#### CSci 5715, Fall 19: Homework 2

Due on 10/8 before class

In-class students: hard copy; UNITE students: Email.

#### Table of Participation

|  |  |  |
| --- | --- | --- |
| Question ID | Answer drafted by | Answer reviewed by |
| 1 | Alex |  |
| 2 | Alex |  |
| 3 | Alex |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |

#### Chapter 2: Spatial Concepts and Data Models

**Q1:** The following figure (Figure 1) shows a part of the Entity-Relationship (ER) Diagram of a database for turf grass evaluation system where experiment is performed in the **State** of Minnesota across multiple **Sites** where each State with their respective sites are assigned with their unique ids. The database records multiple sites where each site is partition into multiple **Fragments** (identified with a unique\_ID) which are further divided into multiple **Plots**. Each plot has a unique id which and multiple plots from each fragment is stored as a single **Entry** in the database.

A trail **Experiment** (with information such as trail year and trail name) is conducted on multiple entries where each entry is tested on multiple ratings which include attributes based on rating types and timestamp (e.g. Month, Day and Year). The ratings are subdivided based on Quality, Color, Density, Disease and Insect Damage along with their unique value id respectively (e.g. Quality\_value, Color\_value etc.).

The database also records multiple sponsors along with sponsor names which supports multiple entries and each entry is tested by distinct cultivators with their respective names.

The database is expected to answer questions such as the follows.

1. How many plots are there in the state of Minnesota?
2. How many plots in a particular state are tested while conducting experiments?
3. How can we fetch disease rating on a specific entry?
4. How many entries can be found for a specific experiment?

