# ROUTING ALGORITHM (JAVA)

```
INPUT: A graph G = (V,E), a set of distances between the locations {aij : (i,j) ∈ V}, an
       origin s ∈ V, a destination  t ∈V.
       Weight of the orders, Pickup and Drop Locations, Capacity of the truck, pickup times.

INITIALIZE: rem_drop[V], feasible[V], U=source node.

for each vertex V in graph do begin
    visited[V] ← false
end for
for each neighbour vertex v of U in graph do begin
    if (time_taken >= pickup_time[v]) , then
        add v to the feasible[] list
    end if
end for
visited[U] = true
ver = vertex in feasible[] with minimum distance from U
if (node v contains drop order), then
    remove drop_order[v] from the truck
end if
if (weight[v] >= truck_capacity), then
    for each vertex i in rem_drop do begin
        x ← vertex with min distance from i
        add x to the route
    end for
    print route
    print truck_capacity
    update the remaining nodes to next truck
end if
if (weight[v]!=0) and (weight[v] < truck_capacity), then
    add the vertex v to the route
    add the weight[v] to the truck
    add drop location and weight of v to rem_drop
end if
```

## OUTPUT:

(i) The route containing locations

(ii) Capacity of the truck

(iii) File output containing locations, weight of orders and trucks.