CptS 415 Big Data: Assignment 2

Md Muhtasim Billah

10/22/2020

Question 1.

1(a)

The XML document to represent the tuples as the fact about the airports should be as follows.

```
<Airports>

   <Airport>
      <AirportID> 507 </AirportID>
      <Name> London Heathrow Airport </Name>
      <City> London </City>
      <Country> United Kingdom </Country>
      <IATA> LHR </IATA>
      <ICAO> EGLL </ICAO>
      <Latitude> 51.4706 </Latitude>
      <Longitude> - 0.461941 </Longitude>
      <Altitude> 83 </Altitude>
   </Airport>

   <Airport>
      <AirportID> 26 </AirportID>
      <Name> Kugaaruk Airport </Name>
      <City> Pelly Bay </City>
      <Country> Canada </Country>
      <IATA> YBB </IATA>
      <ICAO> CYBB </ICAO>
      <Latitude> 68.534401 </Latitude>
      <Longitude> - 89.808098 </Longitude>
      <Altitude> 56 </Altitude>
   </Airport>

   <Airport>
      <AirportID> 3127 </AirportID>
      <Name> Pokhara Airport </Name>
      <City> Pokhara </City>
      <Country> Nepal </Country>
      <IATA> PKR </IATA>
      <ICAO> VNPK </ICAO>
      <Latitude> 28.200899 </Latitude>
      <Longitude> 83.982101 </Longitude>
      <Altitude> 2712 </Altitude>
   </Airport>

   <Airport>
```

```
      <AirportID>  8810 </AirportID>
      <Name> Hamburg Hbf </Name>
      <City> Hamburg </City>
      <Country> Germany </Country>
      <IATA> ZMB </IATA>
      <ICAO> \N </ICAO>
      <Latitude> 53.552776 </Latitude>
      <Longitude> 10.006683 </Longitude>
      <Altitude> 30 </Altitude>
   </Airport>

</Airports>
```

1(b)

The XML schema for the relation `Airport` is as follows.

```
<xs:element name="Airport" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="AirportID" type="xs:positiveInteger"/>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="City" type="xs:string"/>
      <xs:element name="Country" type="xs:string"/>
      <xs:element name="IATA" type="xs:string"/>
      <xs:element name="ICAO" type="xs:string"/>
      <xs:element name="Latitude" type="xs:decimal"/>
      <xs:element name="Longitude" type="xs:decimal"/>
      <xs:element name="Altitude" type="xs:positiveIntegerr"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The XML schema for the relation `Airline` is as follows.

```
<xs:element name="Airline" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="AirlineID" type="xs:positiveInteger"/>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="Alias" type="xs:string"/>
      <xs:element name="IATA" type="xs:string"/>
      <xs:element name="ICAO" type="xs:string"/>
      <xs:element name="Callsign" type="xs:string"/>
      <xs:element name="Country" type="xs:string"/>
      <xs:element name="Active" type="xs:string"/>
        <xs:complexType>
          <xs:choice>
            <xs:element name="Y" type="xs:string"/>
            <xs:element name="N" type="xs:string"/>
          </xs:choice>
```

```
        </xs:complexType>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The XML schema for the relation `Route` is as follows.

```
<xs:element name="Route" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Airline" type="xs:string"/>
      <xs:element name="AirlineID" type="xs:positiveInteger"/>
      <xs:element name="SourceAirport" type="xs:string"/>
      <xs:element name="SourceAirportID" type="xs:positiveInteger"/>
      <xs:element name="DestinationAirport" type="xs:string"/>
      <xs:element name="DestinationAirportID" type="xs:positiveInteger"/>
      <xs:element name="Codeshare" type="xs:string" nillable="true"/>
      <xs:element name="Stops" type="xs:positiveInteger"/>
      <xs:element name="Equipment" type="xs:positiveInteger"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

For XML documents, the purpose of the key and foreign key is served by the attribute(s) of the elements which are given by `ID`, `IDREF` and `IDREFS` in the data type definition (DTD) of the XML schema (XSD). Since these attribute values are unique for each element, refencing is easy.

For example, for the XML schema of the Airline dataset, the `AirlineID` can be used as the attribute (ID) for each `Airline` element. To connect one airline (thus, an element) with another (or multiple) airline their corresponding `AirlineID` can be considered as the attribute values (IDREF or IDREFS) to be referred.

1(c)

The RDF schema for the provided natural language sentences is as follows.

