ChatGPT | DAY-4[1].pdf | Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

main.py | Share | Run

```
1  import math
2  def euclidean_distance(point1, point2):
3      return math.sqrt((point1[0] - point2[0])**2 + (point1[1] -
           point2[1])**2)
4  def closest_pair_brute_force(points):
5      min_distance = float('inf')
6      closest_pair = None
7      for i in range(len(points)):
8          for j in range(i + 1, len(points)):
9              distance = euclidean_distance(points[i], points[j])
10             if distance < min_distance:
11                 min_distance = distance
12                 closest_pair = (points[i], points[j])
13     return closest_pair, min_distance
14 points = [(1, 2), (4, 6), (7, 1), (3, 4), (5, 9)]
15 closest_pair, min_distance = closest_pair_brute_force(points)
16 print(f"The closest pair of points is: {closest_pair} with a distance
       of {min_distance}")
17
```

Output | Clear

```
The closest pair of points is: ((4, 6), (3, 4)) with a distance of 2
    .23606797749979

=== Code Execution Successful ===
```


88°F Partly sunny | Search | ENG IN | 15:54 17-10-2024

DAY-4[1].pdf | Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

main.py | Share | Run

```python
import math
def euclidean_distance(point1, point2):
    return math.sqrt((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)
def closest_pair_brute_force(points):
    min_distance = float('inf')
    closest_pair = None
    for i in range(len(points)):
        for j in range(i + 1, len(points)):
            distance = euclidean_distance(points[i], points[j])
            if distance < min_distance:
                min_distance = distance
                closest_pair = (points[i], points[j])

    return closest_pair, min_distance
points = [(1, 2), (4, 6), (7, 1), (3, 4), (5, 9)]
closest_pair, min_distance = closest_pair_brute_force(points)
print(f"The closest pair of points is: {closest_pair} with a distance of {min_distance}")

```

Output | Clear

```
The closest pair of points is: ((4, 6), (3, 4)) with a distance of 2
    .23606797749979

=== Code Execution Successful ===
```