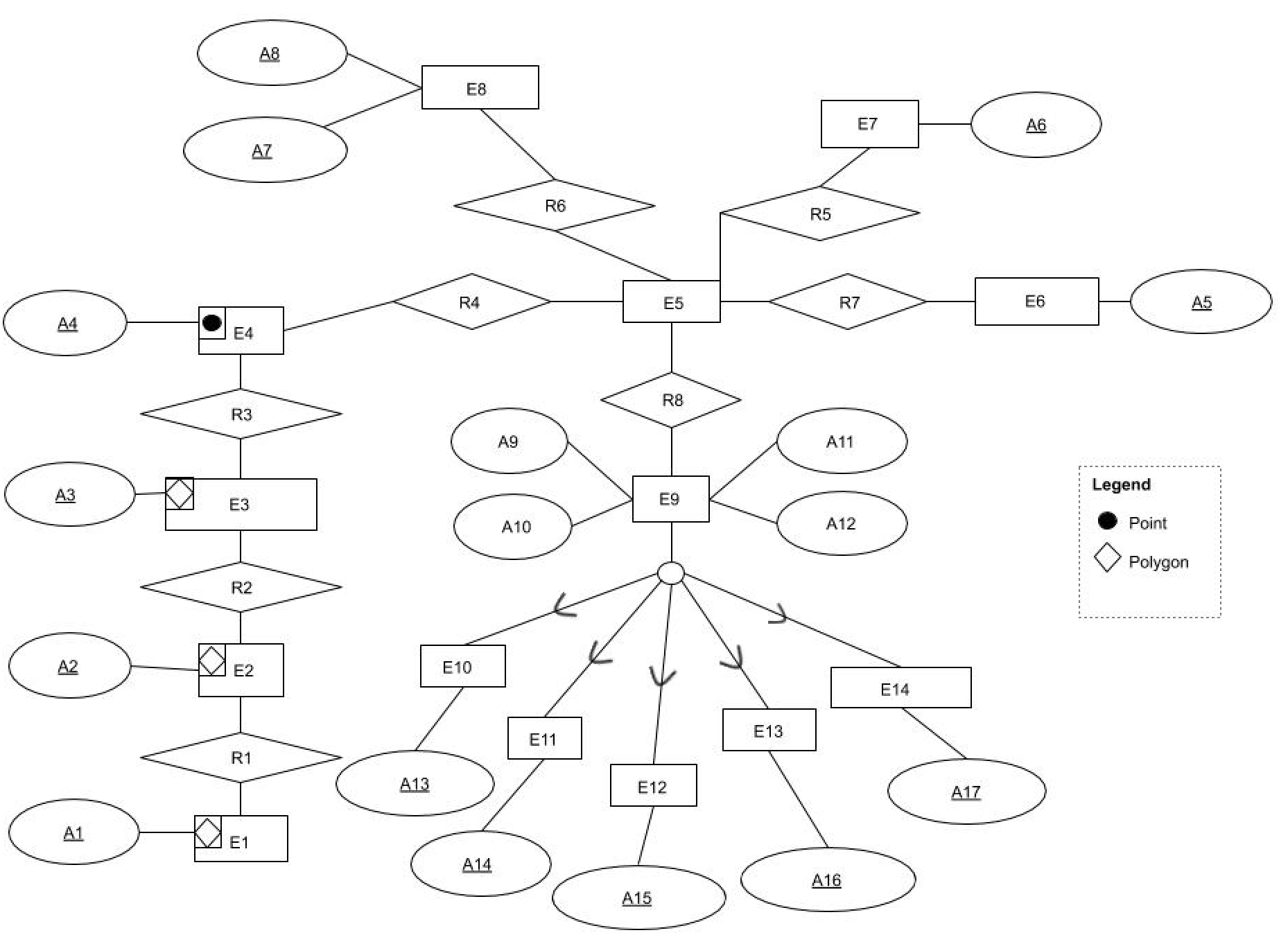


Figure 1. E-R Diagram for Turf Grass Experiment

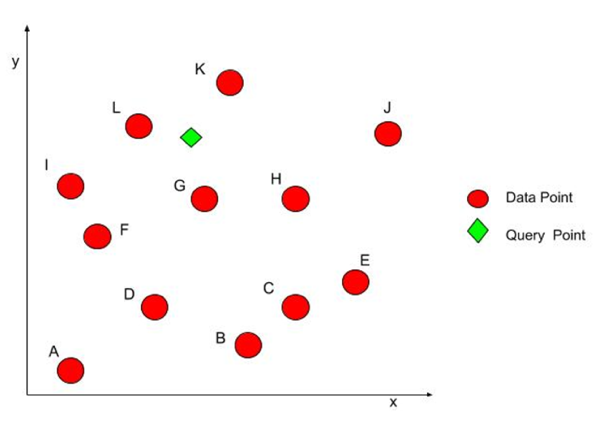
Complete the ER-diagram with entity and attribute names and the names and cardinality of relationships.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entities | |  | Entities | |
| E1 | State |  | E8 | Experiments |
| E2 | Site |  | E9 | Rating |
| E3 | Fragment |  | E10 | Quantity |
| E4 | Plot |  | E11 | Color |
| E5 | Entry |  | E12 | Density |
| E6 | Sponser |  | E13 | Disease |
| E7 | Cultivator |  | E14 | Insect Damage |
| Attributes | |  | Attribute | |
| A1 | State\_ID |  | A10 | Rating\_Month |
| A2 | Site\_ID |  | A11 | Rating\_Year |
| A3 | Fragment\_ID |  | A12 | Rating\_Day |
| A4 | Plot\_ID |  | A13 | Quantity\_Value |
| A5 | Sponser\_Name |  | A14 | Color\_Value |
| A6 | Cultivator\_Name |  | A15 | Density\_Value |
| A7 | Trial\_Year |  | A16 | Disease\_Value |
| A8 | Trail Name |  | A17 | Insect\_Damage\_Value |
| A9 | Rating\_Type |  |  |  |

|  |  |  |
| --- | --- | --- |
| Relationships | | |
|  | Name | Cardinality |
| R1 (E1🡪E2) | Contains | One to Many |
| R2 (E2🡪E3) | Contains | One to Many |
| R3 (E3🡪E4) | Contains | One to Many |
| R4 (E4🡪E5) | Member of | Many to One |
| R5 (E5🡪E7) | Tested by | Many to One |
| R6 (E5🡪 E8) | Member of | Many to One |
| R7 (E5 🡪 E6) | Sponsered by | Many to Many |
| R8 (E5 🡪 E9) | Tested on | One to Many |

**Chapter 4: Spatial Storage and Indexing**

**Q2.** Given 12 points A through H in space shown in Figure 2, which are stored in a spatial database. Assume that one data page can store the information of at most one point.

