Rusu Wu
CptS 415
Assignment 2
10/26/2021

1.

a)

Follow is the four airport instances showing with XML document format.

```
<Airports>
    <Airport>
        <Airport id = “3467”>
        <Name> Spokane International Airport </Name>
        <City> Spokane </City>
        <Country> US </Country>
        <IATA> GEG </IATA>
        <ICAO> KGEG </ICAO>
        <Latitude> 47.61989974975586</Latitude>
        <Longitude> -117.53399658203125 </Longitude>
        <Altitude> 2376 </Altitude>
        <Timezone> -8 </Timezone>
        <DST> US/Canada </DST>
        <Tz Database time zone> Unknow</Tz Database time zone>
        <Type> Airport </Type>
        <Source> OpenFlight </Source>
    </Airport>

        <Airport>
        <Airport id = “3370”>
        <Name> Guangzhou Baiyun International Airport</Name>
        <City> Guang zhou </City>
        <Country> CHN</Country>
        <IATA> ZGGG </IATA>
        <ICAO> ZGGG </ICAO>
        <Latitude> 23.39240074157715</Latitude>
        <Longitude> 113.29900360107422 </Longitude>
        <Altitude> 50</Altitude>
        <Timezone>8 </Timezone>
        <DST> Unknow</DST>
        <Tz Database time zone> Unknow</Tz Database time zone>
        <Type> Airport </Type>
        <Source> OpenFlight </Source>
    </Airport>
```

```
        <Airport>
            <Airport id = “3577”>
            <Name> Seattle Tacoma International Airport </Name>
            <City> Seattle </City>
            <Country> US </Country>
            <IATA> SEA </IATA>
            <ICAO> KSEA</ICAO>
            <Latitude> 47.449 001</Latitude>
            <Longitude> -122.308998 </Longitude>
            <Altitude> 433 </Altitude>
            <Timezone> -8 </Timezone>
            <DST> US/Canada </DST>
            <Tz Database time zone> Unknow</Tz Database time zone>
            <Type> Airport </Type>
            <Source> OpenFlight </Source>
        </Airport>

        <Airport>
            <Airport id = “3484”>
            <Name> Los Angeles International Airport </Name>
            <City> Los Angele </City>
            <Country> US </Country>
            <IATA> LAX </IATA>
            <ICAO> KLAX </ICAO>
            <Latitude> 33.942 50107</Latitude>
            <Longitude> -118.4079971</Longitude>
            <Altitude> 125 </Altitude>
            <Timezone> -8 </Timezone>
            <DST> US/Canada </DST>
            <Tz Database time zone> Unknow</Tz Database time zone>
            <Type> Airport </Type>
            <Source> OpenFlight </Source>
        </Airport>
    <Airports>
```

b) RDF schema is shown as below.

