CS 58000_01/02I Design, Analysis and Implementation Algorithms (3 cr.)
Assignment As_03 (Exam 02)
Student Name: Truc Huynh

Problem 1[30 points]:

1a. Show that for any real constants $a < 0$ and $b > 0$,

$$(n + a)^b = \Theta(n^b).$$

ANSWER:

To prove (n+a)^b = O(n^b), we must prove that there is exist constant c_1, c_2, $n_0 > 0$ such that 0 < c_1*n^b <= (n+a)^b <= c_2 * n^b for all n>= n0

n+a <= 2*n, when |a| <= n

n+a>= ½*n, when |a| <= n/2

Therefore n >= 2|a| and 0<= n/2 <= (n+a) <= 2n

As b>0, we raise all the terms of the previous inequality to the power of b without breaking the inequality.

0^b <= (n/2)^b <= (n+a)^b <= (2n)^b

0 <= (1/2)^b * n^b <= (n+a)^b <= 2^b*n^b

0 <= 1/(2^b)*n^b <= (n+a)^b <= 2^b*n^b

Therefore there is exists c_1 = 1/(2^b), c_2 = 2^b, and n_0 = 2|a|

1b. Explain why the statement "The running time of algorithm A is at least O(n2)" is meaningless.

ANSWER:

T(n): running time of Algo A.

T(n) The statement is: T(n) >= O(n^2). To decide the meaning of a function we need to decide the upper bound and lower bound of it.

Upper bound: Because T(n) >= O(n^2), then there's no information about upper bound of T(n)

Lower bound: Assume f(n) = O(n^2), then the statement is: T(n) >= f(n), but f(n) could be anything that is "smaller" than n^2. Ex: constant, n,..., So there's no conclusion about lower bound of T(n) too

Therefore, The statement is meaningless

Reference:

big o - Running time of algorithm A is at least O(n²) - Why is it meaningless? - Stack Overflow

1c. Is 2n+1 = O(2n)? Justify your answer.

Is 22n = O(2n)? Justify your answer.

ANSWER:

Here

Is $2^{n+1} = O(2^n)$? Justify your answer

n	2^(n+1)	2^n	compare
0	2	1	not equal
1	4	2	not equal
2	8	4	not equal

Therefore 2^n+1 is not equal to O(2^n) and

Is $2^{2n} = O(2^n)$? Justify your answer.

n	2^(n+1)	2^n	compare
0	1	1	equal
1	4	2	not equal

Therefore, 2^{2n} is not equal $O(2^n)$

Problem 2 [30 points]:

Order of the following functions according to their order of growth (from the lowest to the highest)

(n – 2)!, 22n, 0.002 n4 + 3n2 +1, 2^n, e^n, $n2^n$, 1n2n, 3√ n, 3n, $2^{\log n}$, n^2, $4^{\log n}$, $\sqrt{\log n}$

{Hint: 1n2 n = (loge n) (loge n) where e = 2.71828.}

ANSWER:

	Functions		Similarity group
F(1)	(n – 2)!	(n-2)! => O((n-2)!) This a factorial and can implement by a recursive call from 1 to n-2	
F(2)	2^2n	2^2n = (2^2) ^n = 4^n => O(4^n)	Group 1
F(3)	0.002*n^4 + 3*n^2 +1	0.002*n^4 + 3*n^2 +1 = infinity + infinity + constant = infinity + infinity + 0	Group 2

		=> Therefore, the time complexity will be O(n^4) as n^4 is the highest	
F(4)	2^n	2^n => O(2^n)	Group 1
F(5)	e^n	e^n => O(e^n) => e is a constant there for e^n belongs to Group 1	Group 1
F(6)	n*2^n	n*2^n => O(n2^n) => Due to the time complexity of this function is based on ^n. Therefore it belongs to group 1	Group 1
F(7)	1*n^2*n	1*n^2*n = n^2n =>, since 2 is a constant therefore the time complexity of this function, is O(n^2n)	
F(8)	3√ n	3√ n => O(n^1/3)	Group 2
F(9)	3^n	3^n => O(3^n)	Group 1
F(10)	2^log n	2^log n => O(2^log n) the logarithmic function takes less time complexity than other given function	Group 3
F(11)	n^2	n^2 => O(n^2)	Group 2
F(12)	4^log n	4^log n => O(4^log n) the logarithmic function takes less time complexity than other given function	Group 3
F(13)	√log n	√log n = O(log n^1/2)	

Group:

- Group 1: F(2), F(4), F(5), F(6), F(9)
- Group 2: F(3), F(11), F(8)
- Group 3: F(10), F(12)
- Standalone: F(1), F(7), F(13)

Total Comparison:

F(1) = O((n-2)!)
F(7) = O(n^2n)
F(13) = O(log n^1/2): will take less than other functions because ^½ is a constant. Therefore, O(log n^1/2) is equal to O(log n) (in the order of growth)
Group 3 sampling: F(10) = O(2^log n): second smallest in the order of growth in logarithmic function takes less than other given function
Group 1 sampling: F(2) = O(4^n)
Group 2 sampling: F(3) = O(n^4): third smallest

F(13) < Group3< Group2< Group1< F(7)< F(1)

Group 1 Comparision:

We have Group 1: F(2), F(4), F(5), F(6), F(9)
F(2) = O(4^n)
F(4) = O(2^n)
F(5) = O(e^n) = O(2.71828^n)
F(6) = O(n2^n)
F(9) = O(3^n)
Therefore, the order of growth within group 1 should be: F(4)<F(5)<F(9)<F(2)<F(6)

Group 2 Comparision:
We have Group 2: F(3), F(8), F(11)
F(3) = O(n^4)
F(8) = O(n^1/3)
F(11) = O(n^2)
Therefore, the order of growth within group 1 should be: F(8)<F(11)<F(3)

Group 3 Comparision:
We have Group 3: F(10), F(12)
F(10) = O(2^log n)
F(12) = O(4^log n)
Therefore, the order of growth within group 1 should be: F(10)<F(12)
Finally: F(13) < Group3< Group2< Group1< F(7)< F(1)

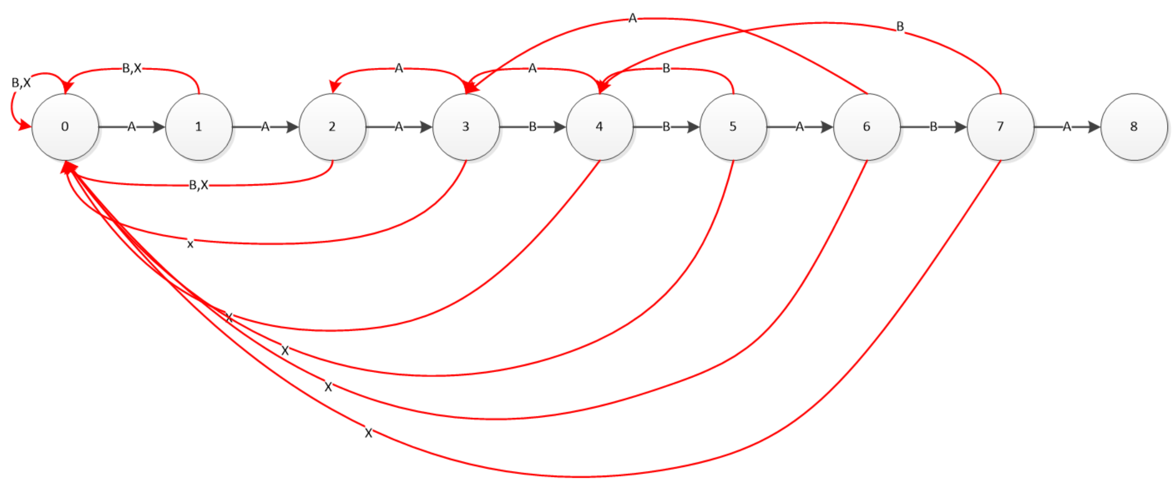
Final Result:
F(13)<F(10)<F(12)<F(8)<F(11)<F(3)<F(4)<F(5)<F(9)<F(2)<F(6) <= F(7)<F(1)
Therefore:
[√log n] < [2^log n] < [4^log n] < [3√n] < [n^2] < [0.002*n^4 + 3*n^2 +1] < [2^n] < [e^n] < [3^n] < [2^2n]< [n*2^n] = [1*n^2*n] < (n-2)!

Problem 3[40 points]:
Construct the string-matching automaton for the pattern P = aaabbaba over the alphabet Σ = {a, b, x |x is any letter other than a and b}; and illustrate its operation on the text string T = aaaabbabaaabbaaabbababaab.

3a. Construct the string-matching automation for the pattern P over the alphabet Σ = {a, b, x}in terms of the state transition table (Complete the state transition table)

ANSWER:

	input			P
state	a	b	x	
0	1	0	0	a
1	2	0	0	a
2	3	0	0	a
3	2	4	0	b
4	3	5	0	b
5	6	4	0	a
6	3	7	0	b
7	8	4	0	a
8				



3b. Show the operation on the text string T, computed by the state transition table. Complete the following table, in which T[i] is the letter at the position i of the text string; and State Φ(T[i]) stands for the state transition Φ (s, T[i]) = s'.

text string T = aaaabbabaaabbaaabbababaab.

ANSWER:

Stop when hit step 8, I found the pattern matches the text string at index i=9. I then stop to fill in the table because I have already found the matched string there is no need to keep going

i		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T[i]		a	a	a	a	b	b	a	b	a	a	a	b	b	a	a	a	b	b	a	b	a	a	b	
State Φ(T[i])	0	1	2	3	3	4	5	6	7	8															

3c. Complete the following sentence.

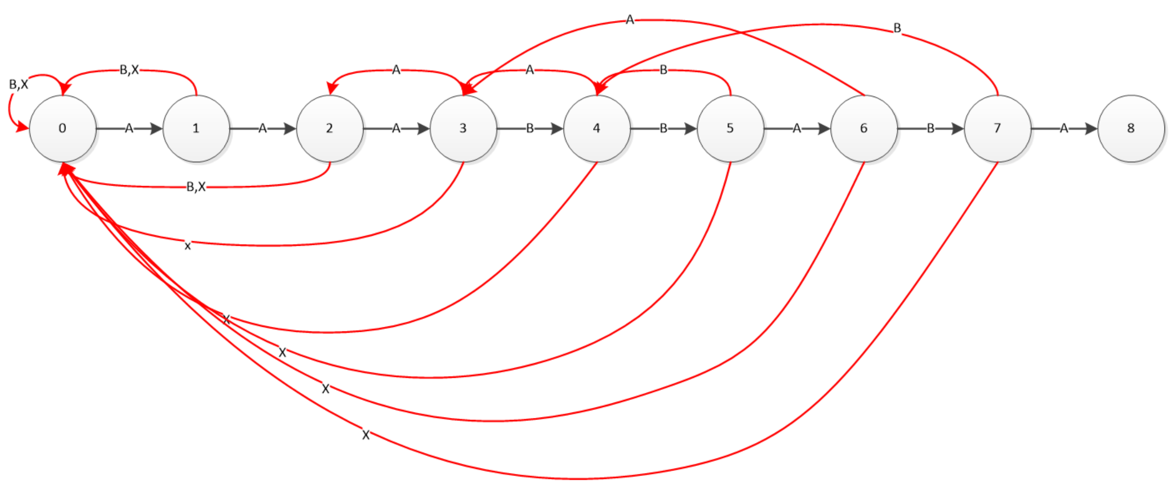
ANSWER:

The result is text string T = “aaaabbabaaabbaaabbabaab” matches the pattern at shift = 1 (i =2) and shift = 8 (i =9). (Note that shift = i – 1)

3d. Draw a state transition diagram for a string-matching automaton for the pattern P over the alphabet Σ = {a, b, x |x is any letter other than a and b}.

