CPT_S 415

Big Data, Fall 2020

Homework # 2

Tayyab Munir

11716089

Question 1:

a) Consider the database instance you gave in Assignment 1 Question 2 (a). Assume now that you don't have any schema. Give an XML document to represent the tuples as the fact about the airports.

```xml
<!-- Airport Table-->
 <Airport>
    <AirportId>17</AirportId>
    <Name>San Jose International</Name>
    <City>California </City>
    <Country> United States of America</Country>
    <IATA> SJI</IATA>
    <ICAO> GELL</ICAO>
    <Latitutde>56.5606 </Latitutde>
    <Longitude>11.461941 </Longitude>
    <Altitude> 53 </Altitude>
    <Timezone>0</Timezone>
    <DST>A</DST>
    <Tzdatabasetimezone>America/California </Tzdatabasetimezone>
    <Type>Airport </Type>
    <Source>ourAirport </Source>

    <AirportId>515</AirportId>
    <Name>London Heathrow Airport</Name>
    <City>London</City>
    <Country>United Kingdom</Country>
    <IATA>LHR</IATA>
    <ICAO>EGLL</ICAO>
    <Latitutde>51.4706</Latitutde>
    <Longitude>-0.461941</Longitude>
    <Altitude>83</Altitude>
    <Timezone>0</Timezone>
    <DST>E</DST>
    <Tzdatabasetimezone>Europe/London</Tzdatabasetimezone>
    <Type>airport</Type>
    <Source>OurAirports</Source>

    <AirportId>8900</AirportId>
    <Name>Seattle Tacoma</Name>
    <City>Seattle</City>
    <Country>United States of America</Country>
    <IATA>SJI</IATA>
    <ICAO>GELL</ICAO>
    <Latitutde>56.5606</Latitutde>
    <Longitude>11.461941</Longitude>
    <Altitude>53</Altitude>
    <Timezone>0</Timezone>
    <DST>A</DST>
   <Tzdatabasetimezone>America/California</Tzdatabasetimezone>
    <Type>airport</Type>
```

```xml
    <Source>OurAirports</Source>
  </Airport>
```

b) Consider the relational schemas you gave in Assignment 1 Question 2 (b). Give an XML schema representation of each relational schema. How do you encode keys? Foreign keys?

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Airport Table-->
 <xsd:element name="Airport">
   <xsd:complexType>
     <xsd:attribute name="AirportId" type="sqltypes:int" use="required" />
     <xsd:key name="AirportId">
        <xs:selector xpath="Airport/AirportId"/>
        <xs:field xpath="."/>
      </xs:key>
     <xsd:attribute name="Name" use="required">
       <xsd:simpleType>
         <xsd:restriction base="sqltypes:nvarchar">
           <xsd:maxLength value="100" />
         </xsd:restriction>
       </xsd:simpleType>
     </xsd:attribute>
     <xsd:attribute name="City" use="required">
       <xsd:simpleType>
         <xsd:restriction base="sqltypes:nvarchar">
           <xsd:maxLength value="50" />
         </xsd:restriction>
       </xsd:simpleType>
     </xsd:attribute>
     <xsd:attribute name="Country" use="required">
       <xsd:simpleType>
         <xsd:restriction base="sqltypes:nvarchar">
           <xsd:maxLength value="50" />
         </xsd:restriction>
       </xsd:simpleType>
     </xsd:attribute>
     <xsd:attribute name="IATA">
       <xsd:simpleType>
         <xsd:restriction base="sqltypes:nvarchar">
           <xsd:maxLength value="3" />
         </xsd:restriction>
       </xsd:simpleType>
     </xsd:attribute>
     <xsd:attribute name="ICAO">
       <xsd:simpleType>
         <xsd:restriction base="sqltypes:nvarchar">
           <xsd:maxLength value="4" />
         </xsd:restriction>
       </xsd:simpleType>
     </xsd:attribute>
```

