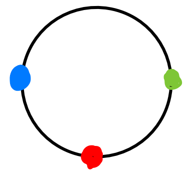
TRAVERSIERUNG:

DEPTH-FIRST = STACK
BREADTH-FIRST = QUEUE

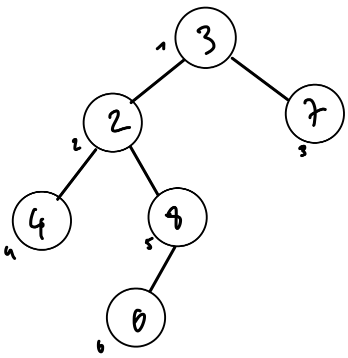
NODE:



⇒ PRÄORDER $O(n)$
⇒ INORDER $O(n)$
⇒ POSTORDER $O(n)$

} DEPTH-FIRST

TREE:



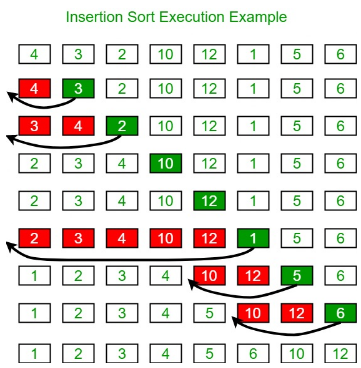
OUTPUT: 3, 2, 7, 4, 8, 6

QUEUE:
1. (3)
2. (2, 7)
3. (7, 4, 8)
4. (4, 8)
5. (8)
6. (6)
7. ()

} BREADTH-FIRST (LEVEL-ORDER) $O(n)$

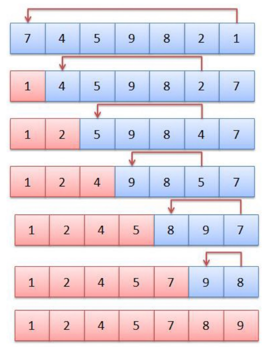
SORTS:

INSERTION SORT:



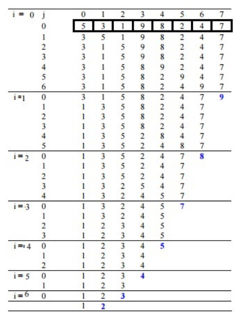
> STARTE LINKS UND VERGLEICHE DAS NÄCHSTE ELEMENT
  ⇒ WENN DIE REIHENFOLGE PASST, MARKIERE ALS SORTIERT
  ANSONSTEN SWAPE MIT LINKEN ELEMENT, BIS REIHENFOLGE PASST
> $\boxed{O(n^2)}$
> IN-PLACE UND STABIL

SELECTION SORT:



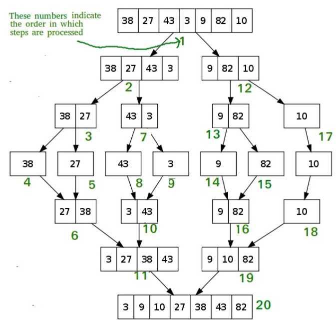
> STARTE MIT ZWEI POINTER, P1: MINIMUM P2: CURRENT ELEMENT
> TAUSCHE INDEX MIT MINIMUM (VERGLEICHE INDEX MIT ALLEN ELEMENTE
> $\boxed{O(n^2)}$
> IN-PLACE UND STANDARDMÄßIG NICHT STABIL

BUBBLE SORT:



> VERGLEICHE IMMER ZWEI ELEMENTE, WENN DAS ZWEITE ELEMENT KLEINER IST
> $\boxed{O(n^2)}$
> IN-PLACE UND STABIL

MERGE SORT:

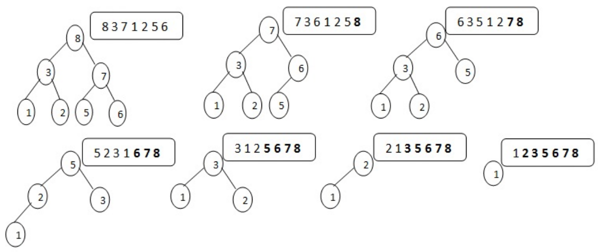


> REKURSIV, DIVIDE & CONQUER
> ZUERST WIRD HALBIERT BIS ALLE ELEMENTE ALLEIN STEHEN, DANN WERDEN IMMER ZWEI SUBARRAYS SORTIERT UND GEMERGED
> $\boxed{O(n \log n)}$
> NOT IN-PLACE UND STABIL

HEAP SORT:

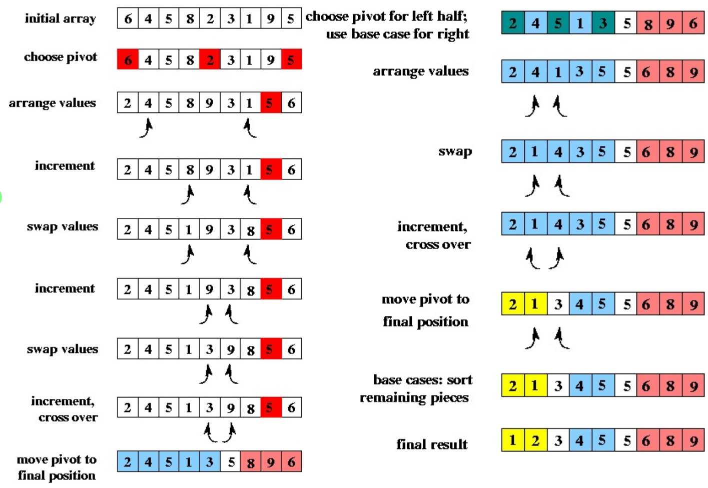
- BUILD MAX HEAP
- REMOVE ROOT AND SWAP IT WITH THE LAST NODE
- CHECK IF IT'S STILL MAX HEAP, IF NOT HEAPIFY
- BUILD MAX HEAP $\boxed{O(n)}$
  HEAPIFY $\boxed{O(\log n)}$
- IN-PLACE AND NOT STABLE

Example:- The fig. shows steps of heap-sort for list (2 3 7 1 8 5 6)



QUICK SORT:

- RECURSIVE USING PIVOT
- CHOOSE PIVOT BY MEDIAN, MOVE ALL ITEMS WHICH ARE SMALLER TO THE LEFT AND THE REST TO THE RIGHT
- WORST CASE: $\boxed{O(n^2)}$
  AVERAGE CASE: $\boxed{O(n \log n)}$



RED-BLACK TREE: (BALANCED SEARCH TREE)

HEIGHT: $O(\log n)$

SEARCH $O(\log n)$
INSERT $O(\log n)$
REMOVE $O(\log n)$

$\}$ REQUIRE ROTATIONS

! ROOT AND LEAVES ARE ALWAYS BLACK !

! IF A NODE IS RED, THEN THE CHILDREN ARE RED!

ALL PATHS FROM A NODE TO ITS LEAVES, HAVE THE SAME NUMBER OF BLACK NODES (ROOT EXCLUDED)