Travelling Salesman Problem Considering the Weights of Vertices

Group 10

Xuenan Wang, Zhenyuan Zhang

Traveling Salesman Problem

- *The **travelling salesman problem** (**TSP**) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?"*

- Directed, complete, positive graph

Modified TSP

- In some cases in daily life, not only distance matters, we might also care about the weight a vehicle is carrying. (garbage collection and package delivery)

- TSP considers only the cost coming from distance

- Our problem also considers the cost from carrying mass while travelling on each edge(u, v):

  $\text{cost 1} = \text{distance(u, v)}$

  $\text{cost 2} = \frac{carried\ weight\ at\ u}{total\ weight} \times \text{distance(u, v)}$

- Total $cost = \sum[\text{distance} + \frac{carried\ weight}{total\ weight} \times distance]$

Algorithm (Brute Force)

- **Brute Force**

1) Set the starting point
2) Generate all permutations of visiting n vertices
3) Keep track of the minimum cost considering weight of both edges and vertices
4) Return the minimum cost and its path

- **Time Complexity:** $O(n!)$

Algorithm (Dynamic)

- Dynamic Programming

1) Set the starting point
2) Start with visiting v=2 vertices in total.(1 starting point and 1 other point)
3) Calculate the cost of each path and store the result.(All vertices visited already, the last vertex visited, total distance cost, total mass cost, mass carried)
4) If more than one path share the same visited vertices and last vertex, only keep the one with minimum cost
5) Continue with v=3 using the result of v=2
6) Repeat the previous steps for v=4, 5, 6… until v=n
7) The last step is adding the edge from the end point back to the starting point and find the minimum cost
8) Return the minimum cost and its path

- Time Complexity: $O(n^2 \times 2^n)$

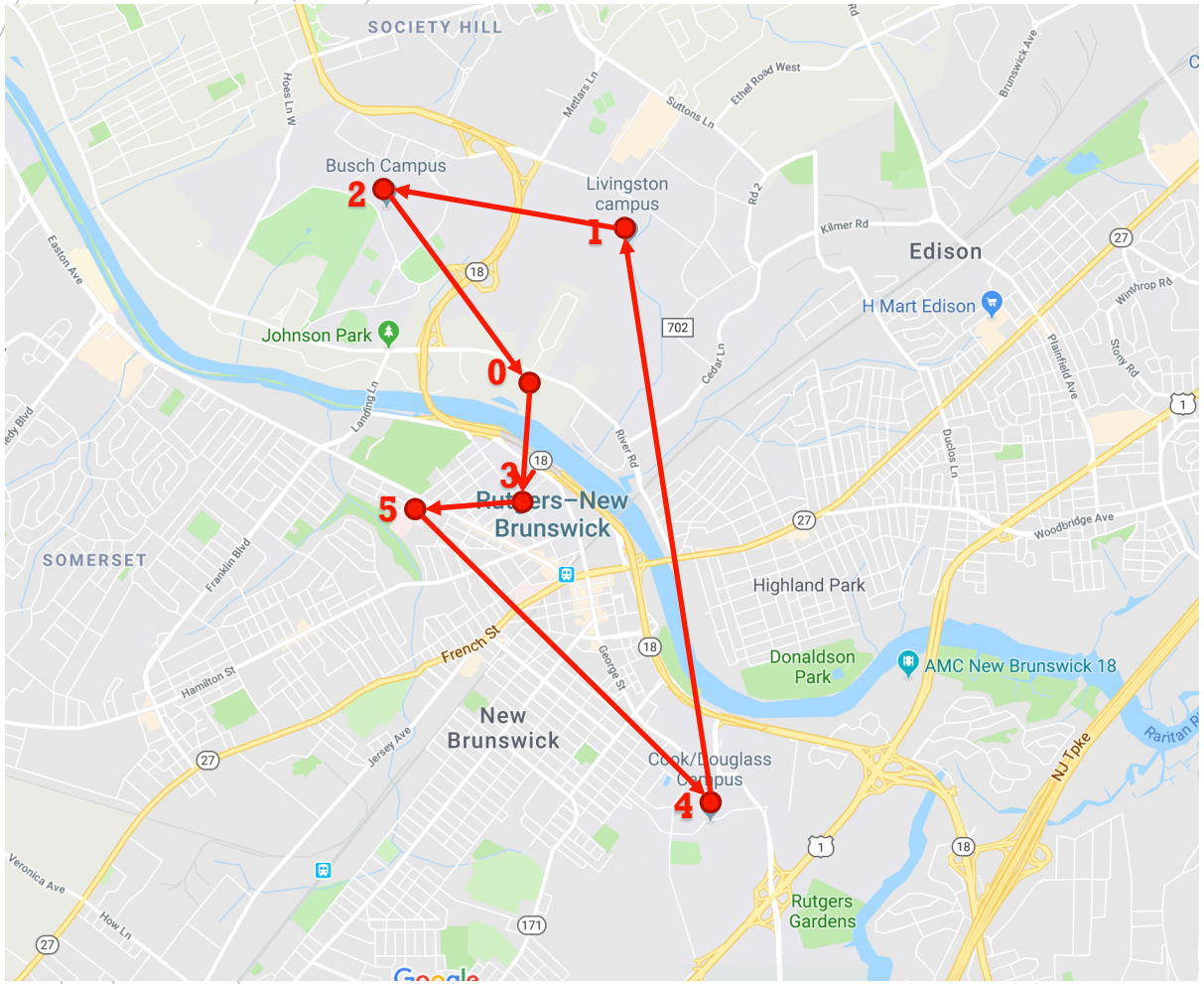
Algorithm (Greedy)

