.cpp files and thier outputs are attached to the zip file

• O1. Analyze the theoretical time complexity of algorithms A and B using Big O.

SOLUTION:

ALGORITHM A:

Nearest Neighbor Algorithm:

- **Looping through nodes:**
  - The nearestNeighborAlgorithm() uses a while loop that executes N times, as each node is visited exactly once.
- **Finding the nearest neighbor:**
  - Inside the while loop, it iterates over all neighbors of the current vertex. In the worst case, a vertex might have N−1 neighbors. Thus, the complexity of finding the nearest unvisited neighbor for each node is O(N).
- **Total Complexity for Nearest Neighbor Search:**
  - Since there are N nodes and each requires an O(N) search through its neighbors, the total complexity of the nearest neighbor search is O(N^2).
- **Overall Time Complexity of algorithm A: O(N^2)** where N is the number of nodes in the graph.
  - Since there are M input files the total time complexity for processing all files becomes: **O(M * N^2)**

ALGORITHM B:

Nearest Insertion Algorithm:

- **Finding the Nearest Unvisited Node:**
  - For each unvisited node, the algorithm must compare distances to each node currently in the tour. This requires O(N) checks per unvisited node.
  - Since this nearest search is done across all unvisited nodes, it requires O(N^2) operations in total each time a node is added.
- **Finding the Best Position for Insertion:**
  - After finding the nearest unvisited node, the algorithm calculates the cost of inserting it between each pair of consecutive nodes in the tour. As the tour grows, this step takes O(N) time for each insertion position.
  - Since each addition to the tour involves calculating the best insertion position, and this is done for each of the N nodes being inserted, it results in O(N^2) operations per insertion.
  - Since both steps are performed within each insertion iteration, the total complexity for constructing the tour is O(N) * O(N^2) = O(N^3)
  - So, the overall time complexity for the Nearest Insertion Algorithm here is indeed: **O(N^3)**
  - Since there are M input files the total time complexity for processing all files becomes: **O(M * N^3)**

• O2. Fill in the required information (“Tour: Sequence of Nodes” and “Total Weight”) for each algorithm in this table.

Algorithm	File	Tour: Sequence of nodes	Total weight
Nearest Neighbour	4n	[0, 1, 2, 3, 0]	2611
	5n	[0, 3, 2, 4, 1, 0]	1290
	6n	[0, 5, 3, 4, 1, 2, 0]	3198
	7n	[0, 6, 4, 3, 1, 5, 2, 0]	2425
	8n	[0, 2, 4, 1, 6, 3, 5, 7, 0]	2898
	9n	[0, 8, 1, 6, 5, 2, 3, 7, 4, 0]	2842
	10n	[0, 7, 8, 4, 2, 6, 5, 3, 1, 9, 0]	3008
	11n	[0, 1, 6, 4, 8, 2, 9, 7, 5, 10, 3, 0]	3350
	12n	[0, 8, 3, 5, 6, 4, 7, 11, 10, 2, 1, 9, 0]	2646
	13n	[0, 9, 2, 11, 7, 12, 3, 6, 1, 5, 4, 8, 10, 0]	3653
Nearest Insertion	4n	[0, 3, 2, 1, 0]	2611
	5n	[2, 4, 1, 0, 3, 2]	1290
	6n	[1, 0, 5, 3, 2, 4, 1]	2455
	7n	[0, 1, 5, 2, 3, 4, 6, 0]	1993
	8n	[3, 7, 1, 6, 0, 2, 4, 5, 3]	2692
	9n	[1, 7, 3, 0, 8, 4, 2, 5, 6, 1]	2588
	10n	[2, 7, 0, 9, 1, 3, 8, 4, 5, 6, 2]	3532
	11n	[0, 3, 5, 2, 9, 7, 8, 4, 10, 6, 1, 0]	2867
	12n	[10, 6, 5, 1, 2, 3, 8, 9, 0, 4, 7, 11, 10]	2930
	13n	[1, 10, 7, 12, 2, 9, 0, 11, 8, 3, 6, 4, 5, 1]	3031
Dynamic Programming	4n	[0, 1, 2, 3, 0]	2611
	5n	[0, 1, 2, 4, 3, 0]	1146
	6n	[0, 1, 4, 2, 3, 5, 0]	2455
	7n	[0, 1, 5, 2, 3, 4, 6, 0]	1993
	8n	[0, 2, 4, 1, 7, 5, 3, 6, 0]	2667
	9n	[0, 3, 2, 4, 1, 6, 5, 7, 8, 0]	2439
	10n	[0, 5, 6, 2, 4, 9, 1, 3, 8, 7, 0]	2953
	11n	[0, 1, 6, 10, 5, 2, 9, 7, 8, 4, 3, 0]	2701
	12n	[0, 6, 4, 7, 5, 3, 8, 1, 2, 10, 11, 9, 0]	1907

	13n	[0, 11, 2, 9, 10, 7, 8, 4, 5, 1, 6, 3, 12, 0]	2523
Branch-and-Bound	4n	[0, 1, 2, 3, 0]	2611
	5n	[0, 3, 4, 2, 1, 0]	1146
	6n	[0, 5, 3, 2, 4, 1, 0]	2455
	7n	[0, 6, 4, 3, 2, 5, 1, 0]	1993
	8n	[0, 6, 3, 5, 7, 1, 4, 2, 0]	2667
	9n	[0, 8, 7, 5, 6, 1, 4, 2, 3, 0]	2439
	10n	[0, 7, 8, 3, 1, 9, 4, 2, 6, 5, 0]	2953
	11n	[0, 1, 6, 10, 5, 2, 9, 7, 8, 4, 3, 0]	2701
	12n	[0, 9, 11, 10, 2, 1, 8, 3, 5, 7, 4, 6, 0]	1907
	13n	[0, 11, 2, 9, 10, 7, 8, 4, 5, 1, 6, 3, 12, 0]	2523