ALGORITHM ANALYSIS AND DESIGN COURCE PROJECT

Firt Semester 2022/2023

GROUB MEMBERS:

- Warood Khalid Alzayer.
- Hana Alomran.
- Reem Shaker Almualem.
- Danah AlMuzel.
- Areej Ahmed Saleh.
- Luluwah Waleed.

SUPERVISED BY:

- Dr. Azza A. Ali.

جامعة الإمام عبدالرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

01. INTRODUCTION

Sorting Algorithms are important as they reduce the complexity, increase the efficiency of the performance. They are now used in our daily lives such as in google search. This project discusses tree types of sorting algorithms; insertion, merge and heap sort. These algorithms were tested, analyzed, and proved theoretically, and empirically by using Java object oriented programming language and all results were represented in graph and compared with each other. The data was sorted in ascending and descending order using these algorithms. Furthermore, an improved divide-and-conquer algorithm was generated and analyzed.

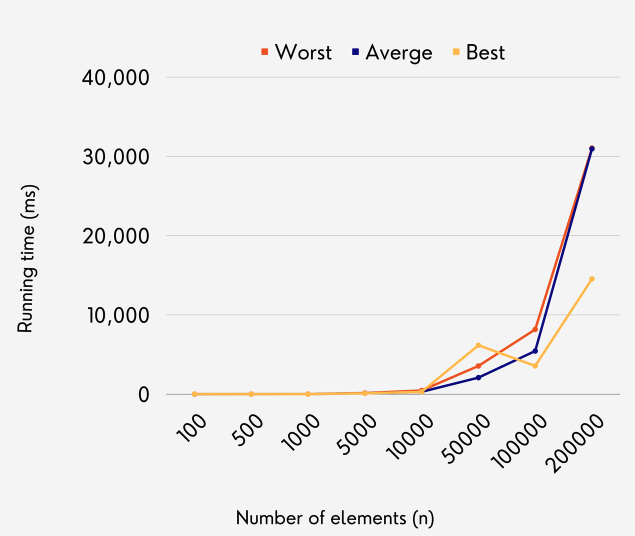
02. OBJECTIVE

This project shall yield the best, average, and worst cases for each insertion, merge, and heap sorting algorithms and for all according to the device's characteristics, input size, and rate of growth of the running time. Also, design an improved divide-and-conquer algorithm.

03. IMPLEMENTATION

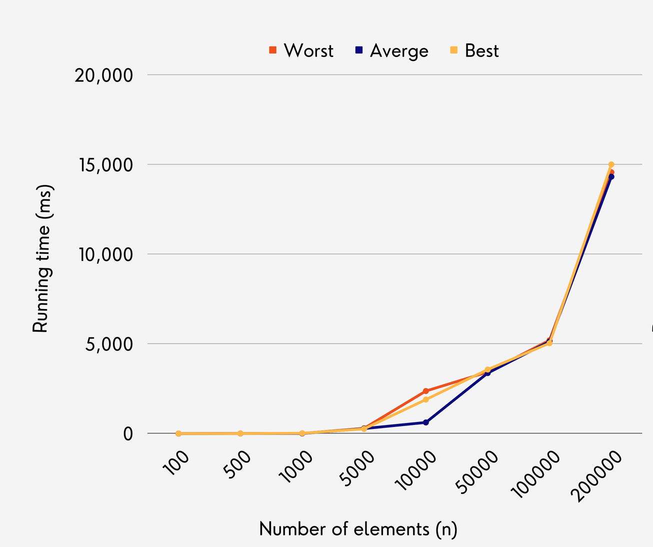
For the implementation, each sorting algorithm best, average and worst cases will be analyzed theoretically by its time complexity equation and practically from the code execution using an input size (n) of: (100,500,1000,5000,10000,50000,10000,20000) and compare between the results.

