BINARY HEAP

NITHIN JOSHUA STEPHEN - CS14M033

Abstract. This document gives a brief overview of binary heap data structure and operations associated with it.

1. Introduction

A **binary heap** is a data structure similar to binary tree with two additional constraints.

1) **Heap Property**: The value of children of an element must be always less than **(max-heap)** or always greater than **(min-heap)** the value of it's parent. However, there is no constraint for values between the siblings of an element.

2) **Shape property**: The binary heap is a *complete* tree where all levels, except the lowest level with height 0 is completely filled.

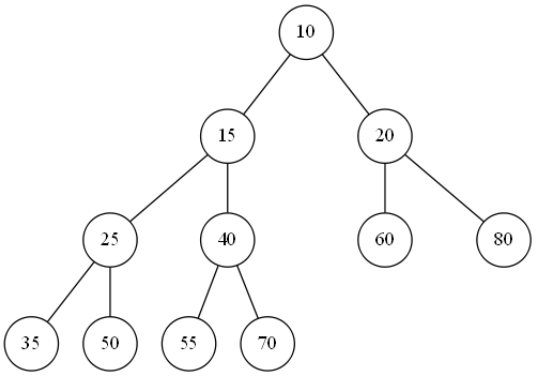

Figure 1. min-heap

2. Operations

A binary minheap has the following operations associated with it

- **MIN-HEAPIFY**: It maintains the *Heap property* where an element's left and right subtrees are valid min-heaps, but the subtree rooted at it is not a min-heap.
- **INSERT**: It inserts the element into a min-heap.
- **EXTRACT-MIN**: It removes the element and returns it.

3. Implementation

A binary heap is an *Abstract Data Type* implemented either using an array or pointers similar to binary tree.

3.1. Using array.

If min-heap is implemented using an array, attributes associated with are

- `size`: It indicates the number of elements in a min-heap.
  The root is `array[0]` and the last element is `array[size-1]`

The auxillary operations associated with it are

- The *parent* of an element of index i is given by $\left\lceil \frac{i}{2} \right\rceil - 1$

Date: 25-Aug-2014.

- The *left child* of an element of index i is given $2 * i + 1$
- The *right child* of an element of index i is given $2 * (i + 1)$

3.2. **Using pointers.** If implemented using pointers, attributes associated with are

(1) `root`: It points to the root of the min-heap.
(2) `tail`: It points to the last element of the min-heap.
(3) `parent`: This is associated with each element and it points to the parent of the element.
(4) `left child`: This is associated with each element and it points to the left child of the element.
(5) `right child`: This is associated with each element and it points to the right child of the element.

4. Height of a heap

The height of an element in heap is the number of edges on the longest downward path from the element to a leaf and the height of the heap is the height of the root. The height h of the heap is $O(log_2 n)$ where n is the number of elements in a heap. The worst case for height h occurs when there is *only 1 element in last level*

No of elements in level $0 = 2^0 = 1$
No of elements in level $1 = 2^1 = 2$
No of elements in level $2 = 2^2 = 4$
No of elements in level $i = 2^i$
No of elements in last but one level $i = 2^{h-1}$
No of elements in last level $= 1$

Adding all the elements at each level

$\sum_{i=0}^{h} = 1 + 2 + 4..2^{h-1} + 1 = 2^h$

So number of elements $n = 2^h \implies h = log_2 n \implies h = O(log_2 n)$

5. Working of INSERT operation

The INSERT operation add the element at the last index, which may cause violation of *Heap property*. The inserted element is compared with its parent. If the parent is greater than the element, they are swapped to maintain *Heap property*. Then the parent is compared with its parent and the swaps continue up the heap until an element is greater than its parent.

Consider inserting 18. The following operations happen during insertion.

1) element 18 is compared with element 60 and they are swapped as $18 < 60$
2) element 18 is compared with element 20 and they are swapped as $18 < 20$
3) element 18 is compared with element 10 and insert operation terminates as $18 > 10$

The INSERT operation happens up the heap and the worst case number of swaps occur when the element inserted is less than the root, where swaps happen all the way up to the root. So maximum number of swaps is equal to height of the heap and hence running time is $O(h)$ or $O(log_2 n)$

6. Working of EXTRACT-MIN operation

EXTRACT-MIN operation removes and returns the minimum element in a min-heap. The minimum element is at the root of the heap. It is swapped with the tail and removed from the heap. The new root is compared with it's children. If it is less, the process is stopped, else the root is swapped with the smallest of the children and this process continues down the heap until the parent element is smaller than it's children. When the swapping process finishes the removed element is returned.