ANSWER:

The state transaction diagram was drawn with Microsoft Visio.



Problem 4[30 points]

Consider the following Algorithm Quicksort:

```
Algorithm  Quicksort(A[p .. r])
    //Quicksort(A[0 .. n - 1]) is the initial call for sorting an entire array A.
    Input:   A subarray  A[p .. r] of  A[0 .. n-1], defined by its left and right indices p and r.
    Output: Subarray  A[p .. r]  sorted in nondecreasing order
    {
    if (p < r )
              {s ← Partition(A[p .. r]);  //s ← j is a split position
               Quicksort(A[p .. s-1]);    //there is s - p elements
               Quicksort(A[s+1 .. r]);}   //there is r - s elements
    }//end of Quicksort()
```

```
Algorithm Partition(A[p .. r])
    //Partitions a subarray by using its first element as a pivot.
    Input:    A subarray A[p .. r] of A[0 .. n-1], defined by its left and right indices p and r,
              (p < r).
    Output:  A partition of A[p .. r], with the split position returned as this function's value
    {
    x ← A[p];       //set x be the leftmost element of A[p .. r].
    i ← p; j ← r+1;  //set left and right pointers pointing at p and r+1
    repeat
        repeat i ← i+1 until A[i] ≥ x; //move i towards right until …
        repeat j ← j-1 until A[j] ≤ x; //move j towards left until …
        swap(A[i], A[j]);
    until i ≥ j;
    swap(A[i], A[j]);
    swap(A[p], A[j]);
    return j;
    }
```

In the Algorithm Partition(A[p .. r]), there are three swap() procedures.

4a. When and why the first swap(A[i], A[j]) is needed?

ANSWER:

- The first swap(A[i], A[j]) is needed because this first swap enables i and j to continue to move.
- The move until i and j cross each other or both i and j point at the same place.
- We also need to increment i and decrement j after each swap.

4b. When and why the second swap(A[i], A[j]) is needed?

ANSWER:

- The second swap (A[i], A[j]) is needed to undo the last swap when i>= j.
- This means when i>= j we need to partition the array after exchanging the pivot with A[j]

4c. When and why the third swap(A[p], A[j]) is needed?

ANSWER:

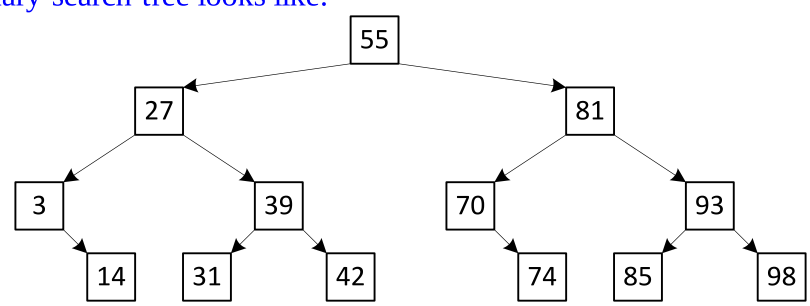
- The third swap is needed to exchange the pivot A[p] with A[j] whenever i>j

Problem 5 [70 points]

Given the following array A[0..15] contains 13 elements.

	3	14	27	31	39	42	55	70	74	81	85	93	98		

It is helpful when answering these questions 5a through 5d to recognize what an equivalent binary search tree looks like:



5a. What is the largest number of key comparisons made by binary search in searching for a key in the following array?

ANSWER:

Both procedures run in O(h) time on a tree of height h = Θ(log n). Θ(log n) in the average case

The number of elements in an array n=13 maximum operation. Thus Cworst(n) = ± $\log_2(n+1)= \log_2(13+1) = 3.907352 = 4$.

5b. List all the keys of this array that will require the largest number of key comparisons when searched for by binary search.

ANSWER:

Value	3	14	27	31	39	42	55	70	74	81	85	93	98
Key	0	1	2	3	4	5	6	7	8	9	10	11	12

According to the tree structure, the lowest level requires the largest key comparison. They are:

Value	14	31	42	74	85	98
Key	1	3	5	8	10	12

5c. Find the average number of key comparisons made by binary search in a successful search in this array. (Assume that each key is searched for with the same probability.)

ANSWER:

The average number of key comparisons made by binary search in a successful search is:

= 1* (1/13) + 2 * (2/13) + 3 * (4/13) + 4 * (6/13) = 1/13 + 4/13 + 12/13 + 24/13 = (1+4+12+24) / 13 = 41/13 =3.2

5d. Find the average number of key comparisons made by binary search in an unsuccessful search in this array. (Assume that searches for keys in each of the 14 intervals formed by the array's elements are equally likely.)

ANSWER:

There are 3 comparisons at position 6, or 7 (the key is at level 0 and between positions 6 and 7). For the remaining 12 elements, there will be 4 comparisons occurring. The average number of key comparisons made by binary search in an unsuccessful search is:

3*(2/14) + 4*(12/14) = (48+6)/14 = 54/14 = 3.9

5e. Assume that the arrival of the elements is in the order 3, 14, 27, …., 98 of a given array A[0..15]. Rearrange the contents of 13 elements such that array A forms an AVL tree. Show step-by-step in terms of the intermediate resulting arrays.

ANSWER:

The design is drawn using MS Visio. The sourcee is here: [Drawing.vsdx]()

AVL Tree is a self-balancing BST. There is 4 possible rotation: LL Rotation, RR Rotation, LR Rotation, RL Rotation

	3	14	27	31	39	42	55	70	74	81	85	93	98		

Insert 3:

Node	Height
3	0

no rotation

	3														

Insert 14:

Node	Height
3	-1
14	0

no rotation

	3		14												

Insert 27:

Node	Height
3	-2
14	-1
27	0

Rotate at 14, and 14 becomes the root

	14	3	27												

Insert 27:

Node	Height
3	0
14	-1
27	-1
31	0

No rotation

	14	3	27				31								

Insert 39:

Node	Height
3	0
14	-2
27	-2
31	-1
39	0

Rotate at 27

	27	14	31	3			39								

Insert 42:

Node	Height
3	
14	
27	
31	-2
39	-1
42	0

Rotate at 31

	27	14	39	3		31	42								

Insert 55:

Node	Height
3	
14	
27	-1
31	-1
39	-1
42	-1
55	0

No rotation I needed

	27	14	39	3		31	42								55

Insert 70:

Node	Height
3	
14	
27	
31	
39	
42	-2
55	-1
70	0

Rotate at 42

	27	14	39	3		31	55							42	70

Insert 74:

Node	Height
3	
14	
27	
31	

39	-2
42	-1
55	
70	-1
74	0

Rotate at 39

	27	14	55	3		39	70					31	42		74

Insert 81:

Node	Height
3	
14	
27	
31	
39	
42	
55	
70	-2
74	-1
81	0

Rotate at 70

	27	14	55	3		39	74					31	42	70	81

Insert 85:

Node	Height
3	
14	
27	-2
31	
39	
42	

55	-1
70	0
74	-1
81	-1
85	0

Rotate at 27

	55	27	74	14	39	70	81	3		31	42				85

Insert 93:

Node	Height
3	
14	
27	
31	
39	
42	
55	
70	
74	
81	-2
85	-1
93	0

Rotate at 81

	55	27	74	14	39	70	85	3		31	42			81	93

Insert 98:

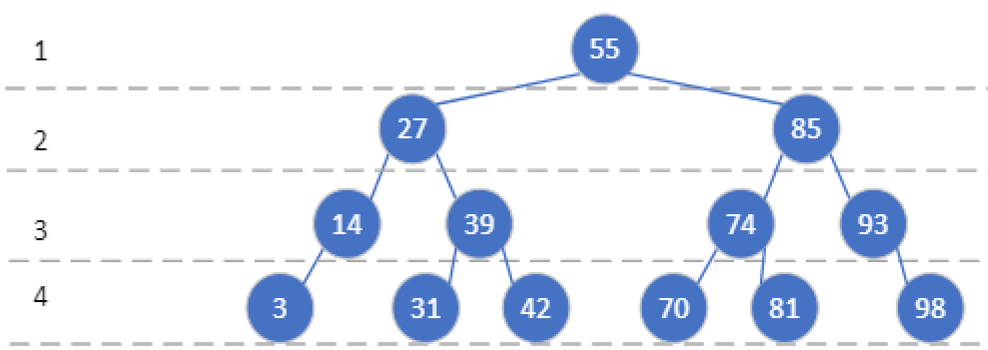
Node	Height
3	
14	
27	
31	

39	
42	
55	
70	
74	-2
81	
85	-1
93	-1
98	0

Rotate at 74

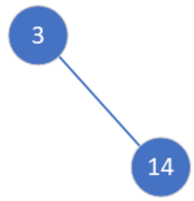
	55	27	85	14	39	74	93	3		31	42	70	81		98

Result:

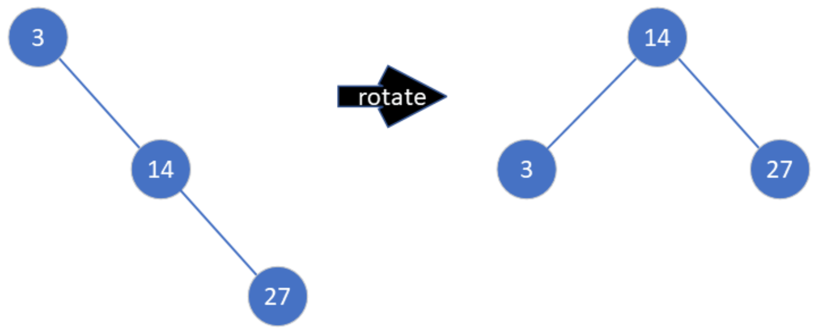


Step 1:Insert 3, 3 now is the root

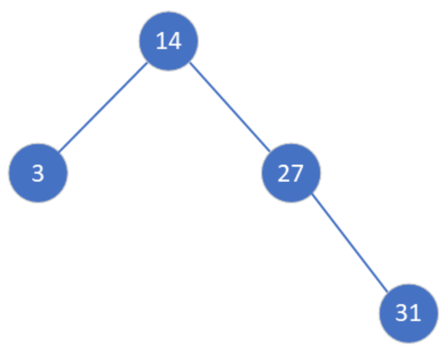
Step 2:Insert 14, 14 is bigger than 3, 14 go to the right of the tree

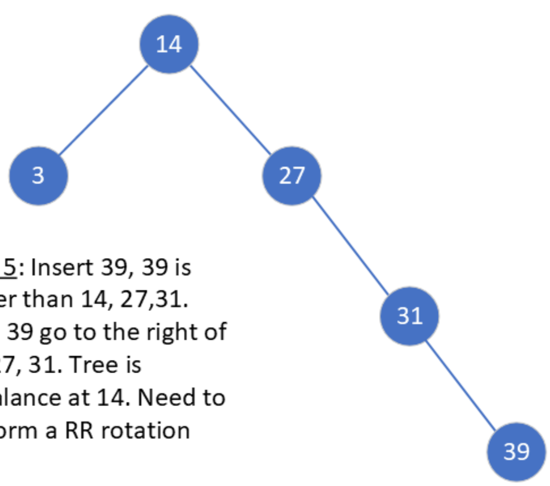


Step 3: Insert 27, 27 is bigger than 3 and 14, 27 go to the right of 14. Now this tree is unbalance and we need to perform a RR rotation

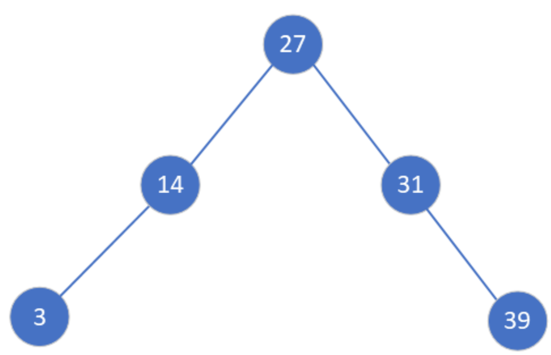


Step 4: Insert 31, 31 is bigger than 14, 27. Thus 31 go to the right of 14, 27. Tree is still balance.

