Module 5 Critical Thinking: Heaps

Donghwan Kim

Colorado State University Global

CSC 506: Design and Analysis of Algorithms

Dr. Lori Farr

May 18, 2022

Module 5 Critical Thinking: Heaps

This assignment is about heap sort based on a heap data structure. It shows the algorithm of heap sort, discusses time complexity, and exemplifies how it works with a list. Heaps are a useful structure, providing another way to find the minimum numbers in a list.

How heapsort works

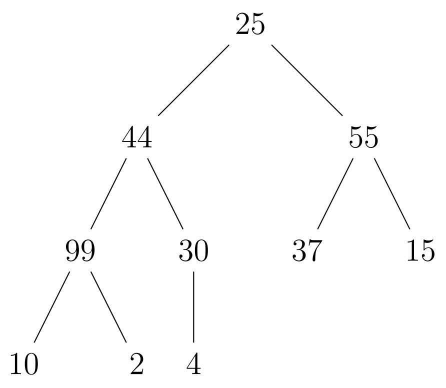
The heap is good for managing the maximum or minimum element of a list and as will be shown it can be used for sort. It is a mimimum-finding algorithm. It also makes an efficient priority queue (Cormen et al. 2009: p. 150). A heap has a binary tree structure but is implemented with a list. The algorithm consists of two functions: The main HeapSort function and the MaxHeapPercolateDown function (Lysecky and Vahid 2019).

```
Algorithm.

1. Function MaxHeapPercolateDown(node_i, x, n):
2.     child_i = 2 * node_i + 1
3.     value = x[node_i]
4.     while child_i < n:
5.         max_v = value
6.         max_i = -1
7.         i = 0
8.         while i < 2 and i + child_i < n:
9.             if x[i + child_i] > max_v:
10.                max_v = x[i + child_i]
11.                max_i = i + child_i
12.                i = i + 1
13.        if max_v == value:
14.            return
15.        x[node_i], x[max_i] = x[max_i], x[node_i]
16.        node_i = max_i
```

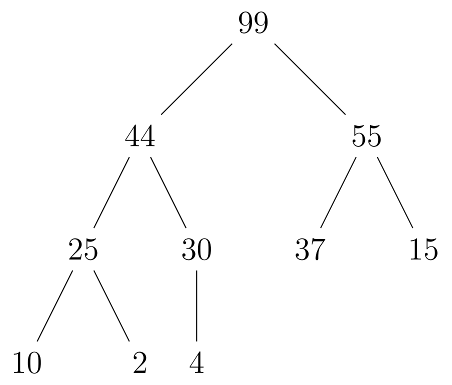
```
        child_i = 2 * node_i + 1
```

```
Algorithm.
Function HeapSort(x):
    // Heapify
    i = len(x) // 2 - 1
    while i >= 0:
        MaxHeapPercolateDown(i, x, len(x))
        i = i - 1

    i = len(x) - 1
    while i > 0:
        x[0], x[i] = x[i], x[0]
        MaxHeapPercolateDown(0, x, i)
        i = i - 1
```

The HeapSort function heapifies an original list first, which performs the MaxHeapPercolateDown function. It builds a max-heap (Levitin A. 2003). A heap of n elements based on a complete binary tree, its height is O(logn) and the basic operations on heaps have O(logn) runtime (Cormen et al. 2009). The Heapsort has O(nlogn) runtime regardless of whether it is ordered. The algorithm is explained with the list [25, 44, 55, 99, 30, 37, 15, 10, 2, 4]. The heap list has the following tree structure:



Heapifying is done with the internal nodes. The number of the internal nodes are floor(n/2) - 1. Since the length of the list is 10, the number of the internal node is 4: 25, 44, 55, 99, and 30. As shown in the algorithm, the MaxHeapPercolateDown function is performed. There is no swap until the internal node becomes zero. At the root node, 25 and 44 are swapped, 25 and 99 are swapped. The heapified list is [99, 44, 55, 25, 30, 37, 15, 10, 4] whose tree structure is:



In the heapified list, the first element is swapped with the element of the end index. And the MaxHeapPorcolateDown function is run for the elements from the index 0 to endIndex - 1. It repeats as the end index goes from len(x) - 1 to 0. The first one is done with the list of [4, 44, 55, 25, 30, 37, 15, 10] where 4 and 55 are swapped and 4 and 37 are swapped. It results in [55, 44, 37, 25, 30, 4, 15, 10]. Then the first and last elements are swapped. The list becomes [10, 44, 37, 25, 30, 4, 15]. The list [55, 99] is the sorted one. It is repeated until the end index is zero. After another procedure, the list consists of the unsorted list, [10, 30, 37, 25, 2, 4, 15], and sorted one, [44, 55, 99]. It finally has [2, 4, 10, 15, 25, 30, 37, 44, 55, 99].

Conclusions

This assignment implements heapsort based on a heap. In heap sort, the right-hand side is sorted as the index moves from the end to the beginning of a list. It is different than insertion sort and quicksort. It is like selection sort in the sense that it uses the maximum-finding algorithm.

References

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. The MIT Press.

Lysecky, R. and Vahid, F. (2019). Design and analysis of algorithms. In Lysecky, R. and Vahid, F., editors, *Data structures essential: Pseudocode with python examples*. Zybooks.

Levitin, A. (2003). *Introduction to the design and analysis of algorithms*. Boston: Pearson.