EXPERIMENT 10

Taniya Ahmed

21BDS0059

1. **Making own dataset.**

CODE:

```
df = data.frame(
 x = c(1, 2, 3, 6, 7, 8, 10, 12, 15, 18, 20, 23, 25, 28, 30, 33, 35, 38, 40, 42),
 y = c(1, 1, 2, 5, 5, 6, 9, 10, 13, 14, 17, 19, 21, 22, 24, 27, 29, 31, 33, 35)
)
print("Taniya Ahmed 21BDS0059")
```

OUTPUT:

```
> df = data.frame(
+   x = c(1, 2, 3, 6, 7, 8, 10, 12, 15, 18, 20, 23, 25, 28, 30, 33, 35, 38,
40, 42),
+   y = c(1, 1, 2, 5, 5, 6, 9, 10, 13, 14, 17, 19, 21, 22, 24, 27, 29, 31, 3
3, 35)
+ )
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
```

2. **Calculating shortest distance between points using Euclidean distance.**

CODE:

```
euclid_distance = function(a, b){
 return(sqrt(sum((a - b) ^ 2)))
}
print("Taniya Ahmed 21BDS0059")
```

OUTPUT:

```
> euclid_dist = function(a, b){
+   return(sqrt(sum((a - b) ^ 2)))
+ }
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
```

3. **Calculating the distance between every pair of points / clusters, putting the values in a distance matrix and returning the same.**

CODE:

```
dist_matrix_computation = function(data){
 n = nrow(data)
 dist_matrix = matrix(0, n, n)


 for(i in 1 : (n - 1)){
  for(j in (i + 1) : n){
   dist_matrix[i, j] = euclid_dist(data[i, ], data[j, ])
   dist_matrix[j, i] = dist_matrix[i, j]
  }
 }


 return(dist_matrix)
}
print("Taniya Ahmed 21BDS0059")
```

OUTPUT:

```
> dist_matrix_computation = function(data){
+   n = nrow(data)
+   dist_matrix = matrix(0, n, n)
+
+   for(i in 1 : (n - 1)){
+     for(j in (i + 1) : n){
+       dist_matrix[i, j] = euclid_dist(data[i, ], data[j, ])
+       dist_matrix[j, i] = dist_matrix[i, j]
+     }
+   }
+
+   return(dist_matrix)
+ }
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
```

4. **Traversing through the distance matrix and returning the minimum distance between pair of data points / clusters.**

CODE:

```
min_dist_computation = function(dist_matrix){
 min_dist = Inf
 n = nrow(dist_matrix)
 cluster_pair = c(0, 0)

 for(i in 1 : (n - 1)){
  for(j in (i + 1) : n){
   if(dist_matrix[i, j] < min_dist && dist_matrix[i, j] != 0){
    min_dist = dist_matrix[i, j]
    cluster_pair = c(i, j)
   }
  }
 }

 return(cluster_pair)
}
print("Taniya Ahmed 21BDS0059")
```

OUTPUT:

```
> min_dist_computation = function(dist_matrix){
+   min_dist = Inf
+   n = nrow(dist_matrix)
+   cluster_pair = c(0, 0)
+
+   for(i in 1 : (n - 1)){
+     for(j in (i + 1) : n){
+       if(dist_matrix[i, j] < min_dist && dist_matrix[i, j] != 0){
+         min_dist = dist_matrix[i, j]
+         cluster_pair = c(i, j)
+       }
+     }
+   }
+
+   return(cluster_pair)
+ }
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
```

5. **Updating the distance matrix any time datapoints/ clusters are merged.**

CODE:

```
dist_matrix_updation = function(dist_matrix, cluster1, cluster2){
 n = nrow(dist_matrix)


 for(i in 1 : n){
  if(i != cluster1 && i != cluster2){
   dist_matrix[cluster1, i] = (dist_matrix[cluster1, i] + dist_matrix[cluster2, i]) / 2
   dist_matrix[i, cluster1] = dist_matrix[cluster1, i]
  }
 }


 dist_matrix[cluster2, ] = 0
 dist_matrix[, cluster2] = 0


 return(dist_matrix)
}
print("Taniya Ahmed 21BDS0059")
```

