+1];

for(int i=1; i<=n; i++) cin >> coin[i];

for(int i=1; i<=n; i++) cin >> book[i];

for(int i=1; i<=n; i++)

{

for(int j=1; j<=x; j++)

{

if(j >= coin[i])

dp[i][j] = max(dp[i-1][j], dp[i-1][j-coin[i]] + book[i]);

else dp[i][j] = dp[i-1][j];

}

}

cout << dp[n][x] << ed;

**Cái túi xịn knapsack**

ll dp[1010][100010];

int main()

{

Hello\_i\_am\_Salmon

int n, x;

cin >> n >> x;

int arr[n];

int brr[n];

for(int i=0; i<n; i++) cin >> arr[i];

for(int i=0; i<n; i++) cin >> brr[i];

vector<int> dp(x+2);

for(int i=0; i<n; i++) {

for(int j=x; j-arr[i]>=0; j--) {

dp[j] = max(dp[j], dp[j-arr[i]]+brr[i]);

}

}

cout<<dp[x]<<'\n';

}

**KMP**

void KMP(string text,string pattern){

int n = text.length(), m = pattern.length(), F[m+2],i,j;

F[0] = F[1] = 0;

for(int i = 2;i<=m;i++){

j = F[i-1];

while(true){

if(pattern[j] == pattern[i-1]) { F[i] = j+1; break;}

else if(j==0) {F[i] = 0; break;}

else j = F[j];

}

}

i = j = 0;

while(j<n){

if(text[j] == pattern[i]){

i++; j++;

if(i==m) printf("%d ",j-i+1);

}

else if(i>0) i = F[i];

else j++;

}

}

int main(){

string a,b;

cin>>a>>b;

KMP(a,b);

// getch();

}

**Convex hull**

struct pt { // Kiểu điểm

double x, y;

};

bool cmp (pt a, pt b)

{ // So sánh theo tọa độ x, trong trường hợp bằng nhau so sánh theo y

return a.x < b.x || a.x == b.x && a.y < b.y;

}

bool cw (pt a, pt b, pt c) { // a -> b -> c đi theo thứ tự xuôi chiều kim đồng hồ

return a.x\*(b.y-c.y)+b.x\*(c.y-a.y)+c.x\*(a.y-b.y) < 0;

}

bool ccw (pt a, pt b, pt c) { // a -> b -> c đi theo thứ tự ngược chiều kim đồng hồ

return a.x\*(b.y-c.y)+b.x\*(c.y-a.y)+c.x\*(a.y-b.y) > 0;

}

void convex\_hull (vector<pt> & a) {

if (a.size() == 1) { // chỉ có 1 điểm

return;

}

// Sắp xếp các điểm theo tọa độ x, nếu bằng nhau sắp xếp theo y

sort (a.begin(), a.end(), &cmp);

pt p1 = a[0], p2 = a.back();

vector<pt> up, down; // chuỗi trên và chuỗi dưới

up.push\_back (p1);

down.push\_back (p1);

for (size\_t i=1; i<a.size(); ++i) { // xét lần lượt các điểm

// Thêm vào chuỗi trên

if (i==a.size()-1 || cw (p1, a[i], p2)) {

while (up.size()>=2 && !cw (up[up.size()-2], up[up.size()-1], a[i]))

up.pop\_back();

up.push\_back (a[i]);

}

// Thêm vào chuỗi dưới

if (i==a.size()-1 || ccw (p1, a[i], p2)) {

while (down.size()>=2 && !ccw (down[down.size()-2], down[down.size()-1], a[i]))

down.pop\_back();

down.push\_back (a[i]);

}

}

// Gộp 2 chuỗi trên và dưới để lấy bao lồi

a.clear();

for (size\_t i=0; i<up.size(); ++i)

a.push\_back (up[i]);

for (size\_t i=down.size()-2; i>0; --i)

a.push\_back (down[i]);

}

**Sinh bit**

string sinhbit(string x)

{

for(int i=sz(x); i>=0; i--)

