1 Heapsort

1.1 Intuition and Description

Heapsort is a sorting algorithm which introduces a data structure, a binary tree (the *heap*). A heap is a nearly complete binary tree. For example, consider the list from previous examples

$$(7, 1, 5, 4, 9, 2, 8, 3, 0, 6).$$

This list can be arranged into a heap simply by placing the first element of the list at the root, then placing the next two elements as its children and so on (see figure 1).

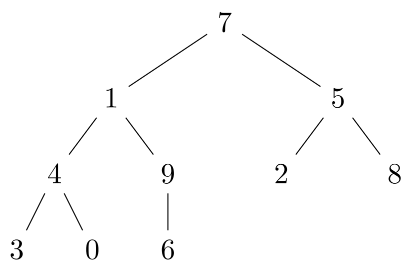

Figure 1: the given list arranged into a heap

In essence, the goal of heapsort is to first sort the heap into a *max-heap* where each parent is larger than each of its children. Then, the root element which by necessity must be the largest element is removed from the heap and the element on the lowest branch (in figure 1 where 6 currently is) is moved to the root.

It turns out however, that not every recursion needs to check if the heap is a max-heap. After sorting the initial heap, one can assume that the heap is already sorted except for the root. The function which partially sorts the heap after the largest element was removed and the element of the lowest branch is moved to the top is called *heapify*.[**?**, p. 135]

As we did with quicksort, we present the pseudocode for heapsort in the following.

```
def heapsort(_list):
    def heapify(_list, _n, _i):
        largest_element_index = _i

        LEFT_CHILD_INDEX = 2 * _i + 1
        if LEFT_CHILD_INDEX < _n:
            if (_list[LEFT_CHILD_INDEX] >
                _list[largest_element_index]):
                largest_element_index = LEFT_CHILD_INDEX

        RIGHT_CHILD_INDEX = 2 * _i + 2
        IF RIGHT_CHILD_INDEX < _n:
            if (_list[RIGHT_CHILD_INDEX] >
                _list[largest_element_index]):
                largest_element_index = RIGHT_CHILD_INDEX

        if largest_element_index != _i:
            _list[_i], _list[largest_element_index] =
                _list[largest_element_index], _list[_i]

            heapify(_list, _n, largest_element_index)

    i = floor(len(_list) / 2) - 1

    while i >= 0:
        heapify(_list, len(_list), i)
        --i

    i = len(_list) - 1
    while i >=0:
        _list[0], _list[i] = _list[0], _list[i]
        heapify(_list, i, 0)
        --i
```