CS1203 - Monsoon 2023 - Question 3

Niyati Rao, Student ID: 1020211512

Collaborators: NONE

Question 3

Comparing the time complexity of merge sort, quick sort and heap sort

- **MERGE SORT:**
  Merge sort is a divide-and-conquer sorting algorithm known for its stability and efficient performance. It works by recursively dividing an array into smaller subarrays, sorting those subarrays, and then merging them to create a single, sorted array. The space complexity therefore is O(N), as it uses additional space for the sub arrays.
  - **Best case, worst and average case:**
    The algorithm continuously divides the array into two smaller sub arrays and therefore the total number of levels is O(log N) and the amount of work done at each level is of order O(N) therefore the time complexity of this sorting algorithm is O(N log N)
- **QUICK SORT:**
  Quick sort is an in-place sorting algorithm unlike the other two and chooses a partitioning element depending on its value elements are continuously put to the left or right of this element and then the algorithm works on the subarray. The space complexity of this algorithm is O(1)
  - **Worst Case:**
    This situation occurs when the partition element chosen is always unbalanced as in the case where there is one element on one side and the rest on the other. This makes the time complexity O(N^2) randomization of choosing this partitioning element improve the worst case.
  - **Best and average case:** The algorithm partitions within the array and these fragments need to be solved the total levels of fragmentation are of the order O(log N). At each stage of fragmentation, however, the amount of work done is O(N) as all the elements are chosen to be compared against partition elements. This makes the time complexity O(N log N)

- **HEAP SORT:**
  Heap sort is a sorting algorithm that uses properties of a binary heap. It first creates a max-heap/min-heap from the unsorted array and then repeatedly extracts the top element from the heap (the maximum or minimum) and places it at the end of the array until the entire array is sorted. It is an in-place sorting algorithm, but it's not as commonly used as some other sorting methods in practice. It has two main steps in the algorithm:

  1. **Building the heap**
  2. **Extracting maximum element**, which is then placed at the end of array until array is sorted.

  The worst-case time complexity of heap-sort is $O(nlogn)$.

  Building heap from an unsorted array (this is binary) would take O(n) time, as each element must be inserted into the heap and then sifted up to its correct position.

  Extraction of the maximum element from the heap takes O(log n) time and is then placed at the top of the heap, further the heap must be restructured to maintain properties after each extraction.

  Therefore, heap-sort would perform n extractions, each taking $O(logn)$ time.

  Worst-case time complexity of heap-sort is **O(n log n)**.

```
PS C:\Users\91636\OneDrive\Desktop\Assignment 3\question3> gcc .\mergesort.c
PS C:\Users\91636\OneDrive\Desktop\Assignment 3\question3> ./a.exe
Given array is
8 19 9 11 12 11 13 5 6 7 3 17

Sorted array is
3 5 6 7 8 9 11 11 12 13 17 19
Copy count 55, Comparison Count 33
PS C:\Users\91636\OneDrive\Desktop\Assignment 3\question3> gcc .\quicksort.c
PS C:\Users\91636\OneDrive\Desktop\Assignment 3\question3> ./a.exe
Unsorted array:
8 19 9 11 12 11 13 5 6 7 3 17
Sorted array:
3 5 6 7 8 9 11 11 12 13 17 19
Swap 28 Comparison 38
PS C:\Users\91636\OneDrive\Desktop\Assignment 3\question3> gcc .\heapsort.c
PS C:\Users\91636\OneDrive\Desktop\Assignment 3\question3> ./a.exe
Unsorted array:
8 19 9 11 12 11 13 5 6 7 3 17
Sorted array:
3 5 6 7 8 9 11 11 12 13 17 19
Swap 31 Comparison 74
```

Figure 1: An instance of execution of merge quick and heap sort