88°F Partly sunny | Search | ENG IN | 16:04 17-10-2024

DAY-4[1].pdf | Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

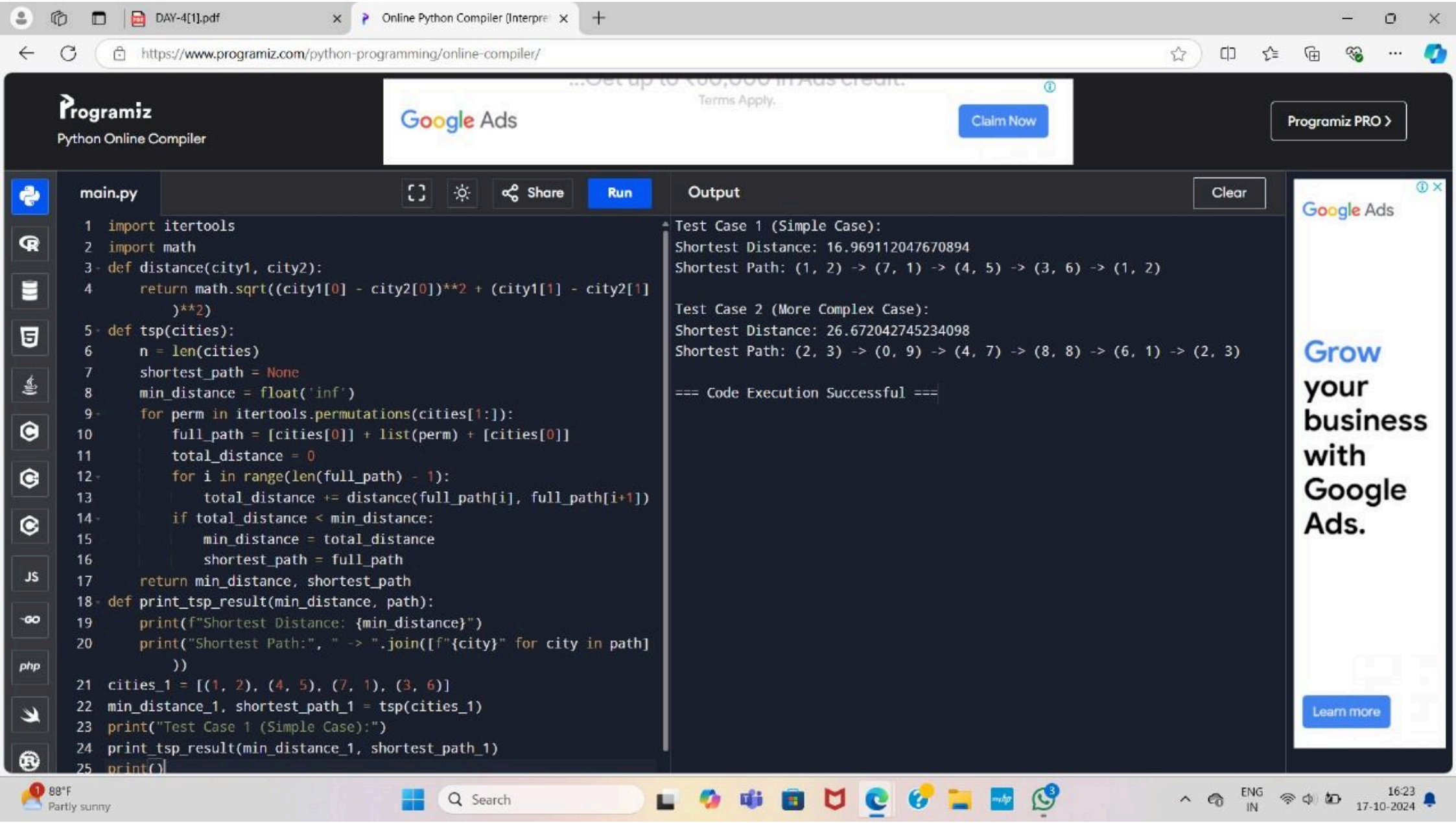
main.py | Share | Run

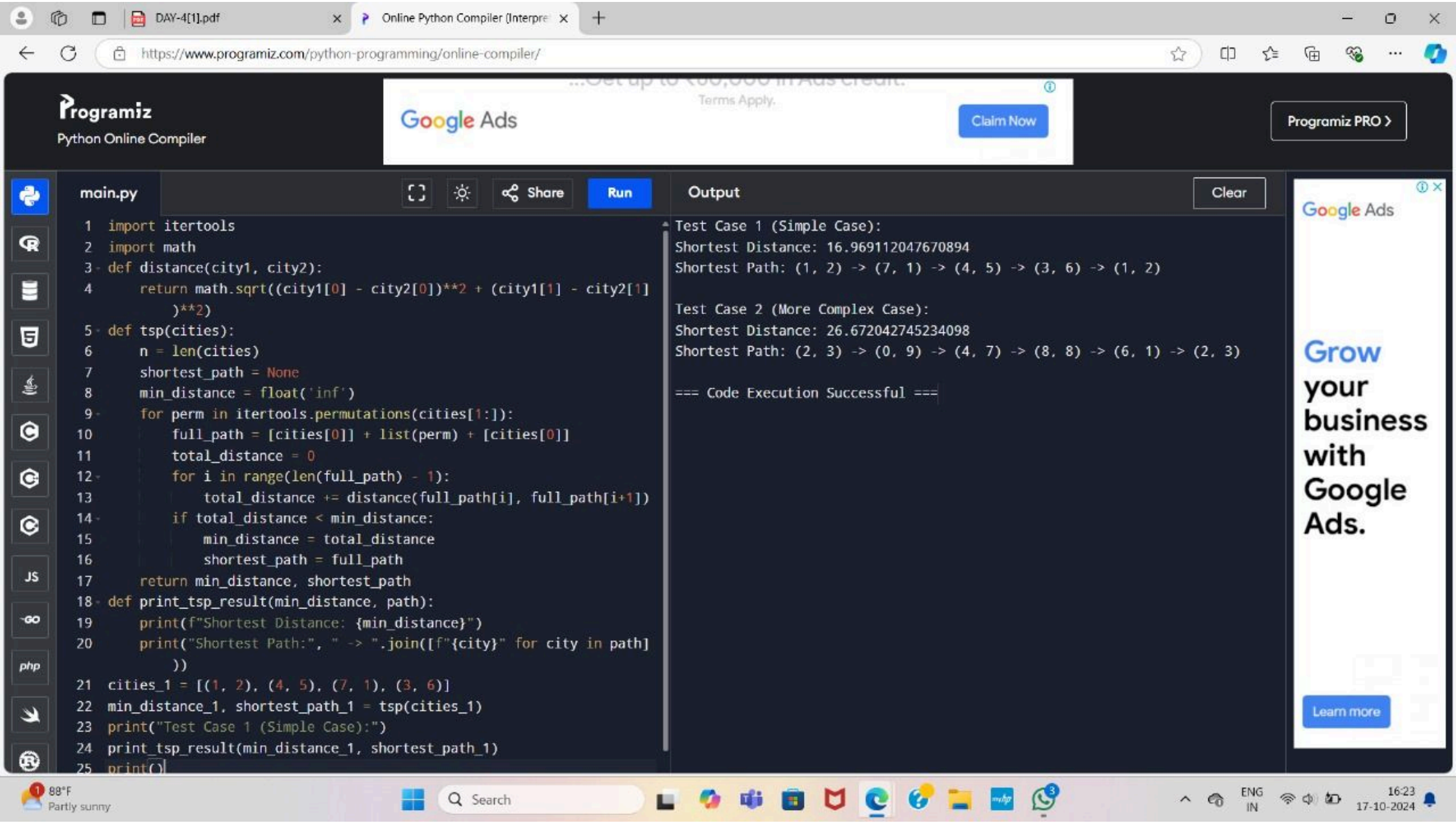
```
    left = right = False
    for p in points:
        if p == p1 or p == p2:
            continue
        orient = orientation(p1, p2, p)
        if orient == -1:
            left = True
        elif orient == 1:
            right = True
        if left and right:
            return False
    return True
def convex_hull_brute_force(points):
    hull = set()

    for i in range(len(points)):
        for j in range(i + 1, len(points)):
            if is_convex_edge(points, points[i], points[j]):
                hull.add(points[i])
                hull.add(points[j])
    return sorted(hull)
points = [(10, 0), (11, 5), (5, 3), (9, 3.5), (15, 3), (12.5, 7), (6
    , 6.5), (7.5, 4.5)]
convex_hull = convex_hull_brute_force(points)
print("The convex hull points are:", convex_hull)