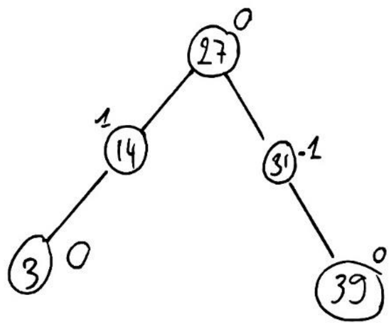




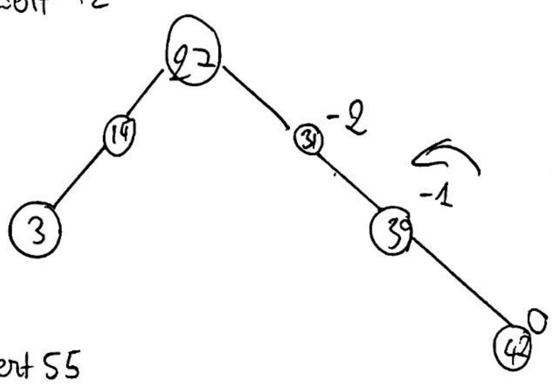
Step 5: Insert 39, 39 is bigger than 14, 27,31. Thus 39 go to the right of 14, 27, 31. Tree is unbalance at 14. Need to perform a RR rotation



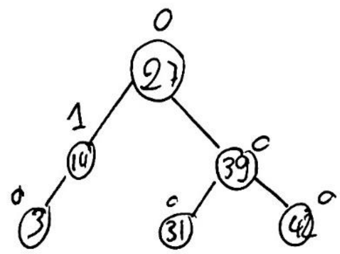
$B = L - R$



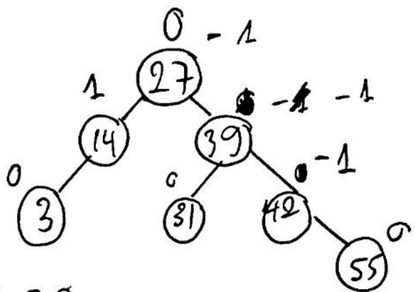
insert 42



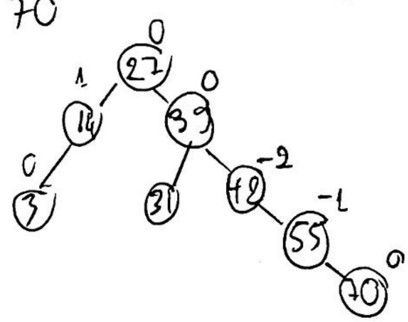
rotate at 31 →



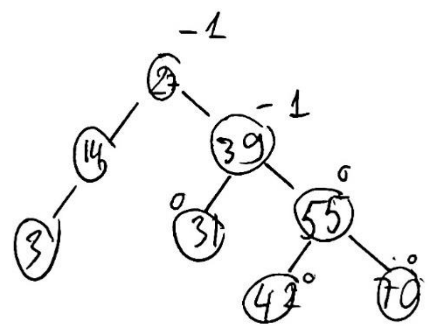
insert 55



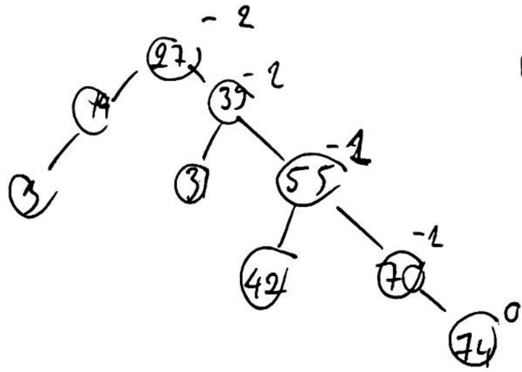
insert 70



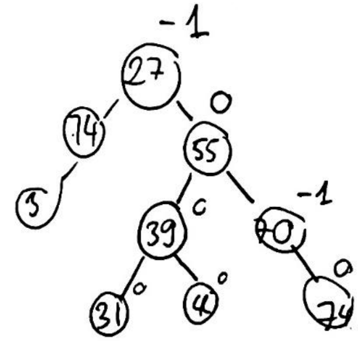
rotate at 42 →



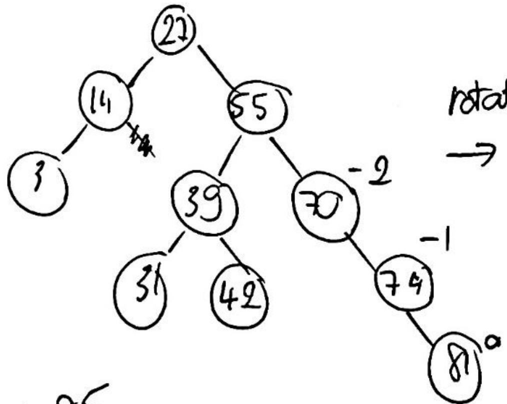
Insert 74 :



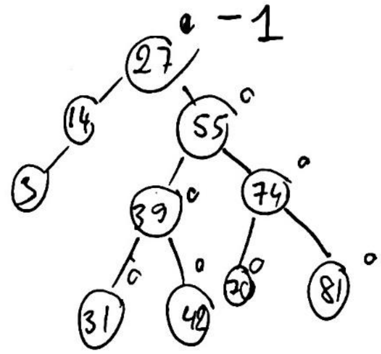
rotate at 39 ->



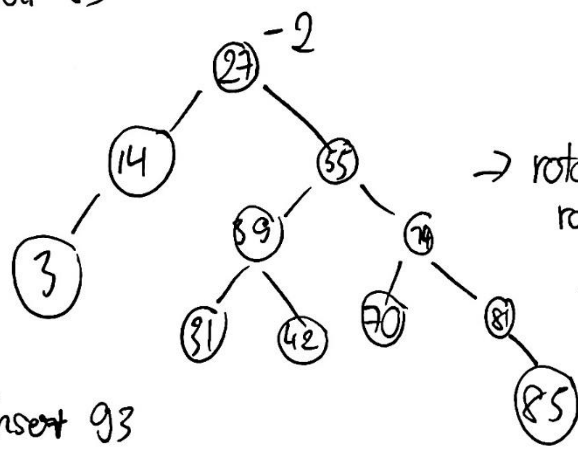
Insert 81



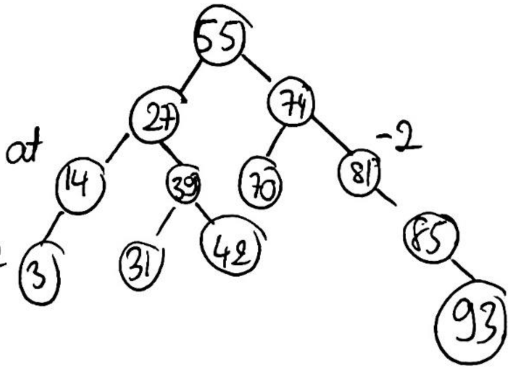
rotate at 70 ->



Insert 85

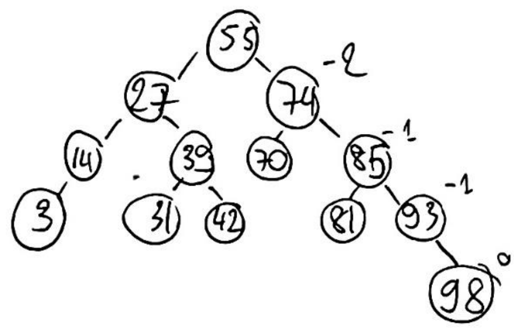


-> rotate at root 27

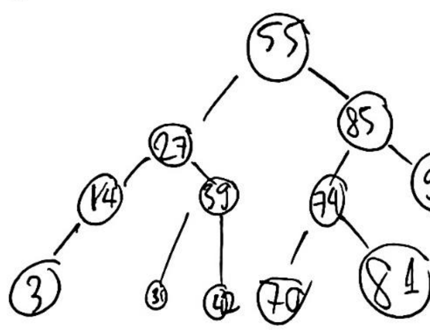


Insert 93

After insert at 93 rotate at 81.



Insert 98



5f. What is the largest number of key comparisons in searching for a key in array A which has an AVL tree?

ANSWER:

Both procedures run in O(h) time on a tree of height h = Θ(log n). Θ(log n) in the average case. The number of elements in an array n=13 maximum operation. Thus largest number of key comparision = $\log_2(n) = \log_2(13) = 3.7 = 4$

5g. List all the keys of this array that will require the largest number of key comparisons when searched for by binary search.

ANSWER:

Key	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Value	55	27	85	14	39	74	93	3		31	42	70	81		98
	55	27	85	14	39	74	93	3		31	42	70	81		98

According to the tree structure, the lowest level requires the largest key comparison. They are:

Value	7	9	10	11	12	14
Key	3	31	42	70	81	98

Problem 6 [20 points]

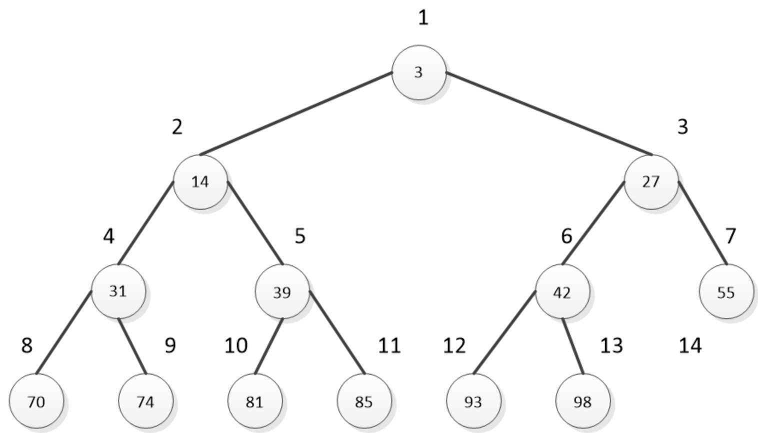
Given the following array A[0..15] contains 13 elements.

	3	14	27	31	39	42	55	70	74	81	85	93	98		

6a. Use Max Heapify(A, i), 0< i to maintain given array A[0..15] a max-heap.

ANSWER:

This is the heap structure before apply the MaxHeapify



This structure violate the definition of maxheap. Therefore, we need to reconstruct

	3	14	27	31	39	42	55	70	74	81	85	93	98		

<u>1st run:</u>

Run Heapify from $\lfloor$ lengthA[ ]/2 $\rfloor$ to 1, to determine their children node, A[2i] or A[2i+1]. Therefore, we start at index i= 12/2 = 6 and 42, 93 < 98. Therefore swap 98 and 42.

