The Traveling Salesman Problem

A quick breakdown of an approximate solution

Approximate Solution

Pseudocode

```
FUNCTION nearest_neighbor():
    start_node = randomly select a node from the graph keys
    num_nodes = length of the graph keys
    curr_node = start_node
    visited = set containing curr_node
    path = list containing curr_node
    total_cost = 0

    FOR _ in range(num_nodes):
        min_cost = positive infinity
        min_node = NULL

        FOR each u in neighbors of curr_node:
            IF u is not in visited AND the cost from curr_node to u is less than min_cost:
                min_cost = cost from curr_node to u
                min_node = u

        IF min_node is not NULL:
            curr_node = min_node
            add curr_node to visited
            add curr_node to the path
            add min_cost to total_cost

    add start_node to the end of the path
    add the cost from curr_node to start_node to total_cost

    RETURN path and total_cost
```

Approximation Strategies

Utilizing Greedy Local Choices

```
        FOR each u in neighbors of curr_node:
            IF u is not in visited AND the cost from curr_node to u is less than min_cost:
                min_cost = cost from curr_node to u
                min_node = u
```

- Greedily choose the shortest edge from the current node
- Increases efficiency, but lessens effectiveness

Run-time Analysis

- Dictionary:
    - k in d = O(1)
    - Get item = O(1)
    - Set item[key] = O(1)
- Outer Loop: O(N)
- Inner Loop: O(N)

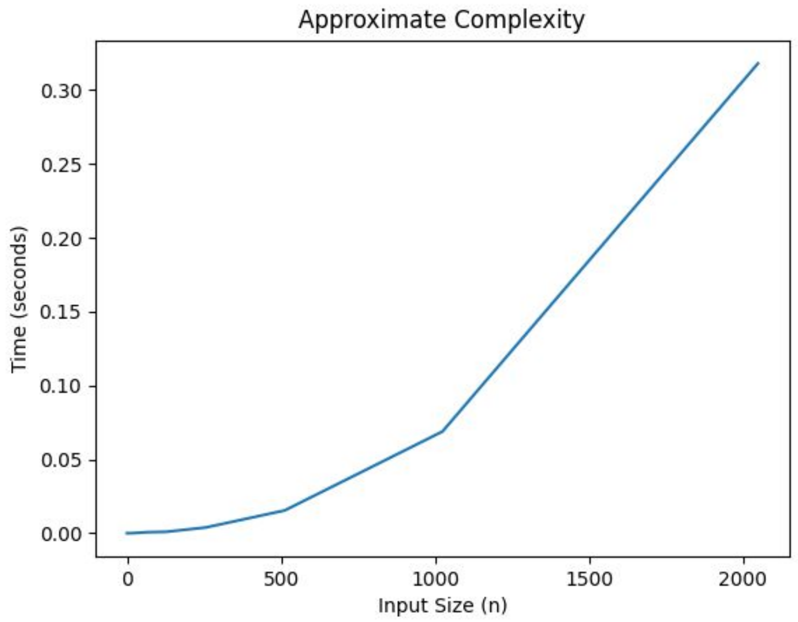
Total = $O(N^2)$

```
FUNCTION nearest_neighbor():
    start_node = randomly select a node from the graph keys
    num_nodes = length of the graph keys
    curr_node = start_node
    visited = set containing curr_node
    path = list containing curr_node
    total_cost = 0

    FOR _ in range(num_nodes):
        min_cost = positive infinity
        min_node = NULL

        FOR each u in neighbors of curr_node:
            IF u is not in visited AND the cost from curr_node to u is less than min_cost:
                min_cost = cost from curr_node to u
                min_node = u

        IF min_node is not NULL:
            curr_node = min_node
            add curr_node to visited
            add curr_node to the path
            add min_cost to total_cost

    add start_node to the end of the path
    add the cost from curr_node to start_node to total_cost

    RETURN path and total_cost

```

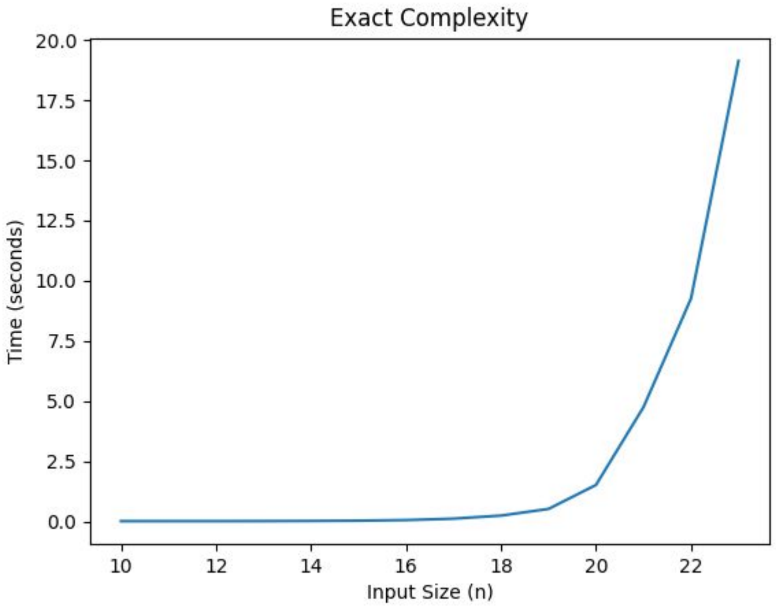
Lower Bound Analysis

- The lower bound is NP-hard because there is no known polynomial-time algorithm to solve it optimally for all cases.
- Our approximation algorithm aims to find solutions close to the exact.
- The best-known approximation has a ratio of 3/2.
- Verifying a solution's optimality requires exponential time due to the number of possible routes that grow exponentially with the number of cities.

Run-time



Approximate Complexity
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Time (seconds)
0
500
1000
1500
2000
Input Size (n)



Exact Complexity
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Time (seconds)
10
12
14
16
18
20
22
Input Size (n)

Comparing Results