Test 1

#include <bits/stdc++.h>

using namespace std;

// Function to check if a value lies within any segment

bool isInSegment(const set<pair<int, int>> &segments, int val) {

auto it = segments.upper\_bound({val, INT\_MAX});

if (it != segments.begin()) {

--it;

if (it->first <= val && val <= it->second)

return true;

}

return false;

}

int countPointsOnPath(vector<int> &Xq, vector<int> &Yq, vector<int> &Xp, vector<int> &Yp) {

int M = Xp.size(), N = Xq.size();

map<int, set<pair<int, int>>> vertical, horizontal;

// Collect all segments

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

int x1 = Xp[i - 1], y1 = Yp[i - 1];

int x2 = Xp[i], y2 = Yp[i];

if (x1 == x2) // Vertical segment

vertical[x1].insert({min(y1, y2), max(y1, y2)});

else // Horizontal segment

horizontal[y1].insert({min(x1, x2), max(x1, x2)});

}

// Check query points

int count = 0;

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

if (isInSegment(vertical[Xq[i]], Yq[i]) || isInSegment(horizontal[Yq[i]], Xq[i]))

count++;

}

return count;

}

int main() {

int T;

cin >> T;

while (T--) {

int N, M;

cin >> N >> M;

vector<int> Xq(N), Yq(N), Xp(M), Yp(M);

for (int i = 0; i < N; i++) cin >> Xq[i];

for (int i = 0; i < N; i++) cin >> Yq[i];

for (int i = 0; i < M; i++) cin >> Xp[i];

for (int i = 0; i < M; i++) cin >> Yp[i];

cout <<"#"<<countPointsOnPath(Xq, Yq, Xp, Yp) << endl;

}

return 0;

}

Test 2

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int mx\_day, target;

int removed\_max(vector<vector<int>> &dp, vector<int> &stock, vector<int> &inflow, int idx, int day) {

int n = stock.size();

if (idx == n || day == mx\_day) return 0;

if (dp[idx][day] != -1) return dp[idx][day];

int ans = removed\_max(dp, stock, inflow, idx + 1, day); // Option 1: Skip current item.

// Option 2: Export current item.

int total\_item\_stock = stock[idx] + (day + 1) \* inflow[idx];

ans = max(ans, removed\_max(dp, stock, inflow, idx + 1, day + 1) + total\_item\_stock);

return dp[idx][day] = ans;

}

// Checks if it is possible to reduce the total stock to the target within 'day' days.

bool possible(vector<int> &stock, vector<int> &inflow, int day) {

int n = stock.size(), total = 0;

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

total += stock[i] + day \* inflow[i];

}

vector<vector<int>> dp(n, vector<int>(day + 1, -1));

mx\_day = day;

return (total - removed\_max(dp, stock, inflow, 0, 0)) <= target;

}

// Finds the minimum number of days required to meet the target.

int solve(vector<int> &stock, vector<int> &inflow) {

int n = stock.size();

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

if (possible(stock, inflow, day))

return day;

}

return -1;

}

int main() {

int t;

cin >> t;

while (t--) {

int n;

cin >> n >> target;

vector<pair<int, int>> items(n);

// Input is given as (stock, inflow) per item.

for (auto &item : items) {

cin >> item.second >> item.first;

}

sort(items.begin(), items.end());

vector<int> stock, inflow;

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

stock.push\_back(items[i].second);

inflow.push\_back(items[i].first);

}

cout << solve(stock, inflow) << endl;

}

return 0;

}

Test 3

#include <iostream>

#include <map>

#include <string>

using namespace std;

int main(){

string s;

cin >> s;

int n = s.size(), maxLen = 0, prefix = 0;

map<int, int> first;

first[0] = -1;

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

prefix += (s[i] == 'R' ? 1 : -1);

if (first.find(prefix) == first.end())

first[prefix] = i;

else {

int len = i - first[prefix];

if (len > maxLen) maxLen = len;

}

}

cout << n - maxLen << "\n";

return 0;

}

Test 4 Q1

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

typedef long long ll;

const ll INF = 1e9;

const ll dx[] = {0, 1, -1, 0}, dy[] = {1, 0, 0, -1};

void bfs(int sx, int sy, vector<vector<ll>> &dist, const vector<vector<ll>> &mat) {

int h = mat.size(), w = mat[0].size();

queue<pair<ll, ll>> q;

dist[sx][sy] = 0;

q.push({sx, sy});

while (!q.empty()) {

auto [x, y] = q.front(); q.pop();

for (int i = 0; i < 4; i++) {

int nx = x + dx[i], ny = y + dy[i];

if (nx >= 0 && ny >= 0 && nx < h && ny < w && mat[nx][ny] != 1 && dist[nx][ny] > dist[x][y] + 1) {

dist[nx][ny] = dist[x][y] + 1;

q.push({nx, ny});

