Nama: Benhard David Hamonangan Tampubolon
NPM: 140810170042
Kelas: B

Tugas Pengganti Liburan Praktikum Analgo

Heap Sort

Pseudocode

```
procedure heapsort(a, count) is
    input: an unordered array a of length count

    (Build the heap in array a so that largest value is at the root)
    heapify(a, count)

    (The following loop maintains the invariants that a[0:end] is a heap
and every element
     beyond end is greater than everything before it (so a[end:count] is
in sorted order))
    end ← count - 1
    while end > 0 do
        (a[0] is the root and largest value. The swap moves it in front of
the sorted elements.)
        swap(a[end], a[0])
        (the heap size is reduced by one)
        end ← end - 1
        (the swap ruined the heap property, so restore it)
        siftDown(a, 0, end)
(Put elements of 'a' in heap order, in-place)

procedure heapify(a, count) is
    (start is assigned the index in 'a' of the last parent node)
    (the last element in a 0-based array is at index count-1; find the
parent of that element)
    start ← iParent(count-1)

    while start ≥ 0 do
        (sift down the node at index 'start' to the proper place such that
all nodes below
        the start index are in heap order)
        siftDown(a, start, count - 1)
        (go to the next parent node)
        start ← start - 1
    (after sifting down the root all nodes/elements are in heap order)

(Repair the heap whose root element is at index 'start', assuming the heaps rooted at its children are valid)

procedure siftDown(a, start, end) is
    root ← start

    while iLeftChild(root) ≤ end do    (While the root has at least one child)
        child ← iLeftChild(root)   (Left child of root)
        swap ← root                (Keeps track of child to swap with)

        if a[swap] < a[child]
            swap ← child
        (If there is a right child and that child is greater)
        if child+1 ≤ end and a[swap] < a[child+1]
            swap ← child + 1
        if swap = root
            (The root holds the largest element. Since we assume the heaps rooted at the
            children are valid, this means that we are done.)
            return
        else
            swap(a[root], a[swap])
            root ← swap            (repeat to continue sifting down the child now)
```

Hitung Manual

Terdapat 6 angka : 1 4 3 2 6 5

1 4 5 2 6 3

1 6 5 2 4 3

6 4 5 2 1 3

3 4 5 2 1 6

1 4 3 2 5 6

1 2 3 4 5 6

1 2 3 4 5 6

1 2 3 4 5 6

1 2 3 4 5 6

Hasil : 1 2 3 4 5 6

Analisa Big-O:
Langkah-langkah heapsort:

1. Mengerjakan fungsi heapfy(), yang membutuhkan waktu operasi O(n)
2. Menukar elemen pertama dalam list dengan elemen terakhir, dan mengurangi range operasi sebanyak: 1
3. Memanggil fungsi siftDown() ke list yang untuk menggeser elemen awal yang baru ke index list heap yang sebenarnya
4. Kembali ke step 2 hingga range operasinya tinggal 1

heapfy() membutuhkan waktu operasi sebanyak O(n) dan hanya dipanggil sekali
siftDown() membutuhkan waktu operasi sebanyak O(logn) dan dipanggil sebanyak n-kali

Sehingga big-O: O(n + nlogn) = O(nlogn)