```
#classes and subclasses

<rdfs:Class rdfs:about="Human">
    <rdfs:comment>
        The class of Human. This class indicates a human being.
        Superclass of all Man and Human classes.
    </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdfs:about="AnotherHuman">
    <rdfs:comment>
        The class of AnotherHuman.
        This class indicates a different human being.
        This might also have sub classes Man and Woman (not cosidered here).
    </rdfs:comment>
</rdfs:Class>
```

```
<rdfs:Class rdfs:about="Man">
    <rdfs:comment>
        The class of Man. Subclass of Human.
    </rdfs:comment>
    <rdfs:subClassOf rdfs:resource="Human"/>
</rdfs:Class>

<rdfs:Class rdfs:about="Woman">
    <rdfs:comment>
        The class of Woman. Subclass of Human.
    </rdfs:comment>
    <rdfs:subClassOf rdfs:resource="Human"/>
</rdfs:Class>

<rdfs:Class rdfs:about="Literals">
    <rdfs:comment>
        The class of Literals.
        Holds information of birth year for Human class.
    </rdfs:comment>
</rdfs:Class>

#properties and subproperties

<rdfs:Property rdfs:about="canBe">
    <rdfs:domain rdfs:resource="Human" />
    <rdfs:range rdfs:resource="Man" />
    <rdfs:range rdfs:resource="Woman" />
</rdfs:Property>

<rdfs:Property rdfs:about="canMarry">
    <rdfs:domain rdfs:resource="Human" />
    <rdfs:range rdfs:resource="AnotherHuman" />
</rdfs:Property>

<rdfs:Property rdfs:about="canLike">
    <rdfs:comment>
        Inherits its domain ("Human") and range ("AnotherHuman")
        from its superproperty "canMarry".
    </rdfs:comment>
    <rdfs:domain rdfs:resource="Human" />
    <rdfs:range rdfs:resource="AnotherHuman" />
    <rdfs:subPropertyOf rdfs:resource="canMarry"/>
</rdfs:Property>

<rdfs:Property rdfs:about="isParentOf">
    <rdfs:domain rdfs:resource="Human" />
    <rdfs:range rdfs:resource="AnotherHuman" />
</rdfs:Property>

<rdfs:Property rdfs:about="canFather">
    <rdfs:domain rdfs:resource="Man" />
    <rdfs:range rdfs:resource="AnotherHuman" />
    <rdfs:subPropertyOf rdfs:resource="isParentOf"/>
```

```
</rdfs:Property>

<rdfs:Property rdfs:about="canMother">
    <rdfs:comment>
        Inherits its domain ("Woman") and range ("AnotherHuman")
        from its superproperty "isParentOf".
    </rdfs:comment>
    <rdfs:domain rdfs:resource="Woman" />
    <rdfs:range rdfs:resource="AnotherHuman" />
    <rdfs:subPropertyOf rdfs:resource="isParentOf"/>
</rdfs:Property>

