Assignment 3

Data Structure and Algorithms
COMP 352
Section AA

By
Chloe Hei Yu Law
ID: 40173275
Concordia University
June 2021

Question 1

Assume a hash table utilizes an array of 13 elements and that collisions are handled by separate chaining.

Considering the hash function is defined as: h(k) = k mod (13).

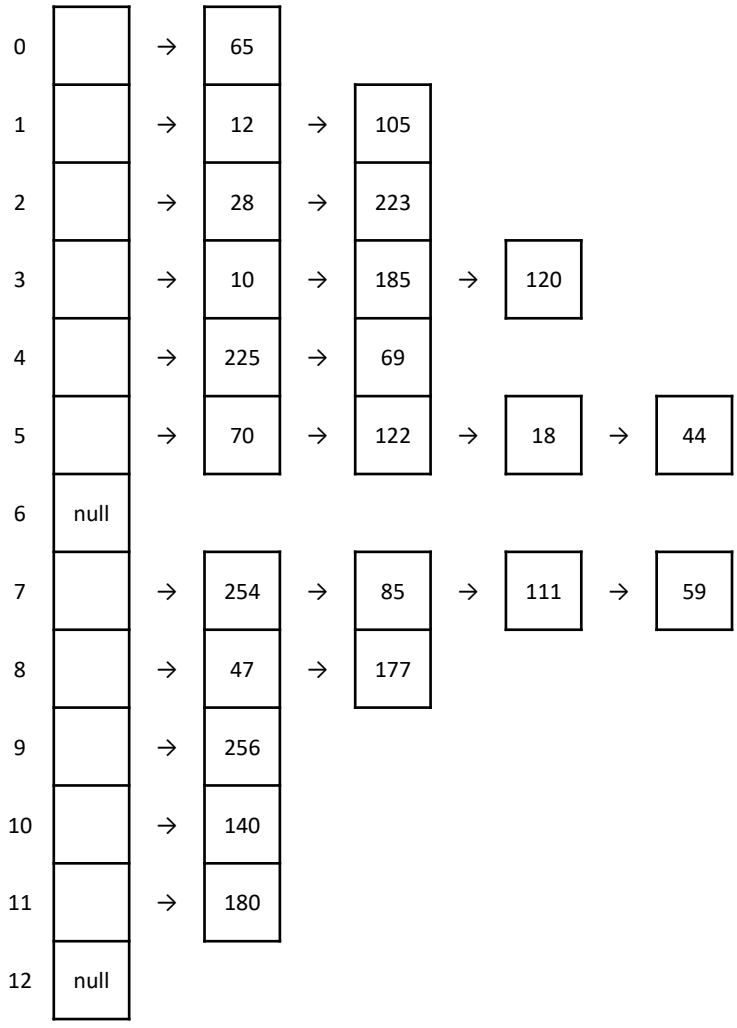
a) Draw the contents of the table after inserting elements with the following keys:

{245, 28, 10, 49, 70, 225, 122, 12, 180, 140, 177, 65, 223, 85, 111, 256, 18, 69, 59, 185, 105, 120, 44}.

Hash table size = 13

Hash function → h(k) = k mod (13)

h(254) = 254 mod (13) → 7
h(28) = 28 mod (13) → 2
h(10) = 10 mod (13) → 3
h(47) = 47 mod (13) → 8
h(70) = 70 mod (13) → 5
h(225) = 225 mod (13) → 4
h(122) = 122 mod (13) → 5
h(12) = 12 mod (13) → 1
h(180) = 180 mod (13) → 11
h(140) = 140 mod (13) → 10
h(177) = 177 mod (13) → 8
h(65) = 65 mod (13) → 0
h(223) = 223 mod (13) → 2
h(85) = 85 mod (13) → 7
h(111) = 111 mod (13) → 7
h(256) = 256 mod (13) → 9
h(18) = 18 mod (13) → 5
h(69) = 69 mod (13) → 4
h(59) = 59 mod (13) → 7
h(185) = 185 mod (13) → 3
h(105) = 105 mod (13) → 1
h(120) = 120 mod (13) → 3
h(44) = 44 mod (13) → 5



b) What is the total number of collisions caused by the above insertions?

