**PART A**

(PART A : TO BE REFFERED BY STUDENTS)

**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. 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. C056 | Name: Malav Mehta |
| Class : B | Batch : EB2 |
| Date of Experiment: 09/01/24 | Date of Submission: 09/01/24 |
| Grade : |  |

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

***(Paste your code completed during the 2 hours of practical in the lab here)***

**CODE 1 :**

def min\_operations\_to\_balance(nums, length):

memo = [[0 for i in range(2000)] for i in range(2000)]

toggle\_flag = 1

total\_sum = 0

for i in range(length):

total\_sum += nums[i]

for i in range(-total\_sum, total\_sum + 1):

memo[0][i] = 10\*\*9

memo[0][0] = 0

for i in range(1, length + 1):

for j in range(-total\_sum, total\_sum + 1):

memo[toggle\_flag][j] = 10\*\*9

if (j - nums[i - 1] <= total\_sum and j - nums[i - 1] >= -total\_sum):

memo[toggle\_flag][j] = memo[toggle\_flag ^ 1][j - nums[i - 1]]

if (j + nums[i - 1] <= total\_sum

and j + nums[i - 1] >= -total\_sum

and memo[toggle\_flag ^ 1][j + nums[i - 1]] != 10\*\*9):

memo[toggle\_flag][j] = min(memo[toggle\_flag][j],

memo[toggle\_flag ^ 1][j + nums[i - 1]] + 1)

toggle\_flag = toggle\_flag ^ 1

for i in range(total\_sum + 1):

if (memo[toggle\_flag ^ 1][i] != 10\*\*9):

return memo[toggle\_flag ^ 1][i]

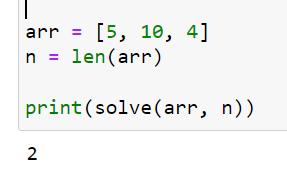
return length - 1

numbers = [5, 10, 4]

array\_length = len(numbers)

print(min\_operations\_to\_balance(numbers, array\_length))

**OUTPUT:-**



--------------------------------------------------------------------------------------------------------------------------------------------

**CODE 2:**

from collections import deque as queue

NUM\_ROWS = 3

NUM\_COLS = 3

def is\_valid(position):

return (position[0] >= 0) and (position[1] < NUM\_COLS)

def find\_max\_cost(matrix):

max\_values = [[0 for i in range(NUM\_COLS)] for i in range(NUM\_ROWS)]

max\_values[NUM\_ROWS - 1][0] = matrix[NUM\_ROWS - 1][0]

source = [NUM\_ROWS - 1, 0]

q = queue()

q.appendleft(source)

while len(q) > 0:

current = q.pop()

up = [current[0] - 1, current[1]]

if is\_valid(up):

max\_values[up[0]][up[1]] = max(max\_values[up[0]][up[1]], matrix[up[0]][up[1]] + max\_values[current[0]][current[1]])

q.appendleft(up)

right = [current[0], current[1] + 1]

if is\_valid(right):

max\_values[right[0]][right[1]] = max(max\_values[right[0]][right[1]], matrix[right[0]][right[1]] + max\_values[current[0]][current[1]])

q.appendleft(right)

return max\_values[0][NUM\_COLS - 1]

matrix\_input = [

[20, -10, 0],

[1, 5, 10],

[1, 2, 3]

]

print("Given matrix is ")

for i in range(NUM\_ROWS):

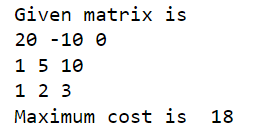
for j in range(NUM\_COLS):

print(matrix\_input[i][j], end=" ")

print()

print("Maximum cost is ", find\_max\_cost(matrix\_input))

**OUTPUT:-**

****

----------------------------------------------------------------------------------------------------

**CODE 3:**

def find\_max\_npr(elements):

elements.sort(reverse=True)

max\_base, max\_exponent = elements[0], elements[1]

max\_npr\_result = max\_base \*\* max\_exponent

for base in range(max\_base, 0, -1):

for exponent in range(max\_exponent, 0, -1):

npr\_result = base \*\* exponent

if npr\_result > max\_npr\_result:

max\_base, max\_exponent = base, exponent

max\_npr\_result = npr\_result

return max\_base, max\_exponent

elements\_list = [5, 2, 3, 4, 1]

print(find\_max\_npr(elements\_list))

**OUTPUT:-**



-------------------------------------------------------------------------------------------------------------------------------

**CODE 4:**

from collections import defaultdict

def minimum\_operations(input\_array):

occurrence\_dict = defaultdict(list)

for index, element in enumerate(input\_array):

occurrence\_dict[element].append(index)

queue = []

visited\_set = set()

operation\_count = 0

for i in range(len(input\_array) - 1):

if input\_array[i] not in visited\_set and input\_array[i] != 0:

queue.append(input\_array[i])

visited\_set.add(input\_array[i])

if input\_array[i] > input\_array[i + 1]:

operation\_count += len(queue)

while queue:

top = queue.pop(0)

if top in occurrence\_dict:

# Set all occurrences of previously assigned zero elements to zero

for idx in occurrence\_dict[top]:

input\_array[idx] = 0

return operation\_count

input\_arr = [4, 1, 5, 3, 2]

print(minimum\_operations(input\_arr))

**OUTPUT:-**



**B.2 Observations and learning:**

***(Students are expected to comment on the output obtained with clear observations and learning for each task/ sub part assigned)***

By means of this experiment, I learnt to perform matrix operation, find minimum cost path, find maximum in an integer array, and array sorting.

**B.3 Conclusion:**

*(****Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.3)***

In this experiment, by solving 4 problems, I learnt about matrices and their operations and finding min cost path, max in integer array and sorting an array accordingly.