	3	14	27	31	39	98	55	70	74	81	85	93	42		

Run Heapify again since i-- now i=5 and 39,81 < 85. Therefore swap 39 and 85

	3	14	27	31	85	98	55	70	74	81	39	93	42		

Run Heapify again since i-- now i=4 and 31,70 < 74. Therefore swap 31 and 74

	3	14	27	74	85	98	55	70	31	81	39	93	42		

Run Heapify again since i-- now i=3 and 27,55 < 98. Therefore swap 27 and 98

	3	14	98	74	85	27	55	70	31	81	39	93	42		

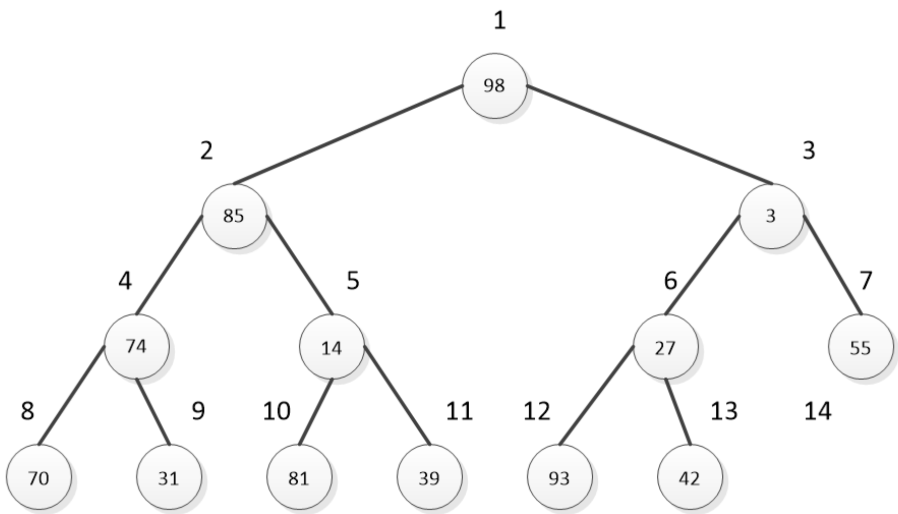
Run Heapify again since i-- now i=2 and 14,74 < 85. Therefore swap 14 and 85

	3	85	98	74	14	27	55	70	31	81	39	93	42		

Run Heapify again since i-- now i=1 and 3,85 < 98. Therefore swap 3 and 98

	98	85	3	74	14	27	55	70	31	81	39	93	42		

This is the heap structure after 1st round



At this point we run function to validate if this is a maxheap. And it is not therefore, we recursive run the heapify again.

	98	85	3	74	14	27	55	70	31	81	39	93	42		

<u>2nd run:</u>

Run Heapify from └ lengthA[ ]/2 ┘ to 1, to determine their children node, A[2i] or A[2i+1]. Therefore, we start at index i= 12/2 = 6 and 27, 42 < 93. Therefore swap 93 and 27.

	98	85	3	74	14	93	55	70	31	81	39	27	42		

Run Heapify again since i-- now i=5 and 14,39 < 81. Therefore swap 14 and 81

	98	85	3	74	81	93	55	70	31	14	39	27	42		

Run Heapify again since i-- now i=4 and 70,31 < 74. DO NOTHING

	98	85	3	74	81	93	55	70	31	14	39	27	42		

Run Heapify again since i-- now i=3 and 3,55 < 93. Swap 93 and 3

	98	85	93	74	81	3	55	70	31	14	39	27	42		

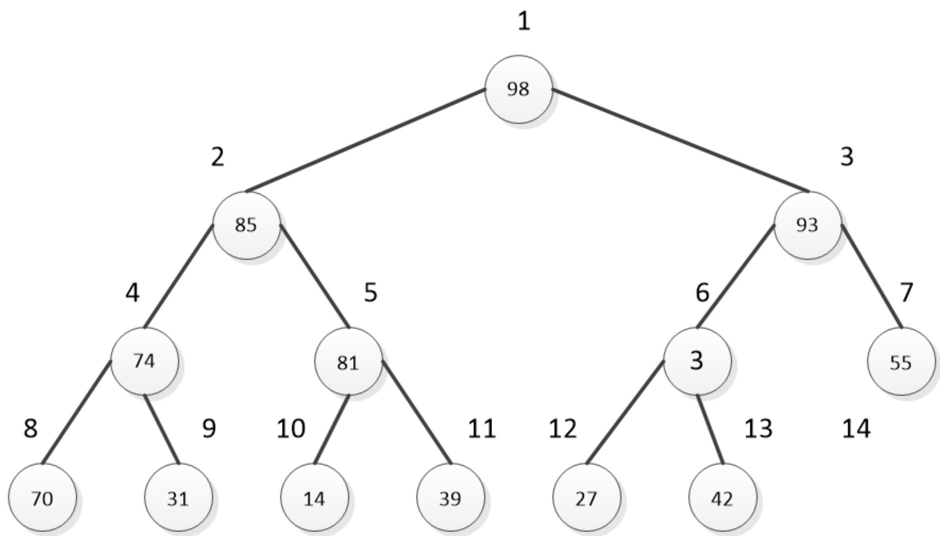
Run Heapify again since i-- now i=2 and 74,81 < 85. Do nothing

	98	85	93	74	81	3	55	70	31	14	39	27	42		

Run Heapify again since i-- now i=1 and 98 is the biggest. Do nothing

	98	85	93	74	81	3	55	70	31	14	39	27	42		

At this point, we runa function to validate if this is a maxheap. And it is not therefore, we recursive run the heapify again.



<u>3rd run:</u>

Start with i= 6 and 3, 27 < 42. Swap 3 and 42.

	98	85	93	74	81	42	55	70	31	14	39	27	3		

After that when i=5,4,3,2,1 do nothing (no swap occurs). Therefore, the final result in the heap is:

	98	85	93	74	81	42	55	70	31	14	39	27	3		

6b. Using the resultant max-heap array obtained from 6a, apply the heapsort algorithm to obtain a sorted array A in descending order. Show step-by-step in terms of the intermediate resulting arrays.

ANSWER:

We have the current max heap

	98	85	93	74	81	42	55	70	31	14	39	27	3		

I draw a tree structure on my whiteboard to help me visualize and work easier. For every time the max heap swap and restructure I re-draw the tree.

Swap the first and last node

	3	85	93	74	81	42	55	70	31	14	39	27	98		

And take 98 out of the maxheap and put it in the sorted array. Please note the one in red is marked as not existing in the maxheap but exists in the sorted array. The max heap now looks like 3, 85, 93, 74, 81, 42, 55, 70, 14, 39, 27 (1).

Restructure the max heap

	93	85	55	74	81	42	3	70	31	14	39	27	98		

Swap the first and last node, similar to (1)

	27	85	55	74	81	42	3	70	31	14	39	93	98		

Restructure the max heap

	85	81	55	74	39	42	3	70	31	14	27	93	98		

Swap the first and last node, similar to (1)

	27	81	55	74	39	42	3	70	31	14	85	93	98		

Restructure the max heap

	81	74	55	70	39	42	3	27	31	14	85	93	98		

Swap the first and last node, similar to (1)

	14	74	55	70	39	42	3	27	31	81	85	93	98		

Restructure the max heap

	74	70	55	31	39	42	3	27	14	81	85	93	98		

Swap the first and last node

	14	70	55	31	39	42	3	27	74	81	85	93	98		

Restructure the max heap, similar to (1)

	70	39	55	31	14	42	3	27	74	81	85	93	98		

Swap

	27	39	55	31	14	42	3	70	74	81	85	93	98		

Restructure the max heap, similar to (1)

	55	39	42	31	14	27	3	70	74	81	85	93	98		

Swap

	3	39	42	31	14	27	55	70	74	81	85	93	98		

Restructure the max heap, similar to (1)

	42	39	27	31	14	3	55	70	74	81	85	93	98		

Swap

	3	39	27	31	14	42	55	70	74	81	85	93	98		

Restructure the max heap, similar to (1)

	39	31	27	3	14	42	55	70	74	81	85	93	98		

Swap

	14	31	27	3	39	42	55	70	74	81	85	93	98		

Restructure the max heap, similar to (1)

	31	14	27	3	39	42	55	70	74	81	85	93	98		

Swap

	3	14	27	31	39	42	55	70	74	81	85	93	98		

Restructure the max heap, similar to (1)

	27	14	3	31	39	42	55	70	74	81	85	93	98		

Swap

	3	14	27	31	39	42	55	70	74	81	85	93	98		

Restructure the max heap, similar to (1)

	14	3	27	31	39	42	55	70	74	81	85	93	98		

Swap

	3	14	27	31	39	42	55	70	74	81	85	93	98		

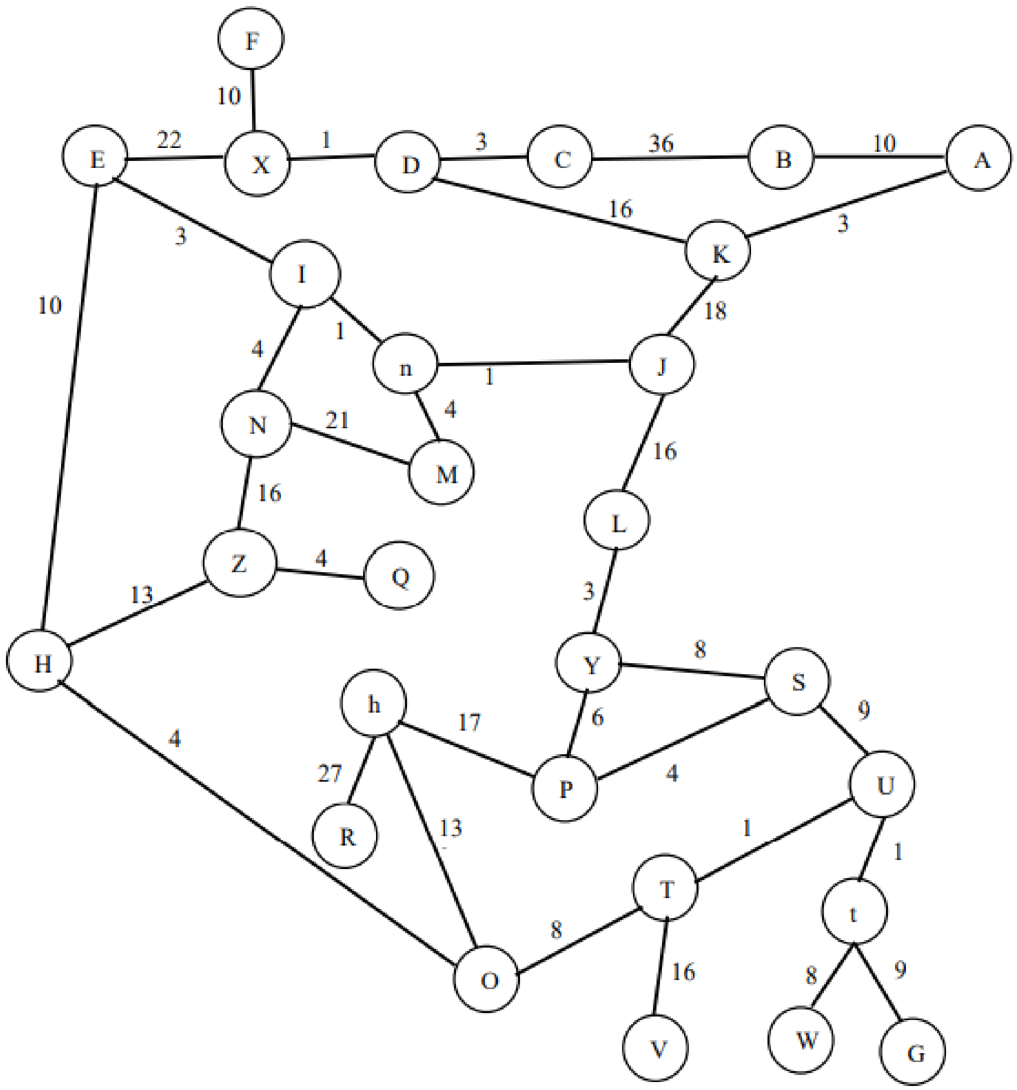
Terminate now we got an ascending order; reversing the array to get a descending order.

