```
Heap Sort
  def heapity (arr, n,i):
largest = i
        left = 2 * i + 1
        right = 2*1+2
if left < n and arr [left] > arr [longest]:
largest = left
if right < n and arr[right] > arr [largest]:

largest = right
if largest 1=i:
    arr[i]; arr[largest] = arr[largest], arr[i
   heapily (arring largest)
def heap-sort (arr):
          n= Len (ars)
         in range (n//2-1,-1,-1):
            heapity (arronoi)
   for i in range (n-1,0,-1)?

arr[i], arr[o] = arr[o], arr[i]
```

heaptly (arr, i, o) arr = [12, 11, 13, 5, 6, 7] heap-sort (arr) print (" Sorted array: ", arr) Time Complexity?

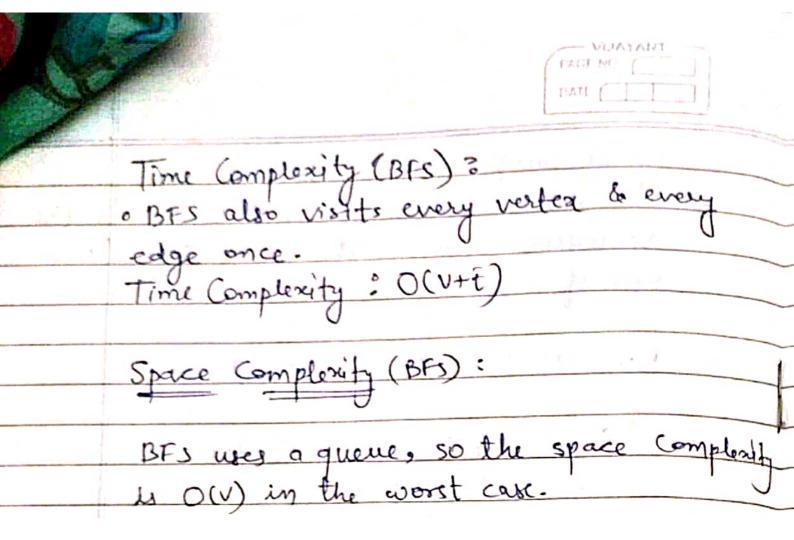
• Heapify? O (logn)

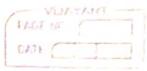
• Building the Heap? O(n)

• Sorting Process: O(nlogn) Overall, time complexity is O(n logn) Space Complexity: Heap sort is an in-place sorting algorithm, so the space complexity 40(M)

Depth First Search (DFS) det des (graph, start, visited = None): if visited 4 None: visited = set () visited add (start) print (start, end = ") for neighbor in graph [start]: if neighbor not in visited: Ats (graph, neighbor, visited) graph = 1 'A' = ['B', 'C'], 'B' & ['D', 'E'], 'c' # ['F'], 'p' + [], F' & ['F'] 'F' + [) off (graph, 'A') · DFS visits every vertex & every edge once · TC = O(V+E) where V is the number of vertices and E is the number of edges

Space C	emplexity (DFS)
. The spa	ce complexity is 060 due to the
recursi	on stock in the coast case (in
	lesp recursion).
Breadt	h First Search (Pfs)
	collections import deque
det	- bfs (graph , start):
	visited = set ()
	queue = doque ([stast])
,	Visted add (stout)
	tite
while	queuo:
	vertex = queue. poplet()
	point (vertex, end = ")
	for neighbor in graph Treater :
	if neighbor not in visited:
-1.	if neighbor not in visited: visited add (neighbor)
	queue. append (neighbor)
graph = 1	
0 ' 'A'	=['B', '(')],
'B'	['D', 'f'],
٠,	; ['f'],
, D	': [],
°E.	': ['f'],
	': []
	(graph, ch')
613	(July 1)





def mage-sort (are) & left-hay = arr [: mid] sort (left-half left-half[i] < sight-half[j]: arr [K) = left_hay [i] [+=10 else: gor[k) - right - half [j] K+=1 < len (left-baly): ou [x] = left - half [i 1=1+ len (right-half) and [k] = right - hall 11,13,5,6,7 point ("Sorted array : " are)

	PAGE NO.: DATE:
a manual dispra	Time Complexity Splitting the array: O(logn)
	Time Complexity Splitting the array: $O(logn)$ Merging process: $O(n)$ Thus, the overall time Complexity of Merge Sort $O(nlogn)$
	Space Complexity
	Merge Sort require O(n) auxilary space the temporary arrays used during the
A Company of the Comp	muge process.