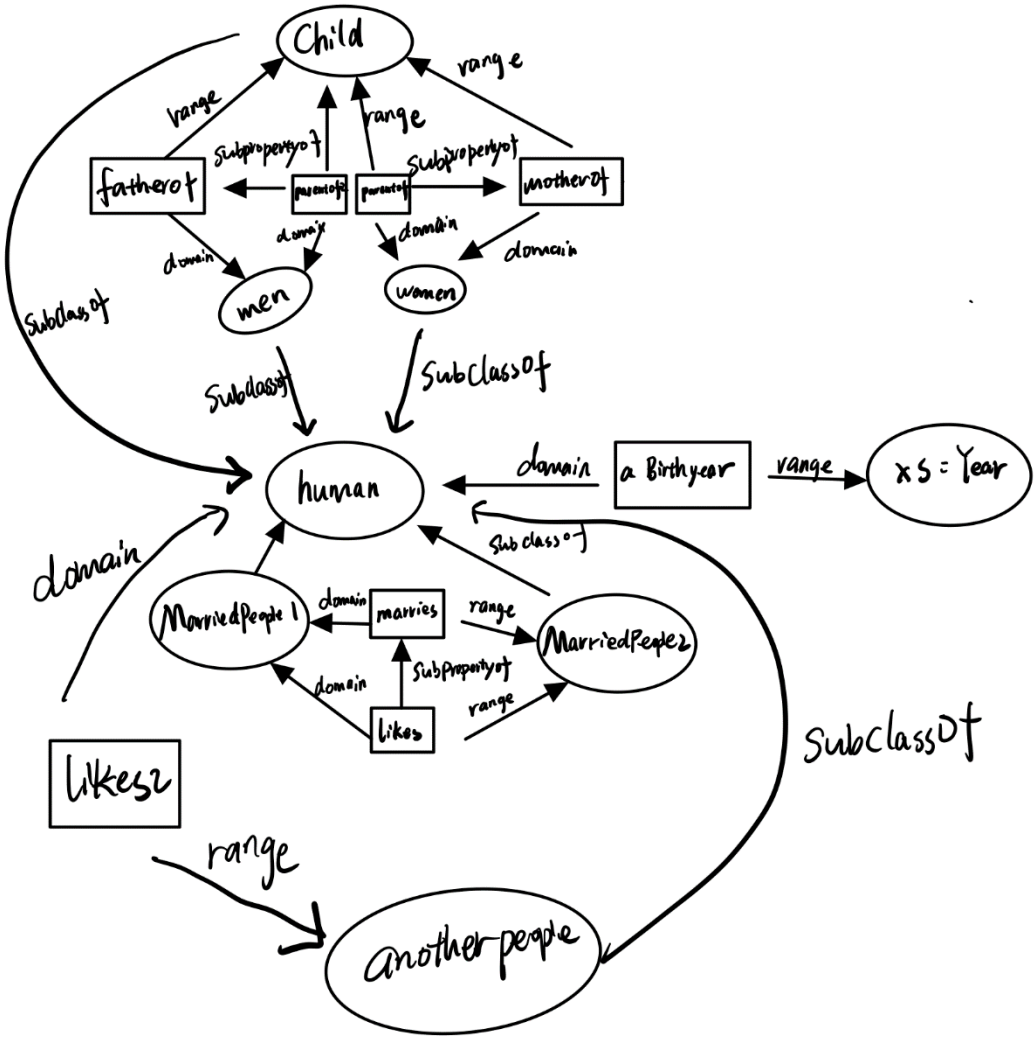
```
<rdfs:Class rdf:about="human" />
<rdfs:Class rdf:about="man">
    <rdfs:comment>
        A human can have a sex property of a man.
    <rdfs:subClassOf rdf:resource="human"/>
</rdfs:Class>
```

```
<rdfs:Class rdf:about="woman">
    <rdfs:comment>
        A human can have a sex property of a woman.
    </rdfs:comment>
    <rdfs:subClassOf rdf:resource="human"/>
</rdfs:Class>
</rdfs:Class>
<rdfs:Class rdf:about="Child ">
    <rdfs:comment>
    A human can be the father or mother of another human.
    </rdfs:comment>
    <rdfs:subClassOf rdf:resource="human "/>
</rdfs:Class>
</rdfs:Class>
<rdfs:Class rdf:about="xs:year">
    <rdfs:comment>
    A human can have a BirthYear property of type“xs:Year”..
    </rdfs:comment>
</rdfs:Class>
</rdfs:Class>
<rdfs:Class rdf:about="MarriedPeople1 ">
    <rdfs:comment>
    A human can be married to another human
    </rdfs:comment>
    <rdfs:subClassOf rdf:resource="human "/>
</rdfs:Class>
<rdfs:Class rdf:about="MarriedPeople2 ">
    <rdfs:comment>
    A human can be married to another human
    </rdfs:comment>
    <rdfs:subClassOf rdf:resource="human "/>
</rdfs:Class>
<rdfs:Class rdf:about="anotherpeople ">
    <rdfs:comment>
    A human can like another human
    </rdfs:comment>
    <rdfs:subClassOf rdf:resource="human "/>
</rdfs:Class>



<rdf:Property rdf:about="a Birthyear ">
    <rdf:domain rdf:resource="human" />
    <rdf:range rdf:resource="xs:year" />
```

```
</rdf:Property>
<rdf:Property rdf:about="marries ">
      <rdf:domain rdf:resource="MarriePeople1 " />
      <rdf:range rdf:resource="MarriedPeople2 " />
</rdf:Property>
<rdf:Property rdf:about="likes">
      <rdfs:comment>
      If a human is married to another, then they like eachother.
      </rdfs:comment>
      <rdfs:subPropertyOf rdf:resource="marries"/>
</rdf:Property>
<rdf:Property rdf:about="likes2 ">
      <rdf:domain rdf:resource="human " />
      <rdf:range rdf:resource="anotherpeople " />
      <rdfs:comment>
      A human can like another human.
      </rdfs:comment>
</rdf:Property>
<rdf:Property rdf:about="fatherof ">
      <rdf:domain rdf:resource="man " />
      <rdf:range rdf:resource="child " />
      <rdfs:comment>
      A man can be the father of anotherhuman.
      </rdfs:comment>
</rdf:Property>
  <rdf:Property rdf:about="motherof ">
      <rdf:domain rdf:resource="women " />
      <rdf:range rdf:resource="child " />
      <rdfs:comment>
      A man can be the mother of anotherhuman.
      </rdfs:comment>
</rdf:Property>
</rdf:Property>
  <rdf:Property rdf:about="parentsof ">
      <rdf:domain rdf:resource="women " />
      <rdf:range rdf:resource="child " />
      <rdfs:comment>
      A man can be the mother of anotherhuman.
      </rdfs:comment>
      <rdfs:subPropertyOf rdf:resource="motherof"/>
</rdf:Property>
<rdf:Property rdf:about="parentsof2 ">
      <rdf:domain rdf:resource="women " />
      <rdf:range rdf:resource="child " />
```

