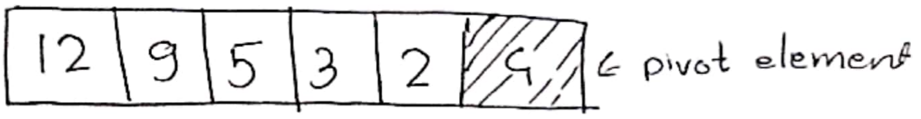
Sumit Manandhar

1 Short an array A = [12, 9, 5, 3, 2, 4] using quickshort.

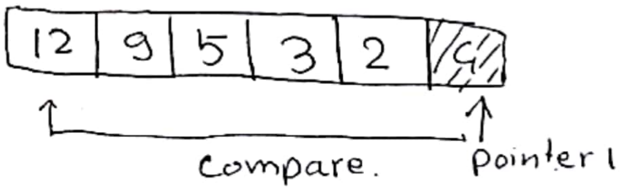
Given array A = [12, 9, 5, 3, 2, 4]

Step1: Selection of pivot element:
We will select right most element as pivot element.

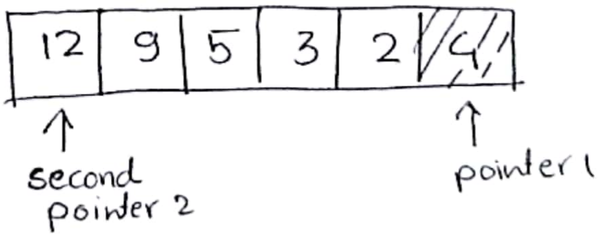


Step2: Rearrangement of an array

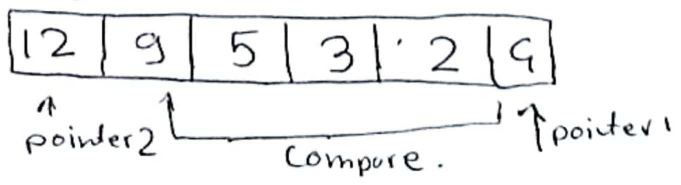
a) a ~~m~~ pointer is fixed at pivot element. The pivot element is then compared with elements beginning from first index.



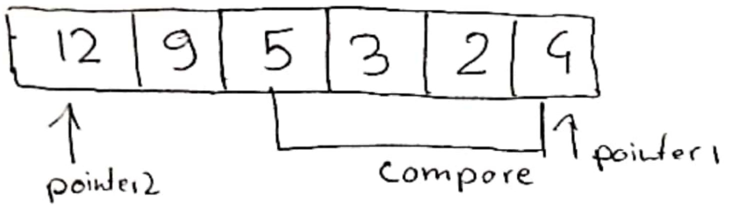
b) If element is greater than pivot element, Second pointer is placed for that element



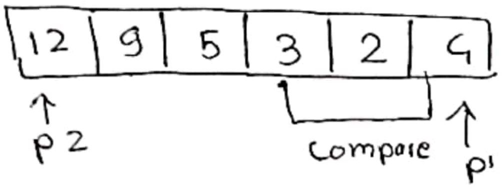
c) pivot is compared with other elements. If an element is smaller than pivot element, it is swapped with greater element ie with pointer 2



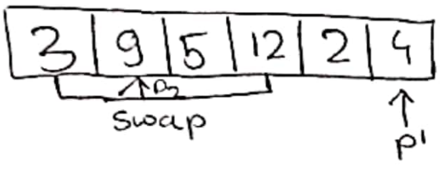
ii)



iii)

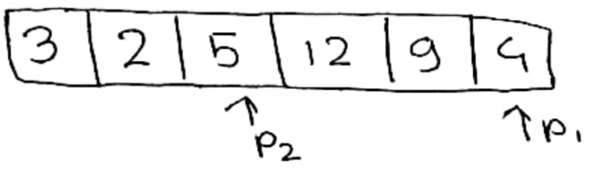


iv)

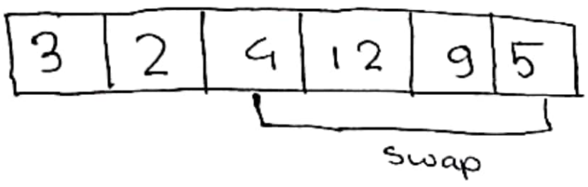


(P 2 is moved to next element after swapping)

d) Now, again step c) is repeated until second last element is reached. i.e.



