Variation of DBSCAN

Problem Statement:
Implement given algorithm in Scheme programming language for clustering data points given in a dataset D. Assume an ordering of D where i^{th} data point has index i.

Parameters:

- K
- ϵ **(**$\leq K$)
- $MinPts$ **(**$\leq K$)

Algorithm:

1. Sparsify Similarity Matrix
    a. Construct the similarity matrix M between every pair of data points. The value of $M[i][j] = d(p_i, p_j)$ where d is the Euclidean distance function. Assume $d(p_i, p_i) = \infty \ \forall\, p_i \in D$.
    b. Sparsify the matrix by keeping only K- nearest neighbours for each data point. Each row $M[i]$ should contain K points with least Euclidean distance from point p_i. If two points have same Euclidean distance then data point with lower index will be preferred. Let $KNN(i) = M[i]$.
2. Construct the shared neighbor graph G
    a. The graph is given by $G = (V, E)$ where V is the set of indices of data points in D and edge relation E is defined as :
    $\forall\, p, q \in D \ (p, q) \in E \ iff \ [(p \neq q) \ and \ p \in KNN(q) \ and \ q \in KNN(p)]$
    b. If $(p, q) \in E$ then the weight of this edge (p, q) is
    $W(p, q) = |KNN(p) \cap KNN(q)|$. Let similarity between two data points be given by $sim(p, q) = W(p, q)$.
3. Identify core points
    a. The density of each data point is :
    $density(p) = |\{q \mid (p, q) \in E(G) \ and \ W(p, q) \geq \epsilon\}|$ where $E(G)$ represents the edge set of the shared neighbour graph G.
    b. p is a core point $iff \ density(p) \geq MinPts$.
4. Form clusters using core points
    a. Two core points c_0 and c_k belong to the same cluster iff there exists a sequence of core points $c = (c_0, c_1, \dots, c_{k-1}, c_k)(k \geq 1)$ where $\forall\, i \in [0, k-1]$
    $(c_i, c_{i+1}) \in E(G) \ and \ W(c_i, c_{i+1}) \geq \epsilon$.
    b. Clusters are numbered from 1 to n_c where n_c are the total number of clusters. Let $Core$ be the set of indices of all core points sorted in ascending order, the numbering of clusters can be done as -

```
index = 1
while(Core ≠ φ)
     c = Core.first
     cluster[index++] = Cluster(c)
     Core = Core – Cluster(c)
```

5. Identify noise points
    a. A point p is a noise point iff p is not a core point and $\{q \mid (p,q) \in E(G)\ and\ W(p,q) \geq \epsilon\} = \phi$
    b. A noise point does not belong to any cluster.
6. Assign border points to clusters
    a. A point is a border point if it is neither a core point nor a noise point.
    b. Assign the border point p to the cluster of the core point c if the $sim(p,c)$is maximum among all such core points. If multiple core points have the same similarity with border point p, assign it to the core point with lower index.

Input Format:

Each input would be given in a file in the following format. First line would contain five space separated parameters as follows.

$N\ D\ K\ \varepsilon\ MinPts$

N = Number of points in the dataset

D = Dimensions of each point

$K, \varepsilon, MinPts$ are parameters for the algorithm

This is followed by N lines each representing a data point of D dimensions.

Sample Input:

```
20 6 5 3 5
3.83 28.87 7.2 26.6 6.19 37.01
2.89 20.1 -11.71 24.4 5.17 26.51
2.86 69.05 12.32 25.7 7.04 36.51
2.92 65.4 14.28 25.7 7.1 40.7
3.06 29.59 6.31 25.4 6.15 37.1
2.07 44.82 6.16 21.6 6.41 33.9
2.52 77.37 12.7 24.9 6.86 41.8
2.45 24.67 -0.17 25.01 5.78 33.4
3.13 65.01 9.85 26.6 6.51 41.01
2.44 9.99 -0.05 28.01 5.57 37.2
2.09 12.2 -12.86 23.51 5.62 23.3
2.52 22.55 0.92 23.6 5.34 35.2
2.22 14.3 4.77 24.51 5.8 34.9
2.67 31.79 -0.96 25.8 6.19 33.1
2.71 11.6 -16.04 25.2 5.62 22.7
3.14 68.47 10.62 25.01 6.94 39.7
3.54 42.64 2.66 25.01 6.33 31.8
2.52 16.7 -10.99 24.8 6.01 31.7
2.68 86.27 15.03 25.51 7.51 43.1
2.37 76.73 12.77 24.51 6.96 41.01
```