7 + 3 + 2 = 12 collisions

Question 2

Assume an open addressing hash table implementation, where the size of the array N = 19, and the double hashing is performed for collision handling. The second hash function is defined as: d(k) = q - k mod q, where k is the key being inserted in the table and the prime number q = 11. Use simple modular operation (k mod N) for the first hash function.

a) Show the content of the table after performing the following operations, in order:

put(37), put(17), put(24), put(36), put(62), put(28), put(58),put(47), put(19).

Hash table size = 19

First hash function → h(k) = k mod(19)

Second hash function → d(k) = q – k mod (q) where q = 11

h(37) = 37 mod(19) → 18

h(17) = 17 mod(19) → 2

h(24) = 24 mod(19) → 5

h(36) = 36 mod(19) → 17

h(62) = 62 mod(19) → 5

d(62) = 11 – 62 mod(11) → 4

5 + 4 = 9

h(28) = 28 mod(19) → 9

d(28) = 11 – 28 mod(11) → 5

h(9+5) = (9+5) mod(19) → 16

h(58) = 58 mod(19) → 1

h(47) = 47 mod(19) → 9

d(47) = 11 – 47 mod(11) → 8

h(9+8) = (9+8) mod(19) → 2

h(9+2*8) = (9+2*8) mod(19) → 6

h(19) = 19 mod(19) → 0

0	19
1	58
2	17
3	Null
4	Null
5	24
6	47
7	Null
8	Null
9	62
10	Null
11	Null
12	Null
13	Null
14	Null
15	Null
16	28
17	36
18	37

b) What is the size of the longest cluster caused by the above insertions? 2

c) What is the number of occurred collisions as a result of the above operations? 4

d) What is the current value of the table's load factor? 9 / 19

Question 3

Assume the utilization of linear probing instead of double hashing for the implementation given in Question 2.

Still, the size of the array N = 19, and that simple modular operation (k mod N) is used for the hash function.

a) Show the contents of the table after performing the following operations, in order:

put(37), put(17), put(24), put(36), put(62), put(28), put(58), put(47), put(19).

Hash table size = 19

Hash function → h(k) = k mod(19)

put(37) = 37 mod(19) → 18

put(17) = 17 mod(19) → 2

put(24) = 24 mod(19) → 5

put(36) = 36 mod(19) → 17

put(62) = 62 mod(19) → 5

24 already at position 5

insert in next free position → 6

put(28) = 28 mod(19) → 9

put(58) = 58 mod(19) → 1

put(47) = 47 mod(19) → 9

28 already at position 9

insert a next free position → 10

put(19) = 19 mod(19) → 0

0	19
1	58
2	17
3	Null
4	Null
5	24
6	62
7	Null
8	Null
9	28
10	47
11	Null
12	Null
13	Null
14	Null
15	Null
16	28
17	36
18	37

b) What is the size of the longest cluster caused by the above insertions? Using Big-O notation, indicate the complexity of the above operations.

The size of the longest cluster is 3.

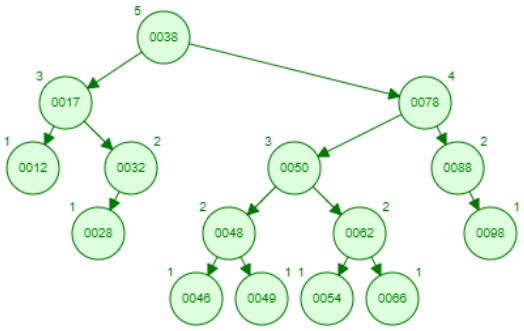
Best case complexity is O(1) when there is no collision.