```

Output | Clear

```
The convex hull points are: [(5, 3), (6, 6.5), (10, 0), (12.5, 7), (15, 3)]

=== Code Execution Successful ===
```


88°F Partly sunny | Search | ENG IN | 16:12 17-10-2024


DAY-4[1].pdf
Online Python Compiler (Interpre
https://www.programiz.com/python-programming/online-compiler/
Programiz
Python Online Compiler
Google Ads
Terms Apply.
Claim Now
Programiz PRO
main.py
Share
Run
Output
Clear
1 import itertools
2 import math
3 def distance(city1, city2):
4 return math.sqrt((city1[0] - city2[0])**2 + (city1[1] - city2[1])**2)
5 def tsp(cities):
6 n = len(cities)
7 shortest_path = None
8 min_distance = float('inf')
9 for perm in itertools.permutations(cities[1:]):
10 full_path = [cities[0]] + list(perm) + [cities[0]]
11 total_distance = 0
12 for i in range(len(full_path) - 1):
13 total_distance += distance(full_path[i], full_path[i+1])
14 if total_distance < min_distance:
15 min_distance = total_distance
16 shortest_path = full_path
17 return min_distance, shortest_path
18 def print_tsp_result(min_distance, path):
19 print(f"Shortest Distance: {min_distance}")
20 print("Shortest Path:", " -> ".join([f"{city}" for city in path]))
21 cities_1 = [(1, 2), (4, 5), (7, 1), (3, 6)]
22 min_distance_1, shortest_path_1 = tsp(cities_1)
23 print("Test Case 1 (Simple Case):")
24 print_tsp_result(min_distance_1, shortest_path_1)
25 print()
Test Case 1 (Simple Case):
Shortest Distance: 16.969112047670894
Shortest Path: (1, 2) -> (7, 1) -> (4, 5) -> (3, 6) -> (1, 2)
Test Case 2 (More Complex Case):
Shortest Distance: 26.672042745234098
Shortest Path: (2, 3) -> (0, 9) -> (4, 7) -> (8, 8) -> (6, 1) -> (2, 3)
=== Code Execution Successful ===
Google Ads
Grow your business with Google Ads.
Learn more
88°F
Partly sunny
Search
ENG
IN
16:23
17-10-2024


