Project Part 3:

A Different Type of Greedy Heuristic

Submitted by:

Name: Krisha Patel

Student ID: 40084336

Proposed heuristic: Greedy BRP3

1. **Estimate the number n^{veh} of vehicles (Algorithm 1)**

   Using the same Data Generator as of Part 2. Estimating vehicles on bases of similar algorithm used in Part 2.

2. **Partition the set of bike stations into n^{veh} cluster sets. (Algorithm 2)**

   With the help of no of vehicles obtain in algorithm 1. No of Clusters are the same as no of vehicles.

   The partition of bikes is done in such a way that the set of bikes in each cluster has similar distance from the depot so its easy for the vehicle that is assigned to that cluster will have all the bike stations with similar distance in one cluster. So the time required to complete that cluster will be less.

Question 1. Write a detailed greedy algorithm for each step of Greedy BRP3

estimateVehicles(graph G)

```
for i <- 0 to no of bikestations
    do temp <- Dijkstra(G,depot,i)
start <- min(temp)
cost <- 0
vehicles=1;
for i <- 0 to temp.length
     do src=depot
      des=I
      time <- Dijkstra(G,src,des)
      t <- Dijkstra(G,des,depot)
      cost <- cost+time+t
      if(cost>=480)
        do      vehicles++
          cost <- 0
          scr=depot
          des=i
    if end
 cost <- cost-t
 src <- des
 return vehicles.
```

Complexity: **O(n)** (for loop) + **O(1)** (Line 3-5) +

O(n) (for loop) * O(E * T(extract_min)) + O(V* T(dec.Key)) +

O(log n) (for if condition) + **O(1)** (last 3 Lines)

<u>getCentroid(data, noofcluster, centroid)</u> //data= bikestation, noofcluster= no of vehicles

```
    distance[][] <- [noofcluster][data.length]
    cluster[] <- data.length

    //…1….new data with bikestation and distance from depot……..…//
    For i <- 0 to data.length
        Temp <- Dijkstra(G,depot,i)
        (data[i],Temp) insert in HashMap Mcluster

    //…2………………New Centroid for 1st time…………………………………//
    For i <-  0 to noOfCluster
        Do centroid[i] <- getRandom(143,220)

    //3Creating temp cluster for difference of centroid and bikestation//
    For i <-  0 to noOfCluster
        For j <- 0 to Mcluster.key
            Diff <- centroid[1][i]-Mcluster.keyValue
            (j,diff) insert in ArrayList mpp
        (i,mpp) insert in Hashmap cls(i)

    //…4…………………….Creating Final cluster………………………………//
    Map FinalCluster(Int,ArrayList)
    smallestDist <- Max.Value
    for i <- Mcluster.key
        for j <- noOfCluster
            if(cls.get(j).get(i) <= smallestDistance) {
                smallestDistance <- cls.get(j).get(i);
                f=j;
        FinalCluster.get(f).add(i);
```

```
//…5……………….New Centroid form final Cluster……………………….//
For  i <- 0 to finalCluster.keySet()
        sum=0, avg=0, count=0;
        for j <- 0 to finalCluster.size
                temp <- finalCluster.get(i).get(j)
                sum <- sum + Mcluster.get(temp)
                count++
                }
                Avg <- sum/count;
                finalcentroid[0][i] <- avg;

//…6…………..Compare previous Centroid and new Centroid……….//
        If( finalCentroid = centroid)
                Return centroid
        Else
                Centroid <- finalCentroid
                getCentroid ( data, noOfCluster, centroid)
```

<u>Complexity:</u> **O(1)** (Line 1-3) +

O(n)*(O(E)+O(V)) (for "**1**" for loop + dijkstra) +

O(n) ("**2**" for loop) +

O(n*n) ("**3**"- 2 for loops) +

O(n*n*logn) ("**4**"- 2 for loops + if conditon) +

O(n*n) ("**5**"- 2 for loops) +

O(logn) ("**6**" if condition)

Inimain()

```
ReadFile (topology, bike_data)
For Each Edge
        Do distance[] <- edge.add
For i <- 0 to bikesStation.length
        Bikes <- insert(data)
Sorted <- bikes.sort()
```

Complexity: O(n) (for loop) + **O(n)** (for loop)

minDistance(distance,graph)

```
for i <- 0 to vertex
if ( distance[i] < min )
        min <- distance[i]
```

Complexity: O(n)

Dijkstra(Graph, src, des)

```
d[s] ←0
for v∈V−{ s}
        do d[v] ←∞
        S←∅
        Q←V
While(Q ≠ ∅)
        Do u ←EXTRACT-MIN(Q)
        S←S∪{u}
        For v ∈ Adj[u]
                Do if (d[v] > d[u] + w(u, v))
                        then d[v] ←d[u] + w(u, v)
```

Complexity: O(E * T(extract_min)) + O(V* T(dec.Key))