Worst case complexity is O(n) when there is a collision because we have to search the entire array for the next free space.

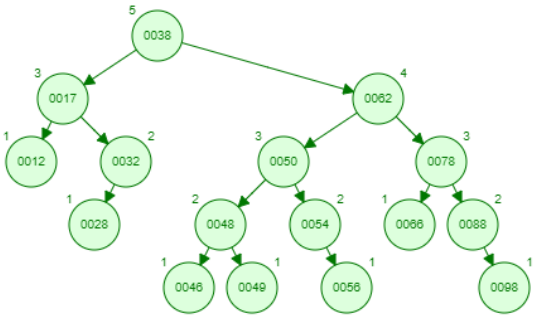
c) What is the number of occurred collisions as a result of the above operations? 2

Question 4

Consider the following AVL tree:



a) Draw the AVL tree resulting from the insertion of an entry with key 56 in the AVL tree shown above.



$56 \geq 38 \rightarrow$ right subtree

$56 < 78 \rightarrow$ left subtree

$56 \geq 50 \rightarrow$ right subtree

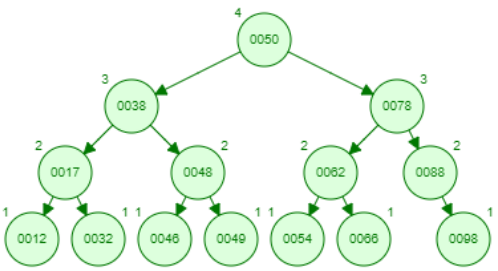
$56 < 62 \rightarrow$ left subtree

$56 \geq 54 \rightarrow$ right subtree

Adjust height → double rotate right

b) Draw the AVL tree resulting from the removal of the entry with key 28 in the AVL tree shown above.

(I did the removal from the initial given tree without key 56)