DAY-4[1].pdf
Online Python Compiler (Interpre
https://www.programiz.com/python-programming/online-compiler/
Programiz
Python Online Compiler
Google Ads
Terms Apply.
Claim Now
Programiz PRO
main.py
Share
Run
Output
Clear
1 import itertools
2 import math
3 def distance(city1, city2):
4 return math.sqrt((city1[0] - city2[0])**2 + (city1[1] - city2[1])**2)
5 def tsp(cities):
6 n = len(cities)
7 shortest_path = None
8 min_distance = float('inf')
9 for perm in itertools.permutations(cities[1:]):
10 full_path = [cities[0]] + list(perm) + [cities[0]]
11 total_distance = 0
12 for i in range(len(full_path) - 1):
13 total_distance += distance(full_path[i], full_path[i+1])
14 if total_distance < min_distance:
15 min_distance = total_distance
16 shortest_path = full_path
17 return min_distance, shortest_path
18 def print_tsp_result(min_distance, path):
19 print(f"Shortest Distance: {min_distance}")
20 print("Shortest Path:", " -> ".join([f"{city}" for city in path]))
21 cities_1 = [(1, 2), (4, 5), (7, 1), (3, 6)]
22 min_distance_1, shortest_path_1 = tsp(cities_1)
23 print("Test Case 1 (Simple Case):")
24 print_tsp_result(min_distance_1, shortest_path_1)
25 print()
Test Case 1 (Simple Case):
Shortest Distance: 16.969112047670894
Shortest Path: (1, 2) -> (7, 1) -> (4, 5) -> (3, 6) -> (1, 2)
Test Case 2 (More Complex Case):
Shortest Distance: 26.672042745234098
Shortest Path: (2, 3) -> (0, 9) -> (4, 7) -> (8, 8) -> (6, 1) -> (2, 3)
=== Code Execution Successful ===
Google Ads
Grow your business with Google Ads.
Learn more
88°F
Partly sunny
Search
ENG
IN
16:23
17-10-2024

DAY-4[1].pdf | Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

Programiz PRO >

main.py | Share | Run

```
    n = len(cities)
    shortest_path = None
    min_distance = float('inf')
    for perm in itertools.permutations(cities[1:]):
        full_path = [cities[0]] + list(perm) + [cities[0]]
        total_distance = 0
        for i in range(len(full_path) - 1):
            total_distance += distance(full_path[i], full_path[i+1])
        if total_distance < min_distance:
            min_distance = total_distance
            shortest_path = full_path
    return min_distance, shortest_path
def print_tsp_result(min_distance, path):
    print(f"Shortest Distance: {min_distance}")
    print("Shortest Path:", " -> ".join([f"{city}" for city in path]
        ))
cities_1 = [(1, 2), (4, 5), (7, 1), (3, 6)]
min_distance_1, shortest_path_1 = tsp(cities_1)
print("Test Case 1 (Simple Case):")
print_tsp_result(min_distance_1, shortest_path_1)
print()
cities_2 = [(2, 3), (8, 8), (0, 9), (4, 7), (6, 1)]
min_distance_2, shortest_path_2 = tsp(cities_2)
print("Test Case 2 (More Complex Case):")
print_tsp_result(min_distance_2, shortest_path_2)

```

Output | Clear

```
Test Case 1 (Simple Case):
Shortest Distance: 16.969112047670894
Shortest Path: (1, 2) -> (7, 1) -> (4, 5) -> (3, 6) -> (1, 2)

Test Case 2 (More Complex Case):
Shortest Distance: 26.672042745234098
Shortest Path: (2, 3) -> (0, 9) -> (4, 7) -> (8, 8) -> (6, 1) -> (2, 3)

=== Code Execution Successful ===
```


88°F Partly sunny | Search | ENG IN | 16:23 17-10-2024

DAY-4[1].pdf | Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

Programiz PRO >

main.py | Share | Run

```
import itertools
def total_cost(assignment, cost_matrix):
    total = 0
    for worker, task in enumerate(assignment):
        total += cost_matrix[worker][task]
    return total
def assignment_problem(cost_matrix):
    num_workers = len(cost_matrix)
    all_tasks = list(range(num_workers))
    min_cost = float('inf')
    best_assignment = None
    for assignment in itertools.permutations(all_tasks):
        current_cost = total_cost(assignment, cost_matrix)
        if current_cost < min_cost:
            min_cost = current_cost
            best_assignment = assignment
    return min_cost, best_assignment
if __name__ == "__main__":
    cost_matrix_1 = [
        [9, 2, 7],
        [6, 4, 3],
        [5, 8, 1]
    ]
    min_cost_1, best_assignment_1 = assignment_problem(cost_matrix_1
        )
    print("Test Case 1:")
    print(f"Minimum cost: {min_cost_1}")
```