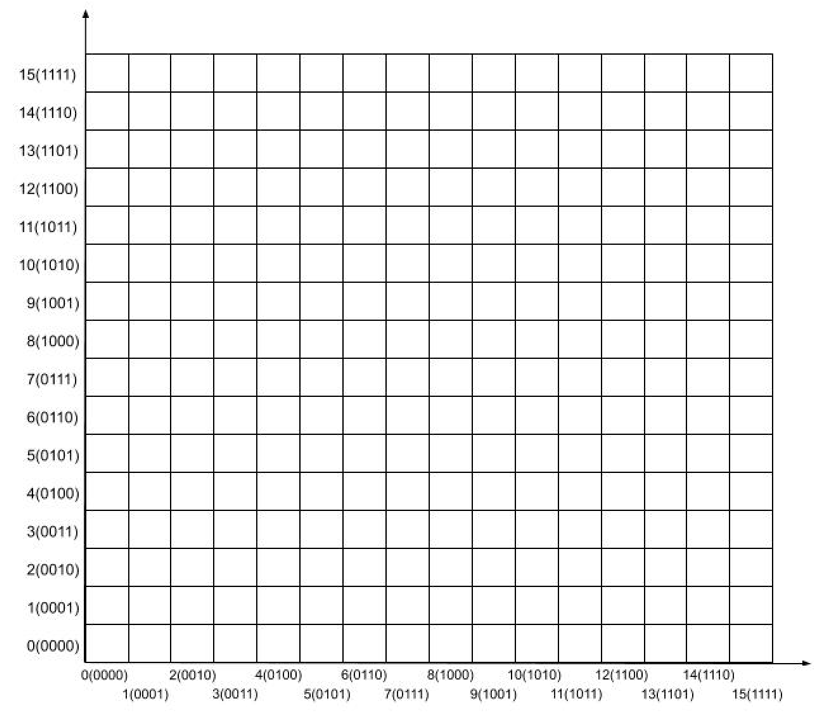
****

**Figure 2.** 12 points in space (best in color)

**Q2(a)**. If the points are saved without order, what is the minimum number of data pages need to be retrieved to know whether there is a point at the query point?

The minimum number of pages needed to be retrieved is 12 (or n = the number of points or pages) when the points are saved without order.

**Q2(b)**. After estimating the data pages needed to retrieve in Q2(a), you decide to add an index to the database to reduce the IO cost of this query. In CSci 5715 you learned that **Hilbert curve** is a space filling curve that may help. Suppose that the area is represented as a grid with cells (shown in Figure 3). Fill out the following table to determine the Hilbert curve value (X-priority and Y-priority, respectively) of the cells. (**Hint:** Review algorithm for Hilbert curve in Chapter 4, pg. 92-93).



