Josh Barber

Assignment 12

April 9th, 2023

My approximation results seem to be better then optimal. I believe this is caused because, when I traverse through my nodes in the Preorder Traversal, it isn't adding the weight between one of the last children into a branch to the next branch's parent. The pathing however is correct

	pa	pb
1.	a→g→b→f→c→e→h→d→i→a	a→g→b→f→c→e→h→d→i→a
2.	a→f→d→h→e→i→b→c→g→a	a→f→d→h→g→e→i→b→c→a
3.	a→d→f→b→h→g→c→e→a	a→d→f→b→h→e→g→c→a
4.	a→b→e→d→f→c→a	a→b→g→e→d→f→c→a
5.	a→b→c→d→e→f→a	a→b→c→d→e→f→a

The brute force algorithm runs in $O(n!)$ as it takes all possible permutations from the root of TSP.

```
TSP(G, r):
    minWeight = inf
    allpermutations = permutations of all vertices in G except r
    for permutation in allpermutations:
        weight = 0
        for i n permutation:
            weight = euclidean distance of i coords to root
        if weight < minWeight:
            minWeight = weight
```

The approximation algorithm is much faster with $O(V^2)$ time. First we find Prim's algorithm's MST_1, then traverse adding weights of all children in a DFS style.

```
Prim(G, r):
    PQ = priority queue
    MST = Graph with G.n vertex
    color[u] = white for all u ∈ G.v
    Visit(r, color, G, PQ)
    while PQ is not empty
        weight, u = PQ.extract min
        if color(u] = white:
            Visit(u, color, G, PQ)
            MST add edge

    return MST

Visit(u, color, G, PQ):
    for each v, weight in G.adjacent(u)
        if color[v] = white:
            PQ.insert(weight, u→v)

    color[u] = black
Approx TSP Tour(G, ):
    root = Prime(G, r)
    H = Traversal(root)
    return H
```

```
Traversal (node):
    if  node is emty
        return

    for each child in node.children
        R = R + Traversal (child)

    return R.
```