$28 < 38 \rightarrow$ left subtree

$28 > 17 \rightarrow$ right subtree

$28 < 32 \rightarrow$ left subtree

$28 == 28 \rightarrow$ found node to delete

The node to delete is a leaf → delete

Adjust height → double rotate left

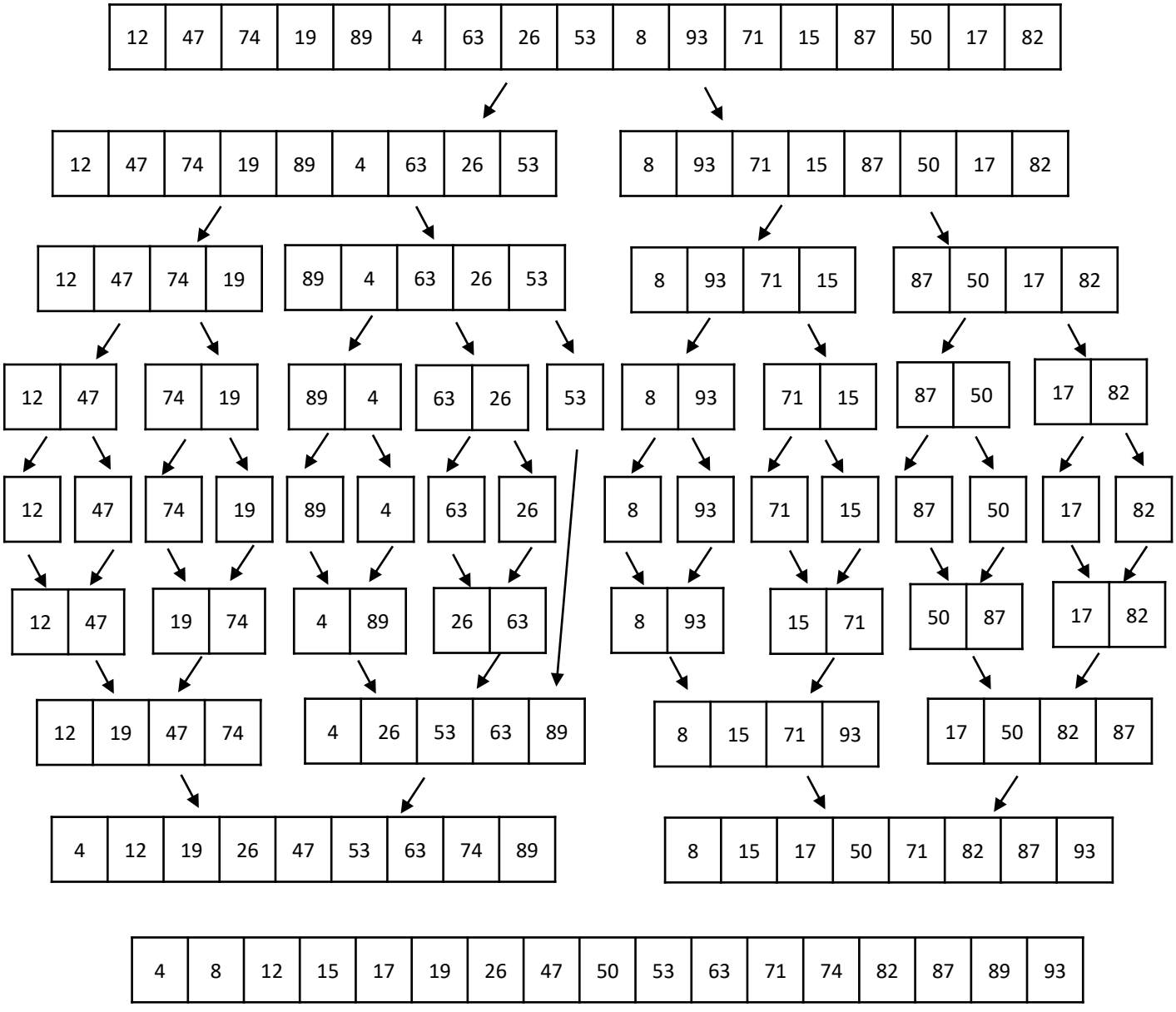
Question 5

Consider the following elements:

12, 47, 74,19, 89, 4, 63, 26, 53, 8, 93, 71, 15, 87, 50, 17, 82

Trace the steps when sorting these values into ascending order using:

a) Merge Sort



First step consists of the divide operation then the second step is the conquer operation which provides the given sorted list

Question 5

b) Quick Sort (using (middle +1) element as pivot point)

quicksort(arr, low, high) → quicksort(arr, 0, arr,length – 1)

1. Initial pivot = middle + 1 = (low + (high– low) / 2) = 8
2. Make left < pivot and right > pivot and let i = low and j = high, while i ≤ j
    1. While element at i < pivot then i++
    2. While element j > pivot then j--
    3. if i ≤ j then swap elements
3. If low < j, then recur with quicksort(arr, low, j)
4. If high > i, then recur with quicksort(arr, i, high)

12	47	74	19	89	4	63	26	53	8	93	71	15	87	50	17	82

Pivot = 8, swap(8, 12), swap(4, 47)

Pivot = 4, swap(4, 8)

Pivot = 93, swap(82, 93)

Pivot = 12, swap(12, 74)

Pivot = 82, swap(17, 89), swap(50, 82)

Pivot = 53, swap(15, 63), swap(50, 53)

Pivot = 15, swap(15, 19)

Pivot = 26, swap(26, 47)

Pivot = 19, swap(19, 26)

Pivot = 19, swap(19, 19)

Pivot = 50, swap(50, 50)

Pivot = 71, swap(63, 74), swap(71, 71)

Pivot = 53, swap(53, 63)

Pivot = 89, swap(89, 89)