<rdfs:comment>
A man can be the mother of anotherhuman.
</rdfs:comment>
<rdfs:subPropertyOf rdf:resource="fatherof"/>
</rdf:Property>

Graphical presentation:



2.

a)
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) Dijkstra algorithm is to find the shortest path

let 1/r to be the unreliability rate of a path, so we find the path with smallest 1/r path. Such that the path has the largest reliability.

Create vertex set Q

For each vertex $v \in V$

d$ist[v]$= ∞

$prev[v] = undefined$

Q.add(v)

$dist[s] = 0$

While Q is not empty:

u = vertex in Q with minimum unreliability distance $dist[u]$

Remove u from Q

Update the distance of each neighbor v of u to (if it is smaller) $dist[v] = dist[u]+w(u,v)$

Return $prev[v]$

Complexity: If Q is a list/array, $O(|E| + |V|^2)$

Complexity with the right data structure (min-priority queue):

$O(|E| + |V| \log |V|)$

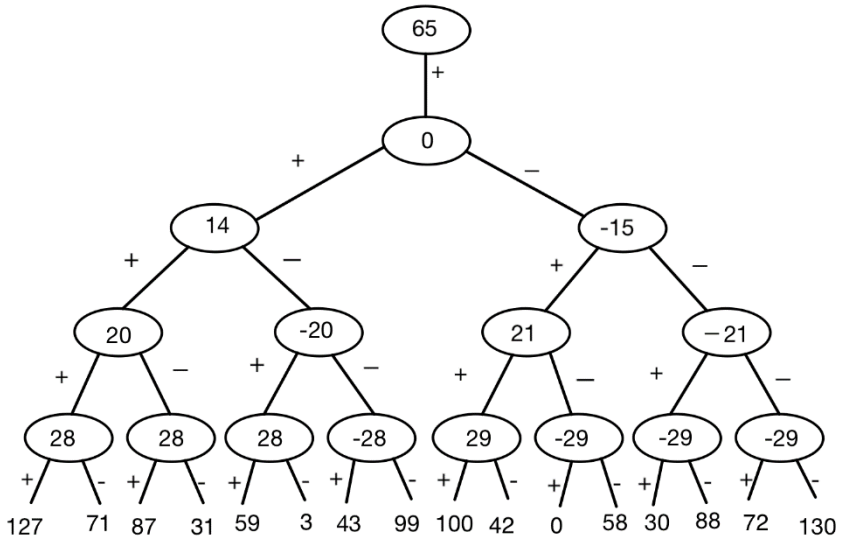
3.

a)