<rdfs:Property rdfs:about="BirthYear">
    <rdfs:domain rdfs:resource="Human" />
    <rdfs:range rdfs:resource="Literals" />
</rdfs:Property>
```

The graphical presentation describing the schema is as follows.

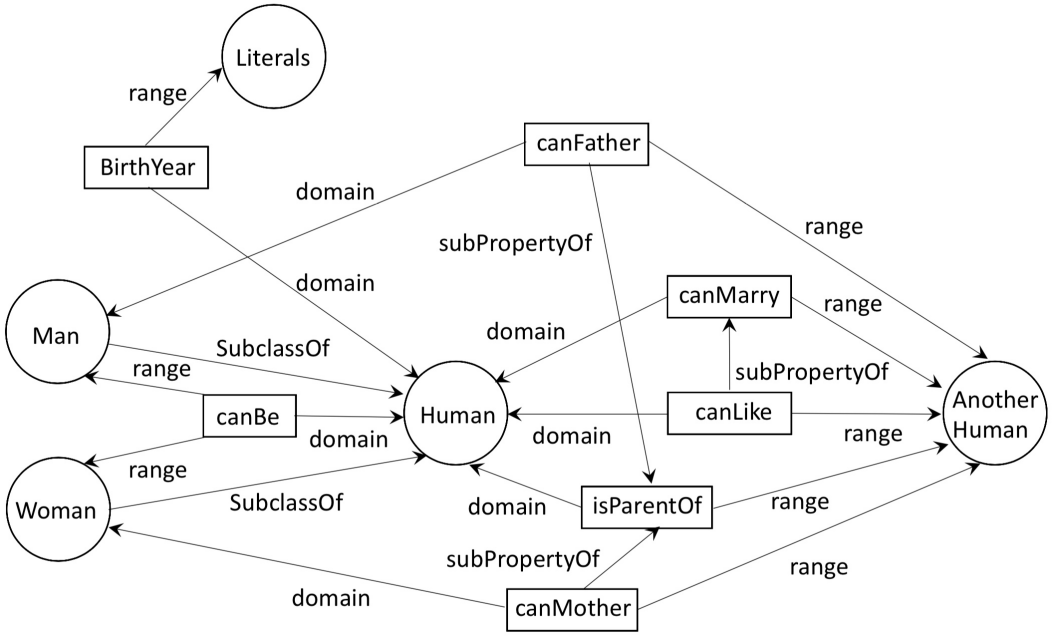


Fig.: The graphical presentation of the RDF schema.

Question 2.

2(a)

The pseudo-code to answer a label constrained reachability query $Q(s, t, M)$ to test 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 is provided below (based on BFS traversal).

```
function Query(s,t,M): # s is source, t is target, G is the graph
  Let Q be a queue
  Q.enqueue(s)
  Let L(e) be the label for the edge e(s,t)
  # the label L(e) is the label for the edge connecting s and t

  While Q is not empty and L(e) is a subset of M:
    v = Q.dequeue()
    If v is the target, return TRUE
    For all edges from v to w in G.adjacentEdges(v) do
      If w is not labelled as discovered:
        Label w as discovered
        w.parent = v
        Q.enqueue(w)
  Return FALSE
```

2(b)

To find the most reliable path between two given servers, we have to find an edge, $e = (u, v)$ that has the maximum value of $r(u, v)$ associated with it. An algorithm for such a query can be derived from the Dijkstra's algorithm for finding the shortest distance between two nodes. But, one modification which is required is setting all the inital weights to non-negative numbers ranging from 0 to 1. The alogorithm can be described as follows.

```
function Dijkstra(Graph, source):
    for each vertex v in Graph:      # Initialization
      dist[v] := values in range[0,1]   # Weights set as non-negative numbers
      previous[v] := undefined  # Previous node in optimal path from source

    dist[source] := 0   # Distance from source to source
    Q := the set of all nodes in Graph  # all nodes in the graph are kept in Q

    while Q is not empty:   # main loop
      u := node in Q with smallest dist[ ]
      remove u from Q
      for each neighbor v of u: # where v has not yet been removed from Q.
        alt := dist[u] + dist_between(u, v)
        if alt < dist[v]
          dist[v] := alt
          previous[v] := u