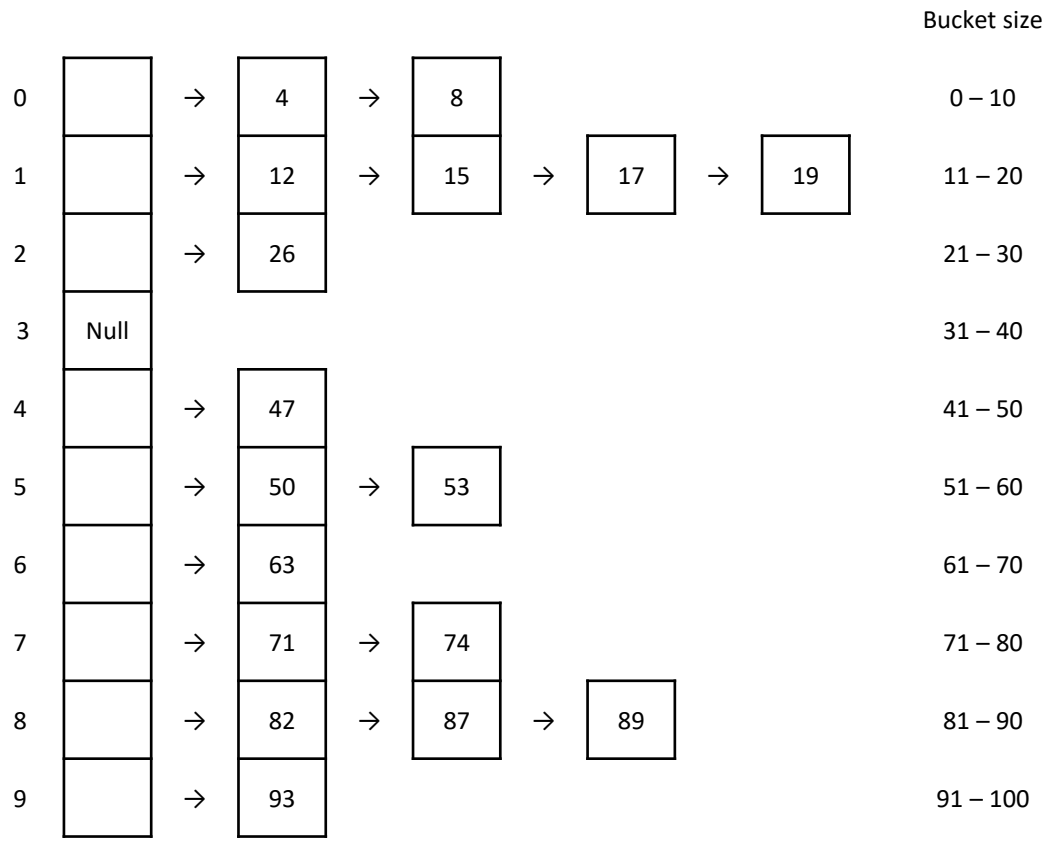
Pivot = 82, swap(82, 87)

4	8	12	15	17	19	26	47	50	53	63	71	74	82	87	89	93

Question 5

c) Bucket Sort – We know that the numbers are less than 99 and there are 10 buckets.

12	47	74	19	89	4	63	26	53	8	93	71	15	87	50	17	82



4	8	12	15	17	19	26	47	50	53	63	71	74	82	87	89	93

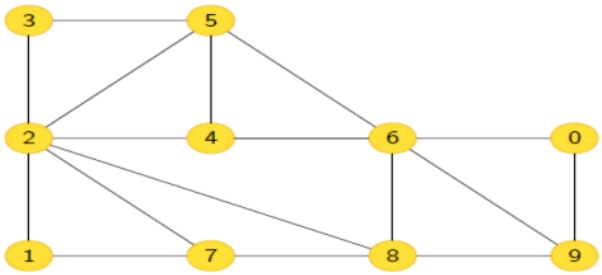
Question 5

d) Radix Sort

12	47	74	19	89	4	63	26	53	8	93	71	15	87	50	17	82
Sorting by the least significant digit (1s place)																
50	71	12	82	53	63	93	4	74	15	26	17	47	87	8	19	89
Sorting by next digit (10s place)																
4	8	12	15	17	19	26	47	50	53	63	71	74	82	87	89	93

