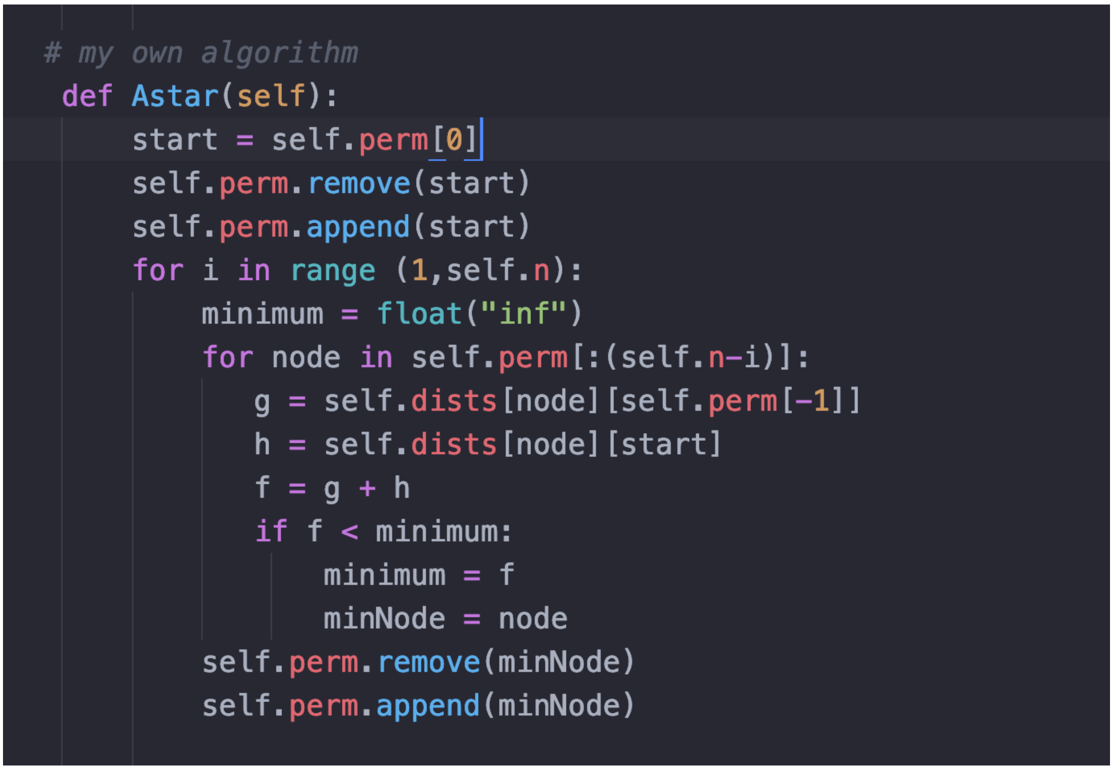
Report of introduction of algorithm and data structure

Part1 my own algorithm

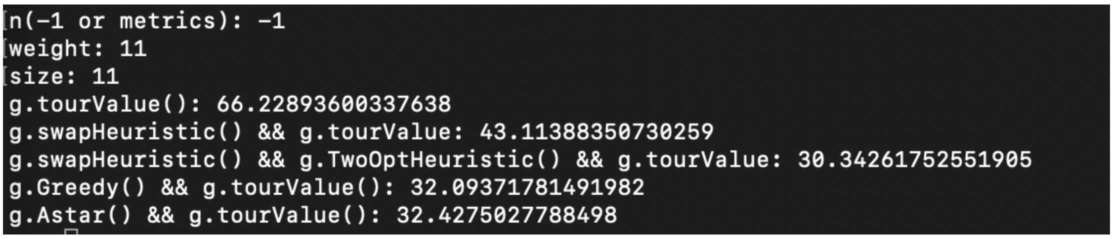
In this part, I implement A star algorithm in graph. A star algorithm is a graph traversal and path search algorithm. It is polynomial time. It evaluates nodes by combining g(n), the cost to reach the next node, and h(n), the cost to get from node to goal f(n) = g(n) + h(n). however, if we implement Astar algorithm in TSP, we would see the result of Astar is greater than the result of greedy, as we set the end node is same as the start node and heuristic of Astar is cost form node to goal node, so Astar is not suitable for this problem.



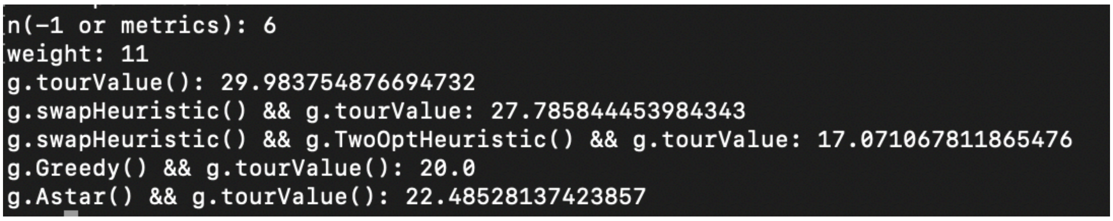
```
# my own algorithm
 def Astar(self):
     start = self.perm[0]
     self.perm.remove(start)
     self.perm.append(start)
     for i in range (1,self.n):
         minimum = float("inf")
         for node in self.perm[:(self.n-i)]:
             g = self.dists[node][self.perm[-1]]
             h = self.dists[node][start]
             f = g + h
             if f < minimum:
                 minimum = f
                 minNode = node
         self.perm.remove(minNode)
         self.perm.append(minNode)
```

Part 2 experiment

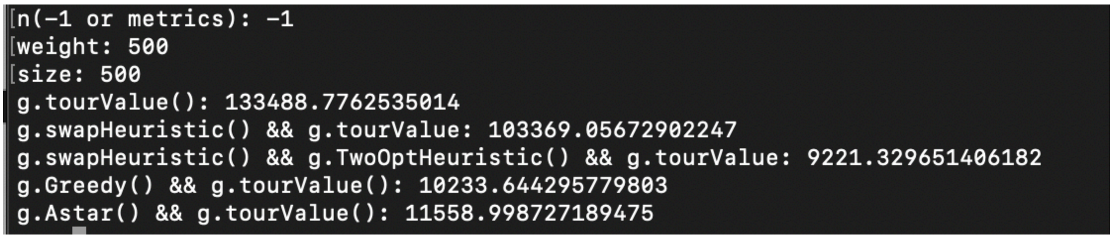
When there are small number of nodes, the distance calculated by greedy and Astar are close, but when nodes increase, we will find that distance by Astar is larger than greedy.



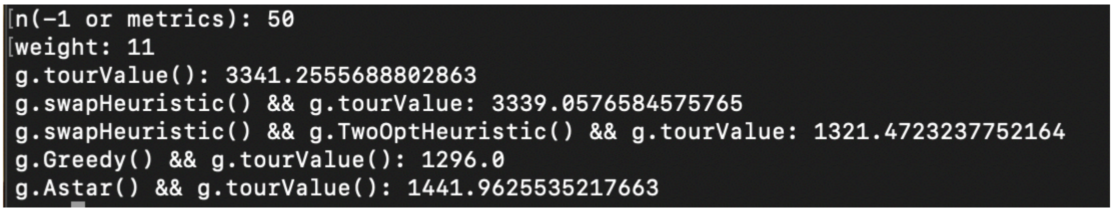
```
[n(-1 or metrics): -1
[weight: 11
[size: 11
 g.tourValue(): 66.22893600337638
 g.swapHeuristic() && g.tourValue: 43.11388350730259
 g.swapHeuristic() && g.TwoOptHeuristic() && g.tourValue: 30.34261752551905
 g.Greedy() && g.tourValue(): 32.09371781491982
 g.Astar() && g.tourValue(): 32.4275027788498
```



```
[n(-1 or metrics): 6
[weight: 11
g.tourValue(): 29.983754876694732
g.swapHeuristic() && g.tourValue: 27.785844453984343
g.swapHeuristic() && g.TwoOptHeuristic() && g.tourValue: 17.071067811865476
g.Greedy() && g.tourValue(): 20.0
g.Astar() && g.tourValue(): 22.48528137423857
```



```
[n(-1 or metrics): -1
[weight: 500
[size: 500
 g.tourValue(): 133488.7762535014
 g.swapHeuristic() && g.tourValue: 103369.05672902247
 g.swapHeuristic() && g.TwoOptHeuristic() && g.tourValue: 9221.329651406182
 g.Greedy() && g.tourValue(): 10233.644295779803
 g.Astar() && g.tourValue(): 11558.998727189475
```



```
[n(-1 or metrics): 50
[weight: 11
 g.tourValue(): 3341.2555688802863
 g.swapHeuristic() && g.tourValue: 3339.0576584575765
 g.swapHeuristic() && g.TwoOptHeuristic() && g.tourValue: 1321.4723237752164
 g.Greedy() && g.tourValue(): 1296.0
 g.Astar() && g.tourValue(): 1441.9625535217663
```