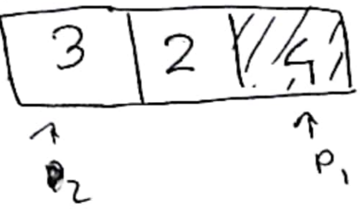
e) Now, pivot element is swapped with second pointer

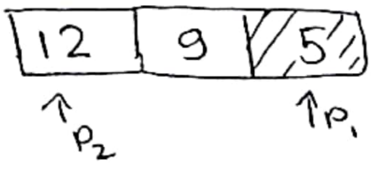


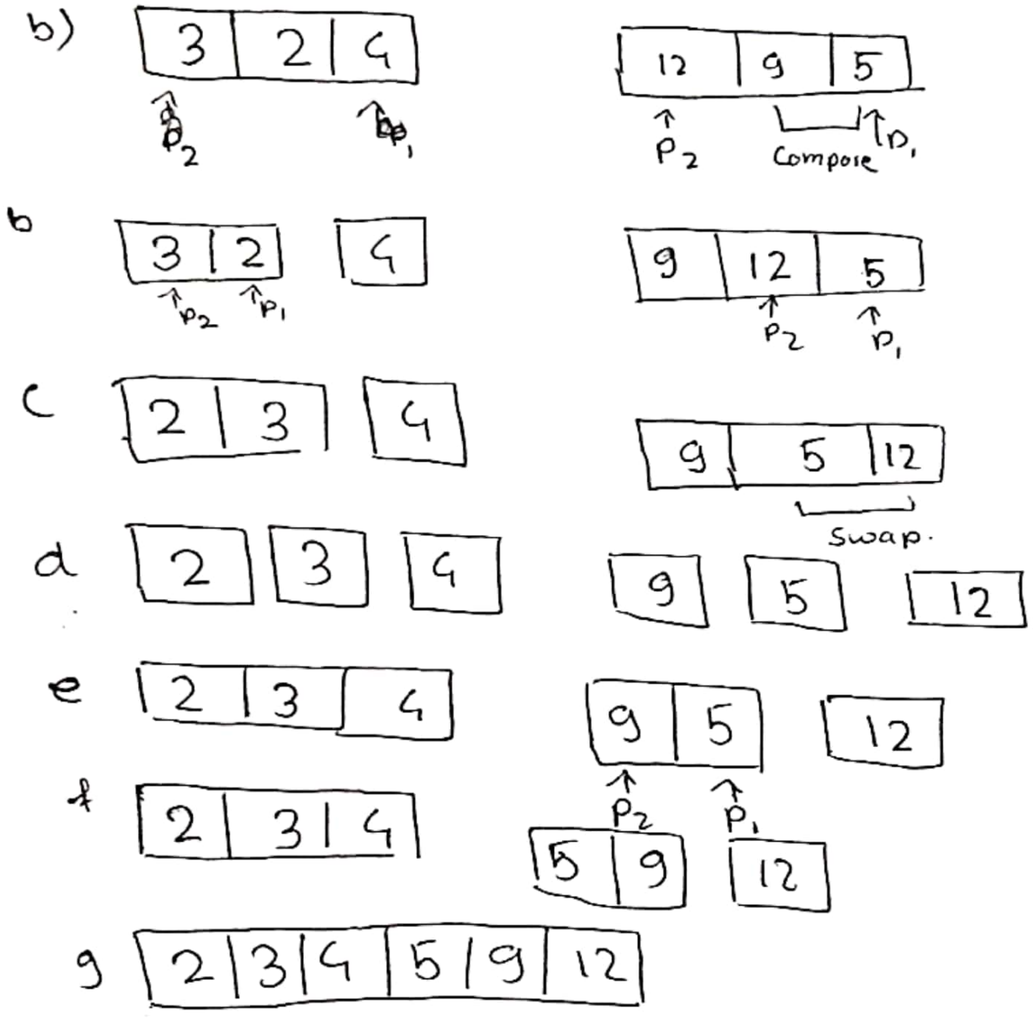
~~f)~~

Step 3 : Now array is divided into sub-array and pivot element is selected for both array as in step1. and process is repeated as step 2 ~~unti~~ and. step 3 until array is sorted

a)







The sorted array. [2, 3, 4, 5, 9, 12]

2 Explain different type of asymptotic notation used in analyzing algorithm

Asymptotic notation is the mathematical representation of algorithm complexity. It is used to described the running time of an algorithm ie how much time an algorithm takes with a given input. There are three types of asymptotic notation used in analyzing algorithm. ie :-

i) Big-oh (O)
ii) Big-omega (Ω)
iii) Big-Theta (ϕ)

i Big-oh (O) :-

This notation describes upper limit or worst case. of running algorithm. It can be calculated by counting how many algorithm it take in worst-case scenario with input 'n'. like we have. O(n) for ~~for~~ a loop.

$f(n) = O(g(n))$ if there exist a positive constant c and non negative integer no. Such that.

$$f(n) \leq cg(n)$$

then, g(n) is said to upper bound of n)

8 ii. Big- omega(-Ω)

It describes the best running time of a program or lower limit or ~~bai~~ best case for an algorithm. We compute big-Ω by counting how many iteration it will take for best case scenario. Like example Big-Ω for shorting will be Ω(1) if the array is already shorted.

iii. Big-Theta (Φ):

It describes enclosed function between upper limit ~~or~~ and lower limit. It describes about average time complexity. ie Big-Φ of selection short ~~hr~~ is N/2.