{

if(x[i] == '0')

{

x[i] = '1';

return x;

}

else x[i] = '0';

}

return "";

}

int main()

{

Hello\_i\_am\_Salmon

int n; cin >> n;

string a = "";

for(int i=0; i<n; i++)

{

a += '0';

}

for(int i=0; i<pow(2, n); i++)

{

cout << a << ed;

a = sinhbit(a);

}

}

long long n; cin >> n;

for(int mask = 0; mask < (1 << n); mask++)

{

for(int j=0; j<n; j++)

{

if(mask & (1 << j))

{

cout << "1";

}

else cout << "0";

}

cout << ed;

}

**Sort struct**

struct book

{

ll t, a, b;

};

bool comp(book x, book y)

{

if(x.t != y.t) return x.t > y.t;

if(x.a != y.a) return x.a > y.a;

return x.b > y.b;

}

sort(A, A+n, comp);

**Z-function**

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;

}

**Spare Table**

void process2(int M[MAXN][LOGMAXN], int A[MAXN], int N)

{

int i, j;

// Khởi tạo M với các khoảng độ dài 1

for (i = 0; i < N; i++)

M[i][0] = i;

// Tính M với các khoảng dài 2^j

for (j = 1; 1 << j <= N; j++)

for (i = 0; i + (1 << j) - 1 < N; i++)

if (A[M[i][j - 1]] < A[M[i + (1 << (j - 1))][j - 1]])

M[i][j] = M[i][j - 1];

else

M[i][j] = M[i + (1 << (j - 1))][j - 1];

}

**Trie**

#include<cstdio>

#include<cstring>

const int N = 4000 + 2, L = 300000 + 2, WL = 100 + 2, MOD = 1337377;

struct TrieNode {

int next[26], nfinish;

} trie[N \* WL];

int nnode = 1, n, f[L];

char tmp[WL], s[L];

void insert(const char \* s) {

int u = 0;

for(int i = strlen(s) - 1; i >= 0; --i) {

int t = s[i] - 0x61;

if(trie[u].next[t] == 0) trie[u].next[t] = nnode++;

u = trie[u].next[t];

}

trie[u].nfinish = 1;

}

void enter() {

scanf("%s%d", s+1, &n);

for(int i = 0; i < n; ++i) {

scanf("%s", tmp);

insert(tmp);

}

}

void solve() {

f[0] = 1; int l = strlen(s + 1);

for(int i = 1; i <= l; ++i) {

int u = 0;

for(int j = i; j && trie[u].next[s[j] - 0x61]; --j)

if(trie[u = trie[u].next[s[j] - 0x61]].nfinish)

f[i] = (f[j-1] + f[i]) % MOD;

}

printf("%d\n", f[l]);

**NCK**

ll nck(ll n, ll k) {

for(int i=k+1; i<=n;i++) {

int res=i;

while(res>1) {

cnt[prime[res]]++;

res/=prime[res];

}

}

for(int i=n-k; i>1;i--) {

int res=i;

while(res>1) {

cnt[prime[res]]--;

res/=prime[res];

}

}

ll q=1;

for(int i=2; i<=n; i++) {

if(cnt[i]>0) {

ll ans=1, count =cnt[i], base= i;

while(count >=1) {

if(count&1)

(ans\*=base)%=mod;

(base\*=base)%=mod;

count>>=1;

}

q\*=ans;

q%=mod;

}

}

return q;

}

}

int main() {

enter();

solve();

return 0;

}

**Modulo Giai thua**

ll giaiThua[N], nghichDao[N];

void TINHGIAITHUA() {

giaiThua[0]=1;

ll i;

for (i = 1; i < N; i++) {

if(i>18)

giaiThua[i] = (i %mod\* giaiThua[i - 1]%mod)%mod;

else

giaiThua[i] = (i \* giaiThua[i - 1]);

}

// i--;

// nghichDao[i] = binpow(giaiThua[i], phi(mod)-1, mod);

