1 Nearest Neighbor Heuristic

The nearest neighbor heuristic is one of the most straightforward approaches. The heuristic is not so unlike how we would browse a museum. Similar to how we would start by picking a point of interest, the nearest neighbor heuristic begins at a single node. Then like touring a museum, we pick the next closest exhibit or display, and continue going through the museum avoiding things weve seen. The nearest neighbor heuristic works the same way, it identifies the unvisited nearest neighbor to the current node, and traverses the graph until all unvisited nodes have been seen once.

1.1 Pseudocode

Algorithm 1 Nearest Neighbor

```
1:  procedure NEAREST NEIGHBOR({G = (V, E), s})
2:      x ← Random Value
3:      L ← {}
4:      s_{key=0}
5:
6:      while |L| ≠ |V| do
7:          for { u ∈ N(v) | v ∈ V } do
8:              x ← min N(v)
9:              if vertex unot visited then
10:             else
11:                 x ← Next shortest adjacency
12:             end if
13:         end for
14:     end while
15:
16:     L ← L ∪ x
17: end procedure
```

1.2 Detail

In additional to being fairly easily implemented, the nearest neighbor heuristic is also a comparatively quick method. A major attraction of the nearest neighbor is that it runs fairly quickly, it has a time complexity of $O(n^2)$ [1].

However there some rather large drawbacks to the approximation. A major flaw is that in its greed, the tour can miss obvious shorter routes that could be identified with more comprehensive algorithms. Additionally for the same reason, the method can miss nodes till the end and need to include them at high cost. In order to improve the path it would then require additional improvement heuristics and approximation algorithms to decrease the cost of the tour and

increase quality. Though it was both simple and clear, for these reasons the approach was not a suitable method for finding an optimal tour.