Assignment 2

CS 514 – Algorithms

Submitted By: Aman Pandita

Onid: panditaa@oregonstate.edu

LOOP INVARIANT:

At the start of each iteration of the for loop, the subarray A[a....i] is a max-heap containing the i smallest elements of A[a....n] , and the subarray A[i+1.....n] contains the n-i largest elements of A[a....n], sorted.

1. Initialization:

Before the for-loop starts (the second for-loop in the `heapSort` function), the function builds a max-heap from the input array. This ensures that every parent node is greater than or equal to its children nodes, with the largest element at the root of the heap. So, before the first iteration of the loop, the entire array is a max-heap, and our invariant holds true as there are no sorted elements yet.

2. Maintenance:

Before the iteration starts, we assume that the subarray ***A[a...i]*** is a max-heap containing the ***i*** smallest elements, and the subarray ***A[i+1...n]*** contains the ***n−i*** largest elements sorted in descending order.

During the iteration, the root of the heap (largest element) is swapped with the element at index i. Now, the subarray ***A[i+1...n]*** has the largest elements in sorted order. ***heapify(arr, i, 0)*** is then called on the reduced heap of size ***i***, restoring the max-heap property in the subarray ***A[a...i]***, leaving the ii smallest elements in a max-heap.

By the end of an iteration, our invariant still holds as the max-heap property is maintained in the subarray ***A[a...i]***, and the ***n–i*** largest elements remain sorted in the subarray ***A[i+1...n]***.

3. Termination:

When the loop terminates (when ***i*** reaches 1), the invariant assures us that the array is split into two parts:

- A single-element max-heap, which is trivially sorted.
- The rest of the elements, which have been sorted in descending order.

Finally, since every step involves swapping the largest remaining element into its correct position, the array is sorted in ascending order, which concludes the proof.