    return previous[ ]
```

Since, all nodes in the graphs are kept in a list, the complexity of this algorithm will be $O(|E| + |V|^2)$ where V is the number of nodes/vertices and E is the number of edges.

Question 3.

3(a)

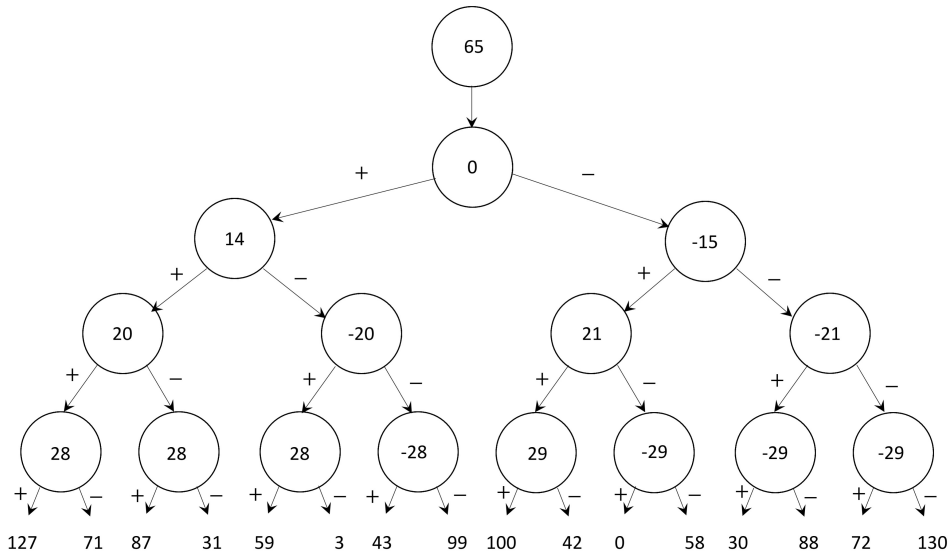
The provided vector of numbers: [127, 71, 87, 31, 59, 3, 43, 99, 100, 42, 0, 58, 30, 88, 72, 130], represents the frequency (total number of communications) of a server in a 5-minute interval. The Haar decomposition and its corresponding error tree for the data vector is as below.

Resolution	Averages (Frequency of communications)	Detailed Coefficients
4	[127, 71, 87, 31, 59, 3, 43, 99, 100, 42, 0, 58, 30, 88, 72, 130]	----------
3	[99, 59, 31, 71, 71, 29, 59, 101]	[28, 28, 28, -28, 29, -29, -29, -29]
2	[79, 51, 50, 80]	[20, -20, 21, -21]
1	[65, 65]	[14, -15]
0	[65]	[0]

Thus, the Haar wavelet decomposition becomes,

$$[65, 0, 14, -15, 20, -20, 21, -21, 28, 28, 28, -28, 29, -29, -29, -29]$$

The error tree diagram is as below.



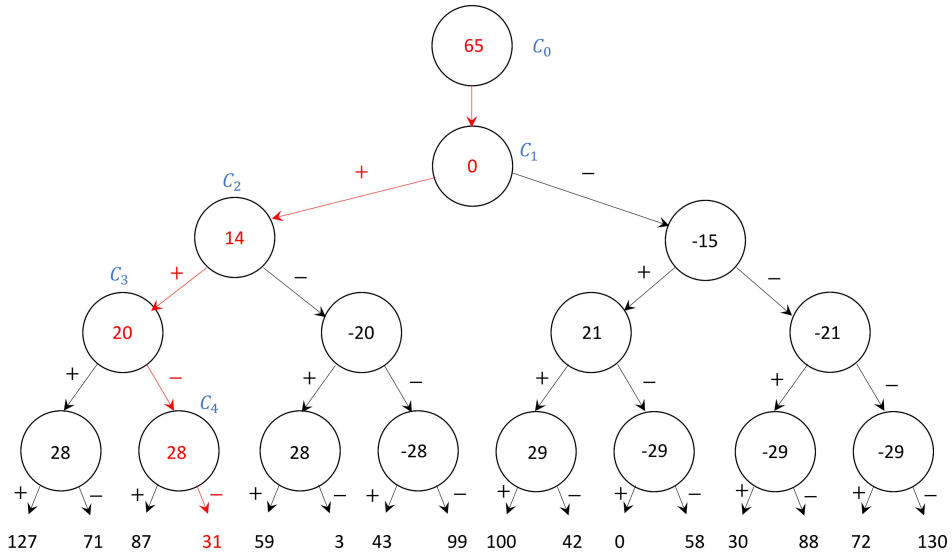
3(b)

The process and result for reconstructing the frequency during time interval [15, 20] is provided using Haar decomposition (the path is explained in a top-down fashion).

The frequency (number of contacts) during the time inteval [15,20] can be found using the following formula and the error tree for the Haar wavelet decomposition.