The result is

	98	93	85	81	74	70	55	42	39	31	27	14	3		

Problem 7 [50 points]

Given a weighted graph G, which is as follows:



7a. Construct

(i) a weighted adjacency list and

(ii) a weighted adjacency matrix

ANSWER:

Vertex List

0	F
1	E
2	X
3	D
4	C
5	B
6	A
7	I
8	K
9	N
10	n
11	J
12	M
13	L
14	Z
15	Q
16	H
17	h
18	Y
19	S
20	R
21	P
22	U
23	O
24	T
25	t
26	V
27	W
28	G

	F	E	X	D	C	B	A	I	K	N	n	J	M	L	Z	Q	H	Y	S	h	R	P	U	O	T	t	V	W	G
F	∞	∞	10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
E	∞	∞	22	∞	∞	∞	∞	3	∞	∞	∞	∞	∞	∞	∞	∞	10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
X	10	22	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
D	∞	∞	1	∞	3	∞	∞	∞	16	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
C	∞	∞	∞	3	∞	36	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
B	∞	∞	∞	∞	36	∞	10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
A	∞	∞	∞	∞	∞	10	∞	∞	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
I	∞	3	∞	∞	∞	∞	∞	∞	∞	4	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
K	∞	∞	∞	16	∞	∞	3	∞	∞	∞	∞	18	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
N	∞	∞	∞	∞	∞	∞	∞	4	∞	∞	∞	∞	21	∞	16	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
n	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞	1	4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
J	∞	∞	∞	∞	∞	∞	∞	∞	18	∞	1	∞	∞	16	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
M	∞	∞	∞	∞	∞	∞	∞	∞	∞	21	4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
L	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	16	∞	∞	∞	∞	∞	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
Z	∞	∞	∞	∞	∞	∞	∞	∞	∞	16	∞	∞	∞	∞	∞	4	13	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
Q	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
H	∞	10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	13	∞	∞	∞	∞	∞	∞	∞	∞	4	∞	∞	∞	∞	∞
Y	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	3	∞	∞	∞	∞	8	∞	∞	6	∞	∞	∞	∞	∞	∞	∞
S	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	8	∞	∞	∞	4	9	∞	∞	∞	∞	∞	∞
h	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	27	17	∞	13	∞	∞	∞	∞	∞
R	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	27	∞	∞	∞	∞	∞	∞	∞	∞	∞
P	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	6	4	17	∞	∞	∞	∞	∞	∞	∞	∞	∞
U	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	9	∞	∞	∞	∞	∞	1	1	∞	∞	∞
O	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	4	∞	∞	13	∞	∞	∞	∞	8	∞	∞	∞	∞
T	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1	8	∞	∞	16	∞	∞
t	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1	∞	∞	∞	∞	8	9
V	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	16	∞	∞	∞	∞
W	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	8	∞	∞	∞
G	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	9	∞	∞	∞
	F	E	X	D	C	B	A	I	K	N	n	J	M	L	Z	Q	H	Y	S	h	R	P	U	O	T	t	V	W	G

The *weight adjacency-matrix*

I used Microsoft Excel to create the table. The original source can be found at: Final.xlsx

F	→	X, 10				
E	→	X, 22	→	I, 3	→	H, 10
X	→	F, 10	→	E, 22	→	D, 1
D	→	X, 1	→	C, 3	→	K, 16
C	→	D, 3	→	B, 36		
B	→	C, 36	→	A, 10		
A	→	B, 10	→	K, 3		
I	→	E, 3	→	N, 4	→	n, 1
K	→	D, 16	→	A, 3	→	J, 18
N	→	I, 4	→	M, 21	→	Z, 16
n	→	I, 1	→	J, 1	→	M, 4
J	→	K, 18	→	n, 1	→	L, 16
M	→	N, 21	→	n, 4		
L	→	J, 16	→	Y, 3	→	
Z	→	N, 16	→	Q, 4	→	H, 13
Q	→	Z, 4				
H	→	E, 10	→	Z, 13	→	O, 4
Y	→	L, 3	→	S, 8	→	P, 6
S	→	Y, 8	→	P, 4	→	U, 9
h	→	R, 27	→	P, 17	→	O, 13
R	→	h, 27				
P	→	Y, 6	→	S, 4	→	h, 17
U	→	S, 9	→	T, 1	→	t, 1
O	→	H, 4	→	h, 13	→	T, 8
T	→	U, 1	→	O, 8	→	V, 16
t	→	U, 1	→	W, 8	→	G, 9
V	→	T, 16				
W	→	t, 8				
G	→	t, 9				

The *weight adjacency-list*
I used Microsoft Excel to create the table. The original source can be found at:
Final.xlsx

Traversing the given graph, based on its weighted adjacency list representation obtained in problem 7a(i), construct its depth-first search tree forest starting from vertex A. In your obtained DFS tree forest, show the tree edges (indicated as solid lines) and back edges (indicated as dotted lines) for your trees. Traversal's stack contains symbols (such as Vi, j, the first subscript number indicates the order in which a vertex V was first visited, say at i, (pushed onto the stack, V), where $0 < i \leq n$; the second one indicates the order in which it became a dead-end, say at j (popped off the stack V), where $0 < j < n$. n is the total number of vertices for the given graph. For simplicity's sake, please use two time-stamps: one is $0 < i \leq n$, the order for pushing a vertex onto the stack counting from 1 through n. The other one is $0 < j \leq n$, the order for popping off a vertex from the stack counting from 1 through n. For this problem, you need to answer 7b through 7e, which are as follows:

7b. Show the traversal's stack with time-stamp, and what are the orderings of vertices yielded by the DFS?

ANSWER:

			$W_{28,10}$	$G_{29,11}$	
		$Q_{24,7}$	$t_{27,12}$	$t_{27,12}$	
		$Z_{23,8}$	$U_{26,13}$	$U_{26,13}$	$V_{30,14}$
		$H_{22,9}$	$T_{25,15}$	$T_{25,15}$	$T_{25,15}$
	$R_{20,6}$	$O_{21,16}$	$O_{21,16}$	$O_{21,16}$	$O_{21,16}$
$A_{13,1}$	$h_{19,17}$	$h_{19,17}$	$h_{19,17}$	$h_{19,17}$	$h_{19,17}$
$B_{12,2}$	$P_{18,18}$	$P_{18,18}$	$P_{18,18}$	$P_{18,18}$	$P_{18,18}$
$C_{11,3}$	$S_{17,19}$	$S_{17,19}$	$S_{17,19}$	$S_{17,19}$	$S_{17,19}$
$D_{10,4}$	$Y_{16,20}$	$Y_{16,20}$	$Y_{16,20}$	$Y_{16,20}$	$Y_{16,20}$
$K_{9,5}$	$L_{15,21}$	$L_{15,21}$	$L_{15,21}$	$L_{15,21}$	$L_{15,21}$
$J_{8,22}$	$J_{8,22}$	$J_{8,22}$	$J_{8,22}$	$J_{8,22}$	$J_{8,22}$
$n_{7,23}$	$n_{7,23}$	$n_{7,23}$	$n_{7,23}$	$n_{7,23}$	$n_{7,23}$
$M_{6,24}$	$M_{6,24}$	$M_{6,24}$	$M_{6,24}$	$M_{6,24}$	$M_{6,24}$
$N_{5,25}$	$N_{5,25}$	$N_{5,25}$	$N_{5,25}$	$N_{5,25}$	$N_{5,25}$
$I_{4,26}$	$I_{4,26}$	$I_{4,26}$	$I_{4,26}$	$I_{4,26}$	$I_{4,26}$
$E_{3,27}$	$E_{3,27}$	$E_{3,27}$	$E_{3,27}$	$E_{3,27}$	$E_{3,27}$
$X_{2,28}$	$X_{2,28}$	$X_{2,28}$	$X_{2,28}$	$X_{2,28}$	$X_{2,28}$
$F_{1,29}$	$F_{1,29}$	$F_{1,29}$	$F_{1,29}$	$F_{1,29}$	$F_{1,29}$

Ordering of verticles yield by DFS:
A, B, C, D, K, R, Q, Z, H, W, G, t, U, V, T, O, h, P, S, Y, L, J, n, M, N, I, E, X, F

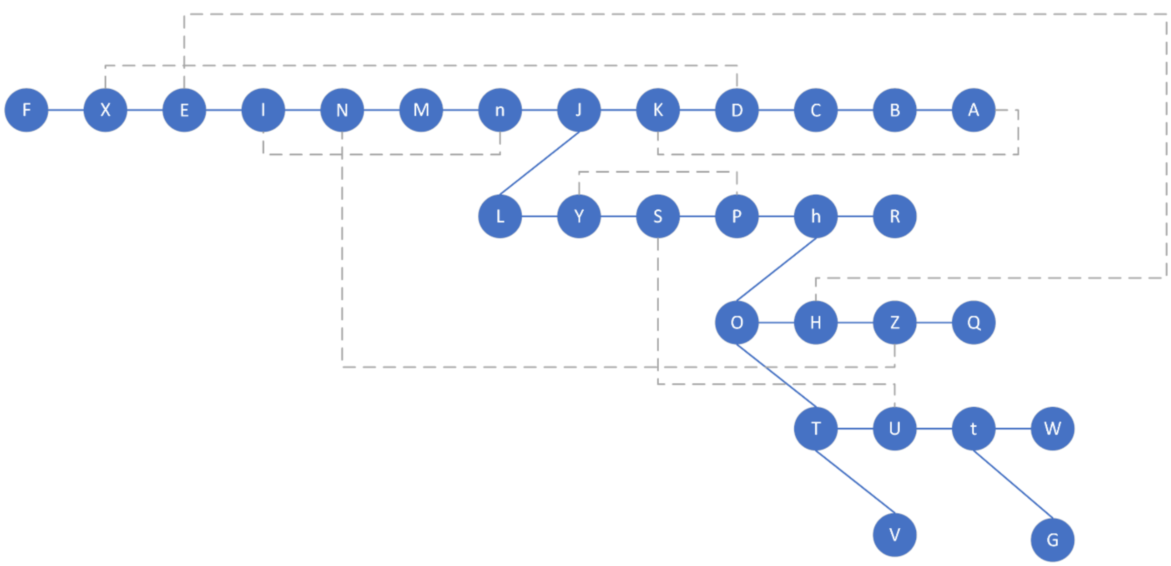
7c. Construct the corresponding depth-first search (DFS) tree forest, with indications of tree edges and back edges.

ANSWER:

I am using Microsoft Visio to construct the DFS tree. Link is here visio-link

Based on the traversal's stack with the time-stamp table, I constructed depth-first search (DFS) tree forest using the following steps:

- At the top level, I have F, X, E, I, N, M, n, J, K, D, C, B, and A are connected by edges
- Since we pop A, B, C, D, K. I connect J to L by an edge
- The next level will be L, Y, S, P, h, and R are connected by edges
- Since we pop R. I connect h to O
- The next level will be O, H, Z, and Q are connected by edges
- Since we will pop Q, Z, H. I connect O to T
- The next level will be T, U, t, and W are connected by edges
- Since we will pop W. I connect G to t, and T to V
- That is finish all the edges of the DFS tree forest
- Now I look at the weight adjacency list to construct the back edges
- Please note **tree edges** are indicated as blue solid lines and **back edges** are indicated as dotted lines



7d. What is the graph called? Is this graph acyclic? Does the graph have articulation points? What is the topological sort ordering for the graph?

ANSWER:

- The graph is called the depth-first forest
- This graph is not acyclic since there are back-edges from some vertex to its ancestor (e.g. X is connected to D via back-edge, E is connected to H via back-edge…)
- The topological sort ordering for the graph is the reverse of pop-off ordering: {F, X, E, I, N, M, n, J, L, Y, S, P, h, O, T, V, U, t, G, W, H, Z, Q, R, K, D, C, B, A}
- Yes, the graph has articulation points.