- Greedy Algorithm

1) Set the starting point and let end point to be the same point
2) Among all the edges coming out of the end point, find the one with minimum total cost expectation. Include that edge into the path and set the other node of the edge to be new end point
3) Repeat the last step with the new end point until all vertices are visited
4) Return the cost and the path

- Time Complexity: $O(n^2)$

Result without considering mass on nodes



node_weight: [0, 0, 0, 0, 0, 0]

self.permutations_list: [(1, 2, 3, 4, 5), (1, 2, 3, 5, 4), (1, 2, 4, 3, 5), (1, 2, 4, 5, 3), (1, 2, 5, 3, 4), (1, 2, 5, 4, 3), (1, 3, 2, 4, 5), (1, 3, 2, 5, 4), (1, 3, 4, 2, 5), (1, 3, 4, 5, 2), (1, 3, 5, 2, 4), (1, 3, 5, 4, 2), (1, 4, 2, 3, 5), (1, 4, 2, 5, 3), (1, 4, 3, 2, 5), (1, 4, 3, 5, 2), (1, 4, 5, 2, 3), (1, 4, 5, 3, 2), (1, 5, 2, 3, 4), (1, 5, 2, 4, 3), (1, 5, 3, 2, 4), (1, 5, 3, 4, 2), (1, 5, 4, 2, 3), (1, 5, 4, 3, 2), (2, 1, 3, 4, 5), (2, 1, 3, 5, 4), (2, 1, 4, 3, 5), (2, 1, 4, 5, 3), (2, 1, 5, 3, 4), (2, 1, 5, 4, 3), (2, 3, 1, 4, 5), (2, 3, 1, 5, 4), (2, 3, 4, 1, 5), (2, 3, 4, 5, 1), (2, 3, 5, 1, 4), (2, 3, 5, 4, 1), (2, 4, 1, 3, 5), (2, 4, 1, 5, 3), (2, 4, 3, 1, 5), (2, 4, 3, 5, 1), (2, 4, 5, 1, 3), (2, 4, 5, 3, 1), (2, 5, 1, 3, 4), (2, 5, 1, 4, 3), (2, 5, 3, 1, 4), (2, 5, 3, 4, 1), (2, 5, 4, 1, 3), (2, 5, 4, 3, 1), (3, 1, 2, 4, 5), (3, 1, 2, 5, 4), (3, 1, 4, 2, 5), (3, 1, 4, 5, 2), (3, 1, 5, 2, 4), (3, 1, 5, 4, 2), (3, 2, 1, 4, 5), (3, 2, 1, 5, 4), (3, 2, 4, 1, 5), (3, 2, 4, 5, 1), (3, 2, 5, 1, 4), (3, 2, 5, 4, 1), (3, 4, 1, 2, 5), (3, 4, 1, 5, 2), (3, 4, 2, 1, 5), (3, 4, 2, 5, 1), (3, 4, 5, 1, 2), (3, 4, 5, 2, 1), (3, 5, 1, 2, 4), (3, 5, 1, 4, 2), (3, 5, 2, 1, 4), (3, 5, 2, 4, 1), (3, 5, 4, 1, 2), (3, 5, 4, 2, 1), (4, 1, 2, 3, 5), (4, 1, 2, 5, 3), (4, 1, 3, 2, 5), (4, 1, 3, 5, 2), (4, 1, 5, 2, 3), (4, 1, 5, 3, 2), (4, 2, 1, 3, 5), (4, 2, 1, 5, 3), (4, 2, 3, 1, 5), (4, 2, 3, 5, 1), (4, 2, 5, 1, 3), (4, 2, 5, 3, 1), (4, 3, 1, 2, 