```
<xsd:attribute name="Latitude_x0020_" use="required">
  <xsd:simpleType>
    <xsd:restriction base="sqltypes:decimal">
      <xsd:totalDigits value="18" />
      <xsd:fractionDigits value="8" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="Longitude" use="required">
  <xsd:simpleType>
    <xsd:restriction base="sqltypes:decimal">
      <xsd:totalDigits value="18" />
      <xsd:fractionDigits value="8" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="Altitude_x0020_" type="sqltypes:int" use="required" />
<xsd:attribute name="Timezone" use="required">
  <xsd:simpleType>
    <xsd:restriction base="sqltypes:decimal">
      <xsd:totalDigits value="18" />
      <xsd:fractionDigits value="8" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="DST" use="required">
  <xsd:simpleType>
    <xsd:restriction base="sqltypes:nvarchar">
      <xsd:maxLength value="5" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="Tz_x0020_database_x0020_time_x0020_zone" use="required">
  <xsd:simpleType>
    <xsd:restriction base="sqltypes:nvarchar">
      <xsd:maxLength value="50" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="Type" use="required">
  <xsd:simpleType>
    <xsd:restriction base="sqltypes:nvarchar">
      <xsd:maxLength value="50" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="Source" use="required">
  <xsd:simpleType>
    <xsd:restriction base="sqltypes:nvarchar">
      <xsd:maxLength value="50" />
    </xsd:restriction>
  </xsd:simpleType>
```

```
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>

<!-- Airline Table-->
  <xsd:element name="Airline">
    <xsd:complexType>
      <xsd:attribute name="AirlineId" type="sqltypes:int" use="required" />
      <xsd:key name="AirlineId">
          <xs:selector xpath="Airline/AirlineId"/>
          <xs:field xpath="."/>
       </xs:key>
      <xsd:attribute name="Name" use="required">
        <xsd:simpleType>
          <xsd:restriction base="sqltypes:nvarchar">
            <xsd:maxLength value="50" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="Alias" use="required">
        <xsd:simpleType>
          <xsd:restriction base="sqltypes:nvarchar">
            <xsd:maxLength value="3" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="IATA">
        <xsd:simpleType>
          <xsd:restriction base="sqltypes:nvarchar">
            <xsd:maxLength value="3" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="ICAO">
        <xsd:simpleType>
          <xsd:restriction base="sqltypes:nvarchar">
            <xsd:maxLength value="4" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="CallSign" use="required">
        <xsd:simpleType>
          <xsd:restriction base="sqltypes:nvarchar">
            <xsd:maxLength value="50" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="Country" use="required">
        <xsd:simpleType>
          <xsd:restriction base="sqltypes:nvarchar">
            <xsd:maxLength value="50" />
```

```
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="Active" type="sqltypes:bit" use="required" />
  </xsd:complexType>
</xsd:element>

