CSci 5715, Fall 19: Homework 3

Due on 11/12 before class

Table of Participation

Question ID	Answer drafted by	Answer reviewed by
1	Alex	Majid
2	Majid	
3	Alex, Majid	Majid
4	Alex (except for c,a), Majid	Majid

Chapter 5: Query processing and optimization

Q1. Given 15 points A through O in space shown in Figure 1.1, which are stored in a spatial database, answer the following questions. Assume that one data page can store the information of at most two points.

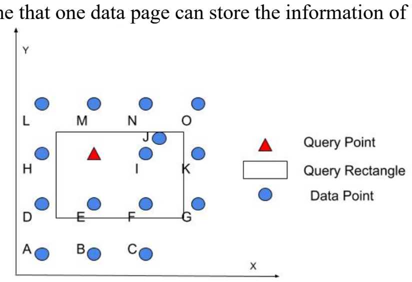


Figure 1.1. 16 points in space (best in color)

Q1(a). If the points are saved without order, what is the minimum number of data pages need to be retrieved to know:

i. Whether there is a point at the query point?

Every page will have to be retrieved. Assuming they are efficiently stored, this is 8 pages.

ii. How many points are there in the query rectangle?

All 8 pages need to be retrieved.

iii. What is the nearest point of the query point?

All 8 pages need to be retrieved.

Q1(b). Now points are saved in the database using an index on their Hilbert value. The Hilbert used is shown in Figure 1.2 as brown dashed lines. What is the minimum number of data pages need to be retrieved to know?

i. Whether there is a point at the query point?

We calculate the Hilbert value of the query point, and then do a search down the B-tree where points are ordered by their Hilbert values. Then we only need to retrieve 1 data page.

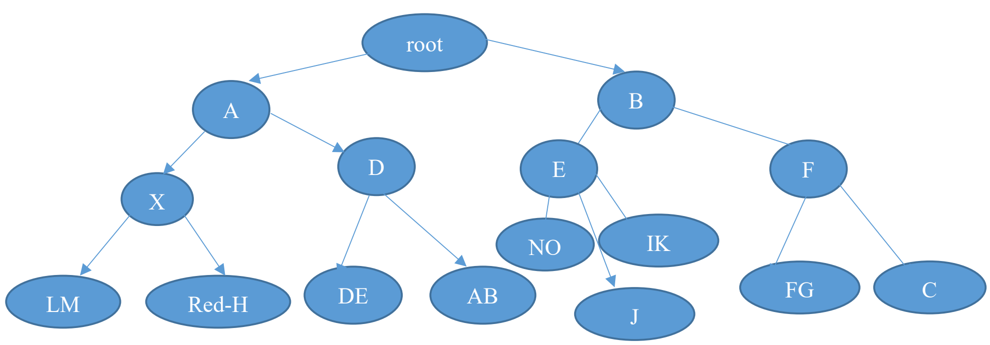
Commented [MF1]: What algorithms choose to do the search?
I think R-tree and Hilbert curve are different?
We should choose on algorithms (linear, binary, R-tree).

ii. How many points are there in the query rectangle?

One way to do this is to split the query rectangle into a series of smaller cells, and then calculate the Hilbert curve value of each of these cells. We order those values, and then join them into a list of ranges to query. We range query the B-tree, and only have to pull in data pages containing those ranges. This means we only end up retrieving 4 data pages.

Commented [MF2]: Same as the above, I think you have to mention what algorithms did you choose? For instance, with R tree we will get 4 data pages, but if we using linear search or binary then it will be more expansive.

Something like this: if you choose the following tree, then we will have 4 data pages.



iii. What is the nearest point of the query point?

Similar to the above answer, we can build small cells surrounding the query point, calculate their Hilbert

curve value, and then point-index the B-tree. Assuming the cells are small enough, we will need to retrieve pages containing points H, M, I, and E, as these are equidistant from the query point. This is a total of 4 data pages.

Commented [MF3]: Similar to previous tree, I drew. First, we need to find, which index does contain the point, then we draw a circle having that point as the center + a define r(radius). Then we draw MOBR which is containing the circle. After that we can calculate the nearest point to (red-Query point).

Briefly explain your reason.