// for (i--; i >= 0; i--)

// {

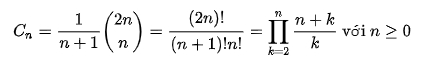
// nghichDao[i] = nghichDao[i + 1] \* (i + 1) % mod;

// }

}

Số nguyên tố lớn: 1073676287, 68718952447, 274876858367, 4398042316799,

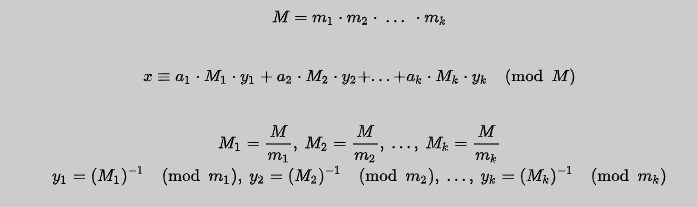
**Số catalan**

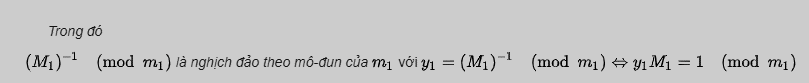


Những số catalan đầu tiên với n = 0, 1, 2, 3, …là:

1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452,

**Số dư trung hoa**



****

**Fenwick Tree 2D**

#include<bits/stdc++.h>

using namespace std;

#define N 4 // N-->max\_x and max\_y

// A structure to hold the queries

struct Query

{

int x1, y1; // x and y co-ordinates of bottom left

int x2, y2; // x and y co-ordinates of top right

};

// A function to update the 2D BIT

void updateBIT(int BIT[][N+1], int x, int y, int val)

{

for (; x <= N; x += (x & -x))

{

// This loop update all the 1D BIT inside the

// array of 1D BIT = BIT[x]

for (int yy=y; yy <= N; yy += (yy & -yy))

BIT[x][yy] += val;

}

return;

}

// A function to get sum from (0, 0) to (x, y)

int getSum(int BIT[][N+1], int x, int y)

{

int sum = 0;

for(; x > 0; x -= x&-x)

{

// This loop sum through all the 1D BIT

// inside the array of 1D BIT = BIT[x]

for(int yy=y; yy > 0; yy -= yy&-yy)

{

sum += BIT[x][yy];

}

}

return sum;

}

// A function to create an auxiliary matrix

// from the given input matrix

void constructAux(int mat[][N], int aux[][N+1])

{

// Initialise Auxiliary array to 0

for (int i=0; i<=N; i++)

for (int j=0; j<=N; j++)

aux[i][j] = 0;

// Construct the Auxiliary Matrix

for (int j=1; j<=N; j++)

for (int i=1; i<=N; i++)

aux[i][j] = mat[N-j][i-1];

return;

}

// A function to construct a 2D BIT

void construct2DBIT(int mat[][N], int BIT[][N+1])

{

// Create an auxiliary matrix

int aux[N+1][N+1];

constructAux(mat, aux);

// Initialise the BIT to 0

for (int i=1; i<=N; i++)

for (int j=1; j<=N; j++)

BIT[i][j] = 0;

for (int j=1; j<=N; j++)

{

for (int i=1; i<=N; i++)

{

// Creating a 2D-BIT using update function

// everytime we/ encounter a value in the

// input 2D-array

int v1 = getSum(BIT, i, j);

int v2 = getSum(BIT, i, j-1);

int v3 = getSum(BIT, i-1, j-1);

int v4 = getSum(BIT, i-1, j);

// Assigning a value to a particular element

// of 2D BIT

updateBIT(BIT, i, j, aux[i][j]-(v1-v2-v4+v3));

}

}

return;

}

// A function to answer the queries

void answerQueries(Query q[], int m, int BIT[][N+1])

{

for (int i=0; i<m; i++)