<!-- Route Table-->
<xsd:element name="Routes">
  <xsd:complexType>
    <xsd:attribute name="Airline" use="required">
      <xsd:simpleType>
        <xsd:restriction base="sqltypes:nvarchar">
          <xsd:maxLength value="4" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="AirlineId" type="sqltypes:int" use="required" />
    <xs:keyref name="Airline1" refer=" AirlineId ">
        <xs:selector xpath="Airline/AirlineId "/>
        <xs:field xpath="."/>
     </xs:keyref>
    <xsd:attribute name="SourceAirport" use="required">
      <xsd:simpleType>
        <xsd:restriction base="sqltypes:nvarchar">
          <xsd:maxLength value="4" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="SourceAirportId" type="sqltypes:int" use="required" />
    <xs:keyref name="Airport" refer=" SourceAirportId ">
        <xs:selector xpath="Airport/AirportId "/>
        <xs:field xpath="."/>
     </xs:keyref>
    <xsd:attribute name="DestinationAirport" use="required">
      <xsd:simpleType>
        <xsd:restriction base="sqltypes:nvarchar">
          <xsd:maxLength value="4" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="DestinationAirportId" type="sqltypes:int" use="required" />
      <xs:keyref name="Airport1" refer=" DestinationAirportId ">
        <xs:selector xpath="Airport/AirportId "/>
        <xs:field xpath="."/>
     </xs:keyref>
    <xsd:attribute name="CodeShare" type="sqltypes:bit" use="required" />
    <xsd:attribute name="Stops" type="sqltypes:int" use="required" />
    <xsd:attribute name="Equipment" type="sqltypes:int" use="required" />
  </xsd:complexType>
```

</xsd:element>

To encode foreign keys and Primary keys in the given schema I have used the following code snippets:

Primary key:

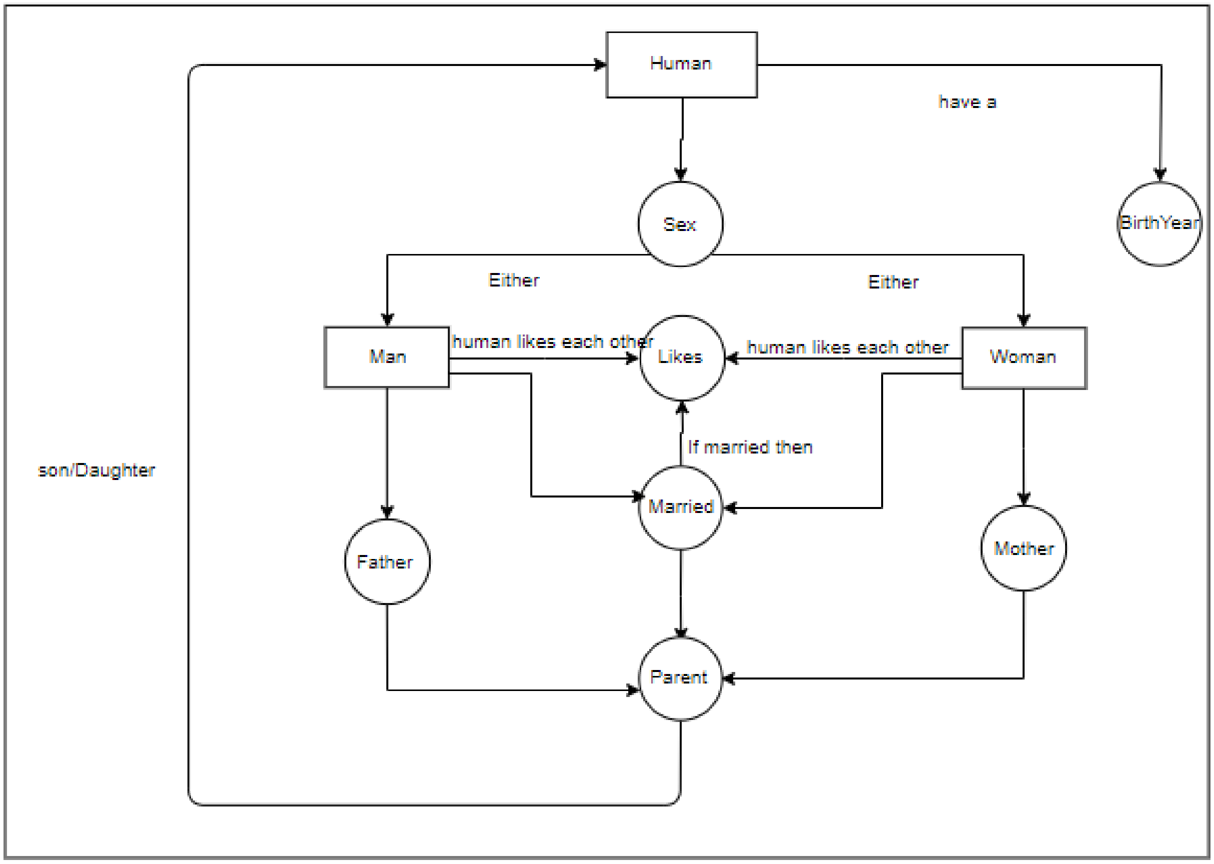
```
<xsd:key name="AirlineId">
   <xs:selector xpath="Airline/AirlineId"/>
   <xs:field xpath="."/>
</xs:key>
```

Foreign Key:

```
<xs:keyref name="Airport1" refer=" DestinationAirportId ">
   <xs:selector xpath="Airport/AirportId "/>
   <xs:field xpath="."/>
</xs:keyref>
```

c) Write a RDF schema and give a graphical presentation to describe these relationships.

Following is the RDF schema of Human.



Question 2:

The following questions test your understanding on basic graph algorithms.

a) Given a directed graph $G(V, E, L)$ with V the node set, E the edge set and L a function that assigns to each edge $e \in E$ a label $L(e)$. A label constrained reachability query $Q(s, t, M)$ tests if there exists a path from a source node s to a target node t, which consists of edges having a label from a label set M. Give an algorithm (pseudo-code) to answer query Q. (hint: A straightforward way is to revise BFS or DFS traversal)

Given:

A directed graph G (V, E, L)

Where

V = Node set
E = Edge set
L = Label of edge *e*, => L(E)

Query Q (S, T, M)

Where

S = Source
T = Target
M = Set of labels

Algorithm:

Breadth-First Search (G, S, t):

1. Start from the source node with initial label O.
   L[S] = O
2. Enqueue S in the queue Q.
   $\therefore$ Enqueue (S).
3. Check: if queue Q is not empty, then:
   i. Dequeue the node V that is on the 1^{st} index at queue = X
   ii. For each vertex *Y* node adjacent to X and which is not previously visited
      - L[Y] = L[X] + 1
      - Prev[Y] = X
      - Enqueue Y
4. Check: whether L[Y] ∈ M