3 ~~RA~~ Describe RAM model of computation

The Random Access Machine Model of computation measures the runtime of an algorithm by summing up the number of steps needed to execute the algorithm of an group of data. For RAM to exist, it has following assumptions:

i) Each registers holds an integer
ii) Program cannot modify itself
iii) Memory instructions are simple arithmetic operations ie (addition, substruction, multiplication, division and control state (got, if then, etc))

The principles involved in ~~Ra~~ operation of RAM model are described as following:

i) Basic logical or arithmetic operations are considered to be simple operation that takes one time steps

ii) Loops and subroutines are complex operation that take multiple time steps

iii) All memory access takes exactly one time step

Eg: Pseudo code cost

Require: Integer array A of length n.

```
S = 0                          O(1) ~~Θ(~~
for i=0, i< n , i++            n time
    S = S+A[i]                 O(1)
return. S                      O(1)
```

Runtime: $O(1) + n.(O(1) + O(1)) = O(n)$

4 Compress the word 'improvement' using Huffman coding

Innitial string:

I	M	P	R	O	V	E	M	E	N	T

Step 1: Calculating the frequency of each charater in the string

1	2	1	1	1	1	2	1	1
I	M	P	R	O	V	E	N	T

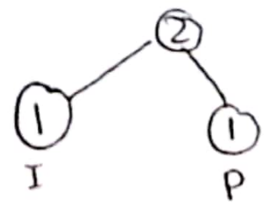
Step 2: Shorting in increasing order

$\overset{1}{I}\,\overset{1}{P}\,\overset{1}{R}\,\overset{1}{O}\,\overset{1}{V}\,\overset{1}{N}\,\overset{1}{T}\ \overset{2}{M}\ \overset{2}{E}$

Step 3: Making each character as unique node:

Step 4: Creating an empty node and assigning the minimum frequency to left child of 2 and assigning second minimum to right. Then setting value of theose sum to empty nodo:

2	1	1	1	1	1	2	2
*	R	O	V	N	T	M	E



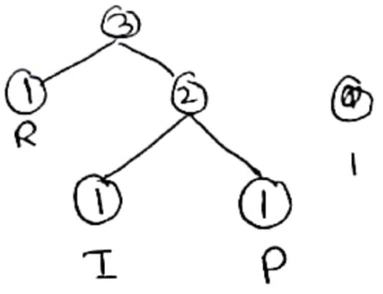
Scanned with CamScanner

Step 5 : Remove these frequency and add the sum into the queue.

Step 6: Repeate these ~~3~~ 2 to 5 steps

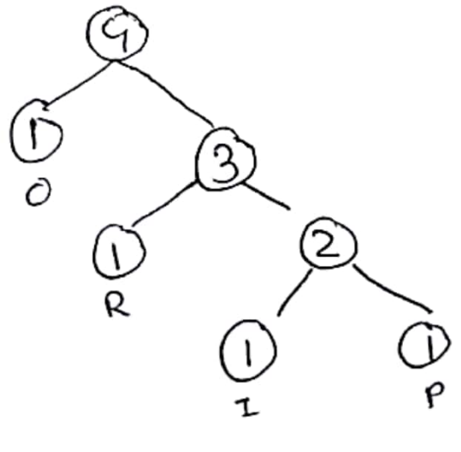
a)

1	1	1	1	1	1	2	2	2
R	O	V	N	T	M	E	*	



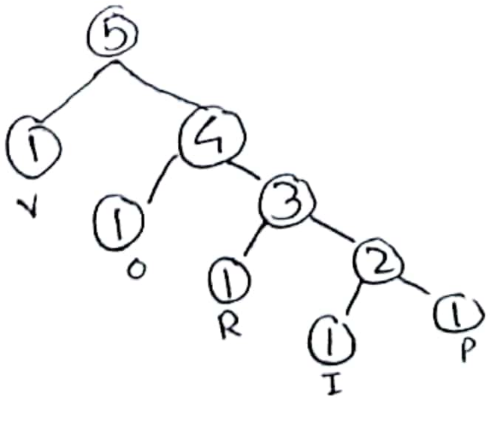
b)

1	1	1	1	2	2	3
O	V	N	T	M	E	*

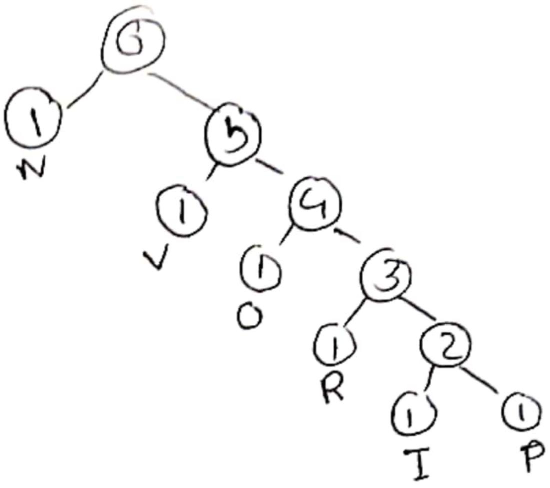


c)