Question 6

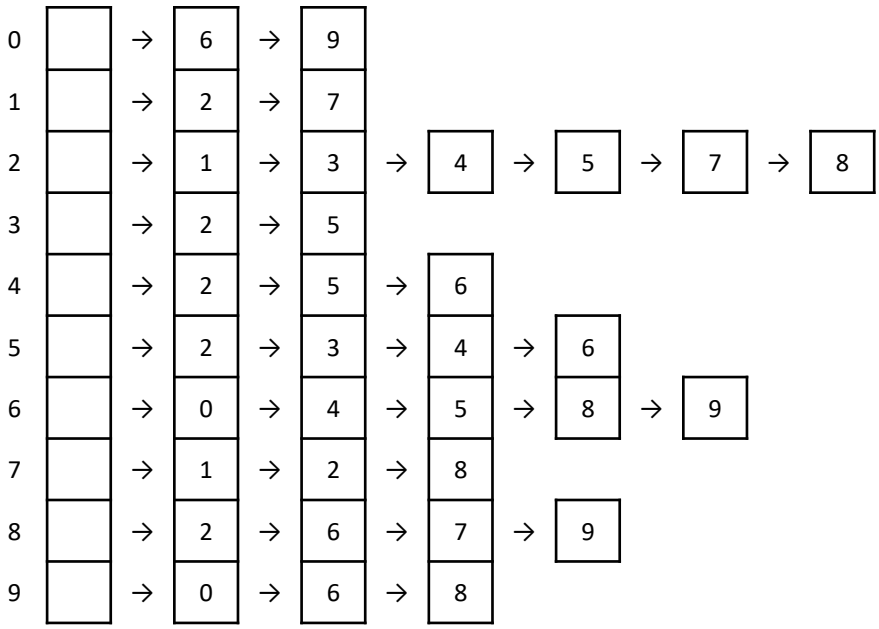
Consider the graph shown below:



1. Give the adjacency matrix representing this graph.

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0	1	0	0
2	0	1	0	1	1	1	0	1	1	0
3	0	0	1	0	0	1	0	0	0	0
4	0	0	1	0	0	1	1	0	0	0
5	0	0	1	1	1	0	1	0	0	0
6	1	0	0	0	1	1	0	0	1	1
7	0	1	1	0	0	0	0	0	1	0
8	0	0	1	0	0	0	1	1	0	1
9	1	0	0	0	0	0	1	0	1	0

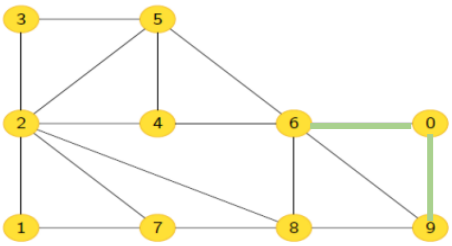
2. Give the adjacency lists representing this graph.