**Figure 3.** Grid with cells

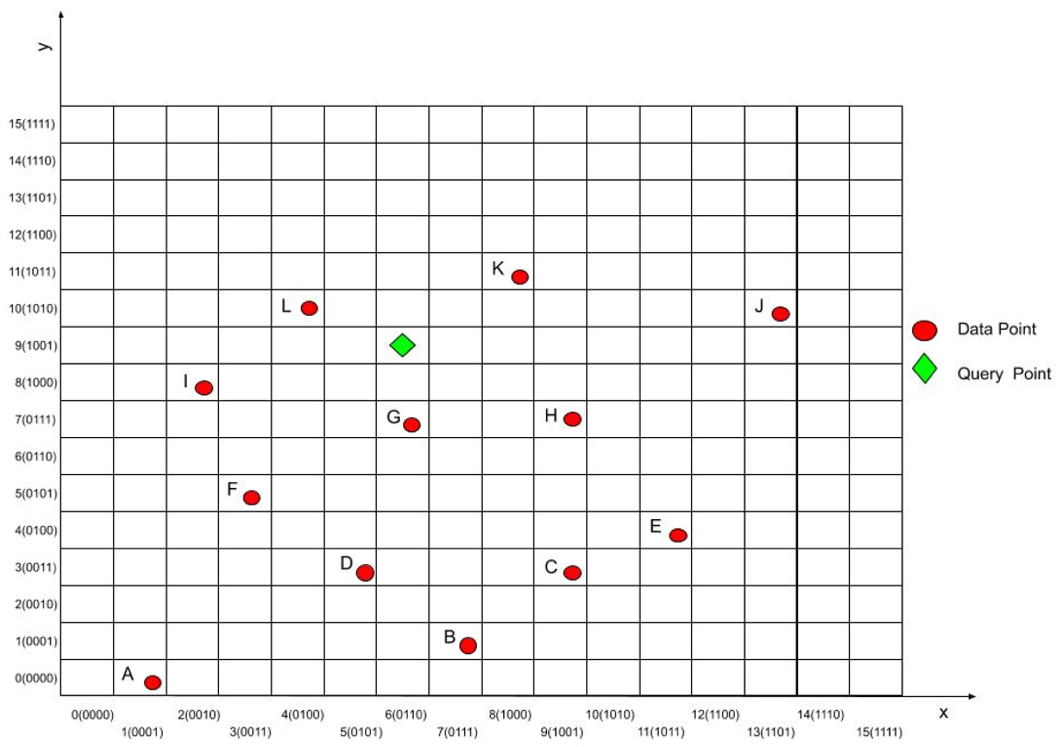
**Table 1.** X-priority Hilbert curve.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cell #  (x, y) | Step 2: Interleave bits of  x and y | Step 3 & 4:  Divide into 2-bit strings and give decimal values | Step 5:  Change the string  based on the jth digit | Hilbert curve value |
| (0001, 0000) | 00000010 | 0003 🡪 0001🡪 0003🡪 0001 | 00000001 | 1 |
| (0111, 0001) | 00101011 | 0223 🡪 0221 🡪 0221 🡪 0221 | 00101001 | 41 |
| (1001, 0011) | 10000111 | 2013 🡪 2013 🡪 2031 🡪 2031 | 10001101 | 141 |
| (0101, 0011) | 00100111 | 0213 🡪 0231 🡪 0231 🡪 0231 | 00101101 | 45 |
| (1011, 0100) | 10011010 | 2122 🡪 2122 🡪 2122 🡪 2122 | 10011010 | 154 |
| (0011, 0101) | 00011011 | 0123 🡪 0321 🡪 0301 🡪 0303 | 00110011 | 51 |
| (0110, 0111) | 00111101 | 0331 🡪 0113 🡪 0113 🡪 0113 | 00010111 | 23 |
| (1001, 0111) | 10010111 | 2113 🡪 2113 🡪 2113 🡪 2113 | 10010111 | 151 |
| (0010, 1000) | 01001000 | 1020 🡪 1020 🡪 1020 🡪 1020 | 01001000 | 72 |
| (1101, 1010) | 11100110 | 3212 🡪 3010 🡪 3030 🡪 3032 | 11001110 | 206 |
| (1000, 1011) | 11000101 | 3011 🡪 3211 🡪 3211 🡪 3211 | 11100101 | 229 |
| (0100, 1010) | 01100100 | 1210 🡪 1210 🡪 1210 🡪 1210 | 01100100 | 100 |

**Table 2.** Y-priority Hilbert curve

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cell #  (x, y) | Step 2: Interleave bits of  x and y | Step 3 & 4:  Divide into 2-bit strings and give decimal values | Step 5:  Change the string  based on the jth digit | Hilbert curve value |
| (0001, 0000) | 00000001 | 0001 🡪 0003 🡪 0001🡪0003 | 00000011 | 3 |
| (0111, 0001) | 00010111 | 0113 🡪 0331 🡪 0331 🡪 0331 | 00111101 | 61 |
| (1001, 0011) | 01001011 | 1023 🡪 1023 🡪 1021 🡪 1021 | 01001001 | 73 |
| (0101, 0011) | 00011011 | 0123 🡪 0321 🡪 0301 🡪 0303 | 00110011 | 51 |
| (1011, 0100) | 01100101 | 1211 🡪 1211 🡪 1211 🡪 1211 | 01100101 | 101 |
| (0011, 0101) | 00100111 | 0213 🡪 0231 🡪 0231 🡪 0231 | 00101101 | 45 |
| (0110, 0111) | 00111110 | 0332 🡪 0112 🡪 0112 🡪 0112 | 00010110 | 22 |
| (1001, 0111) | 01101011 | 1223 🡪 1223 🡪 1223 🡪 1223 | 01101011 | 107 |
| (0010, 1000) | 10000100 | 2010 🡪 2010 🡪 2030 🡪 2032 | 10001110 | 142 |
| (1101, 1010) | 11011001 | 3121 🡪 3101 🡪 3101 🡪 3131 | 11011101 | 221 |
| (1000, 1011) | 11001010 | 3022 🡪 3200 🡪 3200 🡪 3200 | 11100000 | 224 |
| (0100, 1010) | 10011000 | 2120­­­­­ 🡪 2120 🡪 2120 🡪 2120 | 10011000 | 152 |

**Q2(c).** Now points are saved in a database in order of ascending Hilbert curve value.



**Figure 4.** Points in space with Hilbert curve grid

1. Show the sequence of points saved in the database. Use “()” to group the points in to data pages, e.g., (A)(B) ... (H).

|  |  |
| --- | --- |
| Type | Sequence |
| Ascending Hilbert curve value (X-Priority) | (A)(G)(B)(D)(F)(I)(L)(C)(H)(E)(J)(K) |
| Ascending Hilbert curve value (Y-Priority) | (A)(G)(F)(D)(B)(C)(E)(H)(I)(L)(J)(K) |

1. Show the process of determining which data page to retrieve using a balanced binary search tree in which each node is a data block. Recall that the query point is shown as a green diamond in the figure.

