**PART A**

**EXPERIMENT NO. 3**

**A.1 AIM: -** To perform Matrix Operation, find minimum cost path, find maximum in an integer array, and array sorting.

**A.2 Prerequisite**

* Different programming language (Python or Java), Understanding of Machine Learning Algorithms, Machine Learning Algorithms

**A.3 Outcome**

After successful completion of this experiment students will be able to understand working of matrix, find minimum and maximum cost paths

**A.4 Theory**

**Min Cost Path**

The minimum cost path problem in Java is one the most prominent problems that have been asked in the interview. In this problem, a matrix is provided (costMatrix[][]), which represents the cost of each of the cells present in the costMatrix[][]. The task is to go from the top left corner to the bottom right corner such that the cost is minimum. We have to return the minimum cost. The rule from going from one cell to another cell is that one can only go in the left or down or the diagonal direction, with one cell at a time. For example, from the current cell, say costMatrix[x][y], we can only go to one of these cells: costMatrix[x][y + 1] (the left direction), costMatrix[x + 1][y] (the downward direction), and costMatrix[x + 1][y + 1] (the diagonal direction).

For example, in the following matrix

Minimum Cost Path Problem in Java

There are the following paths to go from the top-left cell (of the cost 1) to the bottom-right cell (of the cost 7).

1 -> 6 -> 9 -> 5 -> 7 Total Cost = 1 + 6 + 9 + 5 + 7 = 28

1 -> 6 -> 15 -> 5 -> 7 Total Cost = 1 + 6 + 15 + 5 + 7 = 34

1 -> 6 -> 15 -> 3 -> 7 Total Cost = 1 + 6 + 15 + 3 + 7 = 32

1 -> 6 -> 15 -> 7 Total Cost = 1 + 6 + 15 + 7 = 29

1 -> 6 -> 5 -> 7 Total Cost = 1 + 6 + 5 + 7 = 19

1 -> 2 -> 15 -> 3 -> 7 Total Cost = 1 + 2 + 15 + 3 + 7 = 28

1 -> 2 -> 15 -> 5 -> 7 Total Cost = 1 + 2 + 15 + 5 + 7 = 30

1 -> 2 -> 15 -> 7 Total Cost = 1 + 2 + 15 + 7 = 25

1 -> 2 -> 2 -> 3 -> 7 Total Cost = 1 + 2 + 2 + 3 + 7 = 15

1 -> 2 -> 3 -> 7 Total Cost = 1 + 2 + 3 + 7 = 13

In all the above-mentioned paths, the last path (1 -> 2 -> 3 -> 7, total cost: 13) has the minimum cost. Therefore, 13 is the required answer of the above matrix.

**A5. Task**

Perform Following Operations

1.Write a Python program to find out when given an array of positive elements, you have to flip the sign of some of its elements such that the resultant sum of the elements of array should be minimum non-negative (as close to zero as possible). Return the minimum no. of elements whose sign needs to be flipped such that the resultant sum is minimum non-negative. Note that the sum of all the array elements will not exceed 10^4

Input: arr[] = [14, 10, 4]

Output: 1

Here, we will flip the sign of 14 and the resultant sum will be 0. Note that flipping the signs of 10 and 4 also gives the resultant sum 0 but the count of flipped elements is not minimum.

2. Write a Python program to find out when given a two dimensional grid, each cell of which contains integer cost which represents a cost to traverse through that cell. The task is to find the maximum cost path from the bottom-left corner to the top-right corner.

3. Write a Python program to find out when given an array of non-negative integers arr[], the task is to find a pair (n, r) such that nPr is maximum possible and r ≤ n.

Input: arr[] = {5, 2, 3, 4, 1}

Output: n = 5 and r = 4

5P4 = 5! / (5 – 4)! = 120

which is maximum possible. Input: arr[] = {0, 2, 3, 4, 1, 6, 8, 9} Output: n = 9 and r = 8

4. Write a Python program to find out when given an array of non-negative integers arr[], the task is to find a pair (n, r) such that nPr is maximum possible and r ≤ n.

Function to return the minimum number of given operations required to sort the array

Input: arr[] = {1, 2, 1, 4, 3}

Output: 2

Add 1 to the 3rd element(1) and subtract 1 from the 4th element(4) to get {1, 2, 2, 3, 3} Input: arr[] = {1, 2, 2, 100}

Output: 0 Given array is already sorted.

