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).
Test Cases:
Input
Simple Case: Cost Matrix:
[[3, 10, 7],
[8, 5, 12],
[4, 6, 9]]
More Complex Case: Cost Matrix:
[[15, 9, 4],
[8, 7, 18],
[6, 12, 11]]
Output:
Test Case 1:
Optimal Assignment: [(worker 1, task 2), (worker 2, task 1), (worker 3, task 3)]
Total Cost: 19

AIM: To solve the assignment problem by implementing exhaustive search

PROGRAM:

```
import itertools

def total_cost(assignment, cost_matrix):
    """ Calculate the total cost of an assignment based on the cost matrix. """
    total = 0
    for worker, task in assignment:
        total += cost_matrix[worker][task]
    return total

def assignment_problem(cost_matrix):
    n = len(cost_matrix)
    if n == 0:
        return [], float('inf')
    worker_indices = list(range(n))
    all_permutations = itertools.permutations(worker_indices)

    min_cost = float('inf')
    optimal_assignment = None

    for perm in all_permutations:
        assignment = [(worker, perm[worker]) for worker in range(n)]
        current_cost = total_cost(assignment, cost_matrix)
        if current_cost < min_cost:
```

```
            min_cost = current_cost
            optimal_assignment = assignment

    return optimal_assignment, min_cost

def test_assignment_problem(cost_matrix, case_name):
    print(f"Test Case {case_name}:")
    print(f"Cost Matrix:")
    for row in cost_matrix:
        print(row)
    optimal_assignment, total_cost = assignment_problem(cost_matrix)
    print(f"Optimal Assignment: {[(f'worker {worker+1}', f'task {task+1}') for worker, task in
optimal_assignment]}")
    print(f"Total Cost: {total_cost}\n")

cost_matrix1 = [
    [3, 10, 7],
    [8, 5, 12],
    [4, 6, 9]
]
test_assignment_problem(cost_matrix1, 1)
```

OUTPUT:

```
Cost Matrix:
[3, 10, 7]
[8, 5, 12]
[4, 6, 9]
Optimal Assignment: [('worker 1', 'task 3'),
    ('worker 2', 'task 2'), ('worker 3', 'task
    1')]
Total Cost: 16
```

TIME COMPLEXITY: O(n!*n)