CptS 415 Big Data
Assignment 2
Jinyang Ruan
10/24/2021

1.
    a. The instances I gave in assignment 1 are 4 the airports with the ID 1 to 4.


```
<Airports>
  <Airport>
    <Airport id = “1”>
    <Name> Goroka Airport </Name>
    <City> Goroka </City>
    <Country> Papua New Guinea </Country>
    <IATA> GKA </IATA>
    <ICAO> AYGA </ICAO>
    <Latitude> -6.081689834590001</Latitude>
    <Longitude> 145.391998291 </Longitude>
    <Altitude> 5282 </Altitude>
    <Timezone> 10 </Timezone>
    <DST> U </DST>
    <Tz Database time zone> Pacific/Port Moresby </Tz Database time zone>
    <Type> airport </Type>
    <Source> OurAirports </Source>
  </Airport>

  <Airport>
    <Airport id = “2”>
    <Name> Madang Airport </Name>
    <City> Madang </City>
    <Country> Papua New Guinea </Country>
    <IATA> MAG </IATA>
    <ICAO> AYMD </ICAO>
    <Latitude> -5.20707988739 </Latitude>
    <Longitude> 145.789001465 </Longitude>
    <Altitude> 20 </Altitude>
    <Timezone> 10 </Timezone>
    <DST> U </DST>
    <Tz Database time zone> Pacific/Port Moresby </Tz Database time zone>
    <Type> airport </Type>
    <Source> OurAirports </Source>
  </Airport>

  <Airport>
    <Airport id = “3”>
    <Name> Mount Hagen Kagamuga Airport </Name>
    <City> Mount Hagen </City>
```

```
        <Country> Papua New Guinea </Country>
        <IATA> HGU </IATA>
        <ICAO> AYMH </ICAO>
        <Latitude> -5.826789855957031 </Latitude>
        <Longitude> 144.29600524902344 </Longitude>
        <Altitude> 5388 </Altitude>
        <Timezone> 10 </Timezone>
        <DST> U </DST>
        <Tz Database time zone> Pacific/Port Moresby </Tz Database time zone>
        <Type> airport </Type>
        <Source> OurAirports </Source>
    </Airport>

    <Airport>
        <Airport id = “4”>
        <Name> Nadzab Airport </Name>
        <City> Nadzab Hagen </City>
        <Country> Papua New Guinea </Country>
        <IATA> LAE </IATA>
        <ICAO> AYNZ </ICAO>
        <Latitude> -6.569803 </Latitude>
        <Longitude> 146.725977 </Longitude>
        <Altitude> 239 </Altitude>
        <Timezone> 10 </Timezone>
        <DST> U </DST>
        <Tz Database time zone> Pacific/Port Moresby </Tz Database time zone>
        <Type> airport </Type>
        <Source> OurAirports </Source>
    </Airport>
</Airports>
```

