**CODE QUESTION - 10 MARKS**

**Finding the Minimum Cost Problem – Dynamic Programming**

**Q1.** Find the minimum cost to reach to the last cell of a matrix from its first cell.

You have to provide the no. of rows and columns then enter the values (or cost) to an array, then the method minCost() will calculate the minimum cost to reach to the last cell from the very first cell.

The only constraint is that you can count the cost to right or down direction.

**PUBLIC TEST CASES:**

|  |  |  |
| --- | --- | --- |
| **CASE #01** | **CASE #02** | **CASE #03** |
| **INPUT**  3  3  1 2 3  1 1 2  3 1 1 | **INPUT**  2  2  2 4  1 2 | **INPUT**  4  4  1 2 3 4  4 3 2 1  2 3 4 1  3 4 1 2 |
| **OUTPUT**  5 | **OUTPUT**  5 | **OUTPUT**  12 |

**PRIVATE TEST CASES:**

|  |  |  |
| --- | --- | --- |
| **CASE #01** | **CASE #02** | **CASE #03** |
| **INPUT**  5  5  1 2 3 4 5  1 1 2 2 1  2 2 1 1 3  4 1 1 4 4  5 1 1 1 5 | **INPUT**  3  3  3 4 1  5 2 4  1 3 5 | **INPUT**  1  1  5 |
| **OUTPUT**  14 | **OUTPUT**  17 | **OUTPUT**  5 |

**Driver Code**

//DRIVER CODE

**public** **static** **void** main(String[] args) {

**int** rows = 0;

**int** cols = 0;

Main obj = **new** Main(rows, cols);

obj.minCost();

}

**Default Code**

**import** java.util.Scanner;

**public** **class** Main {

**private** **static** Scanner *input*;

**private** **int** rows, cols;

**private** **int**[][] cost;

**static** {

*input* = **new** Scanner(System.***in***);

}

**public** Main(**int** rows, **int** cols) {

rows = *input*.nextInt();

cols = *input*.nextInt();

**this**.rows = rows;

**this**.cols = cols;

**this**.cost = **new** **int**[rows][cols];

//System.out.println("Enter Cost (Values):");

**for**(**int** i=0;i<rows;i++) {

**for**(**int** j=0;j<cols;j++) {

**this**.cost[i][j] = *input*.nextInt();

}

}

}

**public** **void** minCost() {

**int**[][] dp = **new** **int**[rows][cols];

**for**(**int** i=0;i<rows;i++) {

**for**(**int** j=0;j<cols;j++){

dp[i][j] = -1;

}

}

System.***out***.println(minCost(rows,cols,dp));

}

//LOGIC BUILDING CODE

**public** **int** minCost(**int** m, **int** n, **int**[][]dp){

}

//DRIVER CODE

**public** **static** **void** main(String[] args) {

**int** rows = 0;

**int** cols = 0;

Main obj = **new** Main(rows, cols);

obj.minCost();

}

}

**Solution**

**import** java.util.Scanner;

**public** **class** Main {

**private** **static** Scanner *input*;

**private** **int** rows, cols;

**private** **int**[][] cost;

**static** {

*input* = **new** Scanner(System.***in***);

}

**public** Main(**int** rows, **int** cols) {

rows = *input*.nextInt();

cols = *input*.nextInt();

**this**.rows = rows;

**this**.cols = cols;

**this**.cost = **new** **int**[rows][cols];

//System.out.println("Enter Cost (Values):");

**for**(**int** i=0;i<rows;i++) {

**for**(**int** j=0;j<cols;j++) {

**this**.cost[i][j] = *input*.nextInt();

}

}

}

**public** **void** minCost() {

**int**[][] dp = **new** **int**[rows][cols];

**for**(**int** i=0;i<rows;i++) {

**for**(**int** j=0;j<cols;j++){

dp[i][j] = -1;

}

}

System.***out***.println(minCost(rows,cols,dp));

}

//LOGIC BUILDING CODE

**public** **int** minCost(**int** m, **int** n, **int**[][]dp){

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

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

dp[i][j] = cost[i][j];

**if**(i==0 && j>0) dp[i][j] += dp[i][j-1];

**else** **if**(j==0 && i>0) dp[i][j] += dp[i-1][j];

**else** **if**(j>0 && i>0) dp[i][j] += Math.*min*(dp[i-1][j], dp[i][j-1]);

}

}

**return** dp[m-1][n-1];

}

//DRIVER CODE

**public** **static** **void** main(String[] args) {

**int** rows = 0;

**int** cols = 0;

Main obj = **new** Main(rows, cols);

obj.minCost();

}

}