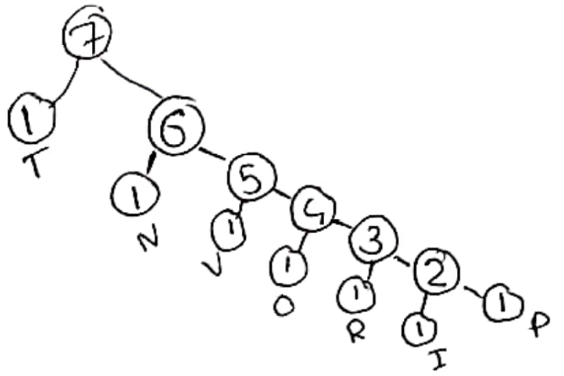
1	1	1	2	2	4
V	N	T	M	E	*



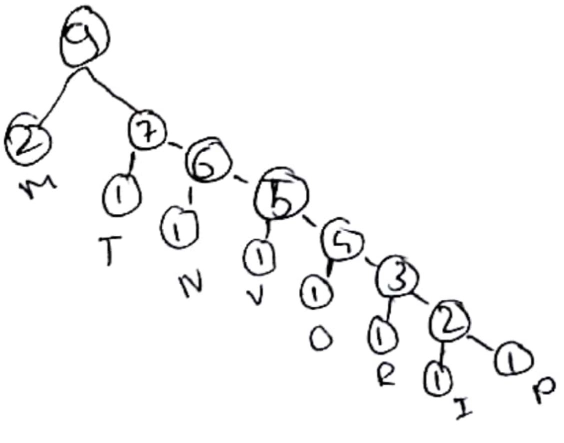
d) $\overset{1}{N}$ $\overset{1}{T}$ $\overset{2}{M}$ $\overset{2}{E}$ $\overset{5}{*}$



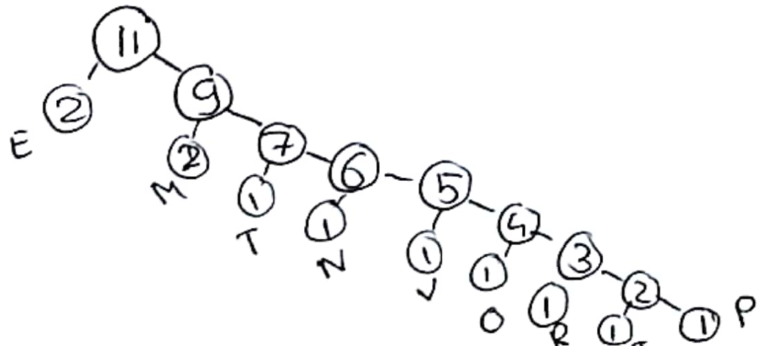
e) $\overset{1}{T}$ $\overset{2}{M}$ $\overset{2}{E}$ $\overset{6}{*}$



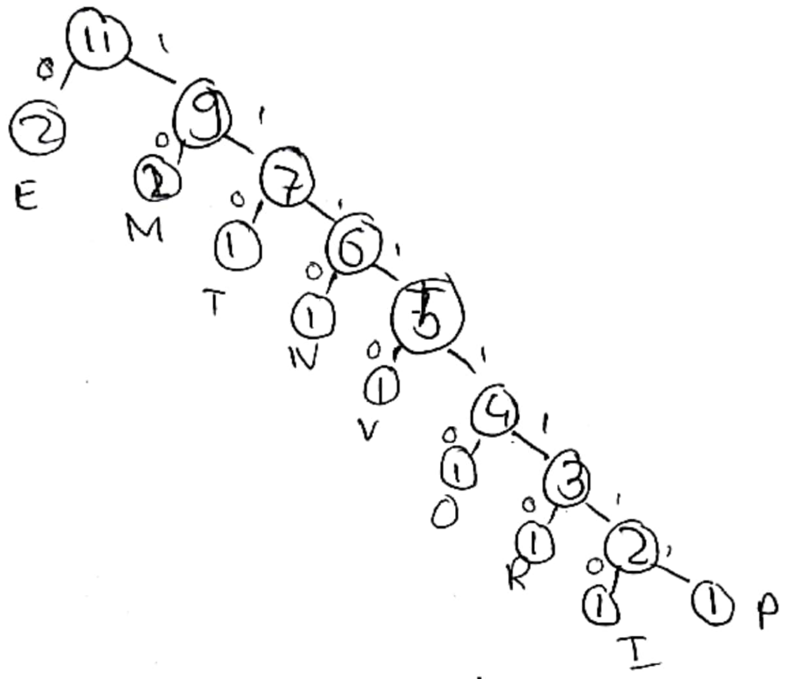
f) $\overset{2}{M}$ $\overset{2}{E}$ $\overset{6}{*}$



g $\overset{2}{E}$ $\overset{9}{*}$



Step 8 : assign 0 to left edge, 1 to right edge
11
*



The Total size is given by.

