Binary Heap Homework

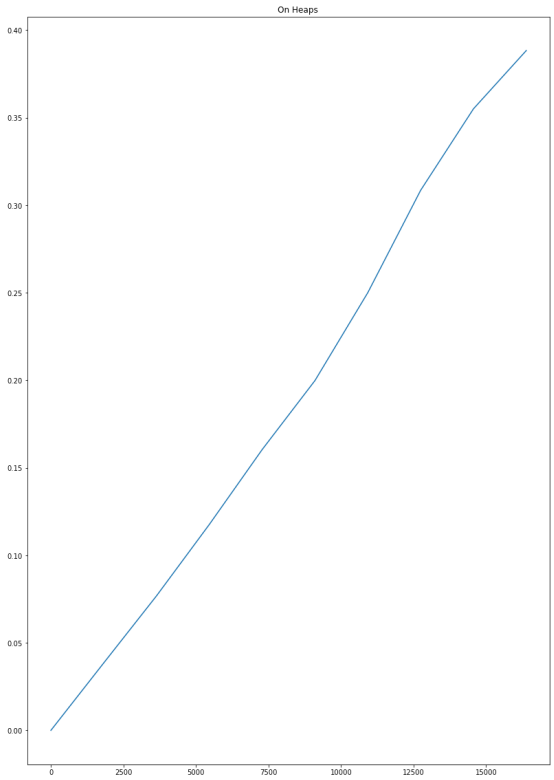
Exercise 3

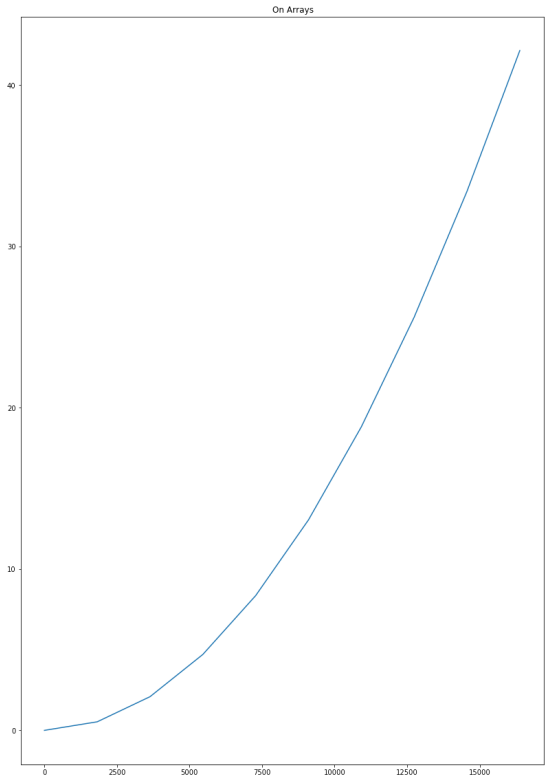
Test the implementation on a set of instances of the problem and evaluate the execution time.

Solution

As we can see from the plots the execution times of the heap implementation increases linearly with the size of the binary heap, whereas the execution time of the array implementation increases exponentially.

In addiction the execution time of the heap implementation is ten time faster then the array implementation.





Exercise 4 (Ex. 6.1-7)

Show that, with the array representation, the leaves of a binary heap containing n nodes are indexed by $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \ldots n$.

Solution

We know that in a binary heap all leaves of the last level are on the left, and that with the array representation the left child of the node in the i-th position has index $2 * i$.

Let's take into consideration the node with index $\lfloor n/2 \rfloor + 1$. Its left child would have index

$$2 * (\lfloor n/2 \rfloor + 1) > 2(n/2 - 1) + 2 = n - 2 + 2 = n \implies \text{LEFT}(\lfloor n/2 \rfloor + 1) > n$$

Since n is the number of elements in the heap, there cannot be a node with index $> n$. Hence the node indexed by $\lfloor n/2 \rfloor + 1$ cannot have any children and so it's a leaf. The same holds for all the nodes with larger indexes ($\lfloor n/2 \rfloor + 2, \ldots n$)

On the other hand, if we consider the node with index $\lfloor n/2 \rfloor$.

$$\lfloor n/2 \rfloor = \begin{cases} n/2, & \text{if } n \text{ is even} \\ (n-1)/2, & \text{if } n \text{ is odd} \end{cases}$$

It will have a left child indexed by

$$\text{LEFT}(\lfloor n/2 \rfloor) = \begin{cases} (n/2) * 2 = n, & \text{if } n \text{ is even} \\ ((n-1)/2) * 2 = n-1, & \text{if } n \text{ is odd} \end{cases}$$

So it is not a leaf.

To conclude, with the array representation, in a binary heap containing n nodes, only the ones with index form $\lceil n/2 \rceil$ to n are leaves.

Exercise 5 (Ex. 6.2-6)

Show that the worst-case running time of HEAPIFY on a binary heap of size n is Ω(log n). (Hint: For a heap with n nodes, give node values that cause HEAPIFY to be called recursively at every node on a simple path from the root down to a leaf.)

Solution

Let's consider a min-heap which has its maximum value in the root. Since that according to the heap property the relation $parent(p) \leq p$ must hold for any node, the root value must be swapped through each level of the heap until it reaches a leaf level and so HEAPIFY will be called recursively at every node on a path from the root down to a leaf. We know that the heap has height $O(\log n)$. Hence the worst-case running time of HEAPIFY on a binary heap of size n is $\Omega(\log n)$.

Exercise 6 (Ex. 6.3-3)

Show that there are at most $\lceil n/2^{h+1} \rceil$ nodes of height h in any n-element heap.

Solution

Let n_h denote the number of nodes at height h.

Let's prove by induction:

- **Base case**: Let's consider a leaf. It is in the last level, so $h = 0$. We know that the leaves have indexes that go from $\lfloor n/2 \rfloor + 1$ to n (as proven before). So there are $\lceil n/2 \rceil$ nodes at level 0.

  Hence the condition $n_0 = \lceil n/2 \rceil \leq \lceil n/2^{0+1} \rceil$ holds.
- **Inductive step**: Suppose the assumption holds for level h: $n_h \leq \lceil n/2^{h+1} \rceil$.

  The nodes at level h will have at most $2 * n_h = 2 * \lceil n/2^{h+1} \rceil$ children. Hence at level $h-1$ there will be at most $2 * \lceil n/2^{h+1} \rceil = \lceil n/2^{(h+1)-1} \rceil = \lceil n/2^{(h-1)+1} \rceil$ nodes.

  The assumption holds also for $h-1$.