1 Greedy Heuristic

Another straightforward approach is the Greedy approach heuristic. The greedy heuristic works by evaluating all the edges to slowly build the tour. First the edges are all sorted and the shortest edges is identified. The approach begins by adding this smallest edge to a possible tour. It then finds the next shortest edges and evaluates it. Does adding this new edge to the tour create a cycle with less than V edges? Does the newly added edge increase the degree of the node to more than 2? Has this edge been used before? The approach checks each edge with these criteria and as long as it does not violate them will add it the tour. It will repeat this process until all of the nodes have a degree of two and there are $|V|$ edges. This means the tour has created a Hamiltonian cycle.

1.1 Pseudocode

Algorithm 1 Greedy

```
1:  procedure GREEDY({G = (V, E), S})
2:      E_sorted ← Sort(E)
3:      x ← min E
4:      T ← T ∪ x
5:
6:      while |V| ≠ |T| do
7:          x ← min E_sorted
8:          if x ∪ T has no cycle with edges > |V| then
9:              if deg(x) ≤ 2 then
10:                 if ∄x ∈ T then
11:                     T ← T ∪ x
12:                 end if
13:             end if
14:         end if
15:     end while
16:
17: end procedure
```

1.2 Detail

A major drawback to this approach is that it does no forecasting and picks the best selection at the moment. This greedy shortsightedness is exactly like nearest neighbor, as they both do not look ahead. The difference between the two greedy approaches is where the focus lies. The nearest neighbor evaluates adjacent nodes, while the greedy approaches focuses on all edges to build a tour piece by piece. As one can see it has a runtime of $O(n^2 log2(n))$[1] where a large portion of the running time is spent ordering the edges. Like any greedy algorithm, it often produces sub-optimal tours and requires additional improvement

heuristics and algorithms to increase quality. For all its simplicity, it also was not a good choice due to its inability to approximate an optimal tou