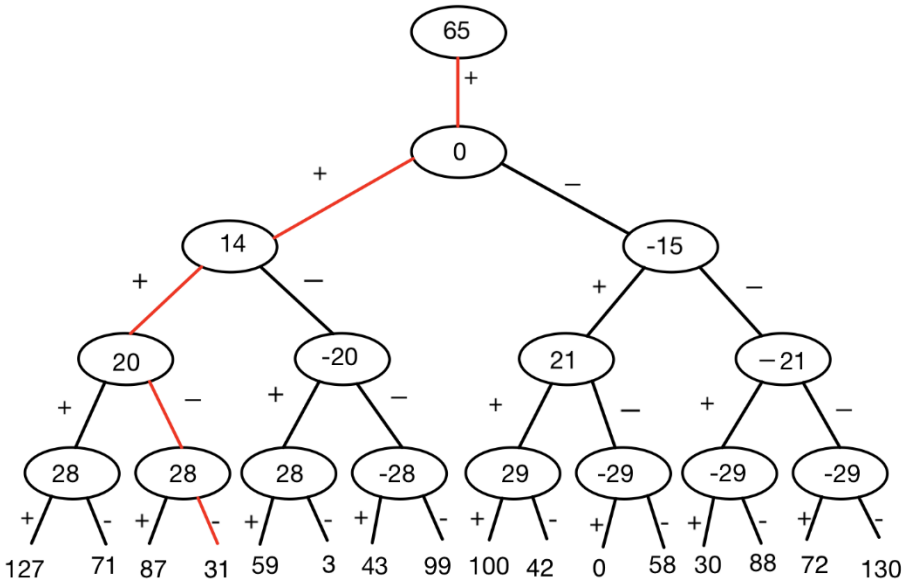
Resolution	Averages	Detail Coefficient
	[127, 71, 87, 31, 59, 3, 43, 99, 100, 42, 0, 58, 30, 88, 72, 130]	----
	[99,59,31,71,71,29,59,101]	[28, 28, 28, -28, 29, -29, -29, -29]
	[79,51,50,80]	[20, -20, 21, -21]
	[65,65]	[14, -15]
	[65]	[0]

Haar wavelets: [65, 0, 14, -15, 20, -20, 21, -21, 28, 28, 28, -28, 29, -29, -29, -29]

Corresponding error tree:

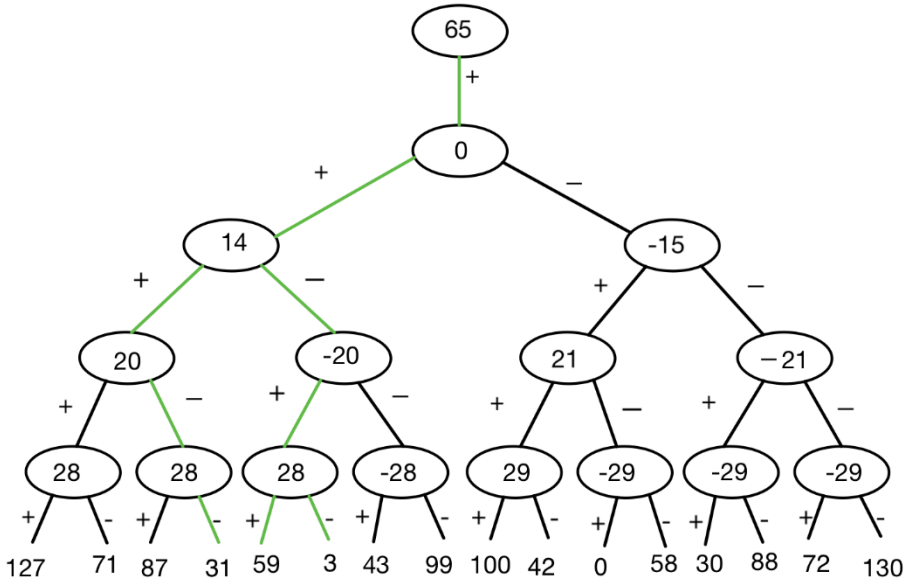


b) The path of interval [15, 20] in a top-down fashion is shown as the red path.



During time interval [15, 20], A[15,20]=65+0+14-20-28=31

c) The path of interval [15, 30] in a top-down fashion is shown as the green path.



A[15,30]=[65+0+14-20-28]+[65+0-14+(-20)+28]+ [65+0-14+(-20)-28]=31+59+3=93