Low\_example (No necessary description, no algorithm procedure and analysis)

CS 415

Assignment 2

1.

A. A block nested loop is an algorithm for relational databases that is used to join 2 relations. Instead of one scan of S for every tuple in R like the traditional nested loop join, this algorithm only scans S once for every group of R tuples. The algorithm reduces the number of scans of S usually.

Cost = N + M \* [ N/(B-2] = 4,200

B. A sort-merge join is an algorithm for relational databases that involves sorting the relations by the join attribute first. This way, linear scan will get to the sets at the same time. The goal is to find the set of tuples in each relation for each distinct value of the join attribute.

Cost = 3\* ( M+N) = 3,600

C. A hash join algorithm is also an algorithm for relational databases that can only be used for equi join. To find the set of tuples in each relation which have the distinct value of the join attribute, the hash joins requires a predicate that compares the values from one table to the other using the equals operator. This algorithm is efficient for larger result sets.

Cost = 3 \* (M+N) = 3600

2.

Q = newQueue();

Q.enqueue(x);

While(Q != empty){

current=q.dequeue();

if( visitied fo the current is equal to false)

set current.visited to true

for each edge current, next

L(e)

If next visited is set to false,

Q.enqueue(next

Q.enqueue(t);

while( Q != empty)

{current = Q.dequeue();

if ( current.visited is set to false )

{current.visited is true;

for ( each edge (current, next))

if( e.label is in M)

if( next is equal to t )

return true

else

Q.enqueue(next)

}

}

return false;

b)

global \_path = new vector()

global \_pathFound = False

global bestProbability = -1

findBestPath ( numberOfHops, probabilitySum, prev\_server, cur\_server)

if ( cur\_server is equal to wanted server)

if( bestProbability < probabilitySum / numberOfHops )

\_path.clear()

\_path.push\_back(cur\_server);

bestProbability = probabilitySum / numberOfHops

return True

else return False

else {

bestPath = False

for each edge e (current, next){

bestPath = findBestPath( numberOfHops + 1, probabilitySum + r(prev\_server,cur\_server ), current, next);

if ( bestPath is True )

\_path.push\_back(cur\_server);

}

}

3a)

[127,71,87,31,59,3,43,99,100,42,0,58,30,88,72,130]

Res: 3,2,1,0

Averages:

[99,59,31,71,71,29,59,101]

[79,51,50,80]

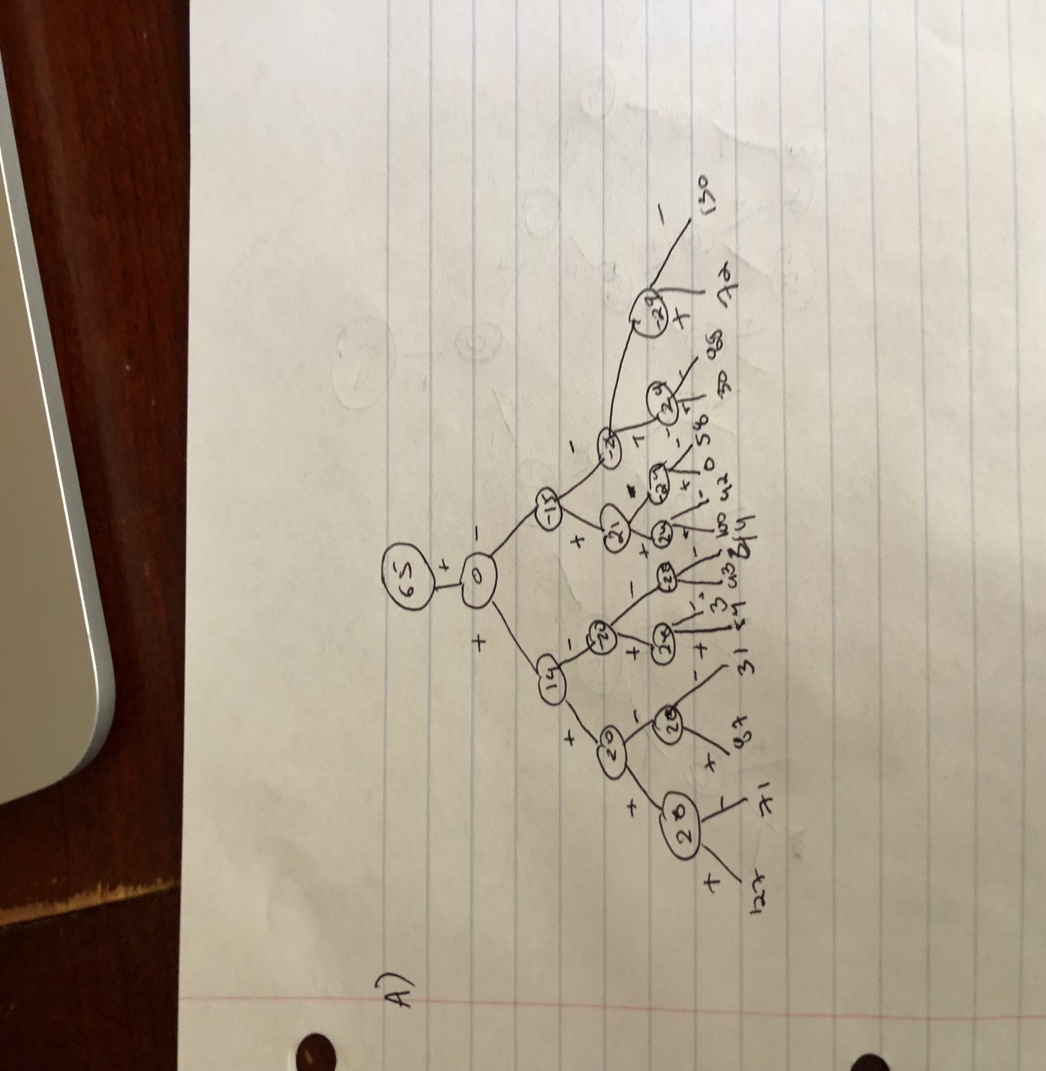
[65,65]

