The construction heuristic we have chosen to implement is nearest neighbor. The algorithm is simple: Start at any location c, it to the path, find the nearest location c_1 to c, and repeat the process with c_1 as the new starting point until there are no locations left to add. The pseudocode is as follows:

```
let start = c
let visited = set()
let path = []
while len(visited) < len(locations):
    visited.add(start)
    path.append(start)
    let start = find_nearest(start)
return path
```

Where the `find_nearest` routine computes the distance, whichever metric is preferred, between `start` and all other locations and returns the location with the least distance from `start`. This requires looping through all locations. Since we must call `find_nearest` once for each location, we will loop through all locations once for each location. Thus, it is clear that this algorithm runs in $O(n^2)$ time, where n is the number of locations. What remains is the problem of choosing c. The algorithm uses a set to hold the visited locations, and a list to hold the total list of locations.

The choice of c can impact performance. When experimenting with different starting points, we found several improvements over the location that was listed first in the list of locations. We performed this evaluation by running the above algorithm with each city as a starting point. This, of course, runs an $O(n^2)$ algorithm n times, which would make trying each city as a starting point $O(n^3)$.

One could also choose a random starting point, which makes no guarantees of finding the most optimal solution (in the context of a nearest-neighbor approximation), but maintains an $O(n^2)$ runtime. Space complexity is simple, and and is irrespective of node selection: $O(n)$. We need a set storing n values and a list storing n values.

The nearest neighbor algorithm does not have a constant approximation ratio[1]. Instead, the approximation ratio depends on the size of the number of nodes in the input graph. We will reproduce the proof from[1] here.

Let NEARNEIBER be the length of the solution obtained by the nearest neighbor algorithm. Then,

$$\frac{\text{NEARNEIBER}}{\text{OPTIMAL}} \leq \frac{1}{2}\lceil log(n) \rceil + \frac{1}{2} \tag{1}$$

First, we must prove an additional property. Suppose that for an n node graph (N, d) there is a mapping assigning each node p a number l_p such that the following two conditions hold:

a. $d(p,q) \geq \min(l_p, l_q)$ for all nodes p and q
b. $l_p \leq \frac{1}{2}$OPTIMAL for all nodes p

Then, $\sum l_p \leq \frac{1}{2}(\lceil log(n) \rceil + 1)$OPTIMAL.

Proof: We can assume without loss of generality that the node set N is $\{i | 1 \leq i \leq n\}$ and that $l_i \geq l_j$ whenever $i \leq j$. We will also assume that

$$\text{OPTIMAL} \geq 2 \sum_{i=k+1}^{\min(2k,n)} l_i \tag{2}$$

for all k satisfying $1 \leq k \leq n$.

In order to prove the above inequality, let H be the complete subgraph of the set of nodes:

$$\{i | 1 \leq i \leq \min(2k, n)\} \tag{3}$$

Let T be the tour in H which visits the nodes of H in the same order as these nodes are visited in an optimal tour of the original graph. Let LENGTH be the length of T. Since the original graphs we are dealing with satisfy the triangle inequality and the edges of T sum to LENGTH and the corresponding paths in the original graph sum to OPTIMAL, we can conclude that

$$\text{OPTIMAL} \geq \text{LENGTH} \tag{4}$$

By condition (a) of the lemma, for each (i,j) in T, $d(i,j) \geq min(l_i, l_j)$. Therefore,

$$\text{LENGTH} \geq \sum_{(i,j) \in T} \min(l_i, l_j) = \sum_{i \in H} \alpha_i l_i \tag{5}$$

where α_i is the number of edges (i,j) in T for which $i > j$. In order to obtain a lower bound on the right hand side of (5), we observe that every α_i is at most 2 because i is the endpoint of only two edges in tour T, as is the definition of tour, and that α_i sum to the number of edges in T. Since k is at least half of the number of edhes in T, assuming the k largest l_i have $\alpha_1 = 0$ and the remaining $\min(2k, n) - k$ of the l_i have an $\alpha_i = 2$ we arrive at a certain lower bound. By our previous sasumption, the k largest are $\{l_i | 1 \leq i \leq k\}$ so, our lower bound is

$$\sum_{i \in H} \alpha_i l_i \geq 2 \sum_{i=k+1}^{\min(2k,n)} l_i \tag{6}$$

(5), (4), and (3) together establish (2).

Summing inequalities (2) for all values of k equal to a power of two less than s

$$\sum_{j=0}^{\lceil log(n) \rceil - 1} \text{OPTIMAL} \geq \sum_{j=0}^{\lceil log(n) \rceil - 1} 2 \times \sum_{i=2^j+1}^{\min(2^{j+1},n)} l_i \tag{7}$$

which reduces to

$$\lceil log(n) \rceil \times \text{OPTIMAL} \geq 2 \times \sum_{i=2}^{n} l_i \tag{8}$$

And condition (b) of the lemma implies

$$\text{OPTIMAL} \geq 2 \times l_i \tag{9}$$

And (8) and (9) combine to conclude the proof of this lemma.

In order to prove the theorem, for each node p let l_p be the length of the edge leaving node p and going to the node selected as the nearest neighbor to p. We want to show that the l_p satisfy the conditions of Lemma 1.

If node p was selected before node q, then q was a candidate for the closest unselected node to p but was not selected. Thus, the edge (p, q) is no shorter than the edge selected and hence

$$d(p, q) \geq l_p \tag{10}$$

The converse is also true. Since one of the nodes was selected before the other, (10) and its converse must hold and condition (a) is satisfied.

To prove condition (b) we must prove that for any edge (p, q)

$$d(p, q) \leq \frac{1}{2} \times \text{OPTIMAL} \tag{11}$$

The optimal tour can be considered to consist of two disjoint parts, each of which is a path between nodes p and q. From the triangle inequality, the length of any path between p and q cannot be less than $d(p, q)$, establishing (11). Because the l_p are the lengths of the pairs comprising tour T,

$$\sum l_p = \text{NEARNEIBER} \tag{12}$$

The conclusion of Lemma 1 together with (12) and the fact that the optimal path length must be greater than 0 prove the inequality of Theorem 1.