}

}

}

}

void solve() {

int t; cin >> t;

while (t--) {

ll h, w, c, gx, gy, ax, ay;

cin >> h >> w >> c;

vector<vector<ll>> mat(h, vector<ll>(w));

vector<pair<ll, ll>> warehouses;

for (ll i = 0; i < h; i++)

for (ll j = 0; j < w; j++) {

cin >> mat[i][j];

if (mat[i][j] == 2) gx = i, gy = j;

if (mat[i][j] == 4) ax = i, ay = j;

if (mat[i][j] == 3) warehouses.emplace\_back(i, j);

}

vector<vector<ll>> ds(h, vector<ll>(w, INF)), dd(h, vector<ll>(w, INF));

bfs(gx, gy, ds, mat);

bfs(ax, ay, dd, mat);

ll max\_goods = 0;

for (auto [wx, wy] : warehouses) {

ll cost = c - ds[wx][wy];

if (cost > 0) {

cost = (cost / dd[wx][wy]) - 1;

if (cost > 0) max\_goods = max(max\_goods, cost);

}

}

cout << max\_goods << "\n";

}

}

int main() {

ios\_base::sync\_with\_stdio(false); cin.tie(NULL);

solve();

}

Test 4 Q2

#include <iostream>

#include <vector>

using namespace std;

typedef long long ll;

#define fastio ios\_base::sync\_with\_stdio(false); cin.tie(NULL); cout.tie(NULL);

void solve() {

int n, k;

cin >> n >> k;

vector<vector<int>> freq(401, vector<int>(401, 0));

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

int x, y;

cin >> x >> y;

freq[x][y]++;

}

for (int i = 0; i < 401; i++) {

for (int j = 0; j < 401; j++) {

if (i > 0) freq[i][j] += freq[i - 1][j];

if (j > 0) freq[i][j] += freq[i][j - 1];

if (i > 0 && j > 0) freq[i][j] -= freq[i - 1][j - 1];

}

}

int low = 0, high = 400, ans = 400;

auto isValid = [&](int mid) {

for (int i = 0; i + mid < 401; i++) {

for (int j = 0; j + mid < 401; j++) {

int cnt = freq[i + mid][j + mid];

if (i > 0) cnt -= freq[i - 1][j + mid];

if (j > 0) cnt -= freq[i + mid][j - 1];

if (i > 0 && j > 0) cnt += freq[i - 1][j - 1];

if (cnt >= k) return true;

}

}

return false;

};

while (low <= high) {

int mid = (low + high) / 2;

if (isValid(mid)) {

ans = mid;

high = mid - 1;

} else {

low = mid + 1;

}

}

cout << ans << "\n";

}

int main() {

fastio;

solve();

return 0;

}

Test 4 Q3

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

long long min\_removal\_cost(int n, const vector<long long>& a, const vector<long long>& b) {

// dp[i][0] = min cost for stones 1..i if stone i+1 is removed before stone i

// dp[i][1] = min cost for stones 1..i if stone i+1 is removed after stone i

vector<vector<long long>> dp(n + 1, vector<long long>(2, 0));

// Base case for the first stone

dp[1][0] = 0;

dp[1][1] = a[0];

// Fill dp table for stones 2..n

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

dp[i][0] = min(dp[i - 1][0] + a[i - 1], dp[i - 1][1] + 0);

dp[i][1] = min(dp[i - 1][0] + b[i - 1], dp[i - 1][1] + a[i - 1]);

}

// The answer is dp[n][0] because stone n+1 doesn't exist

return dp[n][0];

}

int main() {

int n;

cin >> n;

vector<long long> a(n), b(n);

// Read cost array a (cost if stone is removed before its neighbor)

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

cin >> a[i];

}

// Read cost array b (cost if stone is removed after its neighbor)

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

cin >> b[i];

}

cout << min\_removal\_cost(n, a, b) << endl;

return 0;

}

Test 5 Q1

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

#define fastio ios\_base::sync\_with\_stdio(false); cin.tie(NULL); cout.tie(NULL);

ll gcd(ll a, ll b) {

return b == 0 ? a : gcd(b, a % b);

}

int main() {

fastio;

int t;

cin >> t;

while (t--) {

ll n;

cin >> n;

vector<string> v(n);

for (auto &s : v) cin >> s;

vector<vector<ll>> mp(10, vector<ll>(10, 0));

for (ll i = n - 1; i >= 0; i--) {

int first = v[i][0] - '0', last = v[i].back() - '0';

for (ll j = 0; j <= 9; j++) {

if (mp[last][j] != 0) {

mp[first][j] = max(mp[first][j], mp[last][j] + (ll)v[i].length());

}

}

mp[first][last] = max(mp[first][last], (ll)v[i].length());

}

ll res = 0;

for (ll i = 0; i <= 9; i++) {

res = max(res, mp[i][i]);