5), (4, 3, 1, 5, 2), (4, 3, 2, 1, 5), (4, 3, 2, 5, 1), (4, 3, 5, 1, 2), (4, 3, 5, 2, 1), (4, 5, 1, 2, 3), (4, 5, 1, 3, 2), (4, 5, 2, 1, 3), (4, 5, 2, 3, 1), (4, 5, 3, 1, 2), (4, 5, 3, 2, 1), (5, 1, 2, 3, 4), (5, 1, 2, 4, 3), (5, 1, 3, 2, 4), (5, 1, 3, 4, 2), (5, 1, 4, 2, 3), (5, 1, 4, 3, 2), (5, 2, 1, 3, 4), (5, 2, 1, 4, 3), (5, 2, 3, 1, 4), (5, 2, 3, 4, 1), (5, 2, 4, 1, 3), (5, 2, 4, 3, 1), (5, 3, 1, 2, 4), (5, 3, 1, 4, 2), (5, 3, 2, 1, 4), (5, 3, 2, 4, 1), (5, 3, 4, 1, 2), (5, 3, 4, 2, 1), (5, 4, 1, 2, 3), (5, 4, 1, 3, 2), (5, 4, 2, 1, 3), (5, 4, 2, 3, 1), (5, 4, 3, 1, 2), (5, 4, 3, 2, 1)]

cost_array: [14.200000000000001, 12.9, 13.6, 14.2, 12.899999999999999, 14.799999999999997, 18.6, 17.900000000000002, 18.500000000000004, 16.700000000000003, 16.8, 16.5, 17.8, 18.4, 18.5, 15.999999999999998, 18.599999999999998, 16.5, 17.3, 18.6, 17.2, 16.400000000000002, 19.0, 17.1, 14.9, 13.6, 14.200000000000001, 14.8, 13.5, 15.399999999999999, 18.0, 17.3, 16.7, 15.8, 16.400000000000002, 14.2, 16.2, 16.7, 18.0, 15.200000000000001, 18.4, 15.5, 17.1, 18.3, 16.700000000000003, 14.200000000000001, 17.4, 16.1, 15.3, 14.6, 19.500000000000004, 17.700000000000003, 18.400000000000002, 18.1, 16.400000000000002, 15.700000000000001, 18.3, 17.400000000000002, 18.6, 16.400000000000002, 14.0, 16.400000000000002, 16.299999999999997, 17.299999999999997, 13.6, 14.0, 13.7, 17.200000000000003, 14.6, 15.3, 12.0, 13.799999999999999, 14.8, 15.399999999999999, 19.8, 17.3, 19.8, 17.700000000000003, 17.2, 17.7, 20.900000000000002, 18.1, 21.3, 18.4, 16.7, 19.1, 17.8, 18.799999999999997, 14.399999999999999, 14.799999999999999, 17.0, 19.400000000000002, 18.0, 18.6, 14.7, 15.299999999999999, 14.3, 15.600000000000001, 18.7, 17.900000000000002, 19.799999999999997, 17.900000000000002, 15.299999999999999, 16.5, 18.400000000000002, 15.9, 18.5, 17.2, 14.0, 17.500000000000004, 15.1, 15.8, 12.0, 13.799999999999999, 14.600000000000001, 17.0, 17.0, 17.6, 13.899999999999999, 14.5]