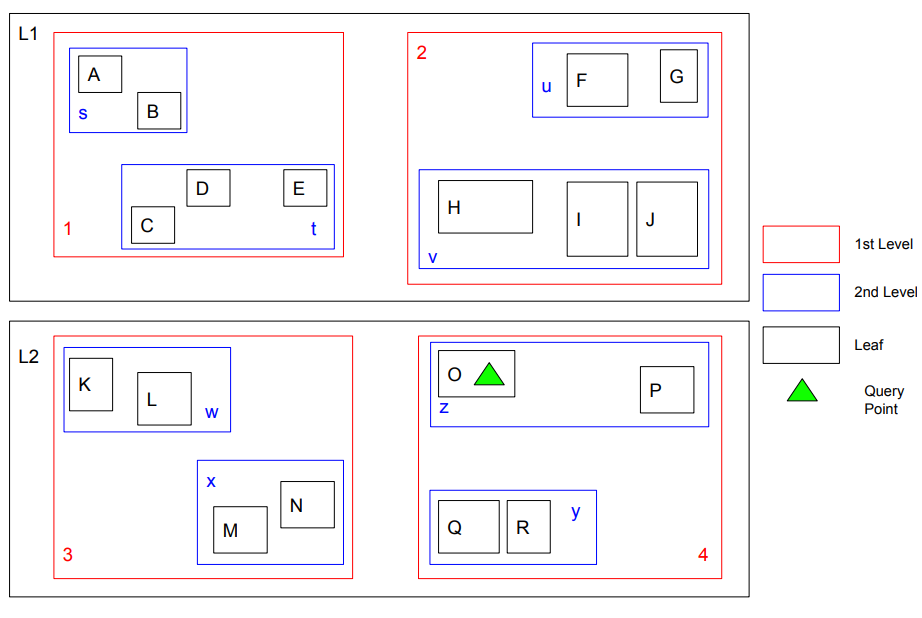
The point query lies at (6,9) or (0110, 1001). We will assume an x-priority Hilbert curve, although the process is very similar for y-priority curves. The bit interleave value is 01101001, the 2-bit string is “1221” which doesn’t change under rotations or reflections. So, the Hilbert curve value is 105.

Our binary tree is constructed as in the picture below. The Hilbert curve value is stored and the node points to the data block on disk. Our retrieval path is traced in blue, obtained by checking the value 105 against each node.

A close up of a map

Description automatically generated

**Q3(a).** Draw an R+-tree for the nested collection of rectangles shown in Figure 5. The root node (level 0) has two children L1 and L2 at level 1 shown using black solid lines. Both L1, and L2 have two children, namely 1, 2, 3 and 4 at level 2 shown using red lines. Third level nodes are s, t, u, v, w, x, y and z which are represented by light blue rectangles. The leaf nodes are represented by the dashed rectangle including A, B, …, R.