7e. What are the time efficiency and space efficiency of the DFS?

ANSWER:

- Time efficiency is a linear-time procedure which means running time increases at most linearly with the size of the input.
- Space efficiency is the total space that needs to store all the data structure below in computer memory:
  - a data structure to store graph G (vertex and edges)
  - a stack data structure (list implementation) to store vertex V of graph G. Length of the stack should be equal to the total vertex in Graph G
  - a global variable count to use for timestamp

Problem 8 [40 points]

Traversing the graph given in Problem 7, based on its weighted adjacency list representation obtained in Problem 7a(i), construct its breath-first search (BFS) tree forest starting from vertex A. For this, you need to use a queue (note the difference from DFS) to trace the operation of breadth-first search, indicating the order in which the vertices {…, V’, V”, …} were visited. i.e., the order of the operation of adding several vertices to, or removing a vertex from the queue $\{V_i^{''}, \ldots, V'_{i+1}, V'_{i+2}, \ldots\}$. The order in which vertices are added to the queue (i.e., enqueue operation) is the same order in which they are removed from it (i.e., dequeue operation). Indicate the tree edges (indicated as solid lines) and cross-edges (indicated as dotted lines) for your trees. For this problem, you need to answer 8a through 7e, which are as follows:

8a. Show the traversal’s queue with a time-stamp indicating the order in which the vertices were visited, and what is the ordering of vertices yielded by the BFS?

ANSWER:

FIFO QUEUE	VISITED NODE	ORDER OF VERTICLES
∞		
A Enqueue : A Dequeue:	A’→B→K	
B K Enqueue : B, K Dequeue: A	A’’→ B →K B’ → C →A K’ → D →A →J	A
K C Enqueue : C Dequeue: B	B’’ → C → A K’ → D → A →J C’ → D → B	A B
C D J Enqueue : D, J Dequeue: K	K’’ → D → A →J C’ → D → B D’ → X → C → K J’ → K → N → L	A B K
D J X n L Enqueue : X, N, L Dequeue: C	C’’ → D → B D’ → X → C → K J’ → K → N → L X’→ F → E → D n’→ I → J → M L’→ J → Y	A B K C
J X n L Enqueue : Dequeue: D	D’’ → X → C → K J’ → K → N → L X’→ F → E → D n’→ I → J → M L’→ J → Y	A B K C D

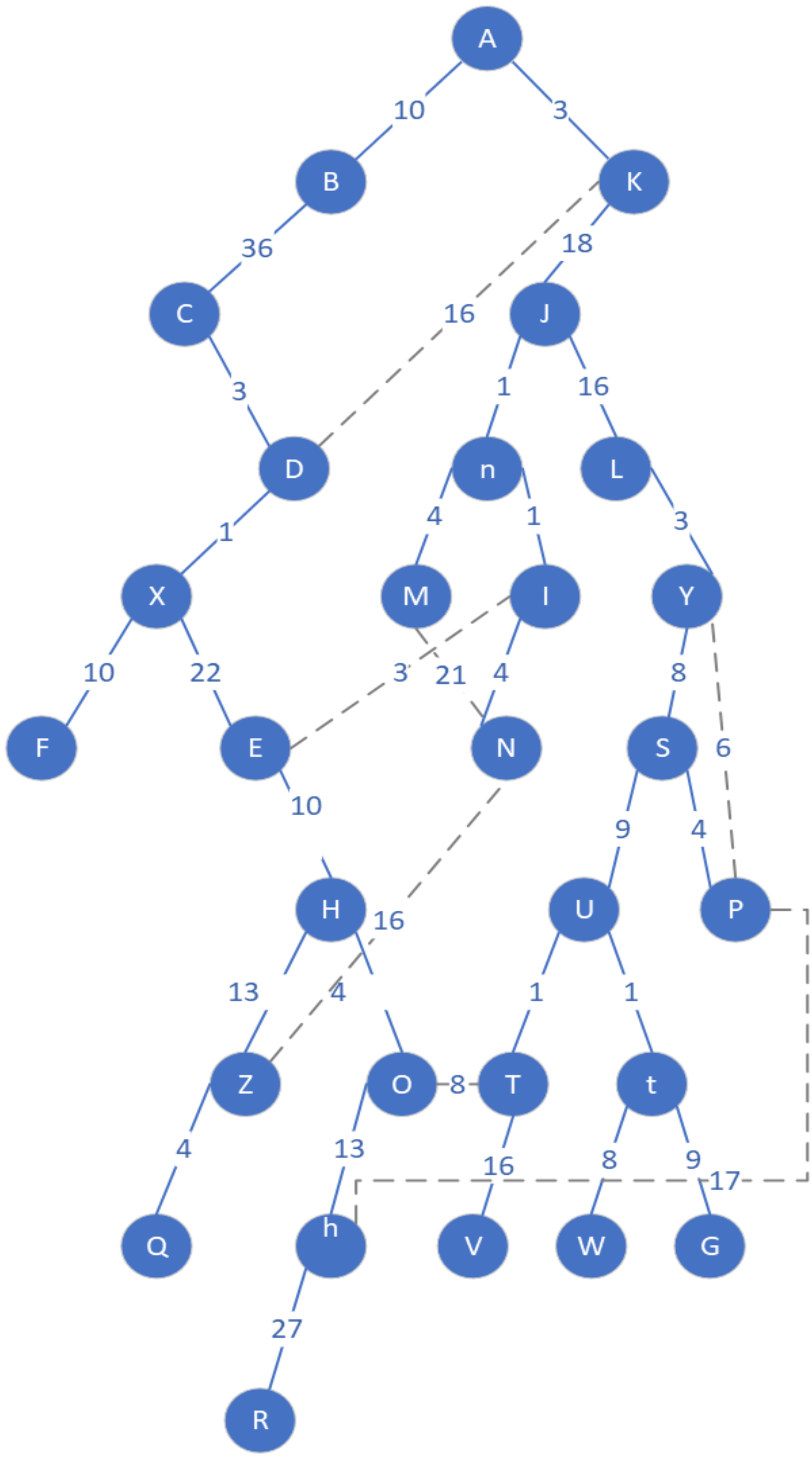
X n L Enqueue : Dequeue: J	J’’ → K → N → L X’→ F → E → D n’→ I → J → M L’→ J → Y	A B K C D J
n L F E Enqueue : F E Dequeue: X	X’’→ F → E → D n’→ I → J → M L’→ J → Y F’ → X E’ → X→ I → H	A B K C D J X
n L F E I M Enqueue : I, M Dequeue: n	n’’→ I → J → M L’→ J → Y F’ → X E’ → X→ I → H I’ → E → N → n M’ → N→ n	A B K C D J X n
F E I M Y Enqueue : Y Dequeue: L	L’’→ J → Y F’ → X E’ → X→ I → H I’ → E → N → n M’ → N→ n Y’ → L→ S → P	A B K C D J X n L
E I M Y Enqueue : Dequeue: F	F’’ → X E’ → X→ I → H I’ → E → N → n M’ → N→ n Y’ → L→ S → P	A B K C D J X n L F
I M Y H Enqueue : H Dequeue: E	E’ → X→ I → H I’ → E → N → n M’ → N→ n Y’ → L→ S → P H’ → E→ Z → O	A B K C D J X n L F E
M Y H N Enqueue : N Dequeue: I	I’’ → E → N → n M’ → N→ n Y’ → L→ S → P H’ → E→ Z → O N’ → I→ M → Z	A B K C D J X n L F E I
Y H N Enqueue : Dequeue: M	M’’ → N→ n Y’ → L→ S → P H’ → E→ Z → O N’ → I→ M → Z	A B K C D J X n L F E I M
H N S P Enqueue : S, P Dequeue: Y	Y’’ → L→ S → P H’ → E→ Z → O N’ → I→ M → Z S’ → Y→ P → U	A B K C D J X n L F E I M Y

	P’ → Y→ S → H	
N S P Z O Enqueue : Z, O Dequeue: H	H’’ → E→ Z → O N’ → I→ M → Z S’ → Y→ P → U P’ → Y→ S → H Z’ → N→ Q → H O’→ H→ h → T	A B K C D J X n L F E I M Y H
S P Z O Enqueue : Dequeue: N	N’’ → I→ M → Z S’ → Y→ P → U P’ → Y→ S → H Z’ → N→ Q → H O’→ H→ h → T	A B K C D J X n L F E I M Y H N
P Z O U Enqueue : U Dequeue: S	S’’ → Y→ P → U P’ → Y→ S → H Z’ → N→ Q → H O’→ H→ h → T U’ → S→ T → t	A B K C D J X n L F E I M Y H N S
Z O U Enqueue : Dequeue: P	P’’ → Y→ S → H Z’ → N→ Q → H O’→ H→ h → T U’ → S→ T → t	A B K C D J X n L F E I M Y H N S P
O U Q Enqueue : Q Dequeue: Z	Z’’ → N→ Q → H O’→ H→ h → T U’ → S→ T → t Q’→ Z	A B K C D J X n L F E I M Y H N S P Z
U Q h T Enqueue : h, T Dequeue: O	O’’→ H→ h → T U’ → S→ T → t Q’→ Z h→ R→ P → O T→ U→ O → V	A B K C D J X n L F E I M Y H N S P Z O
Q h T t Enqueue : t Dequeue: U	U’’ → S→ T → t Q’→ Z h’→ R→ P → O T’→ U→ O → V t’→ U→ W → G	A B K C D J X n L F E I M Y H N S P Z O U
h T t Enqueue : Dequeue: Q	Q’’→ Z h’→ R→ P → O T’→ U→ O → V t’→ U→ W → G	A B K C D J X n L F E I M Y H N S P Z O U Q
T t R Enqueue : R Dequeue: h	h’’→ R→ P → O T’→ U→ O → V t’→ U→ W → G R’→ h	A B K C D J X n L F E I M Y H N S P Z O U Q h

t R V Enqueue : V Dequeue: T	T''→ U→ O → V t'→ U→ W → G R'→ h V'→ T	A B K C D J X n L F E I M Y H N S P Z O U Q h T
t R V Enqueue : V Dequeue: T	T''→ U→ O → V t'→ U→ W → G R'→ h V'→ T	A B K C D J X n L F E I M Y H N S P Z O U Q h T
R V W G Enqueue : W, G Dequeue: t	t''→ U→ W → G R'→ h V'→ T W' → t G' → t	A B K C D J X n L F E I M Y H N S P Z O U Q h T t
V W G Enqueue : Dequeue: R	R''→ h V'→ T W' → t G' → t	A B K C D J X n L F E I M Y H N S P Z O U Q h T t R
W G Enqueue : Dequeue: V	V''→ T W' → t G' → t	A B K C D J X n L F E I M Y H N S P Z O U Q h T t R V
G Enqueue : Dequeue: W	W'' → t G' → t	A B K C D J X n L F E I M Y H N S P Z O U Q h T t R V W
∞ Enqueue : Dequeue: G	G'' → t	A B K C D J X n L F E I M Y H N S P Z O U Q h T t R V W G

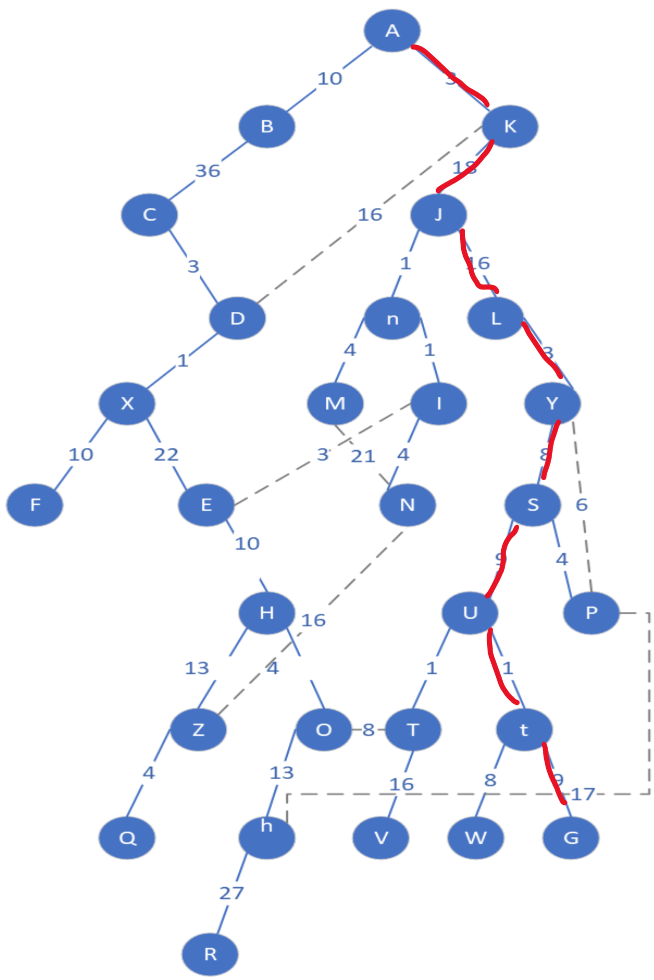
8b. Construct the corresponding breadth-first search (BFS) tree forest, with an indication of tree edges and cross edges in addition to back edges and forward edges)

ANSWER:



8c. From the obtained BFS tree forest, compute the shortest distance (smallest number of edges) from A to vertex G.