{

int x1 = q[i].x1 + 1;

int y1 = q[i].y1 + 1;

int x2 = q[i].x2 + 1;

int y2 = q[i].y2 + 1;

int ans = getSum(BIT, x2, y2)-getSum(BIT, x2, y1-1)-

getSum(BIT, x1-1, y2)+getSum(BIT, x1-1, y1-1);

printf ("Query(%d, %d, %d, %d) = %d\n",

q[i].x1, q[i].y1, q[i].x2, q[i].y2, ans);

}

return;

}

// Driver program

int main()

{

int mat[N][N] = {{1, 2, 3, 4},

{5, 3, 8, 1},

{4, 6, 7, 5},

{2, 4, 8, 9}};

// Create a 2D Binary Indexed Tree

int BIT[N+1][N+1];

construct2DBIT(mat, BIT);

/\* Queries of the form - x1, y1, x2, y2

For example the query- {1, 1, 3, 2} means the sub-matrix-

y

/\

3 | 1 2 3 4 Sub-matrix

2 | 5 3 8 1 {1,1,3,2} ---> 3 8 1

1 | 4 6 7 5 6 7 5

0 | 2 4 8 9

|

--|------ 0 1 2 3 ----> x

|

Hence sum of the sub-matrix = 3+8+1+6+7+5 = 30

\*/

Query q[] = {{1, 1, 3, 2}, {2, 3, 3, 3}, {1, 1, 1, 1}};

int m = sizeof(q)/sizeof(q[0]);

answerQueries(q, m, BIT);

return(0);

}

**Segment tree 2d**

void build(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 sum(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 sum(v\*2, tl, tm, l, min(r, tm))

+ sum(v\*2+1, tm+1, tr, max(l, tm+1), r);

}

void update(int v, int tl, int tr, int pos, int new\_val) {

if (tl == tr) {

t[v] = new\_val;

} else {

int tm = (tl + tr) / 2;

if (pos <= tm)

update(v\*2, tl, tm, pos, new\_val);

else

update(v\*2+1, tm+1, tr, pos, new\_val);

t[v] = t[v\*2] + t[v\*2+1];

}

}

// 2

pair<int, int> t[4\*MAXN];

pair<int, int> combine(pair<int, int> a, pair<int, int> b) {

if (a.first > b.first)

return a;

if (b.first > a.first)

return b;

return make\_pair(a.first, a.second + b.second);

}

void build(int a[], int v, int tl, int tr) {

if (tl == tr) {

t[v] = make\_pair(a[tl], 1);

} else {

int tm = (tl + tr) / 2;

build(a, v\*2, tl, tm);

build(a, v\*2+1, tm+1, tr);

t[v] = combine(t[v\*2], t[v\*2+1]);

}

}

pair<int, int> get\_max(int v, int tl, int tr, int l, int r) {

if (l > r)

return make\_pair(-INF, 0);

if (l == tl && r == tr)

return t[v];

int tm = (tl + tr) / 2;

return combine(get\_max(v\*2, tl, tm, l, min(r, tm)),

get\_max(v\*2+1, tm+1, tr, max(l, tm+1), r));

}

void update(int v, int tl, int tr, int pos, int new\_val) {

if (tl == tr) {

t[v] = make\_pair(new\_val, 1);

} else {

int tm = (tl + tr) / 2;

if (pos <= tm)

update(v\*2, tl, tm, pos, new\_val);

else

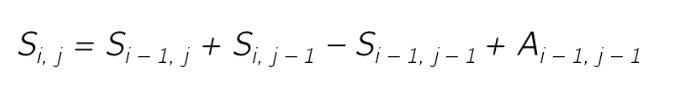
update(v\*2+1, tm+1, tr, pos, new\_val);

t[v] = combine(t[v\*2], t[v\*2+1]);

}

}

**Mảng cộng dồn 2 chiều**

****

**Bell man ford**

const long long INF = 2000000000000000000LL;

struct Edge {

int u, v;

long long w; // cạnh từ u đến v, trọng số w

};

void bellmanFord(int n, int S, vector<Edge> &e, vector<long long> &D, vector<int> &trace) {