**Figure 5.** Rectangles indexed using an R+-tree (Best in color)

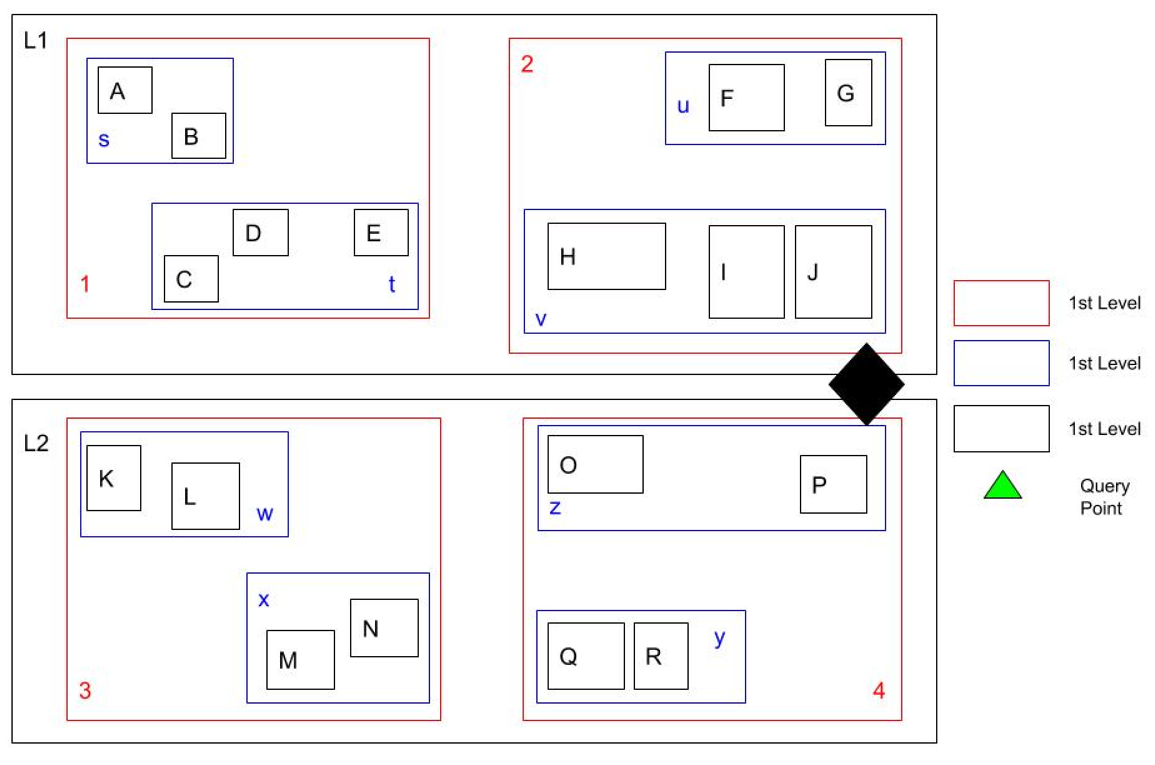
**A close up of a map

Description automatically generated**

**Q3(b).** Consider the R+-tree for Figure 5. List all the R+-tree nodes retrieved by a point query for the location marked by the green triangle (query point).

R, L2, 4, z, (O,P)

**Q3(c).** Suppose a maximum of 4 entries per node is allowed. Consider insertion of a rectangular object “Q” as shown in Figure 6:



**Figure 6.** Insertion of new object “Q” (Best in color)

Insert the new object considering the properties of

1. **R-Tree:** New object goes to the (and only one) nearest leaf node. MOBR of sibling nodes in R-Tree can overlap.

The object “Q” is in the leaf node z. 4 and L2 now overlap with their siblings 2 and L1, respectively.

A close up of a necklace

Description automatically generated

1. **R+-Tree:** New object goes to one or more nearest leaf(ves) MOBR of sibling R+-tree are disjoint.

At this point, we note that in the image provided, the nodes are not minimum bounding rectangles, and that we could redraw the boundaries so that L2, 4, and z fully encapsulate object “Q” without overlapping with L1 or 2. However, if we assume that these really are minimum bounding rectangles, then object “Q” will have to be shared between z and v. L2, 4, and z must expand upwards to encapsulate the lower half of “Q”’s bounding box, and L1, 2, and v must expand lower to encapsulate the upper half of “Q”’s bounding box. Therefore, “Q” will exist in both v and z

A close up of a necklace

Description automatically generated

1. **R\*-Tree:** Reorganize leaves near new object to reduce empty area within MOBRs of R\*-Tree as well as overlap between MOBRs of sibling nodes in R\*-tree.

In order to reduce overlap between minimum bounding rectangles, we can have L1 encapsulate 1 and 3 rather than 1 and 2. Likewise, L2 will encapsulate 2 and 4

**A screenshot of a cell phone

Description automatically generated**

**A close up of a necklace

Description automatically generated**

**Q4**. **Draw a final 2D grid file structures** generated by inserting points with the following sequence of locations: (0,0), (1,1), (2,2), (3,2), (4,5), (5,5), (6,6), (7,6), (8,8), (9,9), (10,11). Assume each disk page can hold at most three records. Clearly show the grid directory, linear scales, and data pages. Splitting policy applies alternating directions, i.e., first x then y, and the splitting boundary divides the overflown bucket into two with the same size.