*********Brute Force********

minimum cost without node weights: 12.0

optimized sequence is: [3, 5, 4, 1, 2]

*********Dynamic********

minimum cost without node weights: 12.0

optimized sequence is: [3, 5, 4, 1, 2]

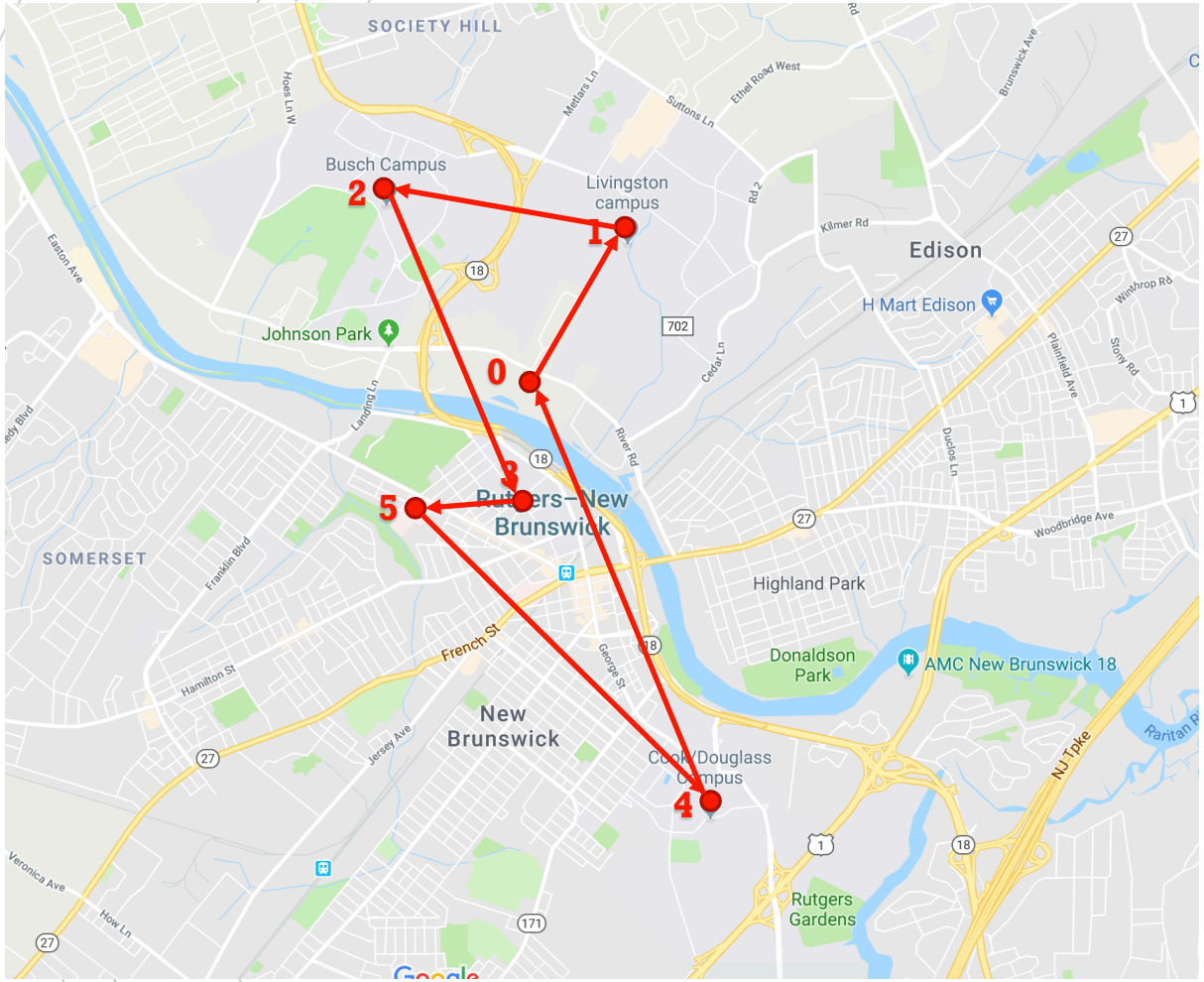
*********Greeedy********

minimum cost without node weights: 12.0

optimized sequence is: [3, 5, 4, 1, 2]

