Ramaneek Gill g3gillra 1000005754
Sept 21 2014

CSSC263 Exercise #2

A) I will be using a min-priority queue and the extracting the first k elements.

```
function(A, k):
    S = new priority_queue //will be storing the priority queue in a heap
    S.heapsize = A.length
    for item in A:
        MIN-HEAP-INSERT(S, item.value)
    B = new array
    for i in range(0, k-1): //assuming 0 is the first element in the array
        B.append(HEAP-EXTRACT-MIN(S))

    return B
```

This algorithm will loop over every item in A and insert it in to a min-priority queue. The MIN-HEAP-INSERT function will input the items in the priority queue S in a sorting fashion in ascending order.

Then the first k elements are extracted from the priority queue and appended to the new array B. The array B is returned which contains all the smallest elements in the priority queue up to l.

B) The running time of this algorithms relies on two main functions MIN-HEAP-INSERT and HEAP-EXTRACT-MIN. MIN-HEAP-INSERT runs at a Θ(log n) time since it bubbles up an insertion along the binary tree called the heap which has a max height of floor(log n).

HEAP-EXTRACT-MIN relies on two things, the first being the extraction of the first element. Which is Θ(1) running time. The second is the function MIN-HEAPIFY that is called within HEAP-EXTRACT-MIN, this runs at Θ(log n) time. Therefore this function's running time is Θ(log n).

The total running time is Θ(log n + klog n) since MIN-HEAP-INSERT runs n times and HEAP-EXTRACT-MIN runs k times. The total worst running time of the algorithm is Θ((n+k)log n) which is Θ(nlog n)

C) I will be using a max-heap and extracting the first k elements of the array (since HEAPSORT creates an array that is in ascending order).

```
function(A):
    HEAPSORT(A):
        BUILD-MAX-HEAP(A)
        for i = A.length downto 2:
```

```
                exchange A[1] with A[i]
                A.heapsize = A.heapsize - 1
                MAX-HEAPIFY(A, 1)

        return A[first_element.....kth element]
```

This function will run a HEAPSORT algorithm on A in place, it will perform the operations only on the array A and use little to no extra memory (at most 3 integers will be stored) this depends on the implementation of user of BUILD-MAX-HEAP and MAX-HEAPIFY.

The way the algorithm works is by creating a max-heap of A onto A itself in place. Then it will exchange the first element of the array with the last. The first element in the array (which is now a max-heap) is the largest. Then it will reduce the heapsize of A by 1, which will make it so that the function MAX-HEAPIFY will not operate on the last element of A.

After MAX-HEAPIFY is run, the largest element in the max-heap is now the root. This entire process is run again and on continually smaller sets until it is empty (to the trivial case of a binary tree with 2 nodes the root being the largest). This algorithm sorts the max-heap array and modifies it in to a sorted array that is in ascending order.

Now that A is sorted the function will return a sliced version of the array from the first element to the k'th element which is the k smallest elements in the array A.

D) HEAPSORT worst case run time relies on two functions: BUILD-MAX-HEAP and MAX-HEAPIFY. BUILD-MAX-HEAP runs at $\Theta(n)$, MAX-HEAPIFY is called n-2 times, MAX-HEAPIFY has a worst case running time of $\Theta(\log n)$.

Returning the sliced list takes O(1) time so it is dwarfed by the larger runtimes defined above. The entire algorithm has a worst case running time of $\Theta(n + (n-2)\log n)$ which is $\Theta(n\log n)$

E)

Algorithm C also uses less memory since the array A is heapified and sorted in place and returns a portion of the array A. Whereas algorithm A creates a new object to store the heap of priority queue and then has to create a new array to return the k smallest elements.

Algorithm C is more efficient than A in runtime since a more correct runtime for it is $\Theta((n+k)\log n)$ and C is $\Theta(n + n\log n - 2\log n)$. $\Theta((n+k)\log n) > \Theta(n + n\log n - 2\log n)$.

SOURCES USED:

The only resources used to finish this exercise was my own lecture notes and the textbook that was assigned to do the readings on.