DOKUZ EYLUL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME 2204 ALGORITHM ANALYSIS

ASSIGNMENT 1 REPORT

by

Muzaffer Sevili

2019510069

Lecturers

Assoc. Prof. Dr. Zerrin IŞIK

Res.Asst.Ali CUVİTOĞLU

IZMIR

09.04.2022

CONTENTS

1.INTRODUCTION

The aim of the project is to sort arrays of different types and sizes according to different sorting algorithms. Then, the analysis of the algorithms is done according to the data results.

Three algorithms were used: Heap Sort, Shell Sort, and Intro Sort.

1.1 Heap Sort

- Heap sort is a comparison-based sorting technique based on Binary Heap data structure.
- Half the array size is taken, and items are compared to their children starting from that index.
- If the parent node is stored at index K, the left child can be calculated by 2 * K + 1 and the right child by 2 * K + 2 (assuming the indexing starts at 0)
- The root-left child-right child, whichever is greater, becomes the root. In other words, the tree is tried to be brought to the Max Heap form.
- If the root element is deleted, the root element is replaced by the element with the largest index (the element with the smallest value) and is deleted from the tree.
- Then the Heapify function is called recursively and the tree is again shaped into the Max Heap structure. The Heapify function makes a comparison between the applied leaf and its right child and left child, if one of the children is greater than the leaf on which the function is applied, it is replaced by this leaf.
- The sorting process is completed by performing the same operations for the new tree.

➢ This algorithm is in-place but not stable.

➢ Worst case time O(n*logn)

BuildHeap(): O(n) time
Heapify() : O(logn) * (n-1 times)
T(n) = O(n) + (n - 1) * O(logn)
T(n) = O(n) + O(n*logn)
T(n) = O(n*logn)

1.2 Shell Sort

- Shell Sort is a variation of Insertion Sort.
- For the sorting process, a jump amount is determined first.
- Although there are many ways to determine the jump amount(k), the simplest method is to start from half of the numbers we have. (Example: 4/2=2 or 13/2=6)
- Each number is compared with the number k after it, and these numbers are ordered among themselves.
- The jump amount is then halved. (Example: 10/2=5, 5/2=2)
- The algorithm ends when all numbers are ordered in a single direction.

- This algorithm is in-place but not stable.
- Worst case time O(n^2)
- Shell sort takes O(1) space

1.3 Intro Sort

- The best versions of Quicksort are competitive with both heap sort and merge sort on the vast majority of inputs. Rarely Quicksort has the worst case of O(N ^ 2) running time and O(N) stack usage.
- Both Heapsort and Merge Sort have O(N log N) worst-case running time, together with a stack usage of O(1) for Heapsort and O(log N) for Merge Sort respectively.
- Also, Insertion sort performs better than any of the above algorithms if the data set is small.
- Combining all the pros of the sorting algorithms, Intro Sort behaves based on the data set.
    - ✓ If the number of elements in the input gets fewer, the Intro Sort performs Insertion sort for the input.
    - ✓ Having the least number of comparisons(Quicksort) in mind, for splitting the array by finding the pivot element, Quicksort is used. Quoted earlier, the worst case of Quicksort is based on the two phases and here is how we can fix them.
        1. Choosing the pivot element: We can use either of median-of-3 concept or randomized pivot concept or middle as the pivot concept for finding the pivot element
        2. Recursion depth during the course of the algorithm: When the recursion depth gets higher, Intro Sort uses Heapsort as it has the definite upper bound of O(N log N).

- This algorithm is in-place but not stable.

- Worst case time O(n*logn)

2.PERFORMANCE MONITORING

	EQUAL INTEGERS			RANDOM INTEGERS			INCREASING INTEGERS			DECREASING INTEGERS		
	1,000	10,000	100,000	1,000	10,000	100,000	1,000	10,000	100,000	1,000	10,000	100,000
heapSort	0.2	0.3	2.2	0.6	2.7	14.1	0.3	1.4	9.5	0.1	1.7	9.8
shellSort	0.3	1.7	7.9	0.2	2	11.7	0.2	1.8	7.3	0.3	1.6	8
introsort	1.1	3.9	13,5	0.5	2	11.9	0.3	1.3	8.2	0.4	1.4	8.5

2.1 Equal Integers

- The best sorting algorithm for equal integers is **Heap Sort**. Because the algorithm does not take time to generate the maximum heap form and working much faster than Shell Sort and Intro Sort.
- The order of the best algorithms according to the measurements in the table is as follows: Heap Sort → Shell Sort → Intro Sort.

2.2 Random Integers

- The best sorting algorithm for random integers is also **Heap Sort**.
- The order of the best algorithms according to the measurements in the table is as follows: Heap Sort → Intro Sort → Shell Sort.

2.3 Increasing Integers

- The best sorting algorithm for increasing integers is **Shell Sort**.
- The order of the best algorithms according to the measurements in the table is as follows: Shell Sort → Intro Sort → Heap Sort.

2.4 Decreasing Integers

- The best sorting algorithm for decreasing integers is **Intro Sort**.
- The order of the best algorithms according to the measurements in the table is as follows: Intro Sort → Heap Sort → Shell Heap Sort.
- But in the case of large sized arrays (by dimensions in the table), the sorting takes the following form: Shell Sort → Intro Sort → Heap Sort.

3. ALGORITHM SCENARIO

- According to the scenario, the preference should be the **Shell Sort** algorithm. Because if we think Turkish Higher Education Institutions Examination, there are so many parameters which can affect the grades of students. That is why it is hard for students to get their grades too equal. This data set will consist mostly of random numbers. Therefore, according to the table, the most correct choice will be Shell Sort.

- But if most of the grades are equal, the **Heap Sort** algorithm will be the best option for us. Because with equal integers, Heap Sort is much faster than other algorithms.

4. REFERENCES

4.1 Heap Sort

- https://www.geeksforgeeks.org/heap-sort/

4.2 Shell Sort

- https://www.geeksforgeeks.org/shellsort/

4.3 Intro Sort

- https://www.geeksforgeeks.org/introsort-or-introspective-sort/