The following operations happen during EXTRACT-MIN operation.

1) element 70 is swapped with element 10(ie root) and removed
2) element 70 is compared with element 15 and they are swapped as $70 > 15$
3) element 70 is compared with element 25 and they are swapped as $70 > 25$

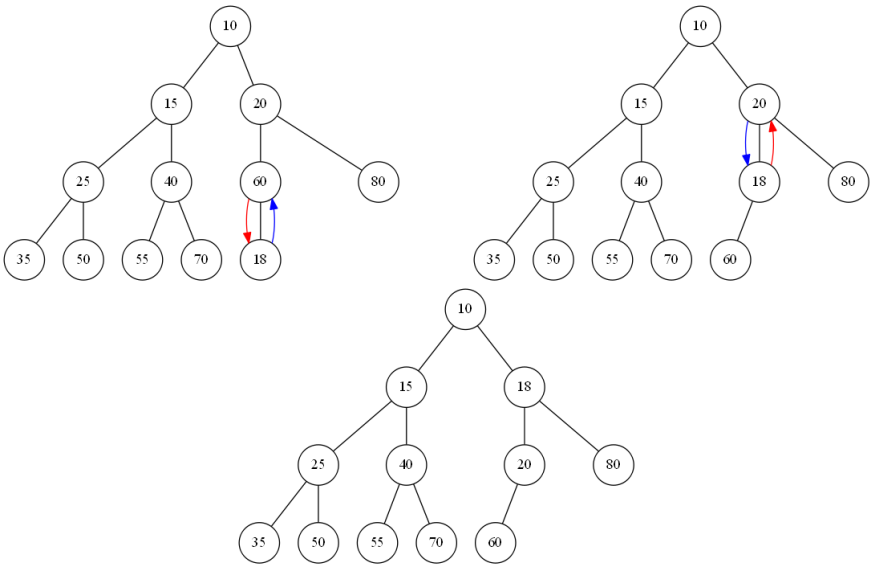

FIGURE 2. Inserting 18 into min-heap

4) element 70 is compared with element 35 and they are swapped as $70 > 35$
5) element 70 is now a leaf, and element 10 is returned

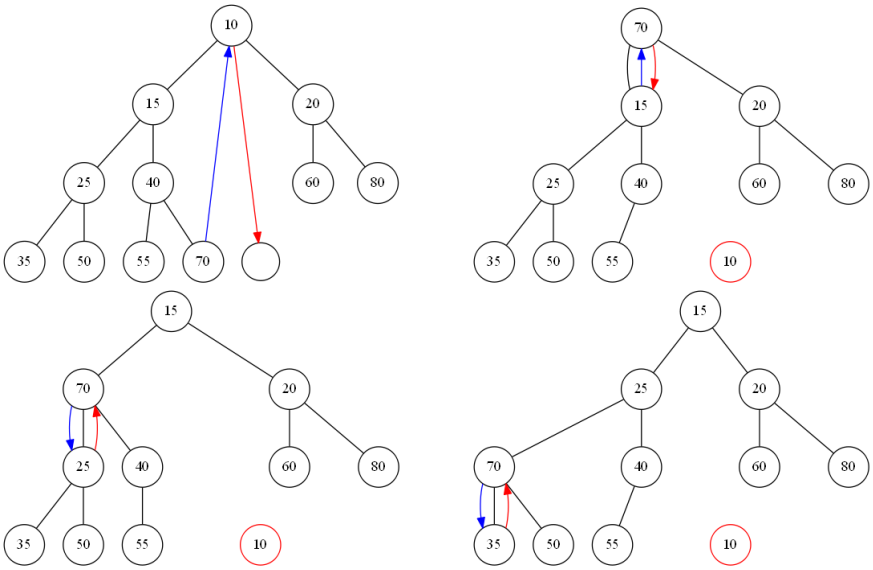

FIGURE 3. extract-min operation

The EXTRACT-MIN operation happens down the heap and the worst case number of swaps occur when the swaps happen all the way up to a leaf. So maximum number of swaps is equal to height of the heap and hence running time is $O(h)$ or $O(log_2 n)$

REFERENCES

[1] Thomas H. Cormen ,Charles E. Leiserson ,Ronald L. Rivest,Clifford Stein, *Introduction to Algorithms, 2nd edition*, 2006
[2] web resource *wikipedia*