ANSWER:



From vertex A to vertex G. The traversal order should be A, K, J, L, Y, S, U, t, G (Visual it from the tree structure).

The shortest distance is the sum of AK + KJ + JL + YS + SU + Ut +tG = 3 + 18 + 16 + 3 + 8 + 9 + 1 + 9 = <u>67</u>

8d. What are the time efficiency and space efficiency of the BFS?

ANSWER:

Analyze the running time of an input graph G = (V, E):

- the total time spent in scanning adjacency lists is O(|E|)
- the total time devoted to queue operation is O(|V|)
- Therefore, the BFS run is linear-time in the size of the adjacency-list representation of G. the total running time of the BFS procedure is O(|V| + |E|).

Space efficiency is the total space that needs to store all the data structures below in computer memory:

- a data structure to store graph G (vertex and edges) (dictionary implementation)
- The color[u] to store the color of each vertex u in V (list implementation)
- The attribute π[u] is the predecessor of vertex u in V (list implementation)
- The attribute d[u] holds the distance from the source s to vertex u computed by the algorithm
- The FIFO queue data structures that containing vertex s.

Problem 9 [30 points]

From the given graph in Problem 7, given a source vertex A, use Prim's algorithm to find the minimum spanning tree for the graph. For each step, state your tree vertices VT and the remaining vertices V - VT. More importantly, you need to give a table stating the tree vertices and remaining vertices with their weights (i.e., the corresponding edges with their weights). You do not have to give the intermediate graph as an "Illustration", but you show the final minimum spanning tree (via highlight edges) within the graph given in problem 7.

9a. Compute the minimum spanning tree of the graph given in Problem 7.

Tree Vertices	Remaining Vertices	VT	V - VT
A(-, -)	B(A, 10), K(A, 3), ?(-, ∞)	{A}	{ B, K, ?}
	. . .		

where VT = { A } and

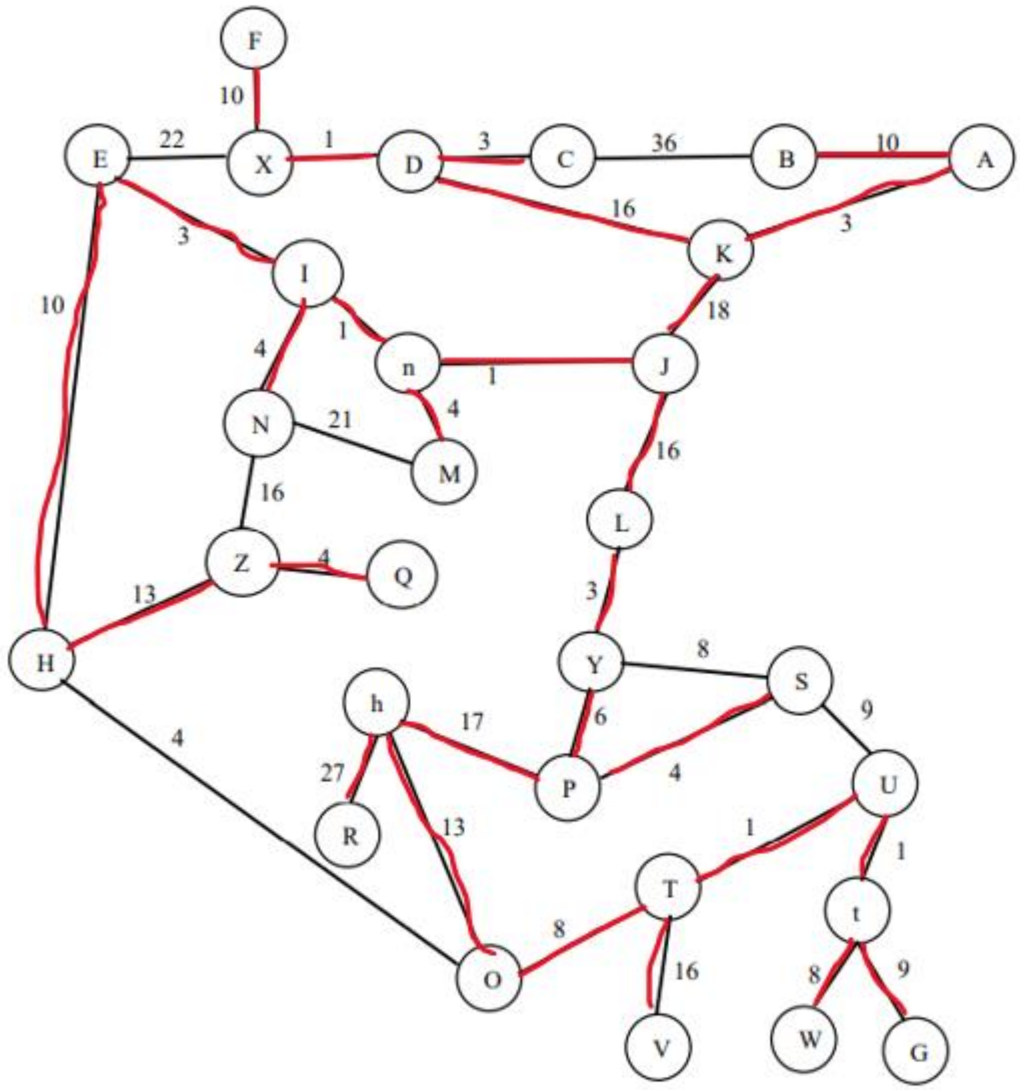
V - VT = { B, K, ?}, where "?" is to denote any vertex in the graph, which is not adjacent to A.

If you wish, use the symbol “?” to denote all the vertices in the graph which are not adjacent to every vertex in VT.

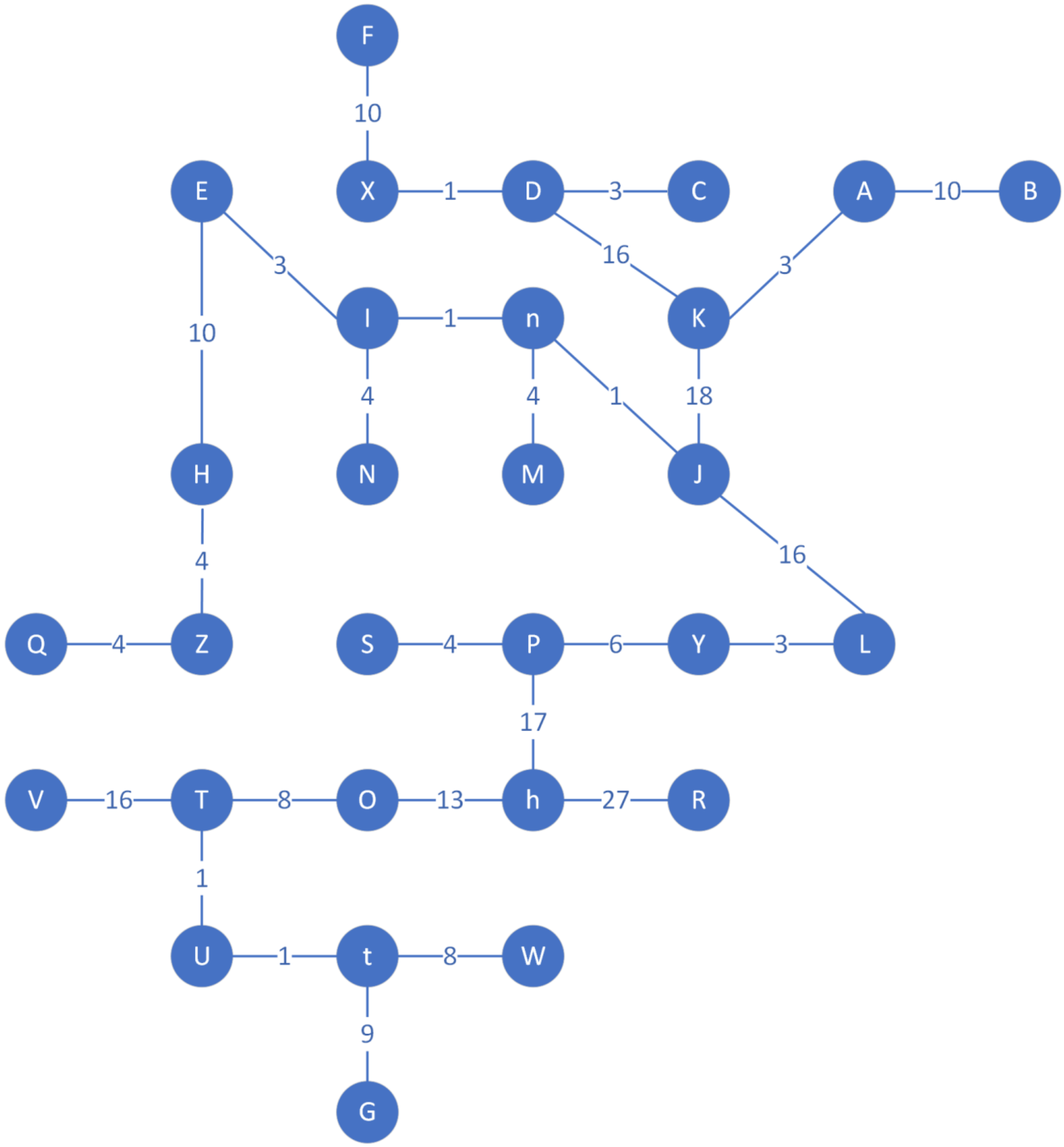
ANSWER:

Tree Vertices	Remaining Vertices (applied min-heap)	V_T	V - V_T
F(-, -)	X(F,10), ?(-,∞)	{F}	{ X, ?}
X(F,10)	D(X,1), E(X,22), ?(-,∞)	{F, X}	{D, E, ?}
D(X,1)	C(D,3), K(D,16), E(X,22), ?(-,∞)	{F, X, D}	{C, K, E, ?}
C(D,3)	K(D,16), E(X,22), B(C, 36), ?(-,∞)	{F, X, D, C}	{K, E, B, ?}
K(D,16)	A(K,3), J(K,18), E(X,22), B(C, 36), ?(-,∞)	{F, X, D, C, K}	{A, J, E, B, ?}
A(K,3)	B(A,10), J(K,18), E(X,22), B(C, 36), ?(-,∞)	{F, X, D, C, K, A}	{B, J, E, ?}
B(A,10)	J(K,18), E(X,22), ?(-,∞)	{F, X, D, C, K, A, B}	{J, E, ?}
J(K,18)	n(J,1), L(J,16), E(X,22), ?(-,∞)	{F, X, D, C, K, A, B, J}	{n, L, E, ?}
n(J,1)	I(n,1), M(n,4), L(J,16), E(X,22), ?(-,∞)	{F, X, D, C, K, A, B, J, n}	{I, M, L, E, ?}
I(n,1)	E(I,3), N(I,4), M(n,4), L(J,16), E(X,22), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I}	{ E, N, M, L, ?}
E(I,3)	N(I,4), M(n,4), H(E,10), L(J,16), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E }	{N, M, H, L, ?}
N(I,4)	M(n,4), H(E,10), L(J,16), Z(N,16), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N }	{ M, H, L, Z, ?}
M(n,4)	H(E,10), L(J,16), Z(N,16), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M }	{H, L, Z, ?}
H(E,10)	Z(H,13), L(J,16), Z(N,16), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H}	{Z, L ?}
Z(H,13)	Q(Z,4), L(J,16), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z}	{Q, L ?}
Q(Z,4)	L(J,16), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q}	{L, ?}
L(J,16)	Y(L,3), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L}	{Y, ?}
Y(L,3)	P(Y,6), S(Y,8), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y }	{P, S ?}

P(Y,6)	S(P,4), h(P,17), S(Y,8), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y, P}	{S, h ?}
S(P,4)	h(P,17), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y, P, S}	{h, ?}
h(P,17)	O(h,13), R(h,27), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y, P, S, h}	{O, R, ?}
O(h,13)	T(O,8), R(h,27), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y, P, S, h, O}	{T, R, ?}
T(O,8)	U(T,1), V(T,16), R(h,27), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y, P, S, h, O, T}	{U, V, R, ?}
U(T,1)	t(U,1), V(T,16), R(h,27), ?(-,∞)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y, P, S, h, O, T, U}	{t, V, R, ?}
t(U,1)	W(t,8), G(t,9), V(T,16), R(h,27)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y, P, S, h, O, T, U, t}	{W, G, V, R}
W(t,8)	G(t,9), V(T,16), R(h,27)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y, P, S, h, O, T, U, t, W}	{G, V, R}
G(t,9)	V(T,16), R(h,27)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y, P, S, h, O, T, U, t, W, G}	{V, R}
V(T,16)	R(h,27)	{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y, P, S, h, O, T, U, t, W, G, V}	{R}
R(h,27)		{F, X, D, C, K, A, B, J, n, I, E, N, M, H, Z, Q, L, Y, P, S, h, O, T, U, t, W, G, V, R}	{}

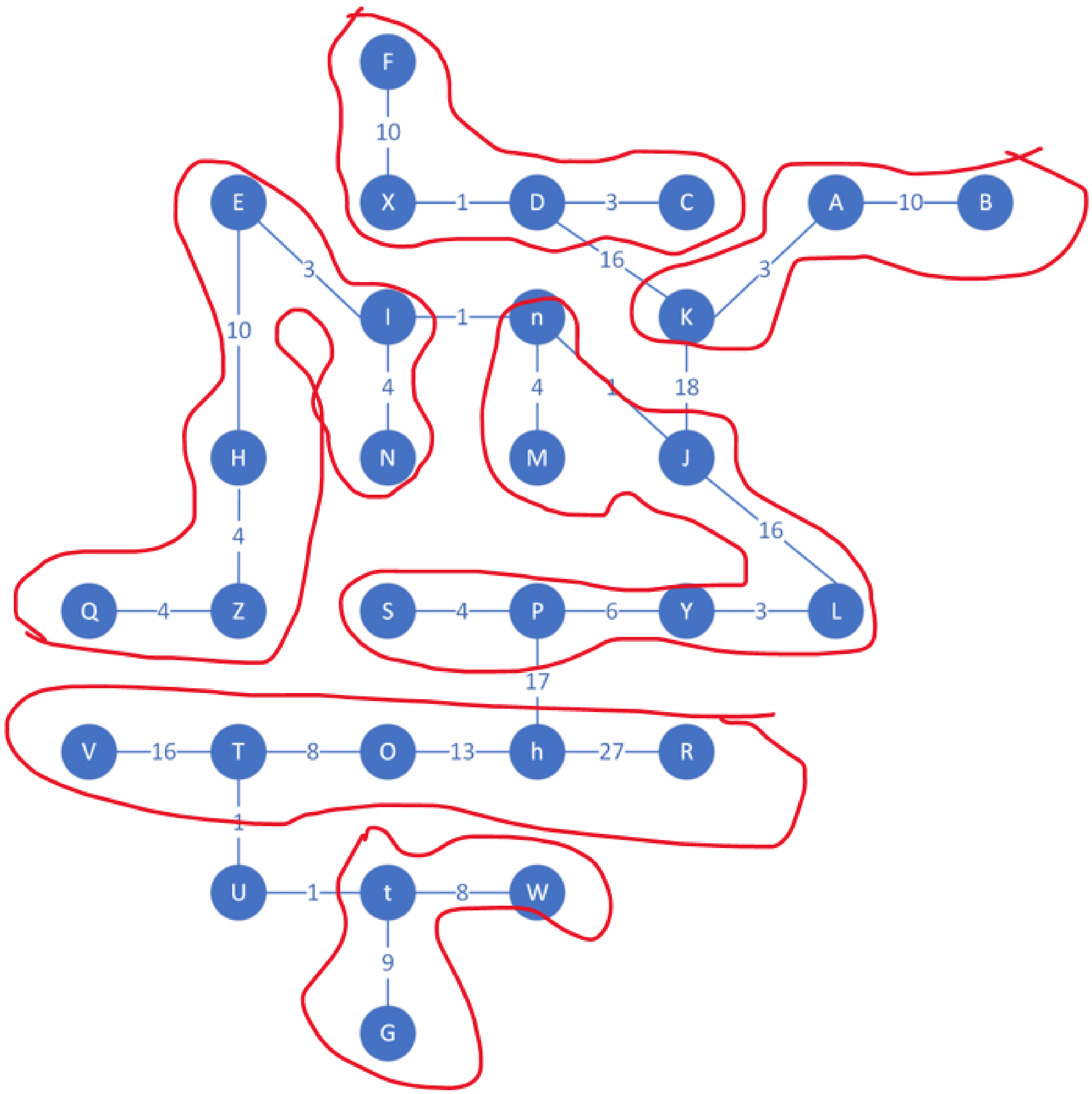


Here is a nice diagram of the above using visual studio [Drawing.vsdx]()



9b. What is your obtained minimum spanning tree with their total weights of branches for the graph given in problem 7? [Highlight the obtained (from 9a) minimum spanning tree in the given graph of Problem 7. For example, the branch of A, K, D, X, F has a weight Secondly, compute the total weight of each branch of the minimum spanning tree. Thirdly, compute the grand total weight of the obtained minimum spanning tree.]

ANSWER:



- The grand total minimum weight of the spanning tree is: 230

Branch	Weight
F, X, D, C	14
K, A, B	13
Q, Z, H, E, I, N	25
M, n, J, L, Y, P, S	34

V, T, O, h, R	64
G, t, W	17

9c. From your obtained minimum spanning tree, what is the minimum distance from vertex A to vertex G?

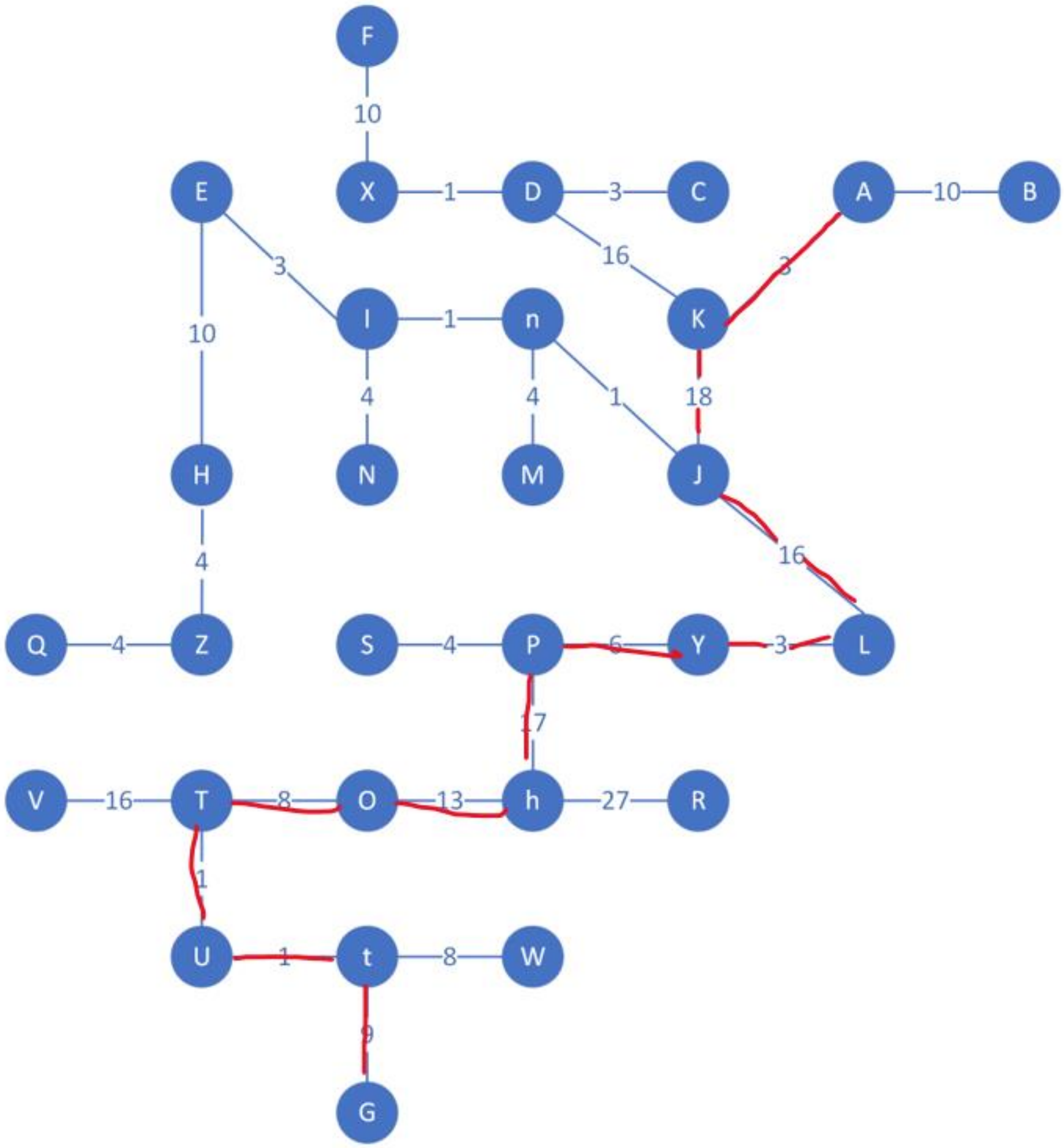
ANSWER:

The shortest path from vertex A to vertex G is following through the below vertex in the following order:

A, K, J, L, Y, P, h, O, T, U, t, G

The minimum distance is the sum of each edge between these vertexes. It is 95

Please refer to the below diagram for verification



Note: Good handwriting is required if you provide your answer in your handwriting. Proper numbering of your answer to each problem is strictly required. The problem's solution must be orderly given. (10 points off if not)