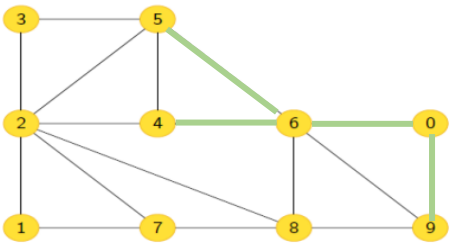
Question 6

3. Show the breadth-first search trees for the graph starting at node 0.

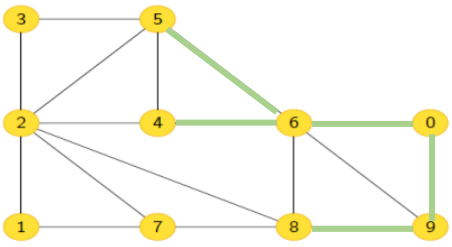
Step 1: From 0 to 6 and 9



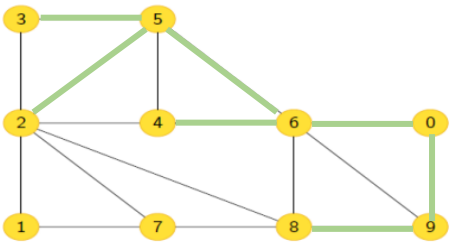
Step 2: From 6 to 5 and 4



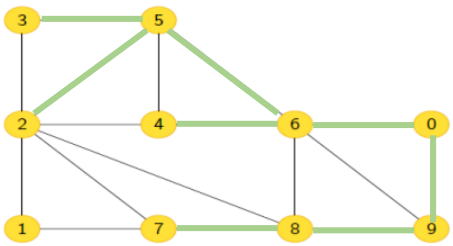
Step 3: From 9 to 8



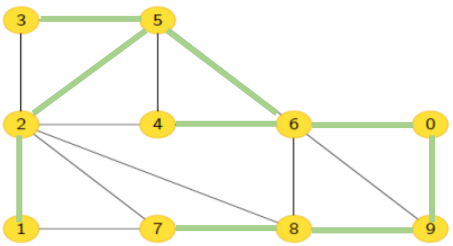
Step 4: From 5 to 3 and 2



Step 5: From 8 to 7



Step 5: From 2 to 1



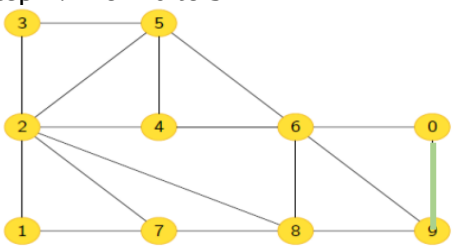
Therefore, the breadth-first search for the graph starting at node 0 is:

0 – 6 – 9 – 5 – 4 – 8 – 3 – 2 – 7 – 1

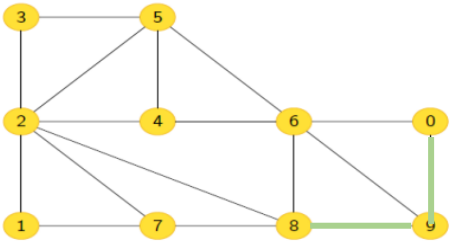
Question 6

4. Show the depth-first search trees for the graph starting at node 0

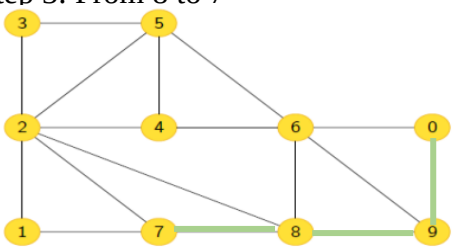
Step 1: From 0 to 9



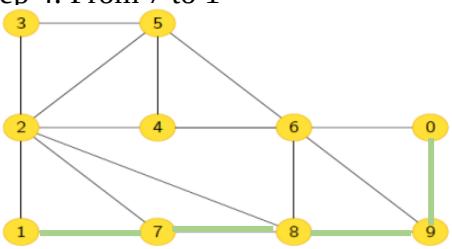
Step 2: From 9 to 8



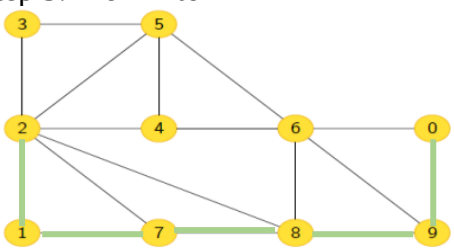
Step 3: From 8 to 7



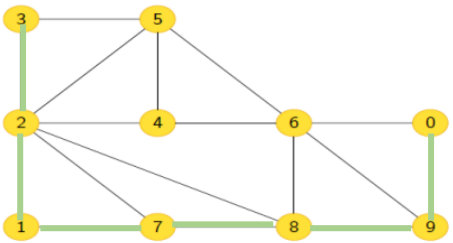
Step 4: From 7 to 1



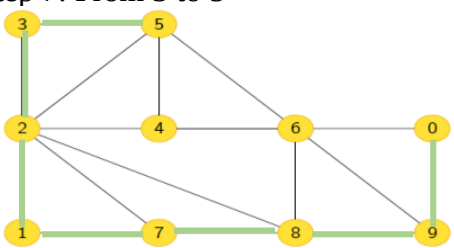
Step 5: From 1 to 2



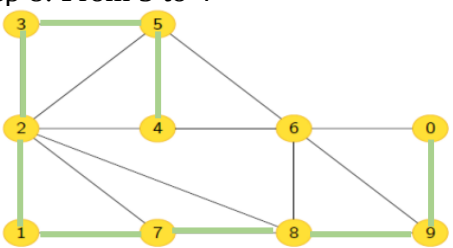
Step 6: From 2 to 3



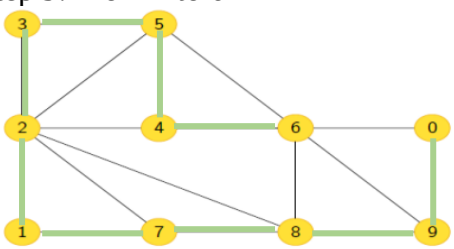
Step 7: From 3 to 5



Step 8: From 5 to 4



Step 9: From 4 to 6

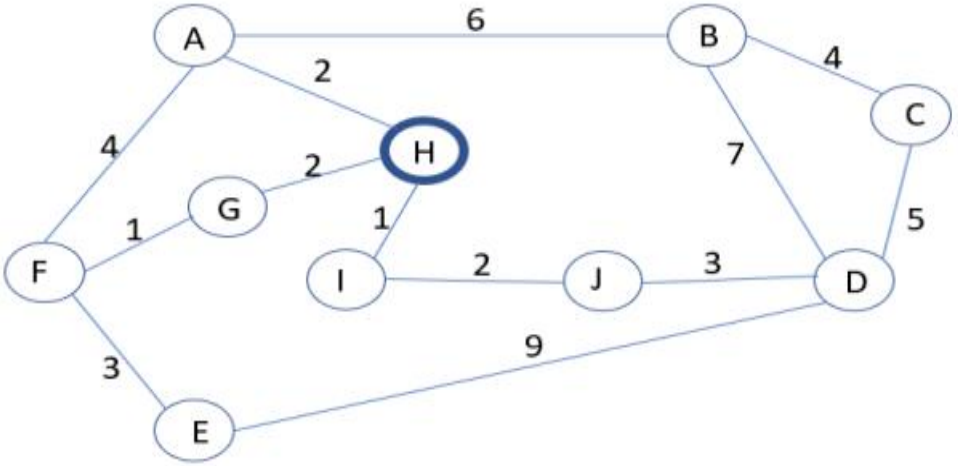


Therefore, the depth-first search for the graph starting at node 0 is:

0 – 9 – 8 – 7 – 1 – 2 – 3 – 5 – 4 – 6

Question 7

Use Dijkstra's Algorithm to find the shortest path of the following graph from node H to each of the other nodes.



The selected node to mark the minimum distance is node C. Therefore, for node C, the distance is 0. For the rest of the nodes, since the minimum distances are unknown, it is initialized to infinity. Compare the minimum distance to infinity and leave the smallest value.

Vertex	Shortest path	Distance
A	H – A	2
B	H – A – B	8
C	H – I – J – D – C	11
D	H – I – J – D	6
E	H – G – F – E	5
F	H – G – F	3
G	H – G	2
H	H	0
I	H – I	1
J	H – I – J	3