$$A[i] = \sum_{C_j \in path\ of\ A[i]} sign_{i,j}\ C_j$$

The path to the time interval [15,20] is shown in red below. Where, C_0, C_1, C_2, C_3 and C_4 are the coefficients.

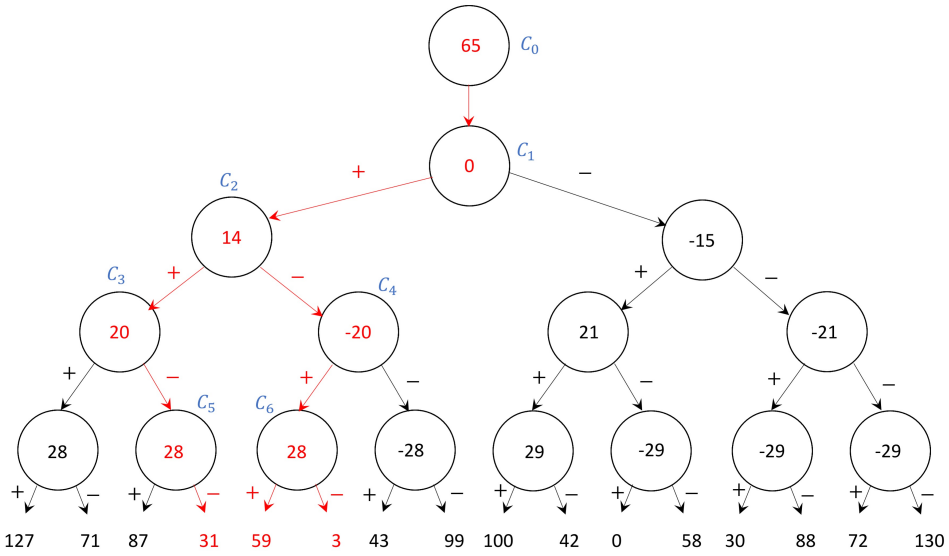


Using the path shown above and the formula, the number of contacts for the time interval [15,20] can be found as below.

$$A_{[15,20]} = C_0 + C_1 + C_2 - C_3 - C_4$$
$$A_{[15,20]} = 65 + 0 + 14 - 20 - 28$$
$$A_{[15,20]} = 31$$

3(c)

Explaining the path in a top-down fashion, the total number of communications between time interval [15, 30] is computed below using the Haar decomposition and corresponding error tree.



Using the path shown above and the formula, the number of contacts for the time interval [15,30] can be found as below.

Number of contacts for the time interval [15,20] (from 3(b)),

$$A_{[15,20]} = 31$$

Number of contacts for the time interval [20,25],

$$A_{[20,25]} = C_0 + C_1 - C_2 + C_4 + C_6$$
$$A_{[20,25]} = 65 + 0 - 14 + (-20) + 28$$
$$A_{[20,25]} = 59$$

Number of contacts for the time interval [25,30],

$$A_{[25,30]} = C_0 + C_1 - C_2 + C_4 - C_6$$
$$A_{[25,30]} = 65 + 0 - 14 + (-20) - 28$$
$$A_{[25,30]} = 3$$

Thus, total number of contacts during the interval [15,30],

$$A_{[15,30]} = A_{[15,20]} + A_{[20,25]} + A_{[25,30]} = 31 + 59 + 3 = 93$$