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No.C015 | Name:Prachi Dave |
| Class :B | Batch :B1 |
| Date of Experiment: 12-01-24 | Date of Submission 12-01-24 |
| Grade : |  |

**B.1 Documentation written by student:**

# %%

def min\_flips\_to\_make\_sum\_non\_negative(arr):

    total = sum(arr)

    n = len(arr)

    dp = [[float('inf')] \* (total + 1) for \_ in range(n + 1)]

    for i in range(n + 1):

        dp[i][0] = 0

    for i in range(1, n + 1):

        for j in range(1, total + 1):

            if arr[i - 1] <= j:

                dp[i][j] = min(dp[i - 1][j], dp[i - 1][j - arr[i - 1]] + 1)

            else:

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

    result = float('inf')

    for j in range(total // 2 + 1):

        result = min(result, dp[n][j])

    return result

arr = [14, 10, 4]

print(min\_flips\_to\_make\_sum\_non\_negative(arr))  # Output: 1

# %%

def max\_cost\_path(grid):

    m, n = len(grid), len(grid[0])

    dp = [[0] \* n for \_ in range(m)]

    for i in range(m - 1, -1, -1):

        for j in range(n):

            if i == m - 1 and j == 0:

                dp[i][j] = grid[i][j]

            elif i == m - 1:

                dp[i][j] = dp[i][j - 1] + grid[i][j]

            elif j == 0:

                dp[i][j] = dp[i + 1][j] + grid[i][j]

            else:

                dp[i][j] = max(dp[i][j - 1], dp[i + 1][j]) + grid[i][j]

    return dp[0][n - 1]

grid = [

    [1, 2, 3],

    [4, 5, 6],

    [7, 8, 9]

]

print(max\_cost\_path(grid))  # Output: 26

# %%

from itertools import combinations

def max\_nPr\_pair(arr):

    n = len(arr)

    max\_nPr = -1

    result = ()

    for i, j in combinations(range(n), 2):

        nPr = arr[i] \* arr[j]

        if nPr > max\_nPr:

            max\_nPr = nPr

            result = (arr[i], arr[j])

    return result

arr = [5, 2, 3, 4, 1]

print(max\_nPr\_pair(arr))  # Output: (5, 4)

# %%

def min\_operations\_to\_sort(arr):

    n = len(arr)

    operations = 0

    for i in range(1, n):

        if arr[i] < arr[i - 1]:

            operations += arr[i - 1] - arr[i]

            arr[i] = arr[i - 1]

    return operations

arr1 = [1, 2, 1, 4, 3]

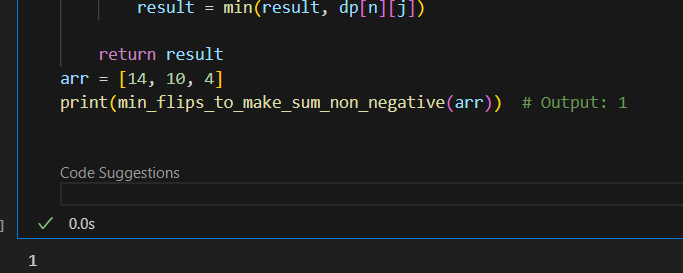
arr2 = [1, 2, 2, 100]

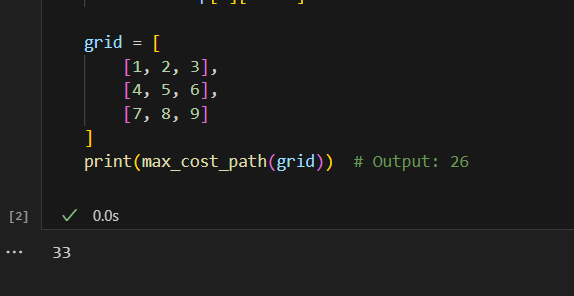
print(min\_operations\_to\_sort(arr1))  # Output: 2

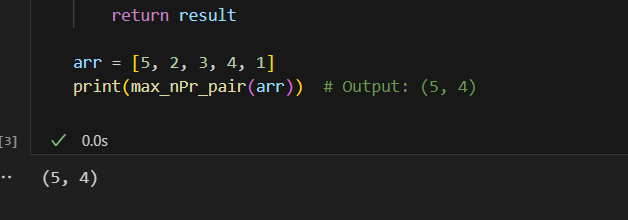
print(min\_operations\_to\_sort(arr2))  # Output: 0

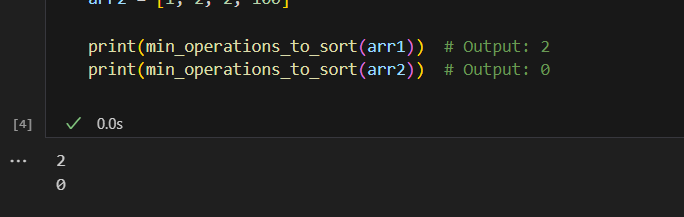
# %%

***Output :***

******

******

******

******

**B.2 Observations and learning:**

understood working of matrix, find minimum and maximum cost paths

**B.3 Conclusion:**

understood working of matrix, find minimum and maximum cost paths