   Then Yes
   Else No

b) Consider a network $G(V, E)$ of servers, where each edge $e = (u, v)$ represents a communication channel from a server u to another server v. Each edge has an associated value $r(u, v)$, which is a constant in [0,1]. The value represents the reliability of the channel, i.e., the probability that the channel from server u to server v will not fail. Assume that these probabilities are independent. Give an algorithm (pseudo-code) to find the most reliable path between two given servers. Give the complexity (in Big O notation) of your algorithm. (hint: transform the weight to non-negative numbers and the problem may become very familiar to you).

Given:

A network G (V, E) of servers

Where

V = Vertices

E = Edges

And each edge (U, V) represents a communication channel from U to V.

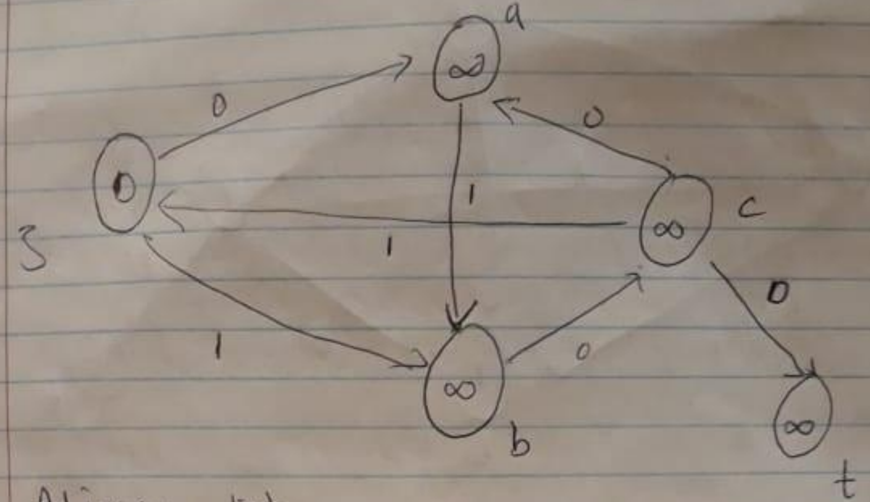
- Each edge in E has a value r.
  r has a constant value [0, 1], which represents the reliability of the channel, i.e. the probability that the channel from U to V will not fail.

Algorithm:

Dijkstra (G, S, r)

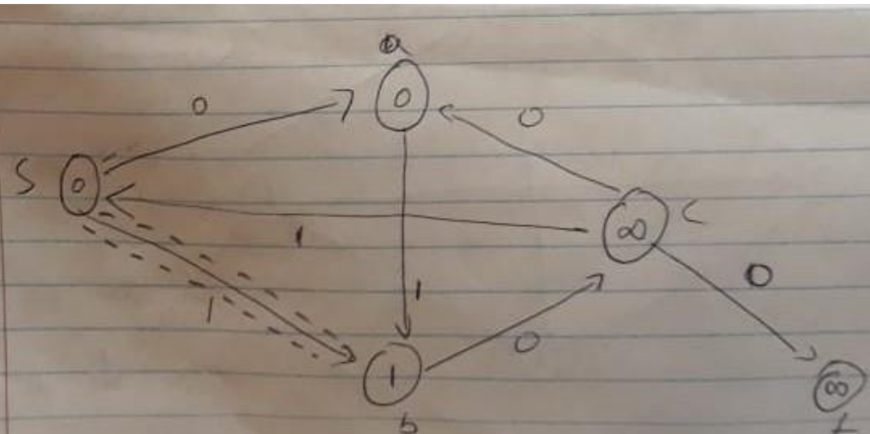
1. Create a set named as shortest path set which will keep the track of vertices that are included in the shortest path. Initially the set is empty.
2. For all nodes *v* in V, do
   d[v] <- ∞
3. Initially, turn the cost of the source node to 0
   d[s] <- 0
   and put the adjacent vertex v in the set.
