Merge Sort & Heap Sort

2021년 1월 17일 일요일 오전 12:44

1. Merge Sort

: merge n sorted elements and n sorted elements to make the 2n elements sorted

ex) [1 3 4 9] [2 5 7 8] — Put the cursor at first place of each set.

⇒ 1 2 3 4 5 7 8 9 — Write the smaller one, and push the cursor pointing to it to right one at a time. Repeat.

ex) 3 7 4 5 1 8 9 2 — $N=8$

3 7 4 5 1 8 2 9

3 4 5 7 1 2 8 9 — $*\log N$ steps, $*N$ comparisons needed per step ⇒ $O(N \log N)$

1 2 3 4 5 7 8 9

→ Disadvantage: need x2 of memory space to write the array data in each step.

2. Heap Sort

* Min Heap (⇔ Max Heap) (a.k.a.) Heap Tree

: Binary Tree. Root is minimum (⇔ maximum)

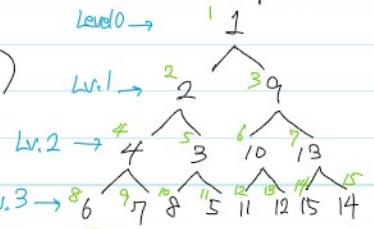
: Every case → Parent Node ≤ Child Node (≥)

* Min-Heap Array example:

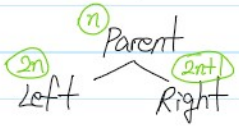
{ ▨, 1, 2, 9, 4, 3, 10, 13, 6, 7, 8, 5, 11, 12, 15, 14 }

(▨: number that stores idx of end of min-heap)

ex) Min Heap (idx)



* Relationship between Parent & Child Nodes: idx of parent = idx of child / 2



★ Heap Sort: with an array given, ① make it a min (or max) heap array ② sort the heap array made in ①

① : insertion algorithm

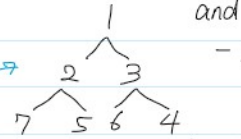
ex) initial array: { (idx) 1, 7, 5, 6, 2, 1, 4, 3 }

↑ min-heap completed until here.

- Compare the newly included number and its parent (= self/2)
- If the number is smaller, swap them and repeat until it goes to root.
- Increment arr[0].

arr = { 7, 1, 2, 3, 7, 5, 6, 4 }

↑ min-heap completed!



* time complexity: $\log 1 + \log 2 + \log 3 + \log 4 + \cdots + \log N$

$= \log(N!) \le \log N^N = N \log N$

* time complexity : $\log 1 + \log 2 + \log 3 + \log 4 + \cdots + \log N$
$= \log(N!) \leq \log N^N = N \log N$

② : deletion algorithm

arr = {7, 1, 2, 3, 7, 5, 6, 4} → min-heap array

- First element (root) is minimum. Swap it with last element of heap tree.
  Decrement arr[0] → last place is now not part of the heap tree.
- When last element became the root, check if it's smaller than child nodes.) make it a min-heap again?
  → should be smaller or equal to min (left child, right child)
  → if not, swap it with min(left, right). And repeat this.
- The back side of array, which is not part of the heap,
  is the sorted results. (in descending order)
  * time complexity : $\log(N!) \leq \log N^N = N \log N$

∴ Time Complexity of Heap Sort : $N \log N + N \log N = 2N \log N \Rightarrow O(N \log N)$
↳ x2 longer than merge sort
* But doesn't require additional memory space!