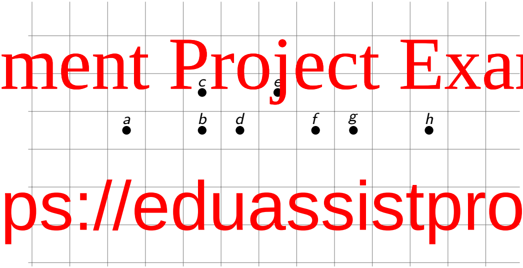
COMP9318 Tutorial 3: Clustering

Q1 I

Consider eight tuples represented as points in the two dimensional space as follows:



Assume that (1) each point lies within the center of the grid; (2) uniform partition of the data space; and (3) each grid is a square with side length 1.

We consider applying DBSCAN clustering algorithm on this dataset. Specifically, we set the minimum number of points (excluding the point in the center) within the ϵ-neighborhood (*MinPts*) to be 3, the radius of the ϵ-neighborhood (ϵ) to be 2, and we adopt the **Manhattan Distance** metric in the computation (i.e., a point p is within the ϵ-neighborhood of point o if and only if the Manhattan distance between p and o is no larger than ϵ).

Q1 II

1. What is the Manhattan distance between point *a* and *e*?
2. List all the *core objects*.
3. What is the clustering result of the DBSCAN algorithm on this dataset if points are accessed following the alphabetical order? You need to write out all the cluster (with points that belonging to the cluster) as well as the outliers (if any).

Solution to Q1 I

1. The Manhattan distance between a and e is 5.
2. Consider each object and list number of points (excluding itself) in the neighborhood.

a	1
b	3
c	3
h	1

Therefore, the core objects are $\{b, c, d, e$

3. DBScan:
   - Start from a, a is not a core object, skip it.
   - Process b, b is a core object, then recursively grow the cluster of b. The final cluster will be $\{b, a, c, d, e, f, g\}$.

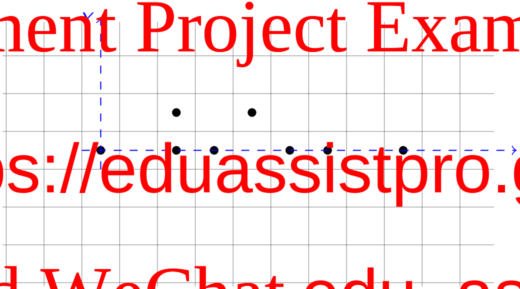
Therefore, the final answer is 1 cluster ($\{b, a, c, d, e, f, g\}$) and 1 outlier (h).

Q2 I

Consider the same dataset as Q1. What is the result of applying **centroid-based** hierarchical clustering algorithm on the dataset (still using the Manhattan distance)?

Solution to Q2 I

1. Let's move the origin of the coordinate to *a* (it is easy to see that the clustering result does not depend on the origin of the coordinate system) and every point is now assign an coordinate.



a	(0, 0)
b	(2, 0)
c	(2, 1)
d	(3, 0)
e	(4, 1)
f	(5, 0)
g	(6, 0)
h	(8, 0)

Solution to Q2 II

2. Centroid-based HAC:
   - Initially, every point is assigned into a distinct cluster.
   - The closest pair of points (under L_1 distance) is one of (b, c), (b, d), and (f, g). Assume we take (b, c). (You may break the tie in any consistent way). We only need to update the distance between (b, c) and d, which is $\frac{1+2}{2} = 1.5$.
   - The next pair to merge is (f, g). After the merge, (f, g) will have the same distance of $\frac{3+2}{2} = 2.5$ to d, e, or h.

[illegible]	[illegible]
$(b, c, d) - (f, g)$	3.50

   - The next pair to merge is (b, c, d) and distance between clusters as

$(a) - (b, c, d,$ [illegible]	[illegible]
$(b, c, d, e) - ($[illegible]	[illegible]

   - The next pair to merge is (f, g) and h. Afterwards, we calculate the new distance between clusters as

$(b, c, d, e) - (f, g, h)$	4.08

   - The next pair to merge is a and (b, c, d, e).

Q3 I

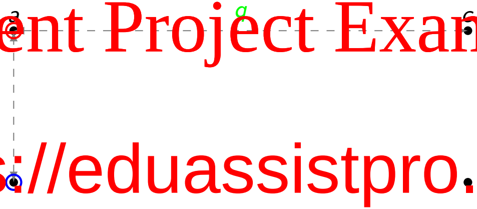
Consider the k-means clustering algorithm.

1. Construct a simple example (with at most four data points) which shows that the clustering result of k-means algorithm can be arbitrarily worse than the optimal clustering results in terms of the cost. (The cost of a clustering is measured as the sum of square distance to the cluster center)

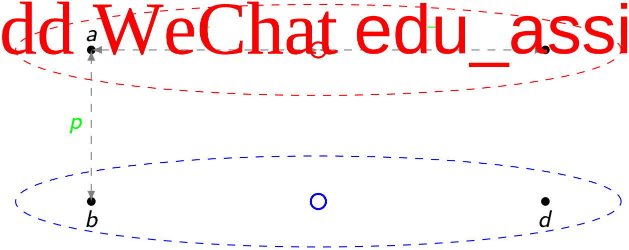
Solution to Q3 I

Consider the k-means clustering algorithm.

1. See the figure below for $k = 2$ and $p < q$ and let the initial cluster centers be a and b.



The final clusters will be:



Solution to Q3 II

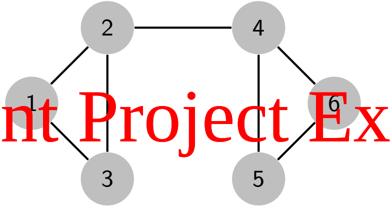
The cost of this clustering result is

$$((q/2)^2 + (q/2)^2) + ((q/2)^2 + (q/2)^2) = q^2$$

The optimal clustering is to group a and b in one cluster and c and d in another cluster. The total cost is

$$((p/2)^2 + (p/2)^2) + ((p/2)^2 + (p/2)^2) = p^2$$

Q3 I



Consi

1. Wr
2. Compute $\mathbf{x}^\top \mathbf{L}\mathbf{x}$ for $\mathbf{x} = \big[1 \quad 1 \quad 1 \quad 1$
3. Show the major steps of embedding the graph into two di

Solution to Q3 I

1. Empty entries in the matrix are 0.

	n_1	n_2	n_3	n_4	n_5	n_6
n_1	2	-1	-1			
n_2	-1	3	-1	-1		
n_3	-1	-1	2			
n_4		-1		3	-1	-1
n_5			-1	-1	3	-1
					1	2

2. 4

3. 2D embeddings: below.

[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	451
2	-0.260956473809	-0.426543475129
3	-0.46470	
4	0.26095	
5	0.46470	
6	0.46470	