evaluation is: 0.0

Result considering mass on nodes



node_weight: [0, 1000, 640, 120, 560, 80]

self.permutations_list: [(1, 2, 3, 4, 5), (1, 2, 3, 5, 4), (1, 2, 4, 3, 5), (1, 2, 4, 5, 3), (1, 2, 5, 3, 4), (1, 2, 5, 4, 3), (1, 3, 2, 4, 5), (1, 3, 2, 5, 4), (1, 3, 4, 2, 5), (1, 3, 4, 5, 2), (1, 3, 5, 2, 4), (1, 3, 5, 4, 2), (1, 4, 2, 3, 5), (1, 4, 2, 5, 3), (1, 4, 3, 2, 5), (1, 4, 3, 5, 2), (1, 4, 5, 2, 3), (1, 4, 5, 3, 2), (1, 5, 2, 3, 4), (1, 5, 2, 4, 3), (1, 5, 3, 2, 4), (1, 5, 3, 4, 2), (1, 5, 4, 2, 3), (1, 5, 4, 3, 2), (2, 1, 3, 4, 5), (2, 1, 3, 5, 4), (2, 1, 4, 3, 5), (2, 1, 4, 5, 3), (2, 1, 5, 3, 4), (2, 1, 5, 4, 3), (2, 3, 1, 4, 5), (2, 3, 1, 5, 4), (2, 3, 4, 1, 5), (2, 3, 4, 5, 1), (2, 3, 5, 1, 4), (2, 3, 5, 4, 1), (2, 4, 1, 3, 5), (2, 4, 1, 5, 3), (2, 4, 3, 1, 5), (2, 4, 3, 5, 1), (2, 4, 5, 1, 3), (2, 4, 5, 3, 1), (2, 5, 1, 3, 4), (2, 5, 1, 4, 3), (2, 5, 3, 1, 4), (2, 5, 3, 4, 1), (2, 5, 4, 1, 3), (2, 5, 4, 3, 1), (3, 1, 2, 4, 5), (3, 1, 2, 5, 4), (3, 1, 4, 2, 5), (3, 1, 4, 5, 2), (3, 1, 5, 2, 4), (3, 1, 5, 4, 2), (3, 2, 1, 4, 5), (3, 2, 1, 5, 4), (3, 2, 4, 1, 5), (3, 2, 4, 5, 1), (3, 2, 5, 1, 4), (3, 2, 5, 4, 1), (3, 4, 1, 2, 5), (3, 4, 1, 5, 2), (3, 4, 2, 1, 5), (3, 4, 2, 5, 1), (3, 4, 5, 1, 2), (3, 4, 5, 2, 1), (3, 5, 1, 2, 4), (3, 5, 1, 4, 2), (3, 5, 2, 1, 4), (3, 5, 2, 4, 1), (3, 5, 4, 1, 2), (3, 5, 4, 2, 1), (4, 1, 2, 3, 5), (4, 1, 2, 5, 3), (4, 1, 3, 2, 5), (4, 1, 3, 5, 2), (4, 1, 5, 2, 3), (4, 1, 5, 3, 2), (4, 2, 1, 3, 5), (4, 2, 1, 5, 3), (4, 2, 3, 1, 5), (4, 2, 3, 5, 1), (4, 2, 5, 1, 3), (4, 2, 5, 3, 1), (4, 3, 1, 2, 5), (4, 3, 2, 1, 5), (4, 3, 2, 5, 1), (4, 3, 5, 1, 2), (4, 3, 5, 2, 1), (4, 5, 1, 2, 3), (4, 5, 1, 3, 2), (4, 5, 2, 1, 3), (4, 5, 2, 3, 1), (4, 5, 3, 1, 2), (4, 5, 3, 2, 1), (5, 1, 2, 3, 4), (5, 1, 2, 4, 3), (5, 1, 3, 2, 4), (5, 1, 3, 4, 2), (5, 1, 4, 2, 3), (5, 1, 4, 3, 2), (5, 2, 1, 3, 4), (5, 2, 1, 4, 3), (5, 2, 3, 1, 4), (5, 2, 3, 4, 1), (5, 2, 4, 1, 3), (5, 2, 4, 3, 1), (5, 3, 1, 2, 4), (5, 3, 1, 4, 2), (5, 3, 2, 1, 4), (5, 3, 2, 4, 1), (5, 3, 4, 1, 2), (5, 3, 4, 2, 1), (5, 4, 1, 2, 3), (5, 4, 1, 3, 2), (5, 4, 2, 1, 3), (5, 4, 2, 3, 1), (5, 4, 3, 1, 2), (5, 4, 3, 2, 1)]