Output Format:
Following are the step numbers and corresponding expected output [The expected output on the sample input provided above is given in the box] -

NOTE: Do not print anything in the entire code. Appropriate command will be used by the automatic evaluator which is of the form (display stepx) for output of step number x.
For example: (display step1) should print the output as shown in step number 1 below.

1. Output the list of points in the dataset D as a list where each point is a list having the first element as index and the second element as a list of values in all dimensions.

```
((1 (3.83 28.87 7.2 26.6 6.19 37.01)) (2 (2.89 20.1 -11.71 24.4 5.17 26.51)) (3 (2.86 69.05 12.32 25.7 7.04 36.51)) (4 (2.92 65.4 14.28 25.7 7.1 40.7)) (5 (3.06 29.59 6.31 25.4 6.15 37.1)) (6 (2.07 44.82 6.16 21.6 6.41 33.9)) (7 (2.52 77.37 12.7 24.9 6.86 41.8)) (8 (2.45 24.67 -0.17 25.01 5.78 33.4)) (9 (3.13 65.01 9.85 26.6 6.51 41.01)) (10 (2.44 9.99 -0.05 28.01 5.57 37.2)) (11 (2.09 12.2 -12.86 23.51 5.62 23.3)) (12 (2.52 22.55 0.92 23.6 5.34 35.2)) (13 (2.22 14.3 4.77 24.51 5.8 34.9)) (14 (2.67 31.79 -0.96 25.8 6.19 33.1)) (15 (2.71 11.6 -16.04 25.2 5.62 22.7)) (16 (3.14 68.47 10.62 25.01 6.94 39.7)) (17 (3.54 42.64 2.66 25.01 6.33 31.8)) (18 (2.52 16.7 -10.99 24.8 6.01 31.7)) (19 (2.68 86.27 15.03 25.51 7.51 43.1)) (20 (2.37 76.73 12.77 24.51 6.96 41.01)))
```

2. Output the similarity matrix M as a list containing Nelements where each element i contains N tuples of the form $(j, d(p_i, p_j))$ $\forall\, j \in [1, N]$.
   NOTE :
   - Use *(expt x 2)* to square the number x and *(sqrt x)* to obtain its square root
   - Use *+inf.0* to express positive infinity

```
(((1 +inf.0) (2 23.484356495335355) (3 40.53850268571842) (4 37.425253506155435) (5 1.8311471814138827) (6 17.126067849918147) (7 49.09683391014129) (8 9.465178286751918) (9 36.465250581889606) (10 20.331197702053856) (11 29.684292142478313) (12 9.70028350101171) (13 15.157644935807145) (14 9.611748020001354) (15 32.35221630738766) (16 39.8830690895272) (17 15.491946294768775) (18 22.631164353607616) (19 58.28730565054452) (20 48.42207244635446)) ((1 23.484356495335355) (2 +inf.0) (3 55.48631543002292) (4 54.16956710183312) (5 22.99821514813704) (6 31.544847439795934) (7 64.130555120005) (8 14.229068838121487) (9 51.948868130114256) (10 19.126902519749507) (11 8.698919473129983) (12 15.551247538380965) (13 19.403102844648327) (14 17.282809378107483) (15 10.314547978462267) (16 54.916567627629455) (17 27.28869362941363) (18 6.325899145576066) (19 73.31764044212005) (20 63.40318525121589)) ((1 40.53850268571842) (2 55.48631543002292) (3 +inf.0) (4 5.892995842523561) (5 39.93096417568701) (6 25.488852465342564) (7 9.906507961940985) (8 46.23297957086477) (9 6.620898730534997) (10 60.409022504920564) (11 63.622829236053306) (12 47.97230138319403) (13 55.321970319214046) (14 39.71224874015572) (15 65.55758613615971) (16 3.7372449745768646) (17 28.53822699468206) (18 57.52391850352338) (19 18.64376571403964) (20 9.005415037631531)) ((1 37.425253506155435) (2 54.16956710183312) (3 5.892995842523561) (4 +inf.0) (5 36.876131847036234) (6 23.53124306108796) (7 12.15914881889353) (8 43.857323219731505) (9 4.590784246727349) (10 57.40883555690709) (11 62.267607951486305) (12 45.304852940937806) (13 52.334369204185506) (14 37.69018837840958) (15 64.34459495559825) (16 4.93665878910017) (17 27.08699688042217) (18 55.61853108452254) (19 21.027153873028073) (20 11.510139008717488)) …. )
```