Let A=(0,0), B=(1,1), C=(2,2), D=(3,2), E=(4,5), F=(5,5), G=(6,6), H=(7,6), I=(8,8), J=(9,9), K=(10,11)

**State 1:**

A screenshot of a cell phone

Description automatically generated

**State 2:**

A screenshot of a cell phone

Description automatically generated

**State 3:**

A screenshot of a cell phone

Description automatically generated

**State 4:**

A screenshot of a cell phone

Description automatically generated

**State 5:**

A screenshot of a cell phone

Description automatically generated

#### Chapter 6: Spatial Network

**Q5.** Given a network shown in Figure 7 with Nodes A through I and the directed edge between them.

**Figure 7.** A directed graph

Assume that the directed edges are stored in a table “EDGE” as shown below.

EDGE

|  |  |
| --- | --- |
| Origin | Destination |
| A | E |
| … | … |
| J | I |

(i) Write a SQL SELECT statement with “WITH RECURSIVE” clause to list the node which can reach Node I in this directed graph.

|  |
| --- |
| WITH RECURSIVE X (end)  AS (SELECT Origin  FROM EDGE  WHERE Destination = ‘I’  )  UNION  (SELECT EDGE.Origin  FROM EDGE, X  WHERE EDGE.Destination = X.end  ); |

(ii) List the results of the following query.

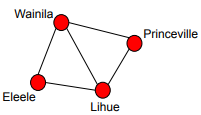
|  |
| --- |
| SELECT ORIGIN  FROM EDGE  CONNECT BY PRIOR ORIGIN = DESTINATION  CONNECT BY DESTINATION <> 'F'  START WITH DESTINATION = 'I' |

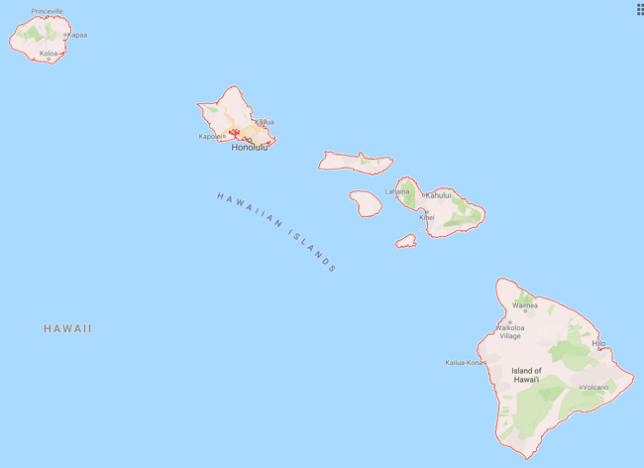
J, H, F, E, B, A

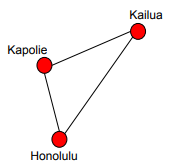
(iii) What is the minimum number of cut-edges to partition the nodes of the directed graph in Figure 7 into 2 equal halves (of 5 nodes each)? List the cut-edges to justify your answer.

The minimum number of cut-edges is 3: E->H, F->J, and G->F OR B->E, E->F, F->J

**Q6**. Hawaii consists of several islands. Assume that the main means of transportation within each island is road transportation, while transportation between islands relies heavily on boats and airplanes. According to the map of Hawaii, we can represent a simplified version of its transportation network as shown in Figure 8, where black solid lines delineate the boundaries of islands (i.e., “A”, “B”, “C”, and “D”), and black solid lines are roads inside islands, and blue dash lines are sea/air routes. Cities connected by sea/air routes are ports, e.g., Honolulu and Hilo.

