In [2]:

```
import random
import math

def objective_function(x):
    """The objective function to be maximized (you can replace this with your 
    return -x**2 + 4*x - 4

def hill_climbing(max_iterations=100, step_size=0.1):
    """Hill climbing algorithm."""
    current_solution = random.uniform(-10, 10)

    for _ in range(max_iterations):
        current_value = objective_function(current_solution)
        neighbor_solution = current_solution + random.uniform(-step_size, step_
        neighbor_value = objective_function(neighbor_solution)
        if neighbor_value > current_value:
            current_solution = neighbor_solution
        return current_solution, objective_function(current_solution)

if __name__ == "__main__":
    max_iterations = 100
    step_size = 0.1
    best_solution, best_value = hill_climbing(max_iterations, step_size)
    print(f"Best solution: {best_solution}")
    print(f"Best value: {best_value}")
```

```
Best solution: -1.7904850631351987
Best value: -14.367777013851052
```

In [1]:

```python
import itertools
import math
def distance(city1, city2):
    """Calculate the Euclidean distance between two cities."""
    x1, y1 = city1
    x2, y2 = city2
    return math.sqrt((x1 - x2)**2 + (y1 - y2)**2)
def total_distance(tour, cities):
    """Calculate the total distance of a tour."""
    total = 0
    for i in range(len(tour) - 1):
        total += distance(cities[tour[i]], cities[tour[i + 1]])
    total += distance(cities[tour[-1]], cities[tour[0]])
    return total
def traveling_salesman_bruteforce(cities):
    """Bruteforce solution for the Traveling Salesman Problem."""
    num_cities = len(cities)
    all_tours = list(itertools.permutations(range(num_cities)))
    best_tour = None
    best_distance = float('inf')
    for tour in all_tours:
        tour_distance = total_distance(tour, cities)
        if tour_distance < best_distance:
            best_distance = tour_distance
            best_tour = tour
    return best_tour, best_distance

if __name__ == "__main__":
    cities = [(0, 0), (1, 2), (2, 4), (3, 1)]
    best_tour, best_distance = traveling_salesman_bruteforce(cities)
    print("Best Tour:", best_tour)
    print("Best Distance:", best_distance)
```

```
Best Tour: (0, 1, 2, 3)
Best Distance: 10.79669127533634
```

In [2]:

```python
def towers_of_hanoi(n, source_rod, target_rod, auxiliary_rod):
    """Solve the Towers of Hanoi problem."""
    if n == 1:
        print(f"Move disk 1 from {source_rod} to {target_rod}")
        return
    towers_of_hanoi(n-1, source_rod, auxiliary_rod, target_rod)
    print(f"Move disk {n} from {source_rod} to {target_rod}")
    towers_of_hanoi(n-1, auxiliary_rod, target_rod, source_rod)
if __name__ == "__main__":
    num_disks = 3
    source_rod = "A"
    target_rod = "C"
    auxiliary_rod = "B"
    print(f"Solving Towers of Hanoi for {num_disks} disks:")
    towers_of_hanoi(num_disks, source_rod, target_rod, auxiliary_rod)
```

```
Solving Towers of Hanoi for 3 disks:
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

In [ ]: