Greedy Algorithm

```
from code import baseobjects

class GreedyFirst(object):
    """
Nodes are sorted by their demand, and then selected i that order to be attended
by first vehicle that has available capacity.
After that every route is sorted (if sort=True) by distance - first nearest node
from the previous node in the route.

So starting from the depot it goes to nearest, and then nearest from that one,
and so on.
    """


    def __init__(self, instance):
        self.instance = instance


    def run(self, sort=False):
        self.instance.network.sort_network_by_demand()
        network = self.instance.network
        fleet = self.instance.fleet
        # assigning nodes so that they fit the capacity restriction
        for node in network:
            if not node.visited:
                for vehicle in fleet:
                    try:
                        vehicle.add_node(node)
                        break
                    except ValueError:
                        continue
        # sorting (if sort=True), and inserting start and end point of the route
- the depot.
        depot = network.get_node(1)
        for vehicle in fleet:
            vehicle.route.insert_node(0, depot)
            if sort:
                vehicle.route = self.sort_by_distance(vehicle.route)
            vehicle.route.append_node(depot)
        return self.instance
```

```
    def sort_by_distance(self, route):
        sorted_route = baseobjects.Route()
        sorted_route.append_node(route.pop_node_id(1))
        while(route):
            source_id = sorted_route[-1].id
            destination_id = self.get_nearest_node(source_id, route)
            sorted_route.append_node(route.pop_node_id(destination_id))
        return sorted_route


    def get_nearest_node(self, source_id, present_nodes):
        minimum = 99999999999
        destination_id = None
        for destination in present_nodes:
            distance = self.instance.distance_matrix[source_id -
1][destination.id - 1]
            if 0 < distance < minimum:
                minimum = distance
                destination_id = destination.id
        return destination_id
```