- This output is incomplete and only $M[1..4]$ is shown.

3. Output the indices of the K nearest neighbours of all points. This list contains Nelements where each element $i$is the list for KNN for data point i sorted in ascending order.

((5 8 12 13 14) (8 11 12 15 18) (4 7 9 16 20) (3 7 9 16 20) (1 8 12 14 17) (1 5 8 14 17) (3 4 16 19 20) (1 5 12 13 14) (3 4 7 16 20) (2 8 12 13 18) (2 8 10 15 18) (1 5 8 13 14) (1 5 8 10 12) (1 5 8 12 17) (2 8 10 11 18) (3 4 7 9 20) (1 5 6 8 14) (2 8 11 12 15) (3 4 7 16 20) (3 4 7 16 19))

4. Output the shared neighbour graph G. The list contains Nelements where each element i contains the outward edges from i. The edges are represented as tuples $(p, w)\ such\ that\ (i, p) \in E(G)\ and\ w = W(i, p)$. This list is further sorted giving more preference to edges with higher weight $w$and in case of clash giving more preference to lower point index.

(((8 4) (12 4) (5 3) (13 3) (14 3)) ((18 4) (11 3) (15 3)) ((4 4) (9 4) (16 4) (7 3) (20 3)) ((3 4) (9 4) (16 4) (7 3) (20 3)) ((14 4) (1 3) (8 3) (12 3) (17 3)) ((17 4)) ((19 4) (20 4) (3 3) (4 3) (16 3)) ((1 4) (12 4) (5 3) (13 3) (14 3)) ((3 4) (4 4) (16 4)) ((13 2)) ((15 4) (2 3) (18 3)) ((1 4) (8 4) (5 3) (13 3) (14 3)) ((1 3) (8 3) (12 3) (10 2)) ((5 4) (1 3) (8 3) (12 3) (17 3)) ((11 4) (2 3) (18 3)) ((3 4) (4 4) (9 4) (7 3) (20 3)) ((6 4) (5 3) (14 3)) ((2 4) (11 3) (15 3)) ((7 4) (20 4)) ((7 4) (19 4) (3 3) (4 3) (16 3)))

5. Output the point density, where the i^{th} element of the list (indexed from 1) gives the density of the i^{th} point.

(5 3 5 5 5 1 5 5 3 0 3 5 3 5 3 5 3 3 2 5)

6. Output the list of indices of core points sorted in ascending order.

(1 3 4 5 7 8 12 14 16 20)

7. Output the core clusters. Each element of the list will have its first element as cluster id (beginning from 1) and sorted list of indices of core points in that cluster.

((1 (1 5 8 12 14)) (2 (3 4 7 16 20)))

8. Output the list of indices of noise points sorted in ascending order.

(10)

9. Output the list of indices of border points sorted in ascending order.

(2 6 9 11 13 15 17 18 19)

10. Output the final clusters formed after merging border points with core clusters. Each element of the list will be a list having its first element as cluster id (beginning from 1) and sorted list of indices of data points in that cluster.

((1 (1 2 5 6 8 11 12 13 14 15 17 18)) (2 (3 4 7 9 16 19 20)))