INSERTION (BEST & WORST & AVERAGE PRACTICAL TIME)



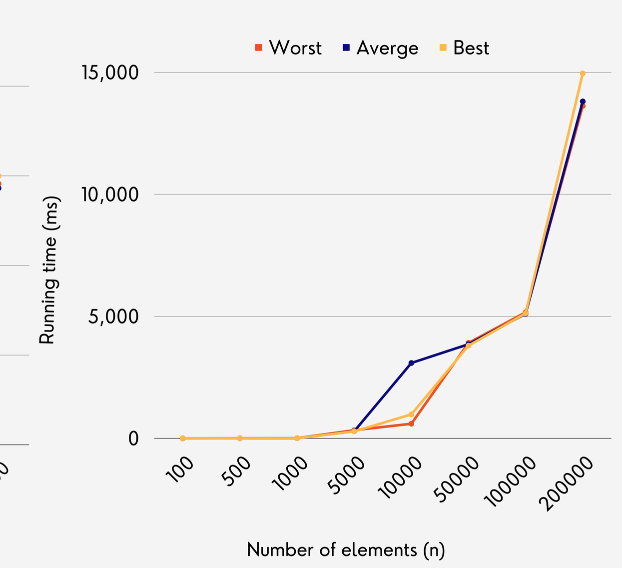
- This graph illustrates the growth of all three cases for the insertion sort algorithm, Where the worst case and average case are approximately similar in growth, while the best case has the smallest growth.

MERGE (BEST & WORST & AVERAGE PRACTICAL TIME)



- This graph illustrates the growth of all three cases for the merge sort algorithm, Where all the points at input size 200000 are approximately similar in growth.

HEAP (BEST & WORST & AVERAGE PRACTICAL TIME)



- This graph illustrates the growth of all three cases for the heap sort algorithm, Where the worst case and average case points are approximately similar in growth, and the best case shows the highest growth.

PART2: DEVIDE-AND-CONQUER

Compute a^n , where n is a natural number. Given two integers a and n, an algorithm is written to solve this problem and give the best running time possible.

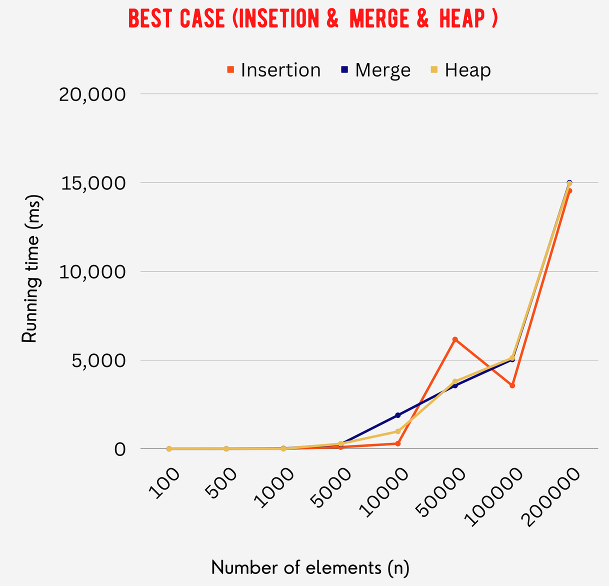
PRACTICAL CODE IN JAVA FOR (IMPROVED DIVIDE-AND-CONQUER):

```
1- STATIC INT POWER (INT A, INT N) {
2- INT TEMP;
3- IF (N == 0)
4- RETURN 1;
5- TEMP = POWER (A, N / 2);
6- IF (N % 2 == 0)
7- RETURN TEMP * TEMP;
8- ELSE
9- RETURN A * TEMP * TEMP;}
```