Character	Frequency.	Code	size.
I	1	11111110	1×8 = 8
M	2	10	2×2 = 4
P	1	11111111	1×8 = 8
R	1	1111110	1×7 = 7
O	1	111110	1×6 = 6
V	1	11110	1×5 = 5
N	1	1110	1×4 = 4
T	1	110	1×3 = 3
E	2	0	2×1 = 2

Sumit Manandhar

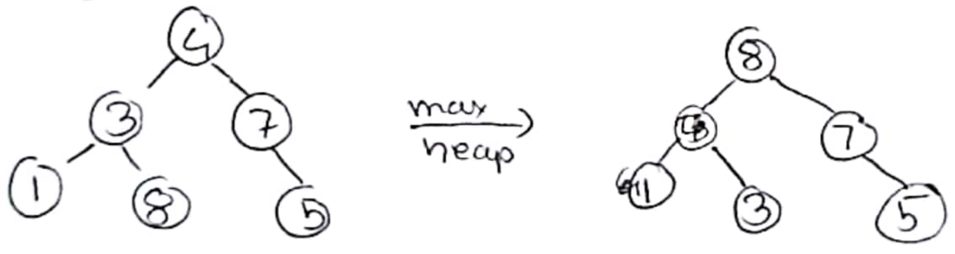
5 Explain Heap sort with example

:) Heap sort is a process of sorting based on binary heap data structure. It is similar to selection sort as we first find minimum element and place the minimum element at the beginning. So, firstly we build a max heap from the inserted data where largest data is at root. Then swap with past item of heap followed by reduced size of heap by 1

Eg:

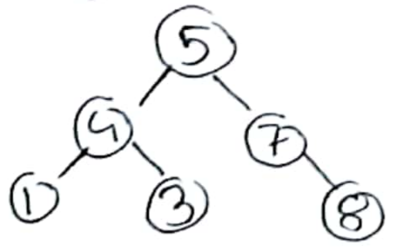
Array : [4, 3, 7, 1, 8, 5]

1 Building heap tree:



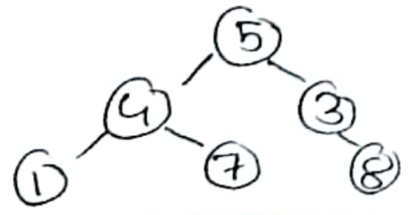
Array becomes: [8 4 7 1 3 5]

2 ~~Finding~~ larg swapping. 8 with 5



Max heap: [5, 4, 7, 1, 3, 8]

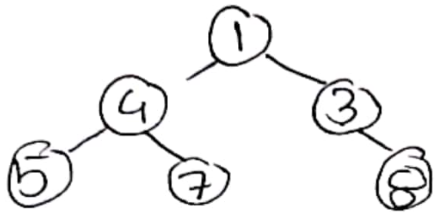
3. Swapping. 7 with 3



Maxheap : [5, 4, 3, 1, 7, 8]

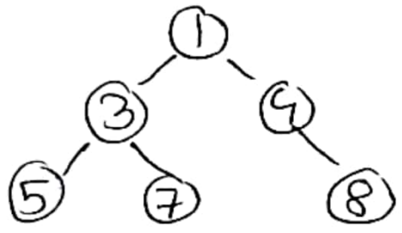
Sumit Manandhar

4 Swapping 5 with 1



Max heap array: [1, 4, 3, 5, 7, 8]

5 swapping 4 with 3



array: [1, 3, 4, 5, 7, 8]

∴ ~~s~~orted array: [1, 3, 4, 5, 7, 8]