Assignment 5

A02092613 Gattu Varun

**Exhaustive algorithm** – This algorithm does the depth first search and takes permutations of all the given input. It takes a very high time because it takes all the permutations. The time complexity for this algorithm is the highest and is hence not an efficient algorithm.

**Global upper bounds** – This algorithm has an upper bound which acts like a threshold value and when any input permutation crosses the upper bound, breaks abruptly and takes the next permutation. This algorithm works a little better compared to the exhaustive algorithm because few of the inputs are not considered.

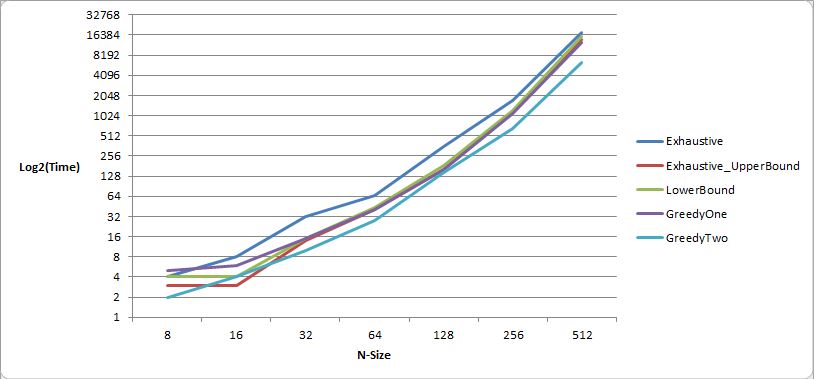
**Local lower bound** – This algorithm makes use of a lower bound which acts as a cut off and acts as a minimum time needed to complete the algorithm for any given input. This works similar to the upper bounds problem with minor deviations.

**Greedy** – This algorithm works keeping the remaining cities in an array list. It calculates the shortest distance from all the cities to other cities. This algorithm works the best compared to other algorithms because it highly reduces the number of permutations.

The following table describes the time complexity for every algorithm:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Exhaustive | Exhaustive\_UpperBound | LowerBound | GreedyOne | GreedyTwo |
| 8 | 4 | 3 | 4 | 5 | 2 |
| 16 | 8 | 3 | 4 | 6 | 4 |
| 32 | 32 | 14 | 15 | 15 | 10 |
| 64 | 65 | 40 | 43 | 41 | 28 |
| 128 | 356 | 163 | 182 | 159 | 143 |
| 256 | 1717 | 1154 | 1232 | 1089 | 655 |
| 512 | 18025 | 14025 | 15444 | 12688 | 6452 |

The graph for the same algorithm looks like:



The exhaustive algorithm takes O(n2\*2n) time.

**Complexity recurrence relation:**

If size of Cities is 2, then Cities must be {1, i},

TSP(Cities, i) = dist(1, i)

Else if size of Cities is greater than 2.

TSP(Cities, i) = min { TSP(Cities-{i}, j) + dis(j, i)} where j belongs to Cities, j != i and j != 1.

**Time function for:**

Exhaustive – O(n2\*2n)

Upper bound – O(n2)

Lower bound – O(nlogn)

Greedy – O(n)

Global upper bound reduces the complexity over the exhaustive using a logic where length is greater or equal to best distance. Lower bound uses a termination technique where the condition checked is distance covered so far should be greater than or equal to best complete solution. Greedy algorithm uses a sort technique over the permutations to sort closest cities to each city. Hence, greedy algorithm gave the best possible result.