Greedy_BRP

```
Inimain()
Veh <- estimateVehicles(G)
For i <- 0 to bikes.length
   Kmp <- bikes.key
Flag <- 0
getCentroid( kmp , veh , centroid, flag)

for i <-0 to finalCluster.keySet()
        ArrayList ar. Add(value(i))
        for j <-0 to  ar.length
                if( bikes. Get(j)!=5)
                        (ar.key,bikes.key.value) insert in hashMap hm
        Sorted <- Hm.sort()
        While(sorted != empty)
                If(flag==0)
                        Scr <- depot
                        Des <- sorted.lastValue
                        Time <- Dijkstra(G,scr,des)
                        Cost <- cost + time
                        Scr <- des
                        Flag <- 1
                        Sorted.remove(last.value)
                        continue
                Else
                        Des <- sorted.lastValue
                        Time <- Dijkstra(G,scr,des)
```

```
        T <- Dijkstra(G,des,depot)
        Cost <- cost + time +t
        If(cost >= 480)
            Scr <- depot
            Cost <- 0
            Vehicle++
            Continue
        Cost <- cost – t
        If( lastValue >= 5 )
            Demand <- bikes.value-5
            Bike <- bike + demand
            Scr <- des
            Update hm
            Update visited
        Else
            Demand <- 5-bike.value
            Bike <- bike – demand
            If(bike<0)
                Break;
            Else
                Update hm
                Update visited
        Scr <- des
    Sorted.remove(lastValue)
Print total Vehicles ,visited Nodes
```

Complexity: **O(n)** (Inimain) +
O(n) * O(E+V) (estimateVehicles) +
O(V) (for loop for nodes) +
O(n^2 * log n) + O(n)*O(E+V) (getCentroid) +
O(n *((n*log n) + log n *log n)) (for loop * (for loop + while condition))

Question 2. Is there an interest of keeping all the bike stations in Step 2 of Greedy BRP3, i.e., even those with a satisfied demand? Justify your answer.

No, if we discard or remove all those bikestations before we start our tour to satisfy all the bike demand bikestations we will have to **cover less nodes** in order this will help us to **decrease the number of vehicles** as well as **time** to complete all the nodes.

The nodes with satisfied demand will increase the efficiency.

Earlier the no of vehicles was more and with that number of clusters were also more with decreased the efficiency and increase time to complete all the nodes to satisfy the bike demand.

Question 3. Perform the complexity analysis of your three algorithms.

EstimateVehicles: O(n) * O(E+V)

O(n) (for loop) + **O(1)** (Line 3-5) +

O(n) (for loop) * O(E * T(extract_min)) + O(V* T(dec.Key)) +

O(log n) (for if condition) + **O(1)** (last 3 Lines)

getCentroid: O(n^2 * log n) + O(n)*O(E+V)

O(1) (Line 1-3) +
O(n)*(O(E)+O(V)) (for "**1**" for loop + dijkstra) +
O(n) ("**2**" for loop) +
O(n*n) ("**3**"- 2 for loops) +
O(n*n*logn) ("**4**"- 2 for loops + if conditon) +
O(n*n) ("**5**"- 2 for loops) +
O(logn) ("**6**" if condition)

Inimain: O(n)

minDistance: O(n)

Dijkstra: O(E * T(extract_min)) + O(V* T(dec.Key))

Greedy_BRP3: O(n)*O(E+V) + O(n^2 * Log n)

O(n) (Inimain) +

$O(n) * O(E+V)$ (estimateVehicles) +
$O(V)$ (for loop for nodes) +
$O(n^2 * \log n) + O(n)*O(E+V)$ (getCentroid) +
$O(n *((n*\log n) + \log n *\log n))$ (for loop * (for loop + while condition))

<u>Question 4. Implement Greedy BRP3 and compare its performance with Greedy BRP2 of Part II.</u>

The time taken for Greedy_BRP3 is bit increased than that of Greedy_BRP2 because of the cluster formation.

The no of vehicles remains same as of both the algorithm.

Question 5. Provide the results of Heuristic BRP3 using the same diagrams as for algorithm Greedy BRP1 in Part I, i.e, Table 1 and Figure 1 should contain the average results over your 10 data sets (generated in Part II). Therefore, it means providing 2 tables and 2 figures for summarizing the results. Use the dat set generated in Part II. You can use them without modifying them, or you can regenerate new data sets with a capacity of the bike stations equal to 12.