// e: danh sách cạnh

// n: số đỉnh

// S: đỉnh bắt đầu

// D: độ dài đường đi ngắn nhất

// trace: mảng truy vết đường đi

// INF nếu không có đường đi

// -INF nếu có đường đi âm vô tận

D.resize(n, INF);

trace.resize(n, -1);

D[S] = 0;

for(int T = 1; T < n; T++) {

for (auto E : e) {

int u = E.u;

int v = E.v;

long long w = E.w;

if (D[u] != INF && D[v] > D[u] + w) {

D[v] = D[u] + w;

trace[v] = u;

}

}

}

}

vector<int> trace\_path(vector<int> &trace, int S, int u) {

if (u != S && trace[u] == -1) return vector<int>(0); // không có đường đi

vector<int> path;

while (u != -1) { // truy vết ngược từ u về S

path.push\_back(u);

u = trace[u];

}

reverse(path.begin(), path.end()); // cần reverse vì đường đi lúc này là từ u về S

return path;

}

**Thuật toán Floyd-Warshall**

void init\_trace(vector<vector<int>> &trace) {

int n = trace.size();

for (int u = 0; u < n; u++) {

for (int v = 0; v < n; v++) {

trace[u][v] = u;

}

}

}

void floydWarshall(int n, vector<vector<long long>> &w, vector<vector<long long>> &D, vector<vector<int>> &trace) {

D = w;

init\_trace(trace); // nếu cần dò đường đi

for (int k = 0; k < n; k++) {

for (int u = 0; u < n; u++) {

for (int v = 0; v < n; v++) {

if (D[u][v] > D[u][k] + D[k][v]) {

D[u][v] = D[u][k] + D[k][v];

trace[u][v] = trace[k][v];

}

}

}

}

}

vector<int> trace\_path(vector<vector<int>> &trace, int u, int v) {

vector<int> path;

while (v != u) { // truy vết ngược từ v về u

path.push\_back(v);

v = trace[u][v];

}

path.push\_back(u);

reverse(path.begin(), path.end()); // cần reverse vì đường đi từ v ngược về u

return path;

}

**Diện tích đa giác**

Hello\_i\_am\_Salmon

ll n; cin >> n;

ll x[n+1], y[n+1];

for(int i=0; i<n; i++)

{

cin >> x[i] >> y[i];

}

x[n] = x[0], y[n] = y[0];

ll ans = 0;

for(int i=0; i<n; i++)

{

ans += (x[i] \* y[i+1] - y[i] \* x[i+1]);

}

cout << abs(ans) << ed;

**Đoạn thẳng giao nhau**

struct toado

{

ll x, y;

};

struct DT

{

ll A, B, C;

};

bool check\_inside(ll x, ll y, ll u1, ll v1, ll u2, ll v2)

{

return (x >= min(u1, u2) && x <= max(u1, u2) && y >= min(v1, v2) && y <= max(v1, v2));

}

toado Cal\_vector(toado A, toado B)

{

toado ans;

ans.x = B.x - A.x;

ans.y = B.y - A.y;

return ans;

}

ll Tich\_co\_huong(toado A, toado B)

{

return A.x \* B.y - A.y \* B.x;

}

int main()

{

Hello\_i\_am\_Salmon

ll t; cin >> t;

while(t--)

{

toado A, B, C, D;

cin >> A.x >> A.y >> B.x >> B.y >> C.x >> C.y >> D.x >> D.y;

toado AB = Cal\_vector(A, B);

toado AC = Cal\_vector(A, C);

toado AD = Cal\_vector(A, D);

toado CD = Cal\_vector(C, D);

toado CA = Cal\_vector(C, A);

toado CB = Cal\_vector(C, B);

ll ans = 0;

if(Tich\_co\_huong(AB, AC) == 0)

if(check\_inside(C.x, C.y, A.x, A.y, B.x, B.y)) ans = 1;

if(Tich\_co\_huong(AB, AD) == 0)