[65]

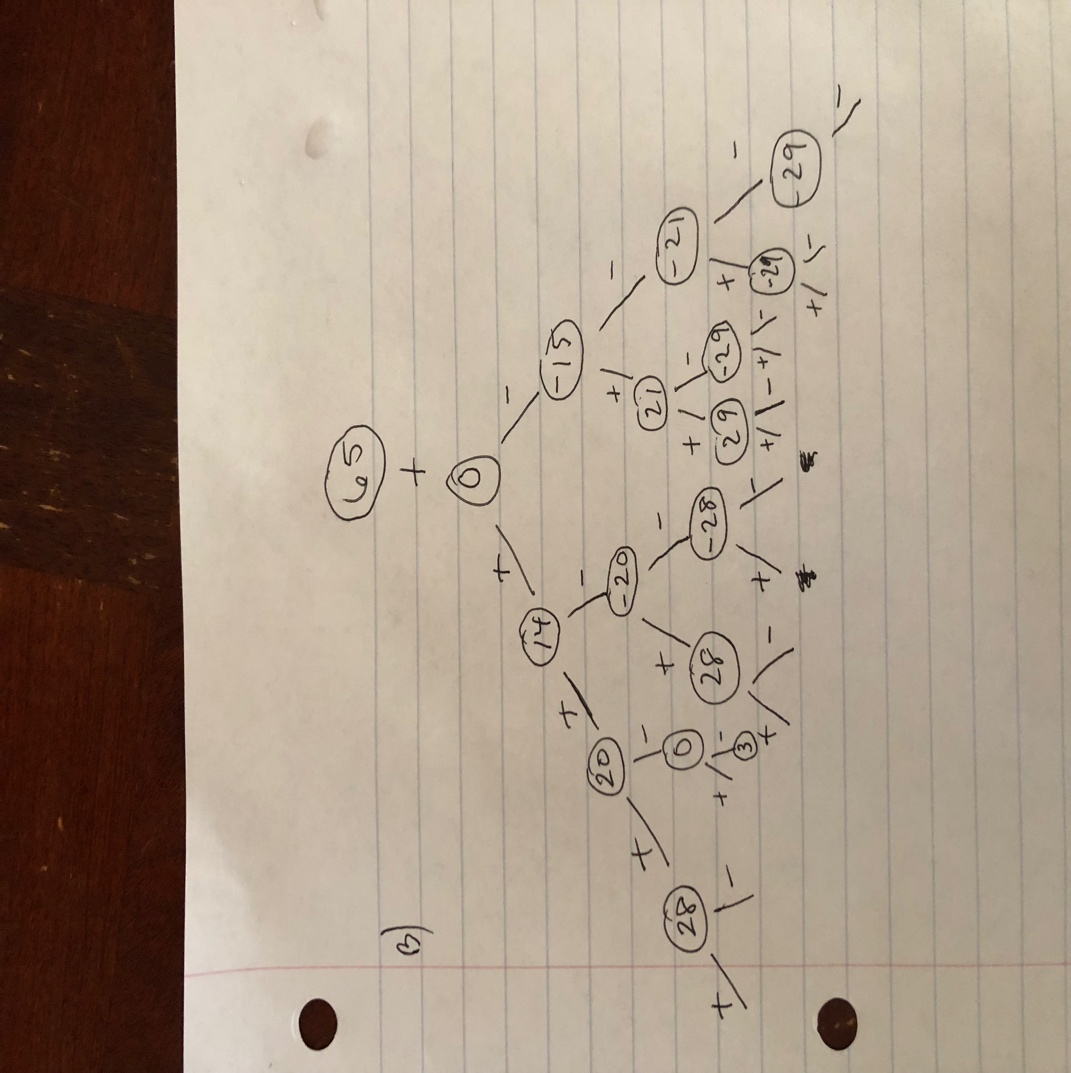
Detail Coefficients: [28,28,28,-28,29,-29,-29,-29] [20,-20,21,-21] [14,-15] [0]

Haar wavelet decomposition:

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



3b)



3c)