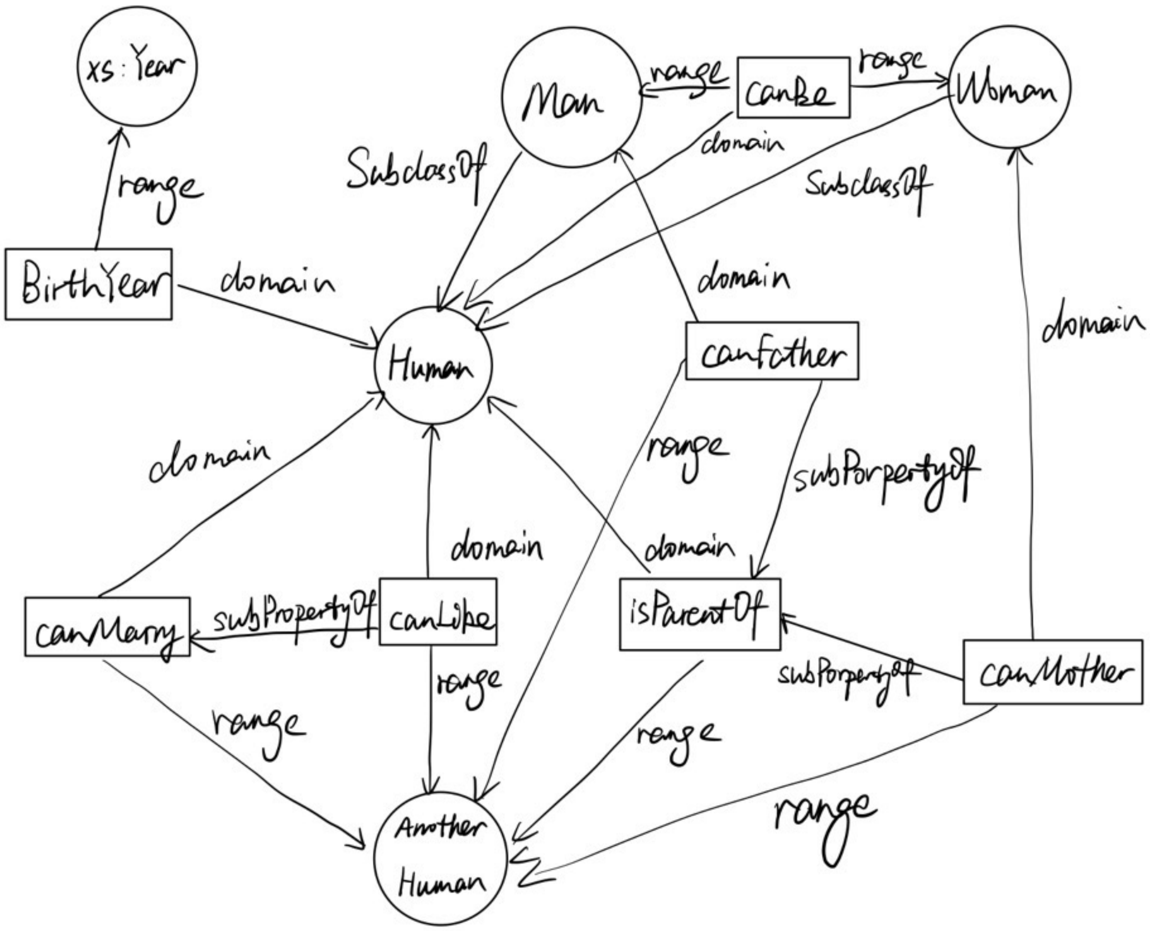
b. The RDF schema is shown as following:

```
#classes
<rdfs:Class rdfs:about="Human">
    <rdfs:comment>
            The class of Human.
    </rdfs:comment>
</rdfs:Class>

<rdfs:Class rdfs:about="AnotherHuman">
    <rdfs:comment>
            The class of another human.
            This class indicates a different person.
    </rdfs:comment>
</rdfs:Class>
```

```
<rdfs:Class rdfs:about="Man">
    <rdfs:comment>
            The class of Man.
    </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="xs:Year">
    <rdfs:comment>
            This class holds information of birth year for Human class.
    </rdfs:comment>
</rdfs:Class>

#properties
<rdfs:Property rdfs:about="canBe">
    <rdfs:comment>
            A human can have a sex property of a man or a woman.
    </rdfs:comment>
    <rdfs:domain rdfs:resource="Human" />
    <rdfs:range rdfs:resource="Man" />
    <rdfs:range rdfs:resource="Woman" />
</rdfs:Property>

<rdfs:Property rdfs:about="canFather">
    <rdfs:comment>
            A man can be the father of another human.
    </rdfs:comment>
    <rdfs:domain rdfs:resource="Man" />
    <rdfs:range rdfs:resource="AnotherHuman" />
    <rdfs:subPropertyOf rdfs:resource="isParentOf"/>
</rdfs:Property>

<rdfs:Property rdfs:about="canMother">
    <rdfs:comment>
            A woman can be the mother of another human.
    </rdfs:comment>
    <rdfs:domain rdfs:resource="Woman" />
    <rdfs:range rdfs:resource="AnotherHuman" />
    <rdfs:subPropertyOf rdfs:resource="isParentOf"/>
```

```
</rdfs:Property>

<rdfs:Property rdfs:about="isParentOf">
    <rdfs:comment>
            If a human is a mother or father, the human is a parent.
    </rdfs:comment>
    <rdfs:domain rdfs:resource="Human" />
    <rdfs:range rdfs:resource="AnotherHuman" />
</rdfs:Property>

<rdfs:Property rdfs:about="canLike">
    <rdfs:comment>
            If a human is married to another, then they like each other.
    </rdfs:comment>
    <rdfs:domain rdfs:resource="Human" />
    <rdfs:range rdfs:resource="AnotherHuman" />
    <rdfs:subPropertyOf rdfs:resource="canMarry"/>
</rdfs:Property>

<rdfs:Property rdfs:about="canMarry">
    <rdfs:comment>
            A human can be married to another human.
    </rdfs:comment>
    <rdfs:domain rdfs:resource="Human" />
    <rdfs:range rdfs:resource="AnotherHuman" />
</rdfs:Property>

<rdfs:Property rdfs:about="BirthYear">
    <rdfs:comment>
            A human can have a BirthYear property of type“xs:Year”.
    </rdfs:comment>
    <rdfs:domain rdfs:resource="Human" />
    <rdfs:range rdfs:resource="xs:Year" />
</rdfs:Property>
```

