**Assignment 12 (Sorting and Searching Algorithms)**

1. Let an array be given as follows:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 34 | 21 | 10 | 45 | 4 | 20 | 15 | 2 | 59 | 12 |

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

Heap Sort algorithm contains two processes: 1) heap building; 2) iteratively swapping and reheaping.

1. Show (*hand-writing is fine*) the array after the heap-building process
2. Show the resulting arrays of four consecutive iterations of the “swapping and reheaping” process.
3. Write a method of Heap class to print the node items in an “inorder” fashion, where “inorder” follows the same definition given in Chapter 8.
4. During the process of Insertion Sort, the sorted portion of the array grows while the portion to be sorted shrinks. For example, an array in progress may have the following configuration:

[1, 3, 6, 10, 2, 9, 5]

when the next iteration would insert ‘2’ into the sorted portion. To insert an item into the sorted portion, one can actually do better by applying binary search algorithm.

1. ***Modify*** the Insertion-Sort so that ***binary search***is used when inserting a new item into the sorted potion. Binary search algorithm is described and coded in Section 6.6.
2. *What would be the big-O efficiency of the approach measured by the number of comparisons*?
3. Merge Sort uses an additional array, which can be costly and even not feasible. Re-implement the Merge Sort in such a way that avoids using another array.

***Hint:*** Basically, you merge the two separately sorted portions *(first…middle)*, and *(middle + 1… last)* within the original array itself. This can be done using roughly the same idea of Insertion Sort. More specifically, you insert each of the items in *(middle + 1… last)* into *(first…middle)* starting at *middle + 1*. The price paid for this approach however is the worsened efficiency. What is the Big-O count with this new implementation in terms of number of comparisons? (In fact, there are two versions: one based on consecutive swapping, and the other based on binary insertion. You may want to implement both for comparison purposes in the next problem.)