   Shortest path set <- V
4. While shortest path set is non-empty, do
   a. V <- ExtractMinimum(ShortestPathSet)
   b. For all nodes v in adj(u), do
      i. If d[v] > d[u] + w (u, v), then,
         d[v] <- d[u] + w (u, v)

The above line states that check if the distance of the V node is greater than the new distance, then change the distance with the shortest one.

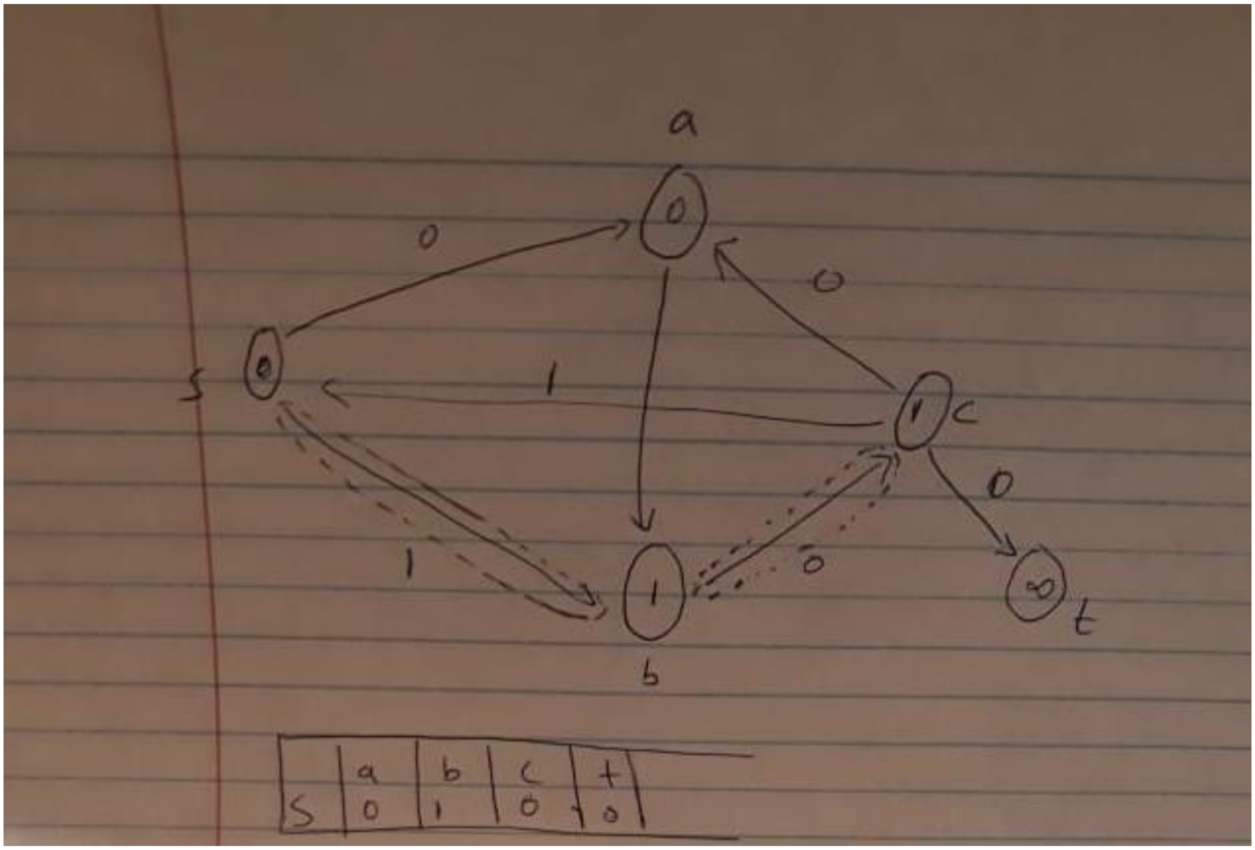


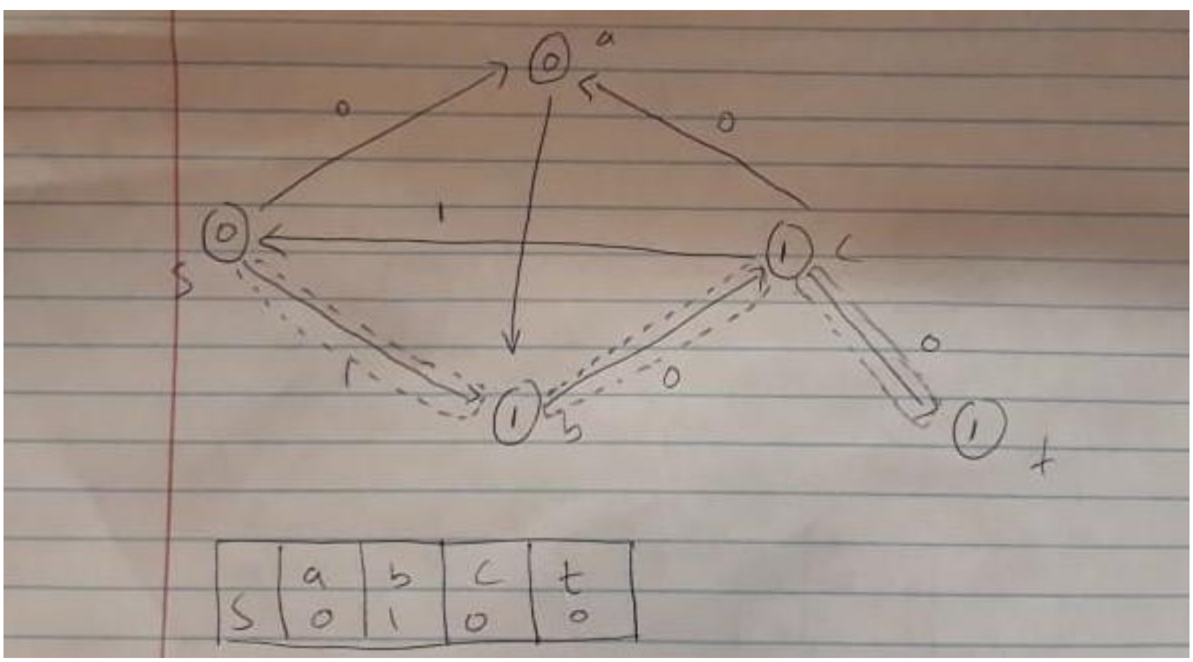
Adjacency list

	a	b	c	t
s	0	1	∞	∞



	a	b	c	t
s	0	1	0	∞





So, the selected shortest path is S -> b -> c -> t with a total probability of 1.

The complexity of this given algorithm is O (|v| + |E|).

Question 3:

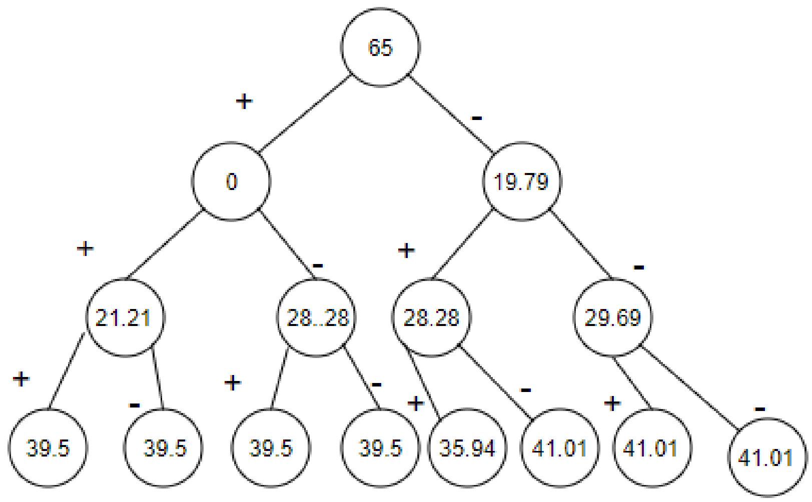
[Approximate query processing]. (25) This question continues our discussion on using data synopsis for query processing based on data-driven approximation. You are given a vector of numbers: [127, 71, 87, 31, 59, 3, 43, 99, 100, 42, 0, 58, 30, 88, 72, 130], each data point records the frequency of communication of a server in a 5-minute interval. For example, in the first 5 minutes, 127 contacts are observed. In the next 5 minutes which is time interval [5, 10], 71 contacts, …

a) Give the Haar decomposition and draw a corresponding error tree for the contacts data vector.

Haar decomposition I the most widely discussed wavelet transformation, which can be constructed in linear time in the size of underlying data array. Following is the haar wavelet decomposition.