****



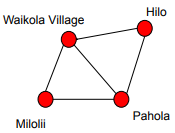
****

**A**

****

**C**

**B**

****

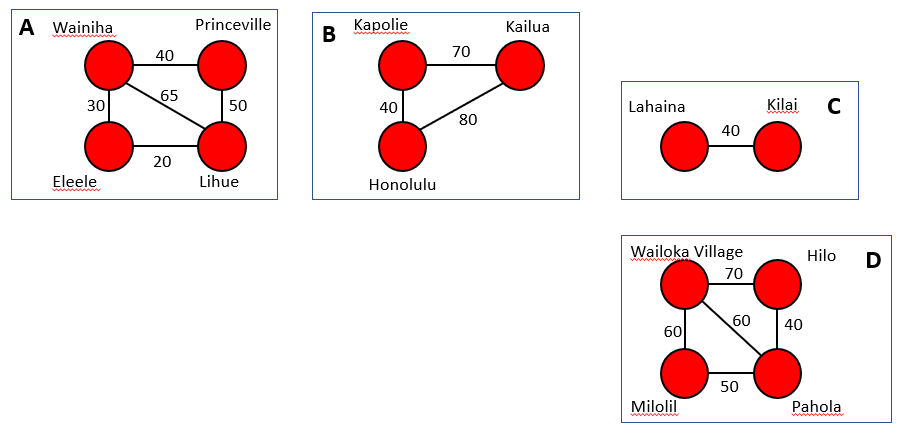
**D**

**Figure 8.** Simplified Transportation Map of Hawaii

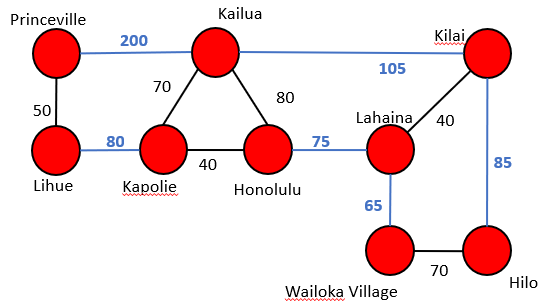
The time cost of traveling between places are shown below. All paths are undirected. Sea/air routes are highlighted as blue.

|  |  |
| --- | --- |
| Path | Time cost (min) |
| Wainilla, Princeville | 40 |
| Wainilla, Lihue | 65 |
| Wainilla, Eleele | 30 |
| Eleele, Lihue | 20 |
| Princeville, Lihue | 50 |
| Kapolie, Kailua | 70 |
| Kapolie, Honolulu | 40 |
| Kailue, Honolulu | 80 |
| Lahaina, Kilai | 40 |
| Waikola Village, Hillo | 70 |
| Waikola Village, Milolii | 60 |
| Waikola Village, Pahola | 60 |
| Milolii, Pahola | 50 |
| Hilo, Pahola | 40 |
| Princeville, Kailua | 200 |
| Lihue, Kapolie | 80 |
| Honolulu, Lahaina | 75 |
| Kailua, Kilai | 105 |
| Lahaina, Waikola Village | 65 |
| Kilai, Hilo | 85 |

1. Suppose fragment graphs are defined by road transportation using each island as a fragment, draw fragment graphs of the transportation network of Hawaii.



1. Suppose boundary graph is defined by sea/air routes, draw boundary graph of the transportation network of Hawaii.



1. Using the hierarchical routing strategy (section 6.4.4, page 170 in the textbook). Describe the steps of finding the shortest path between Wainilia and Pahola briefly.

Briefly, the routing algorithm is:

1. Wainiha and Pahola are both local nodes, so we fall into Case 4 in the hierarchical routing algorithm.
2. We iterate through the boundary nodes of Wainiha’s fragment graph: Princeville and Lihue
   1. We iterate through the boundary nodes of Pahola’s fragment graph: Wailoka Village and Hilo
      1. We do three shortest path queries and add the minimum cost to go from Wainiha to the current boundary, the minimum cost to go from Pahola to its current boundary, and the minimum cost to go from the current boundary of Wainiha’s fragment to the current boundary of Pahola’s fragment.
      2. We store the results of the boundary nodes with the smallest total cost.

So, for our example above, we compute the shortest path from boundary nodes Princeville and Lihue to Hilo and Wailoka.

* Princeville -> Lihue -> Kapolie -> Honolulu -> Lahaina -> Kilai -> Hilo (cost 370)
  + Adding in Wainiha -> Princeville and Hilo -> Pahola adds 80 for a total cost of 450
* Princeville -> Lihue -> Kapolie -> Honolulu -> Lahaina -> Wailoka Village (cost 310)
  + Adding in Wainiha -> Princeville and Wailoka Village -> Pahola adds 100 for a total cost of 410
* Lihue -> Kapolie -> Honolulu -> Lahaina -> Kilai -> Hilo (cost 320)
  + Adding in Wainiha -> Lihue and Hilo -> Pahola adds 105 for a total cost of 425
* Lihue -> Kapolie -> Honolulu -> Lahaina -> Wailoka Village (cost 260)
  + Adding in Wainiha -> Lihue and Wailoka Village -> Pahola adds 125 for a total cost of 385

So, our smallest cost is the path Wainiha -> Lihue -> Kapolie -> Honolulu -> Lahaina -> Wailoka Village -> Pahola for a total cost of 385