cost_array: [19.656666666666666, 18.383333333333333, 19.013333333333335, 19.639999999999997, 18.41, 20.456666666666667, 26.659999999999997, 26.14666666666667, 26.576666666666664, 24.893333333333333, 24.696666666666665, 24.666666666666664, 25.733333333333334, 26.359999999999996, 26.525000000000002, 23.93166666666667, 26.686666666666667, 24.5916666666667, 25.433333333333334, 26.71, 25.315000000000005, 24.535, 27.4, 25.568333333333335, 20.313333333333336, 19.040000000000003, 19.538333333333338, 20.165, 18.935000000000002, 20.981666666666666, 25.835, 25.3216666666667, 24.253333333333334, 24.19, 24.193333333333335, 21.95, 23.546666666666667, 24.064999999999998, 26.115, 23.296666666666667, 26.686666666666664, 23.74833333333333, 25.246666666666666, 26.395, 24.715, 21.976666666666663, 25.406666666666666, 24.814999999999998, 22.215, 21.701666666666668, 28.725000000000005, 27.041666666666668, 27.765, 27.735, 23.685000000000002, 23.17166666666667, 27.490000000000006, 27.426666666666666, 28.440000000000005, 26.196666666666665, 21.33666666666667, 25.113333333333333, 24.609999999999996, 27.301666666666662, 22.403333333333336, 22.743333333333332, 20.696666666666667, 26.366666666666667, 21.74666666666667, 24.28833333333333, 19.736666666666668, 22.43333333333333, 23.070000000000004, 23.696666666666665, 29.666666666666668, 27.073333333333334, 29.693333333333335, 27.598333333333336, 26.566666666666666, 27.084999999999997, 31.984999999999996, 29.166666666666664, 32.556666666666665, 29.618333333333325, 26.345000000000002, 30.121666666666666, 27.558333333333334, 30.25, 24.150000000000002, 24.489999999999995, 26.84333333333333, 30.673333333333336, 27.83333333333333, 30.02833333333333, 24.68166666666667, 25.358333333333334, 21.53, 22.80666666666667, 28.38666666666667, 27.606666666666666, 29.386666666666667, 27.555000000000003, 22.75333333333333, 23.901666666666664, 28.171666666666667, 25.43333333333333, 27.843333333333334, 27.251666666666665, 21.298333333333336, 26.968333333333334, 22.731666666666667, 25.273333333333333, 19.763333333333335, 22.46, 22.496666666666666, 26.326666666666668, 25.91333333333333, 28.10833333333333, 23.108333333333334, 23.784999999999997]

**********Brute Force********

minimum cost with node weights: 18.383333333333333

optimized sequence is: [1, 2, 3, 5, 4]

**********Dynamic********

minimum cost without node weights: 18.383333333333333

optimized sequence is: [1, 2, 3, 5, 4]