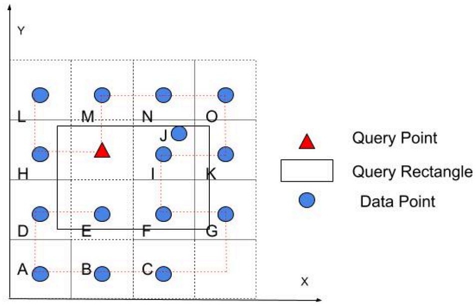


Figure 1.2. Points in space with Hilbert curve grid.

DE 1,AB 2,CG 3,FI 4,KO 5, MN 6, Red_H 7, L 8.

Q2. Figure 2(a) shows a distribution of 10 rectangles in dataset RD1 = {1, 2, …, 10} and minimum orthogonal bounding boxes (MBRs, e.g., X, Y, a, b, c, d) in its R-tree index shown in Figure 2(b).

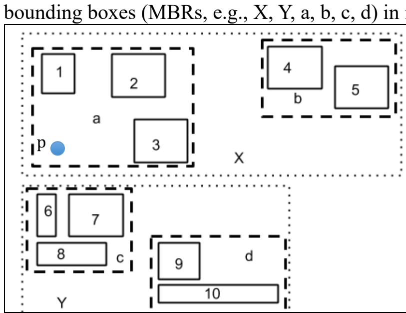


Figure 2(a): Distribution of Rectangles and MBRs

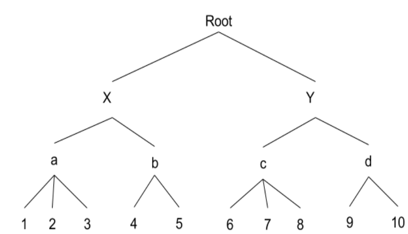


Figure 2(b): R-tree Index

A nearest-neighbor query is performed to find the nearest data rectangle of the query point **p** represented as a circle in Figure 2(a). Assume that the distance between a point p1 and a rectangle R is the distance between the point p1 and the closest point in the rectangle R. The distances from the query point **p** to the closest and the farthest points in each rectangle and MBR are shown in Table 1.

Table 1. Distance from Query Point to Rectangles and MBRs

Rectangle/MBR	Closest distance	Farthest distance
1	4.5	7
2	5.5	10.5
3	5.5	9.5
4	15.5	20
5	20	24
6	2.5	6
7	3	7.5
8	6	8.5

Rectangle/MBR	Closest distance	Farthest distance
9	9.6	13.5
10	12	19
a	0	12.5
b	15	25
c	2.5	9.5
d	9.6	19.5
X	0	25
Y	2.5	19.5

Q1a) Show the execution trace of the query following the **Two-phase** Nearest Neighbor algorithm.
Fill out the following table to list the rectangles and MBRs tested in each phase.

Phase	Rectangles/MBRs
Phase 1	Find the index is containing the query point. In this case, the index is Rectangle (a). The closet MOBR rectangle based on the provided table is [1]. Then we need to create circle (O) by having p its center, and r as its radius. (r = the current closet distance, 4.5), then we will draw the MOBR of circle O. and we test all points in MOBR of circle. Which based on the provided table the closest neighbor from point p is rectangle 6, and its distance is 2.5.
Phase 2	And we test all points in MOBR of circle O. Which based on the provided table the closest neighbor from point p is rectangle 6, and its distance is 2.5.

Q1b) Show the execution trace of the query following the **One-phase** Nearest Neighbor algorithm which recursively checks and eliminate nodes dominated by some other nodes in R-tree index and check the remaining data blocks for nearest neighbor.
Fill out the following table to list the rectangles and MBRs tested in each R-tree level.

Level	Rectangles/MBRs
1st level	We will have X and Y which they do not eliminate anything.
2nd level	C eliminates D, B, A.
Data rectangle	X, Y, C, 6, 7, 8.

Chapter 7: Spatial Data Mining

Q3. Given a distribution for a total of 19 points shown in Figure 3.1, answer the following questions.

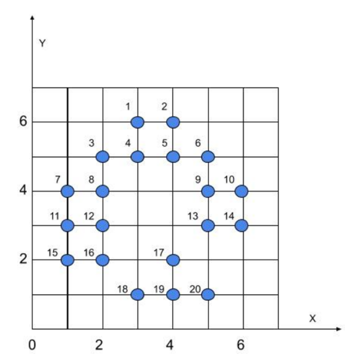


Figure 3.1. Points distribution

Q3(a). Trace execution of K-Means algorithm with $K = 2$ for the points in Figure 3.1 and fill out the following table with results in each iteration. Assume that the seeds are (5,4) and (3,6). (The answers might not fill all the rows of the empty tables.) **Note.** If there are some points that are equi distant from cluster center, you can place the points in any of the two cluster.