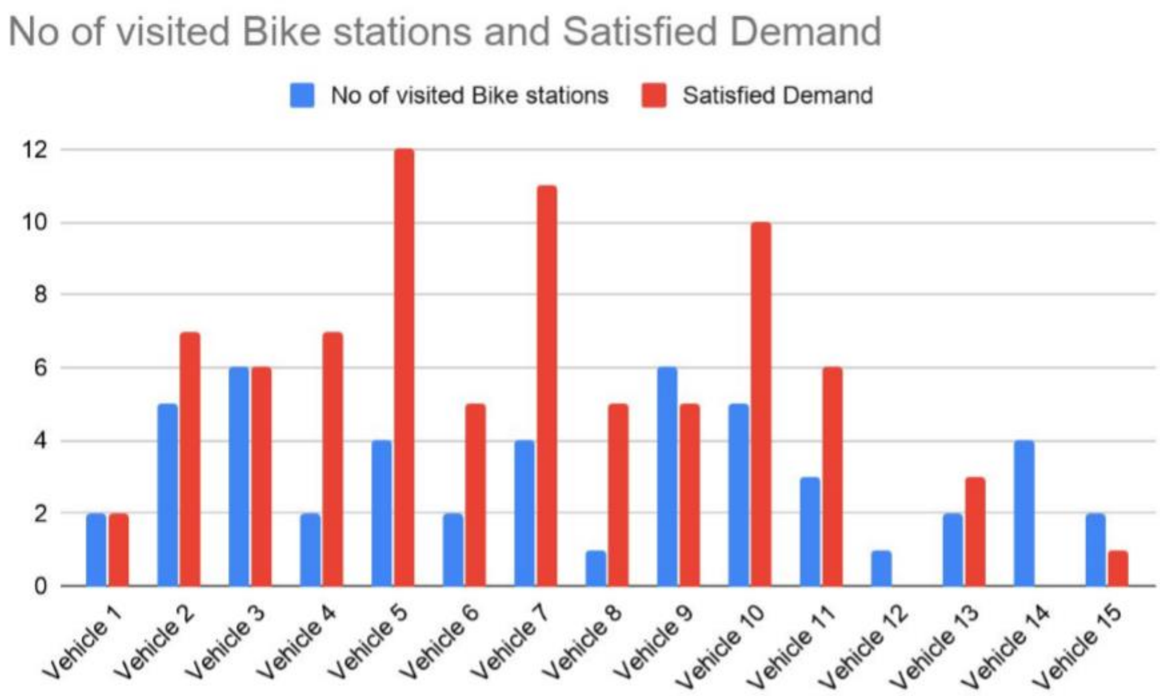
Data Sets are not modified, they are the same as of part 2.

Vehicles in Both the Algorithms remains almost the same bt the time in Greedy_BRP3 is considerating less than that of Greedy_BRP2.

Even more demand of bikes are statisfied than that of Greedy_BRP2.

GREEDY_BRP3

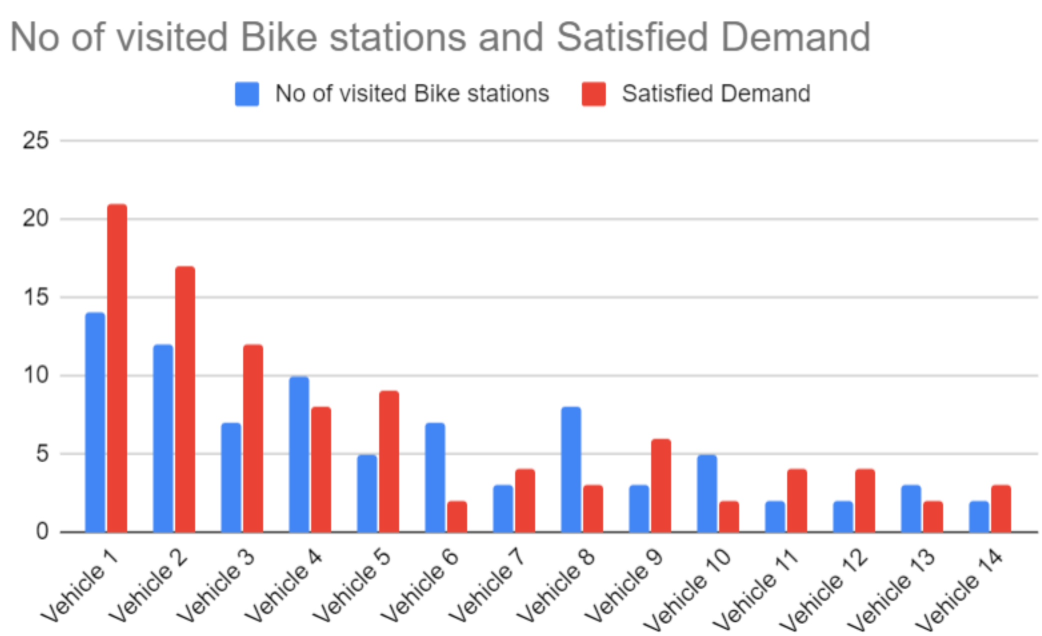
Data Sets	#required vehicles	Computational time (micro seconds)
1	15	250696
2	14	187763
3	14	156694
4	17	208499
5	14	196077
6	15	242323
7	14	162514
8	17	200690
9	17	274945
10	17	284859



GREEDY_BRP2

DataSets	Vehicles	Time(seconds)
1	16	35900
2	13	42900
3	14	35350
4	20	26000
5	14	29700
6	13	37000

7	**11**	**31000**
8	**15**	**26700**
9	**16**	**26100**
10	**13**	**30390**



Refernces:

https://www.youtube.com/watch?v=q3yKyE19OR0

https://www.youtube.com/watch?time_continue=889&v=JSJIolgFYqw

https://www.geeksforgeeks.org/shortest-path-weighted-graph-weight-edge-1-2/

Lecture Slides .