if(check\_inside(D.x, D.y, A.x, A.y, B.x, B.y)) ans = 1;

if(Tich\_co\_huong(CD, CA) == 0)

if(check\_inside(A.x, A.y, C.x, C.y, D.x, D.y)) ans = 1;

if(Tich\_co\_huong(CD, CB) == 0)

if(check\_inside(B.x, B.y, C.x, C.y, D.x, D.y)) ans = 1;

if(((Tich\_co\_huong(AB, AC) < 0 && Tich\_co\_huong(AB, AD) > 0) || (Tich\_co\_huong(AB, AC) > 0 && Tich\_co\_huong(AB, AD) < 0)) && ((Tich\_co\_huong(CD, CA) < 0 && Tich\_co\_huong(CD, CB) > 0) || (Tich\_co\_huong(CD, CA) > 0 && Tich\_co\_huong(CD, CB) < 0))) ans = 1;

cout << (ans ? "YES" : "NO") << ed;

}

}

**Kiểm tra 3 điểm CCW, CW**

ll x1, y1, x2, y2, x3, y3; cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;

ll x12 = x2 - x1;

ll y12 = y2 - y1;

ll x13 = x3 - x1;

ll y13 = y3 - y1;

ll tich\_co\_huong = x12 \* y13 - x13 \* y12;

if(tich\_co\_huong == 0) cout << "TOUCH" << ed;

else if(tich\_co\_huong > 0) cout << "LEFT" << ed;

else cout << "RIGHT" << ed;

**Chuyển nhị phân sang thập phân**

long long BitoDec(string n)

{

long long dec = 0;

long long base = 1;

for(int i=sz(n)-1; i>=0; i--)

{

if(n[i] == '1')

{

dec += base;

}

base \*= 2;

}

return dec;

}

**Chuyển thập phân sang nhị phân**

vector<bool> Binary;

int DectoBi(long long n)

{

int i = 0;

while(n > 0)

{

Binary.pb(n%2);

n /= 2;

i++;

}

reverse(all(Binary));

return i;

}

**Template code**

/\*--------------------------------------------

Competitive programing

Name: Đào Xuân Hoàng Tuấn (Salmon)

Nothing is impossible, as you as you believe you can do it

You can do it!!!

----------------------------------------------\*/

#include <bits/stdc++.h>

// #include <ext/pb\_ds/assoc\_container.hpp>

// #include <ext/pb\_ds/tree\_policy.hpp>

using namespace std;

// using namespace \_\_gnu\_pbds;

#define ordered\_set tree<int, null\_type,less<int>, rb\_tree\_tag,tree\_order\_statistics\_node\_update>

#define Hello\_i\_am\_Salmon ios\_base::sync\_with\_stdio(false);cin.tie(0);cout.tie(0);

#define print\_fix cout << fixed << setprecision(20);

#define show(x) cerr << #x << " -> " << x << endl;

#define show3 cerr << "\*\*\*" << endl;

#define show2 cerr << "\*\*" << endl;

#define show1 cerr << "\*" << endl;

#define all(v) v.begin(), v.end()

#define sz(t) (int) t.size()

#define pb push\_back

#define se second

#define fi first

#define el endl

#define ed '\n'

#define \_ " "

void debug\_out() {cout << '\n';}

template <typename Head, typename ...Tail>

void debug\_out(Head H, Tail ...T)

{

cout << H << ' ';

debug\_out(T...);

}

#define fix(...) cout << "[" << #\_\_VA\_ARGS\_\_ << "]: ", debug\_out(\_\_VA\_ARGS\_\_)

const long long N =1e7+10;

const long long Nn = 1e3+10;

const double PI = atan(1)\*4;

const int MOD = 1e9 + 7;

const long long INF = 1e9 + 7ll;

/\* --- you should drink a cup of milk tea before reading my code ---- \*/

int main()

{

Hello\_i\_am\_Salmon

}

/\* Test case

\*/

/\* My code is very beautiful and artistic \*/