Output | Clear

```
Test Case 1:
Minimum cost: 9
Best assignment: (1, 0, 2)

Test Case 2:
Minimum cost: 140
Best assignment: (2, 1, 0, 3)

=== Code Execution Successful ===
```


88°F Partly sunny | Search | ENG IN | 16:26 17-10-2024

DAY-4[1].pdf | Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

Programiz PRO >

main.py | Share | Run

```
            best_assignment = assignment
    return min_cost, best_assignment
if __name__ == "__main__":
    cost_matrix_1 = [
        [9, 2, 7],
        [6, 4, 3],
        [5, 8, 1]
    ]
    min_cost_1, best_assignment_1 = assignment_problem(cost_matrix_1
        )
    print("Test Case 1:")
    print(f"Minimum cost: {min_cost_1}")
    print(f"Best assignment: {best_assignment_1}")
    print()
    cost_matrix_2 = [
        [82, 83, 69, 92],
        [77, 37, 49, 92],
        [11, 69, 5, 86],
        [8, 9, 98, 23]
    ]
    min_cost_2, best_assignment_2 = assignment_problem(cost_matrix_2
        )
    print("Test Case 2:")
    print(f"Minimum cost: {min_cost_2}")
    print(f"Best assignment: {best_assignment_2}")

```

Output | Clear

```
Test Case 1:
Minimum cost: 9
Best assignment: (1, 0, 2)

Test Case 2:
Minimum cost: 140
Best assignment: (2, 1, 0, 3)

=== Code Execution Successful ===
```


88°F Partly sunny | Search | ENG IN | 16:26 17-10-2024

DAY-4[1].pdf | Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

Programiz PRO >

main.py | Share | Run

```python
import itertools
def total_value(items, values):
    total = 0
    for i in items:
        total += values[i]
    return total
def is_feasible(items, weights, capacity):
    total_weight = 0
    for i in items:
        total_weight += weights[i]
    return total_weight <= capacity
def knapsack_exhaustive(weights, values, capacity):
    n = len(weights)
    best_value = 0
    best_combination = []
    for r in range(n + 1):
        for combination in itertools.combinations(range(n), r):
            if is_feasible(combination, weights, capacity):
                current_value = total_value(combination, values)
                if current_value > best_value:
                    best_value = current_value
                    best_combination = combination

    return best_value, best_combination
if __name__ == "__main__":
    weights_1 = [2, 3, 4, 5]
    values_1 = [3, 4, 5, 6]
```

Output | Clear

```
Test Case 1:
Best Value: 7
Best Combination (item indices): (0, 1)

Test Case 2:
Best Value: 60
Best Combination (item indices): (0, 3)

=== Code Execution Successful ===
```


88°F Partly sunny | Search | ENG IN | 16:28 17-10-2024

DAY-4[1].pdf | Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

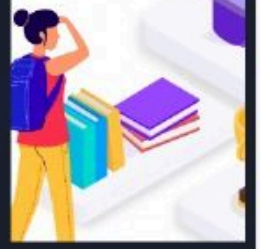
Programiz
Python Online Compiler

main.py | Share | Run

```
            if is_feasible(combination, weights, capacity):
                current_value = total_value(combination, values)
                if current_value > best_value:
                    best_value = current_value
                    best_combination = combination

    return best_value, best_combination
if __name__ == "__main__":
    weights_1 = [2, 3, 4, 5]
    values_1 = [3, 4, 5, 6]
    capacity_1 = 5
    best_value_1, best_combination_1 = knapsack_exhaustive(weights_1
        , values_1, capacity_1)
    print("Test Case 1:")
    print(f"Best Value: {best_value_1}")
    print(f"Best Combination (item indices): {best_combination_1}")
    print()
    weights_2 = [1, 2, 3, 8, 7, 4]
    values_2 = [20, 5, 10, 40, 15, 25]
    capacity_2 = 10
    best_value_2, best_combination_2 = knapsack_exhaustive(weights_2
        , values_2, capacity_2)
    print("Test Case 2:")
    print(f"Best Value: {best_value_2}")
    print(f"Best Combination (item indices): {best_combination_2}")

```

Output | Clear

```
Test Case 1:
Best Value: 7
Best Combination (item indices): (0, 1)

Test Case 2:
Best Value: 60
Best Combination (item indices): (0, 3)

=== Code Execution Successful ===
```

88°F Partly sunny | Search | ENG IN | 16:28 17-10-2024