Data Structures and Algorithms

Prof. Ganesh Ramakrishnan,
Prof. Ajit Diwan,
Prof. D.B. Phatak

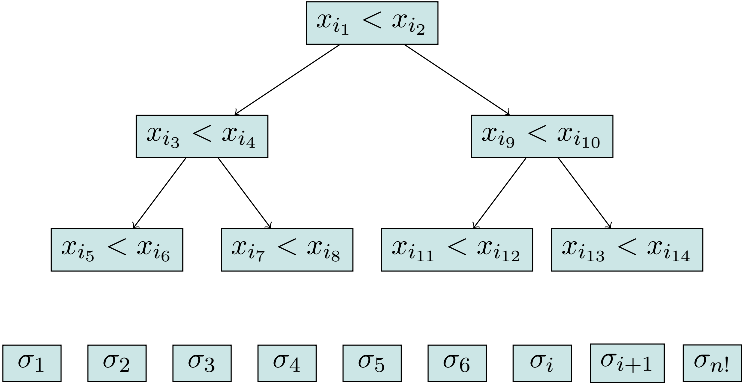
Department of Computer Science and Engineering
IIT Bombay

Session: Heap Based(Heap Sort)

Lower bound for Comparison based Sorting

- Sort by making comparisons between pairs of objects
- Result of each comparison $\Rightarrow$ Yes/No
- Such a sorting algorithm $\equiv$ series of comparisons to decide which permutation of sequence S is the sorted permutation
- Number of permutations $= n!$
- A run of the algorithm $\equiv$ root-to-leaf path in binary decision tree with permutations as the leaves
- $\Longrightarrow$ Number of leaves $= n!$
- $\Longrightarrow$ Minimum height of such a tree $= \log{(n!)} = \Omega(n \log n)$

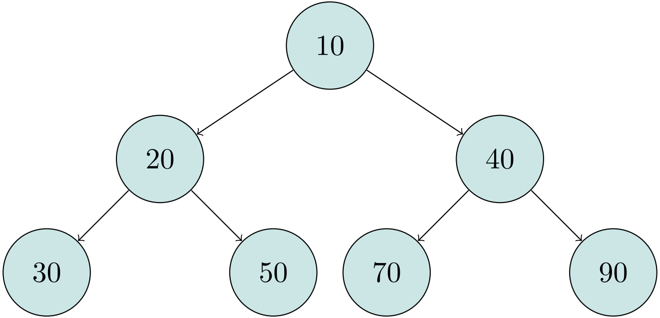
Tree of Permutations



σ_i is the i^{th} permutation of the n unqiue elements in the sequence S. Altogether, there exist $n!$ such unique permutations.

Could a Min-heap ADT be used for Achieving this lower-bound?

Recall for min-heap: $Val[parent(i)] \leq Val[i]$.



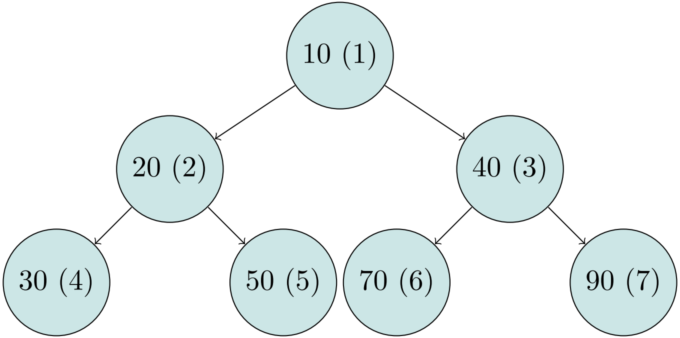
Min-Heap based Sort

Let P be a min-heap

- $insert_h(e, P)$: Insert in order to maintain min-heap nature of list P
  - $\log(1) + \log(2) + \ldots \log(n) = O(n \log(n))$
- $delete(P, m)$: Remove the (min) element m of the heapified list P
  - $\log(n) + \log(n-1) + \ldots \log(1) = O(n \log(n))$
- $\Rightarrow O(n \log(n))$ overall

Min-heap and its Array Representation

For min-heap representation in S: $S[parent(i)] \leq S[i]$



$$[10\ (1),\ 20\ (2),\ 40(3),\ 30(4),\ 50\ (5),\ 70\ (6),\ 90\ (7)]$$

Array-based in-place version of Heap Sort

- Recall a Min-Heap of n elements:
  - A Complete binary tree with all levels except last being full
  - Last level is filled from left to right
  - Value of item at parent $\leq$ values at children
  - Minimum element will be at the root
- In the Array Setup: Children of node k are at $2k$ and $2k+1$, provided the latter two are $\leq n$
- **Heap Sort:** Left portion of S up to index $i-1$ will contain the elements sorted so far, and index i to n to store remaining elements in a Heapified form.

Array-based in-place version of Heap Sort

Algorithm HeapSort(S)
Input: Sequence S
Output: Sequence S sorted in increasing order
Priority Queue: Build-Min-Heap(S)
$i = 1$
while $i \leq length(S) - 1$ **do**
1. Min-Heapify(S,i)
2. $i = i + 1$

end while

Figure: In-place Heap Sort

Min-Heapify Subroutine

Algorithm Min-Heapify(S,i)
l and r are respectively the left and right children of $S[i]$
if $S[l] < S[i]$ **then**
  $min = l$
else
  $min = i$
end if
if $S[r] < S[min]$ **then**
  $min = r$
end if
if $min \neq i$ **then**
  $S[i] \leftrightarrow S[min]$
  Min-Heapify(S, min)
end if

Question: How about a non-recursive variant of Min-Heapify?

Build-Min-Heap Subroutine

Note: Elements in $S\left[\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right) \ldots n\right]$ are all leaves (1 elements heaps) of the tree.

Algorithm Build-Min-Heap(S)
Input: Sequence S
Output: Min-Heapified Sequence S
$i = \left\lfloor \frac{length(S)}{2} \right\rfloor \implies c_1 \quad \times$ 1 times
while $i \geq 1$ **do**

1. Min-Heapify(S,i) $\implies c_2 \quad \times \quad \sum_{i=1}^{n} \log i = O(n \log n)$
2. $i = i - 1$

end while

- Next Session: Merge Sort

Thank you