Resolution	Averages	Detail coefficient
3	[99, 59, 31, 71, 71, 29, 59, 101]	[39.5, 39.5, 39.5, 39.5, 35.94, 41.01, 41.01, 41.01]
2	[79, 51, 50, 80]	[28.28, 28.28, 29.69, 29.69]
1	[65,65]	[19.79, 21.21]
0	[65]	[0]

So, the haar wavelet decomposition is:
[65, 0, 19.79, 21.21, 28.28, 28.28, 29.69, 29.69, 39.5, 39.5, 39.5, 39.5, 35.94, 41.01, 41.01, 41.01]


65
+
-
0
19.79
+
-
+
-
21.21
28..28
28.28
29.69
+
-
+
-
+
-
+
-
39.5
39.5
39.5
39.5
35.94
41.01
41.01
41.01

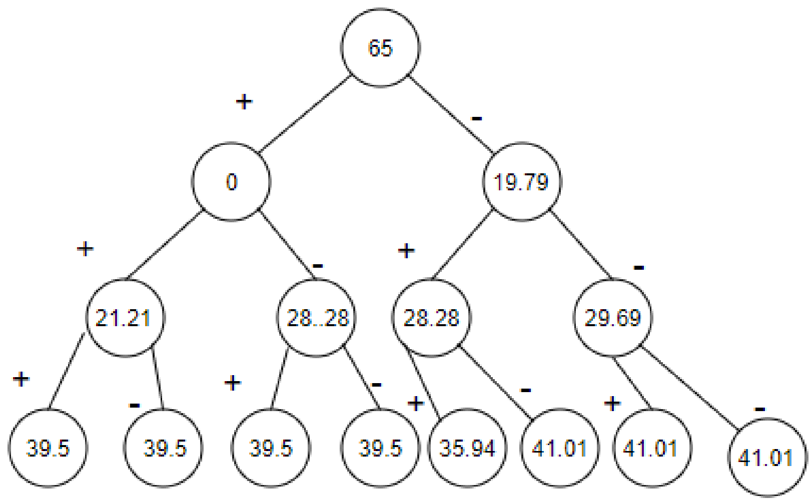
b) Give the process and result for reconstructing the frequency during time interval [15, 20] using Haar decomposition.

To reconstruct the frequency during time interval [15,20] using haar decomposition we have

Resolution	Averages	Detail coefficient
0	59	39.59

c) Use Haar decomposition and error tree to compute the total number of communications between time interval [15, 30].

To calculate the total number of communications between the time interval [15, 30] we have the same error tree, such that



So, from the parent node (I.e. 65) we have two children, 0 is the result of time interval [0, 5] whereas, 19.79 is the result of the time interval [5, 10]. To calculate the calculations between time interval [15,30] we have (65 + 0 + 21.21 + 39.5 – 28.28 + 39.5)

⇨ Around, 136.93 is the total number of communication between the time interval [15, 30].