Iteration ID	Points ∈ Class-1	Center of Class 1	Points ∈ Class-2	Center of Class 2
1	6,9,10,13,14,16,17,18,19,20	(5,4)	1,2,3,4,5,7,8,11,12,15	(3,6)

2	9,10,13,14,17,18,19,20	(4.3,2.3)	1,2,3,4,5,6,7,8,11,12,15,16	(2.3,4.3)
3	9,10,13,14,17,18,19,20	(4.75, 2.38)	1,2,3,4,5,6,7,8,11,12,15,16	(2.5,4.2)

Since no new element added to either cluster, then that is our final solution, with value on row 3.

Q3(b). Density-based spatial clustering of applications with noise (DBSCAN) is another data clustering algorithm. For the purpose of DBSCAN clustering, the points to be clustered are classified as *core points*, (*density-*)*reachable points* and *outliers*, as follows:

- A point p is a core point if at least $minPts$ points are within distance ε (ε is the maximum radius of the neighborhood from p) of it (**including** $\boldsymbol{p}$). Those points are said to be *directly reachable* from p. By definition, no points are *directly reachable* from a non-core point.
- A point q is reachable from p if there is a path p_1, ..., p_n with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i (all the points on the path must be core points, with the possible exception of q).
- All points not reachable from any other point are outliers.

For example, as shown in Figure 3.2 if $minPts = 4$ and $\varepsilon = 1$, red points are core points, and the yellow ones are density-reachable from them, and outliers are represented in green.

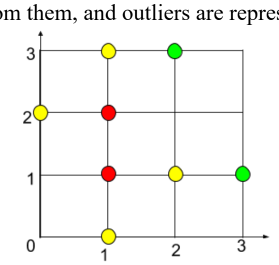


Figure 3.2. A DBSCAN example

Use DBSCAN to cluster the points in Figure 3.1 and fill out Table 1 and 2 with information of the final clusters and outliers. $\boldsymbol{minPts = 4}$ and $\boldsymbol{\varepsilon = 1}$. (The answers might not fill all the rows of the empty tables.)

Table 2. Clusters

Cluster #	Density-reachable points	Core points
1	{1, 3, 2, 6, 7, 15, 16, 10, 13}	{4, 5, 8, 11, 12, 9}
3	{17, 18, 20}	{19}

Table 3. Outliers

Point #	14

Q3(c). In order to detect hotspots with high log likelihood ratio (LogLR), we check the LogLR of four circular areas shown in Figure 3.3 which are potential hotspots. LogLR is calculated as follows.

$$\log LR = c \cdot \log\frac{c}{e} + (c_{tot} - c) \cdot \log\frac{c_{tot} - c}{c_{tot} - e}; \qquad e = c_{tot} \cdot \frac{area_c}{area_{tot}}$$

where c is the number of points (black dots •) inside a candidate region, c_{tot} is total number of points in the study area, e is the expectation of number of points inside the candidate region, $area_c$ is the area of the candidate region and $area_{tot}$ is that of the whole region.

Note: Study area is the surface area of the graph. Here it is 7*7 = 49 units. Similarly, for A it would be the surface area of the circle (Pi r ^2).