}

cout << res << "\n";

}

return 0;

}

Test 5 q2

#include <bits/stdc++.h>

using namespace std;

int maxDiffD(const vector<int> &a, const vector<int> &b)

{

vector<int> sa = a, sb = b;

sort(sa.begin(), sa.end());

sort(sb.begin(), sb.end());

set<int> s;

for (int x : sa)

s.insert(x);

for (int x : sb)

s.insert(x);

vector<int> c;

c.push\_back(\*s.begin() - 1);

for (int x : s)

c.push\_back(x);

c.push\_back(\*s.rbegin() + 1);

int mx = INT\_MIN, bestD = 0;

for (int d : c)

{

int ca = upper\_bound(sa.begin(), sa.end(), d) - sa.begin();

int cb = upper\_bound(sb.begin(), sb.end(), d) - sb.begin();

int as = ca + (sa.size() - ca) \* 2;

int bs = cb + (sb.size() - cb) \* 2;

int diff = as - bs;

if (diff > mx)

{

mx = diff;

bestD = d;

}

}

return bestD;

}

int main()

{

int t;

cin >> t;

while (t--)

{

int n, m;

cin >> n >> m;

vector<int> a(n), b(m);

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

cin >> a[i];

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

cin >> b[i];

cout << "Optimal D: " << maxDiffD(a, b) << endl;

}

return 0;

}

Test 6 Q1

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef long double ld;

int main() {

int t;

cin >> t;

while (t--) {

ll n, m;

cin >> n >> m;

vector<pair<ll, ll>> points(n);

pair<ll, ll> destination;

cin >> destination.first >> destination.second;

for (auto &point : points) {

cin >> point.first >> point.second;

}

set<ll> parities;

for (const auto &point : points) {

parities.insert((abs(destination.first - point.first) + abs(destination.second - point.second)) % 2);

}

if (parities.size() > 1) {

cout << "-1\n";

continue;

}

ll parity = \*parities.begin();

ll max\_distance = 0;

for (const auto &point : points) {

ll distance = abs(destination.first - point.first) + abs(destination.second - point.second);

max\_distance = max(max\_distance, distance);

}

ll lo = 0, hi = (ll)(sqrtl(max\_distance \* 2) + 1), ans = hi;

while (lo <= hi) {

ll mid = lo + (hi - lo) / 2;

ll sum = (mid \* (mid + 1)) / 2;

if (sum >= max\_distance) {

ans = mid;

hi = mid - 1;

} else {

lo = mid + 1;

}

}

ll total\_sum = (ans \* (ans + 1)) / 2;

if ((total\_sum % 2) == parity) {

cout << ans << "\n";

} else {

ll next\_sum1 = ((ans + 1) \* (ans + 2)) / 2;

ll next\_sum2 = ((ans + 2) \* (ans + 3)) / 2;

if ((next\_sum1 % 2) == parity) {

cout << ans + 1 << "\n";

} else {

cout << ans + 2 << "\n";

}

}

}

return 0;

}

Test 7 Q1

#include <bits/stdc++.h>

using namespace std;

int n, m;

int dp[1001][1001];

int helper(int idx, int lastRobot, vector<int> &v) {

if (idx == v.size()) {

return 0;

}

if (dp[idx][lastRobot] != -1) {

return dp[idx][lastRobot];

}

int usePrevRobot = (idx - lastRobot) \* v[idx] + helper(idx + 1, lastRobot, v);

int newRobot = m + helper(idx + 1, idx, v);

return dp[idx][lastRobot] = min(usePrevRobot, newRobot);

}

int main() {

int t;

cin >> t;

while (t--) {

cin >> n >> m;

vector<int> v(n);

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

cin >> v[i];

}

memset(dp, -1, sizeof(dp));

int firstNonZero = -1;

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

if (v[i] != 0) {

firstNonZero = i;

break;

}

}

int ans = 0;

if (firstNonZero != -1) {

ans = m + helper(firstNonZero + 1, firstNonZero, v);

}

cout << ans << endl;

}

return 0;

}

Test 7 Q2

#include <bits/stdc++.h>

using namespace std;

const int MOD = 1e9 + 7;

int dp[101][1001][2];

int solve(int idx, int sum, bool tight, const string &A) {

if (sum < 0) return 0;

if (idx == A.size()) return sum == 0;

if (dp[idx][sum][tight] != -1) return dp[idx][sum][tight];

int limit = tight ? (A[idx] - '0') : 9;

int ans = 0;

for (int digit = 0; digit <= limit; digit++) {

ans = (ans + solve(idx + 1, sum - digit, tight && (digit == limit), A)) % MOD;

}

return dp[idx][sum][tight] = ans;

}

int main() {

string A;

int S;

cin >> A >> S;

memset(dp,-1,sizeof(dp));

cout << solve(0, S, 1, A) << endl;

return 0;

}