OUTPUT:

```
> dist_matrix_updation = function(dist_matrix, cluster1, cluster2){
+   n = nrow(dist_matrix)
+
+   for(i in 1 : n){
+     if(i != cluster1 && i != cluster2){
+       dist_matrix[cluster1, i] = (dist_matrix[cluster1, i] + dist_matrix[c
luster2, i]) / 2
+       dist_matrix[i, cluster1] = dist_matrix[cluster1, i]
+     }
+   }
+
+   dist_matrix[cluster2, ] = 0
+   dist_matrix[, cluster2] = 0
+
+   return(dist_matrix)
+ }
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
>
```

6. **Hierarchical clustering function which combines use of all previous functions and applies them over iterations to the dataframe passed as an argument to the function.**

CODE:

```
hierarchical_clustering = function(data){
 n = nrow(data)
 dist_matrix = dist_matrix_computation(data)
 clusters = as.list(1 : n)

 valid_clusters = 1:n

 while(length(valid_clusters) > 1){
  cluster_pair = min_dist_computation(dist_matrix)

  if(cluster_pair[1] == 0 || cluster_pair[2] == 0){
   break
  }

  cat("Clusters merged are", cluster_pair[1], "and", cluster_pair[2], "\n")
  clusters[[cluster_pair[1]]] = c(clusters[[cluster_pair[1]]], clusters[[cluster_pair[2]]])
  valid_clusters = valid_clusters[valid_clusters != cluster_pair[2]]
  dist_matrix = dist_matrix_updation(dist_matrix, cluster_pair[1], cluster_pair[2])
 }

 return(clusters[valid_clusters])
}
print("Taniya Ahmed 21BDS0059")
```

OUTPUT:

```
> hierarchical_clustering = function(data){
+   n = nrow(data)
+   dist_matrix = dist_matrix_computation(data)
+   clusters = as.list(1 : n)
+
+   valid_clusters = 1:n
+
+   while(length(valid_clusters) > 1){
+     cluster_pair = min_dist_computation(dist_matrix)
+
+     if(cluster_pair[1] == 0 || cluster_pair[2] == 0){
+       break
+     }
+
+     cat("Clusters merged are", cluster_pair[1], "and", cluster_pair[2],
"\n")
+     clusters[[cluster_pair[1]]] = c(clusters[[cluster_pair[1]]], clusters
[[cluster_pair[2]]])
+     valid_clusters = valid_clusters[valid_clusters != cluster_pair[2]]
+     dist_matrix = dist_matrix_updation(dist_matrix, cluster_pair[1], clust
er_pair[2])
+   }
+
+   return(clusters[valid_clusters])
+ }
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
```

7. **Applying the hierarchical clustering to the dataframe.**

CODE:

```
final_clusters = hierarchical_clustering(df)

print(final_clusters)

print("Taniya Ahmed 21BDS0059")
```

OUTPUT:

```
> final_clusters = hierarchical_clustering(df)
Clusters merged are 1 and 2
Clusters merged are 4 and 5
Clusters merged are 1 and 3
Clusters merged are 4 and 6
Clusters merged are 7 and 8
Clusters merged are 12 and 13
Clusters merged are 14 and 15
Clusters merged are 16 and 17
Clusters merged are 18 and 19
Clusters merged are 9 and 10
Clusters merged are 18 and 20
Clusters merged are 9 and 11
Clusters merged are 4 and 7
Clusters merged are 12 and 14
Clusters merged are 16 and 18
Clusters merged are 1 and 4
Clusters merged are 9 and 12
Clusters merged are 9 and 16
Clusters merged are 1 and 9
> print(final_clusters)
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
```