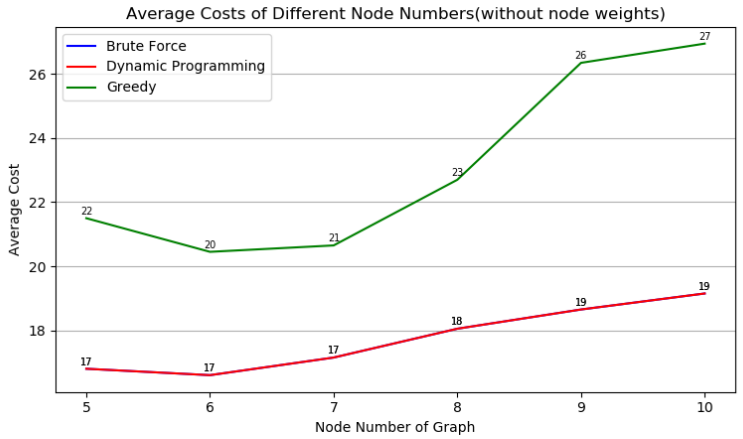
**********Greeedy********

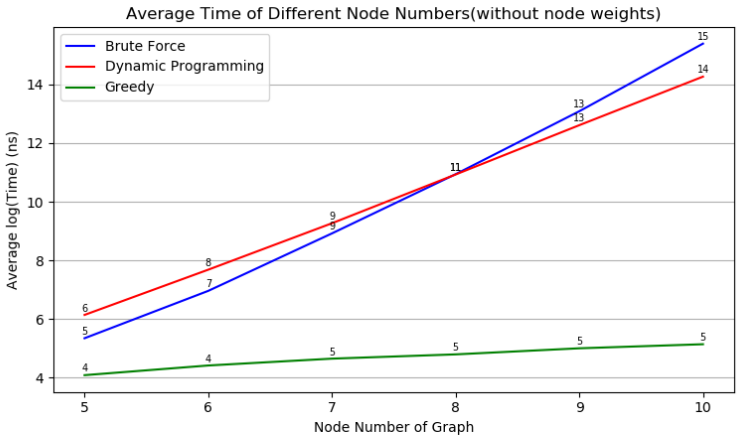
minimum cost with node weights: 22.46

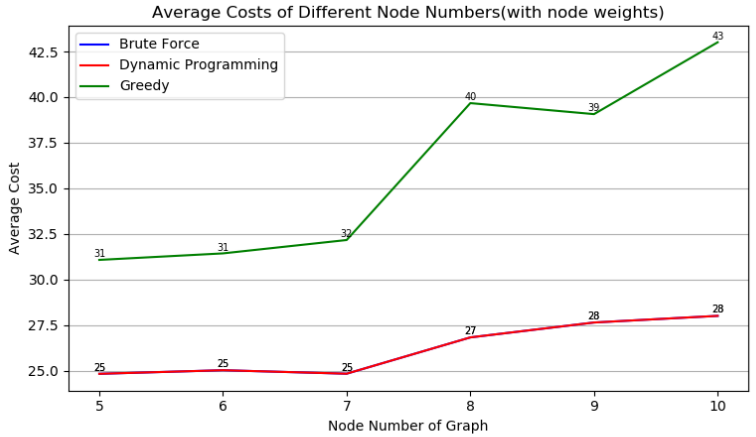
optimized sequence is: [5, 3, 4, 2, 1]

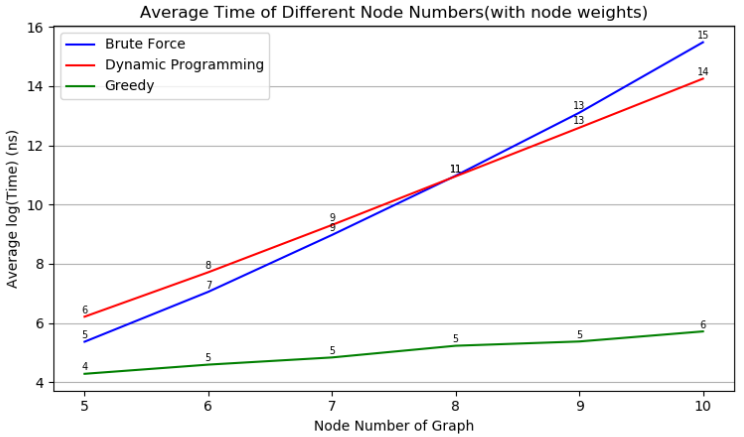
evaluation is: 0.20833333333333334

Greedy algorithm usually does not give best cost









Red curve shows that Dynamic algorithm has exponential complexity.

Thanks For Your Time