Graphical presentation is shown on the next page:


xs:Year
range
BirthYear
domain
Man
range
canBe
range
Woman
domain
SubclassOf
SubclassOf
Human
domain
canFather
domain
range
subPorpertyOf
domain
domain
domain
canMarry
subPropertyOf
canLike
isParentOf
subPorpertyOf
canMother
range
range
range
range
Another Human

2.

a. Function: $Query(s,t,G)$ // where s is source node, t is target node, G is the Graph
    Let Q be a queue
        $Q.enqueue(s)$
    Let $L(e)$ be the label of the edge $e(s,t)$

While Q is not empty and $L(e)$ is a subset of M:
    $v = Q.dequeue()$
    If v is the target node
    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

b. We use Dijkstra algorithm to find the shortest path in a graph. In this case, in order to find the most reliable path between two servers, we can transfer the associated value r to $1/r$, then we use Dijkstra algorithm to find the shortest path with the weight of the edges is $w = 1/r$.

Function $Dijkstra(Graph, source, w)$:
    for each vertex v in Graph: // Initialization
        $dist[v] = \infty$ //Transform weights as non-negative //numbers
        $prev[v] = undefined$
    $dist[source] = 0$
    $Q = the\ set\ of\ all\ nodes\ in\ Graph$
$Q.add(v)$

While Q is not empty:
    $u = node\ in\ Q\ with\ minimun\ dist[u]$
    remove u from Q
    For each neighbor v of u:
    Update the distance of each neighbor v to u (if it is smaller)
    $dist[v] = dist[u] + w(u,v)$
    Return $prev[v]$

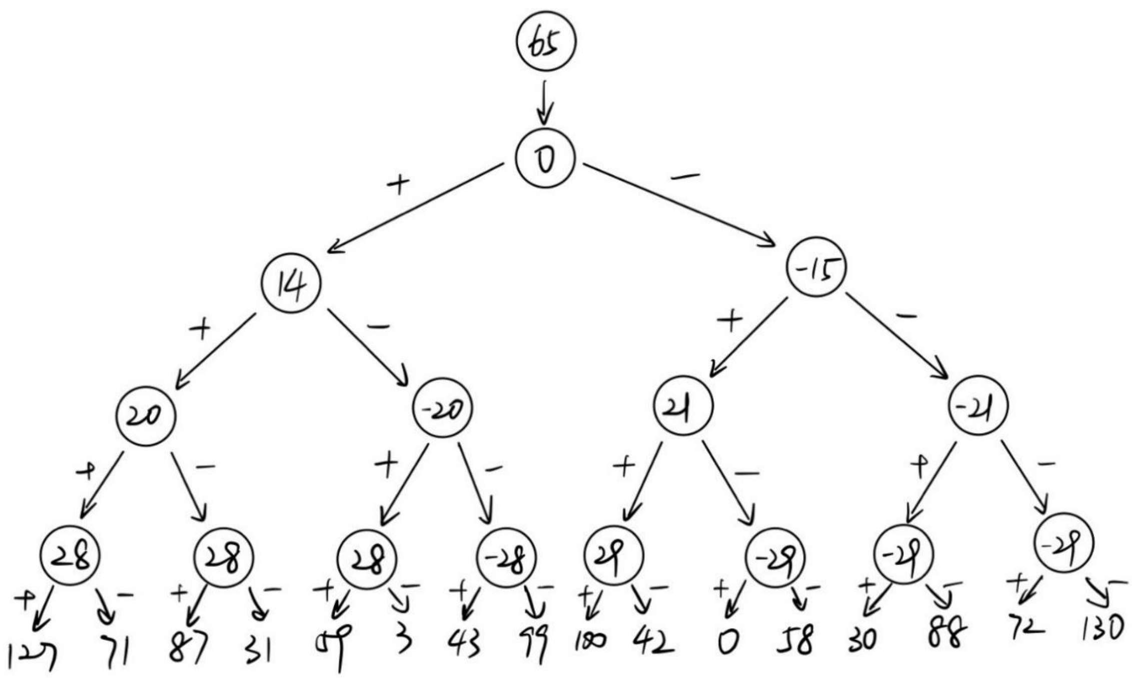
Thus, we can the shortest path with the weight $1/r$ which means the most reliable path between to servers. complexity of the algorithm is $O(|E| + |V|^2)$ where E is the number of edges and V is the number of nodes.

3.