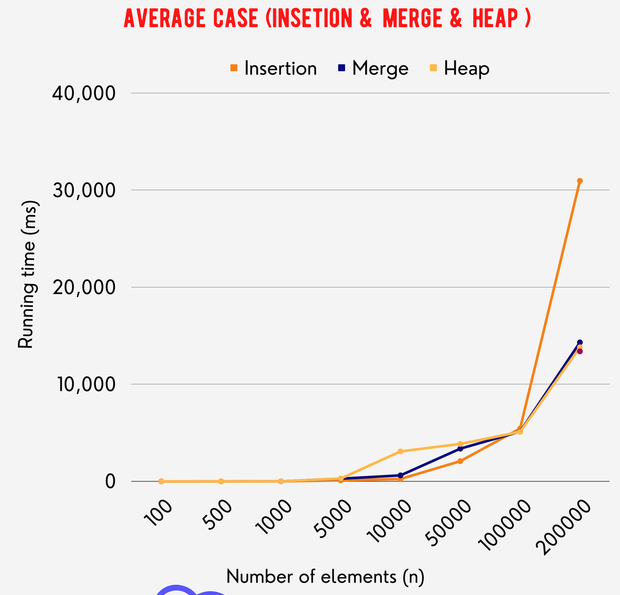
Java

04. RESULTS/FINDINGS

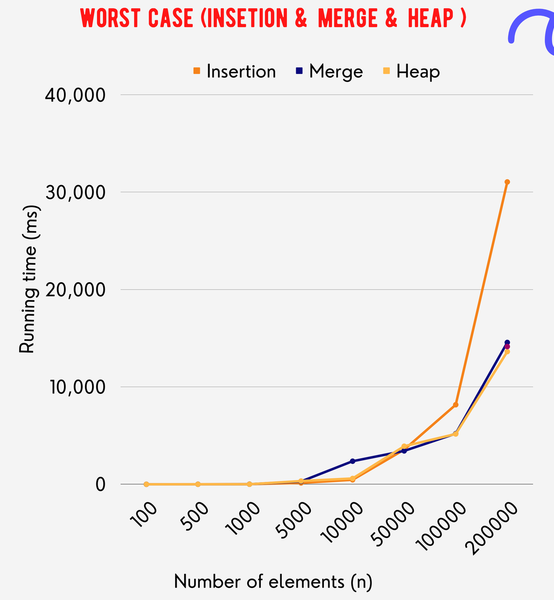
BEST CASE (INSETION & MERGE & HEAP)



AVERAGE CASE (INSETION & MERGE & HEAP)



WORST CASE (INSETION & MERGE & HEAP)



PART2 RESULTS:

improved Divide-and-conquer Steps	Running time	Divide-and-conquer Steps	Running time
static int power(int a, int n)	T(n)	static int power(int x, int y)	T(n)
int temp;	O(1)	if (y == 0)	O(1)
if (n == 0)	O(1)	return 1;	O(1)
return 1;	O(1)	else if (y % 2 == 0)	O(1)
temp = power(a, n / 2);	T(n/2)	return power(x, y / 2) * power(x, y / 2);	2T(n/2)
if (n % 2 == 0)	O(1)	else	O(1)
return temp * temp;	O(1)	return x * power(x, y / 2) * power(x, y / 2);	2T(n/2)
else	O(1)		
return a * temp * temp;	O(1)		
Recurrence relation	T(n)=T(n/2)+1	Recurrence relation	T(n)=2T(n/2)+1

DIVIDE-AND-CONQUER:

T(N)= (1) + (LOG N) + (LOG N) = O (N)

IMPROVED DIVIDE-AND-CONQUER:

T(N)= (LOG N) + (1) = O (LOG2 N)

05. ANALYSIS

For the best performance of the three sorts in the best case is insertion sort with ruining time = (n) .
When it comes to the best performance in the average case both merge sort and heap sort have close results because their complexity is the same = (n log n).
while the insertion sort has the worst performance both in average and worst cases due to its high growth = (n^2) .

06. CONCLUSION

To sum up, this project analyzes three sorting algorithms; insertion, merge and heap. The findings were that in the best case, the insertion has the best running time, and in the average and worst case the merge and heap are better than the insertion, these results were found by implementing each sorting algorithm using different input sizes and comparing the actual and theoretical running time results by creating tables and graphs.

Moreover, this project improved a divide-and-conquer algorithm by trying to find the best running time which was successful in decreasing the running time from (n) to (log n).