Sri Lanka Institute of Information Technology



Foundations of Algorithms – IE2072

Assignment – 2024 May

IT22148018

S. D. P. N. DISANAYAKA

Y2.S2.CS.WD.01

**Question 1**

You are an IT company manager. Based on their performance over the last N working days, you must rate your employee. You are given an array of N integers called workload, where workload[i] represents the number of hours an employee worked on an ith day. The employee must be evaluated.

using the following criteria:

Rating = the maximum number of consecutive working days when the employee has worked more than 6 hours.

**Task**

Determine the employee rating.

**Example**

Assumptions

* N = 12
* workload = [2, 3, 7, 8, 7, 6, 3, 8, 12, 11, 12, 10]

**Approach**

Workload with consecutive hours > 6 = [2, 3, 7, 8, 7, 6, 3, 8, 12, 11, 12, 10] => Longest Interval = [8,12,11,12,10]

Therefore return 5.

**Input format.**

* The first line contains an integer N denoting the number of working days.
* The second line contains a space-separated integer array workload where workload[i] represents the number of hours an employee worked on an ith day.

**Output format.**

Print the employee rating.

**Algorithm**

#include <stdio.h>

int maxConDays(int workload[], int N);

int main() {

int N;

printf("Enter the number of working days: ");

scanf("%d", &N);

int workload[N];

printf("Enter the workload for each day (separated by space): ");

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

scanf("%d", &workload[i]);

}

int rate = maxConDays(workload, N);

printf("Employee rating: %d\n", rate);

return 0;

}

int maxConDays(int workload[], int N) {

int maxCon = 0;

int con = 0;

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

if (workload[i] > 6) {

con++;

if (con > maxCon) {

maxCon = con;

}

} else {

con = 0;

}

}

return maxCon;

}

**Sample output.**

A blue background with white text

Description automatically generated

**Question 2**

You are given a N \* M grid in which each cell consists of either 0 or 1. A cell (i,j) is blocked if its value is 1. Standing at a cell (i,j), you can perform the following steps.

1. You can move right to the very next cell which is not blocked.
2. You can move down to the very next cell which is not blocked.

You are initially located in cell (1,1). Determine the number of ways in which you can reach (N,M) starting from your initial location.

Example: Let 3 \* 3 grid be

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 0 |

If you are standing at cell (1,1), then:

* By performing step 1, you can jump to cell (1,3) and,
* By performing step 2, you can jump to cell (3,1)

The answer will be 2.

**Input format.**

* The first line contains N and M denoting the number of rows and the number of columns.
* Each of the next N lines consists of a string of length M.

**Output format.**

Print a single line containing the number of ways to reach (N,M) from (1,1)

**Algorithm**

#include <stdio.h>

#include <stdlib.h>

void move(int i, int j, int \*\*grid);

int n, m;

int ways = 0;

int main() {

printf("Enter the number of rows (n): ");

scanf("%d", &n);

printf("Enter the number of columns (m): ");

scanf("%d", &m);

int \*\*grid = (int \*\*)malloc(n \* sizeof(int \*));

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

grid[i] = (int \*)malloc(m \* sizeof(int));

}

printf("Enter the grid values (0 or 1):\n");

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

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

scanf("%d", &grid[i][j]);

}

}

n = n - 1;

m = m - 1;

move(0, 0, grid);

printf("Number of ways : %d\n", ways);

return 0;

}

void move(int i, int j, int \*\*grid) {

if (i == n && j == m) {

ways = ways + 1;

} else {

if (i <= n && j <= m && grid[i][j] == 0) {

if (i + 1 <= n) {

if (grid[i + 1][j] == 0) {

move(i + 1, j, grid);

} else {

move(i + 2, j, grid);

}

}

if (j + 1 <= m) {

if (grid[i][j + 1] == 0) {

move(i, j + 1, grid);

} else {

move(i, j + 2, grid);

}

}

}

}

}

**Sample output.**

A blue screen with a black background

Description automatically generated with medium confidence