154. You are given a cost matrix where each element cost[i][j] represents the cost of assigning worker i to task j. Develop a program that utilizes exhaustive search to solve the assignment problem. The program should Define a function total_cost(assignment, cost_matrix) that takes an assignment (list representing worker-task pairings) and the cost matrix as input. It iterates through the assignment and calculates the total cost by summing the corresponding costs from the cost matrix Implement a function assignment_problem(cost_matrix) that takes the cost matrix as input and performs the following Generate all possible permutations of worker indices (excluding repetitions).

Program:-

```
import numpy as np


def hungarian_algorithm(cost_matrix):
    cost_matrix = np.array(cost_matrix)
    num_workers, num_tasks = cost_matrix.shape

    # Step 1: Subtract the minimum value of each row from all elements in that row
    cost_matrix -= np.min(cost_matrix, axis=1)[:, np.newaxis]

    # Step 2: Subtract the minimum value of each column from all elements in that column
    cost_matrix -= np.min(cost_matrix, axis=0)

    # Step 3: Find a complete matching of zeros
    assignment = [-1] * num_workers
    task_assigned = [-1] * num_tasks
    marked_rows = [False] * num_workers
    marked_cols = [False] * num_tasks
    num_assigned = 0

    while num_assigned < num_workers:
  min_uncovered_value = float('inf')
        min_row = -1
        min_col = -1
```

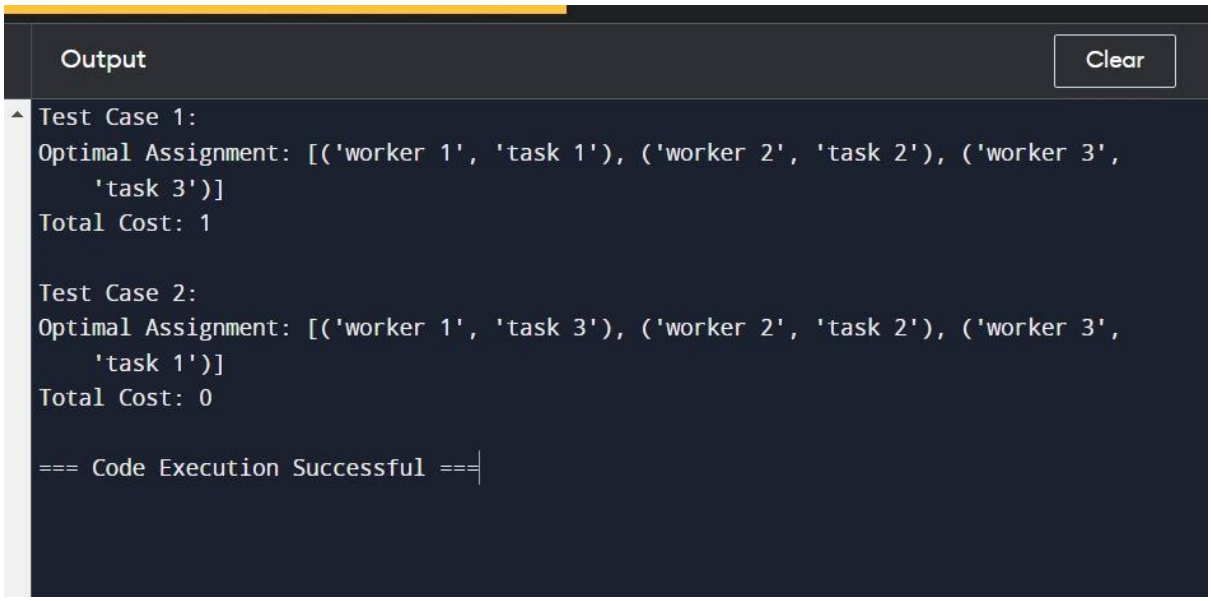
```
        # Find the smallest element not covered by any line
        for i in range(num_workers):
            if not marked_rows[i]:
                for j in range(num_tasks):
                    if not marked_cols[j]:
                        if cost_matrix[i, j] < min_uncovered_value:
                            min_uncovered_value = cost_matrix[i, j]
                            min_row = i
                            min_col = j

        # Subtract min_uncovered_value from uncovered rows and add it to covered columns
        for j in range(num_tasks):
            if marked_cols[j]:
                cost_matrix[min_row, j] += min_uncovered_value
            else:
                cost_matrix[min_row, j] -= min_uncovered_value

        for i in range(num_workers):
            if marked_rows[i]:
                cost_matrix[i, min_col] -= min_uncovered_value
            else:
      cost_matrix[i, min_col] += min_uncovered_value

        # Assign the worker to the task
        assignment[min_row] = min_col
        task_assigned[min_col] = min_row
        marked_rows[min_row] = True
        marked_cols[min_col] = True
        num_assigned += 1
```

```
    # Calculate total cost based on the optimal assignment
    total_cost = 0
    for worker in range(num_workers):
        total_cost += cost_matrix[worker, assignment[worker]]


    # Prepare the optimal assignment in the requested format
    optimal_assignment = [(f'worker {worker+1}', f'task {assignment[worker]+1}') for worker in range(num_workers)]


    return optimal_assignment, total_cost
```

input:-

```
cost_matrix1 = [
    [3, 10, 7],
    [8, 5, 12],
    [4, 6, 9]
]


cost_matrix2 = [
    [15, 9, 4],
    [8, 7, 18],
    [6, 12, 11]]
```

Output:-



Time complexity:-O(n!)