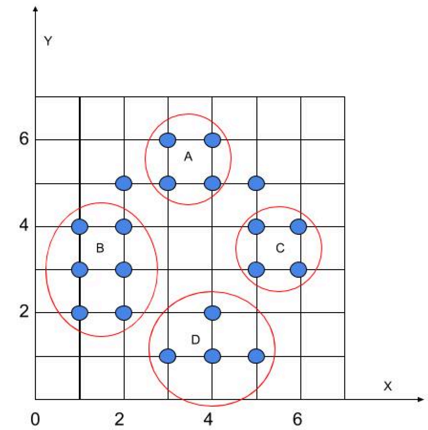


Figure 3.3. Points with circular areas

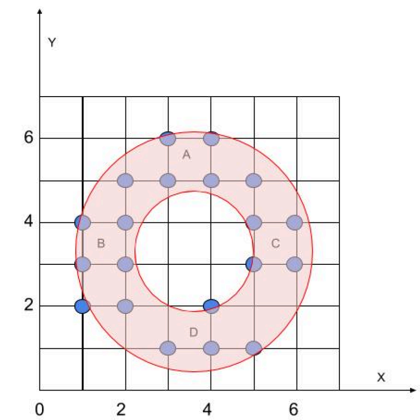


Figure 3.4. Points with a donut-shape area

Table 4: Information on areas

Attribute	Whole region	A	B	C	D
Area	49	1.8	4.5	1.8	4.5

Fill out Table 5 to show the results of log likelihood ratio computation and select one candidate region to be the most likely hotspot.

Table 5. Likelihood ratio results

	A	B	C	D
c	4	6	4	4
e	20*1.8/49=0.735	20*4.5/49=1.837	20*1.8/49=0.735	20*4.5/49=1.837
c_{tot} - c	16	14	16	16
c_{tot} - e	19.265	18.163	19.265	18.163
LogLR	3.805	3.457	3.805	1.084

The most likely candidate region to be a hotspot is B.

Q3(d). After learning the nature of these point records, we propose that maybe donut-shape areas are more suitable for delineating hotspots in this scenario. Compute the LogLR of the area shown in Figure 3.4 and test this hypothesis. The area of the donut-shape region is 12.5.

$$\boldsymbol{LogLR = 18\, ln\left(\frac{18}{5.102}\right) + (20-18)\, ln\left(\frac{2}{20-5.102}\right) = 18.677}$$

This suggests the hypothesis is valid, the LogLR score of a donut-shaped region is much higher than the ellipses.

Q4 Consider 9 cells, namely, A, B, …, I, in a raster dataset. The location and attribute values of these cells are shown in Figure 4. Assume that the neighborhood of a cell consists of the cells sharing an edge with the cell. For example, B and D are neighbors of C.

A	B	C	D	E	F	G	H	I

Figure 4(a). Location of Cells

42	39	37	36	34	41	33	46	41

Figure 4(b). Attribute Values of Cells

Q4a) Consider spatial Z-test, a quantitative test for spatial outliers. Fill out the following table to prepare the calculation of the values of spatial test $Z_s(x)$.

x	A	B	C	D	E	F	G	H	I
$Mean_{y \in N(x)}(f(y))$	**39**	**39.5**	**37.5**	**35.5**	**38.5**	**33.5**	**43.5**	**37**	**46**
$S(x)$	**3**	**-0.5**	**-0.5**	**0.5**	**-4.5**	**7.5**	**-10.5**	**9**	**-5**

Note that x and y are data cells, $N(x)$ is the neighborhood of x, $f(y)$ is the attribute value of cells y, and $S(x)$ is the deviation of x's attribute value from its neighborhood mean value.

Recall that $Z_{s(x)} = |\frac{S(x)-\mu_s}{\sigma_s}|$, where μ_s is the mean of $S(x)$, and σ_s is the standard deviation of $S(x)$ which is 6.148 in this case.
i) What is the value of μ_s? ~~**4.556,**~~ Final answer is: -0.11
ii) Assume the threshold to determine an outlier is $Z_{s(x)} \geq 1.5$. List the cells which are spatial outliers.

x	A	B	C	D	E	F	G	H	I
$Z_{s(x)}$	**0.253**	**0.660**	**0.660**	**0.660**	**0.009**	**0.479**	**0.967**	**0.723**	**0.072**

There are no cells which are spatial outliers given a threshold of 1.5

Q4b) What is the worst-case asymptotic time complexity of the spatial Z-test to identify spatial outliers given *n* cells? Assume that the neighborhood size is bounded by a constant, e.g., 4.

If the cells are organized spatially, then we can identify spatial outliers in O(n) time. However, this also requires an O(n) storage requirement – on the first run-through we can keep a running sum at each cell of its neighbors. We normalize those, and then run down the list three more times calculating the mean, cell deviation, and z score.

Q4c) Recall the Variogram cloud is a graphical test for spatial outliers. What is the worst-case asymptotic time complexity of Variogram cloud when the input raster has *n* cells?

Variogram is the difference between variance of two locations, which we can write for two pairs as (n*(n-1))/2. As the formula is demonstrated the time complexity of Variogram is O(n^2).