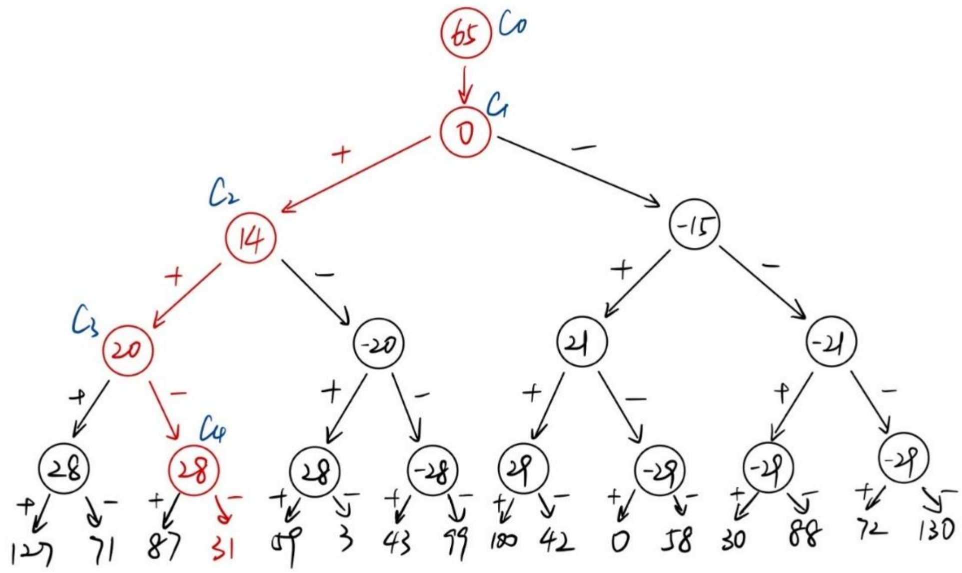
a.

Resolution	Averages	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:



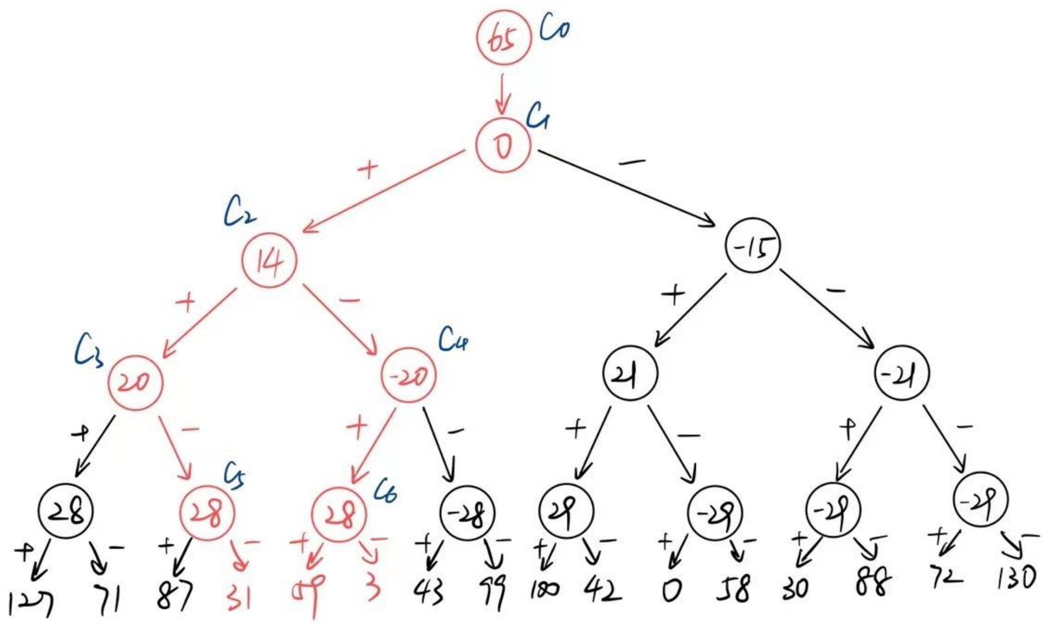
b. The path to the time interval [15, 20] is shown as in red below, where C_0, C_1, C_2, C_3, and C_4 are the coefficients.



The top-down path is down—left—left—right—right.

$$A_{[15,20]} = 65 + 0 + 14 - 20 - 28 = 31$$

c. The path to the time interval [15, 30] is shown as in red below, where C_0, C_1, C_2, C_3, C_4, C_5, and C_6 are the coefficients.



$$A_{[15,30]} = (65 + 0 + 14 - 20 - 28) + (65 + 0 - 14 - 20 + 28) + (65 + 0 - 14 - 20 - 28]$$
$$= 31 + 59 + 3 = 93$$