COMP 6411 Assignment 1 - Sorting Algorithms

Kshitij Yerande - 40194579 (Team Leader)

Mrinal Rai - 40193024

Akshay Dhabale - 40163636

Farheen Jamadar - 40194668

Almas Saba - 40156359

Seung Hyun Hong - 26724701

Ujjawal Aggarwal - 40183962

Nastaran Naseri - 40215694

Sorting Algorithms

Heap Sort

It is a sorting algorithm that uses the Binary Heap data structure to compare items. The Heapsort algorithm involves preparing the list by first turning it into a max heap. The algorithm then repeatedly swaps the first value of the list with the last value, decreasing the range of values considered in the heap operation by one, and sifting the new first value into its position in the heap. In the first step, a heap is built out of the data. The heap is often placed in an array with the layout of a complete binary tree. The complete binary tree maps the binary tree structure into the array indices; each array index represents a node; the index of the node's parent, left child branch, or right child branch are simple expressions.

In the second step, a sorted array is created by repeatedly removing the largest element from the heap (the root of the heap), and inserting it into the array. The heap is updated after each removal to maintain the heap property. Once all objects have been removed from the heap, the result is a sorted array[9].

Binary Tree: A binary tree is a data structure with a maximum of two child nodes.

Heapify: 'Heapify' is the process of transforming a binary tree into a Heap data structure.

BinaryHeap: A Binary Heap is a Complete Binary Tree in which items are stored in such a way that the value of a parent node is bigger (or smaller) than the values of its two offspring nodes.

```
        30(0)
       /   \
    70(1)   50(2)

Child (70(1)) is greater than the parent (30(0))

Swap Child (70(1)) with the parent (30(0))
        70(0)
       /   \
    30(1)   50(2)
```

Figure 1: Example of Heapify

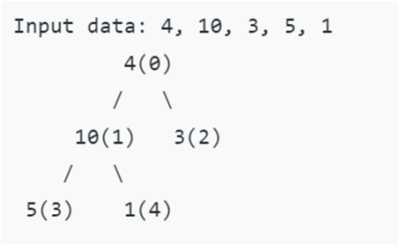
Algorithm for Heap Sort [10]:

Sorting in increasing order using the heap sort algorithm:

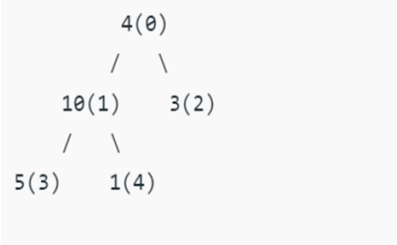
1. Create a maximum heap using the provided data.

2. The largest item is placed at the top of the stack at this point. Replace it with the heap's last item, then reduce the heap's size by one. Finally, heapify the tree's root.
3. Repeat step 2 as long as the heap is larger than 1.

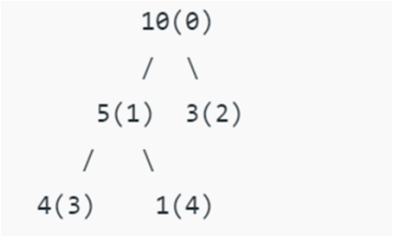
Building a Heap:
Numbers in the brackets alongside are indices in the Array representation:



Applying Heapify at Index 1:



Applying Heapify at Index 0:



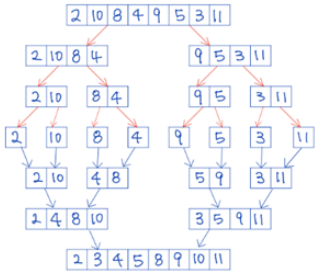
Time Complexity:
Time complexity of heapify is O(Logn). Time complexity of createAndBuildHeap() is O(n) and the overall time complexity of Heap Sort is O(nLogn).

Advantages:
Efficiency - As the number of objects to sort grows, the time required to conduct Heap sort grows logarithmically, whereas alternative methods may grow exponentially slower. This sorting method is really quick [10].
Memory Consumption - Memory usage is modest because it requires no additional memory space to work other than what is required to keep the initial list of objects to be sorted. Because it does not involve difficult computer science concepts like recursion, it is easier to understand than other equally efficient sorting algorithms [10].

Disadvantages:
When working with very complicated data, heap sort is thought to be insecure, expensive, and inefficient [10].

Merge Sort

Merge Sort was invented in 1945 by John von Neumann, and it employs the divide and conquer algorithm[5]. Merge Sort will recursively divide a given list of elements into half until there is a single element. Since the list has a single element considered sorted, merge steps will follow. The sub-lists will merge until there is only one sorted final list.



The diagram below gives an overview of how it works:

To merging two sub-lists, it creates new array list with size of sum of sub-lists, then start comparing each of first elements in the sub-lists and place smaller element into newly created array list until all of elements in two sublists are placed in new array list[6].

```
Merge_Sort(A, l, r)
  If(l<r)
    m = l + (r-l) /2
    Merge_Sort(A,l,m)
    Merge_Sort(A,m+1,r)
    Merge(A,l,m,r)
```

The complexity of merge sort algorithm is O(n*log n) in all three cases (best, average and worst). This is because merge sort always divides the array list until it has one element and merges the sub-lists in linear time[6]. Which means that it always takes the same amount of work and space for any array list given. The space complexity of merge sort is O(n). Because the last step of merging takes two sub-lists that have size n/2.

The merge sort is ideal for larger lists of elements, and it is a stable sorting algorithm. However, the merge sort takes more memory space to store its sub-lists and is slightly slower than other sorting algorithms for smaller data sets[6].

Programming Languages

Java

Java is a simple programming language designed to have as few implementation dependencies as possible.A general-purpose programming language made for developers to write once run anywhere that is compiled Java code can run on all platforms that support[1].

Due to its solidity and scalability, Java is found on mobiles, desktops and large-scale industry servers and applications. Globally, there are around 10 million Java developers, and this community continues to grow on a daily basis. Similar to other open source technologies, Java encourages the values of giving back to the public.

Java has a strong development roadmap with continuous progress in security and performance. While Java does not suit all needs, it's still a popular programming language which is widely used and benefits a lot of people and businesses. Even after 22 years of existence, Java continues to evolve.

Python

Python is among the most convenient programming languages among the other alternatives. Due to the availability of its vast range of applications with in-built solutions to standard web development tasks,the speed of a single project increases by many times. Python is quite popular due to its hundreds of different libraries and frameworks that can be used by developers. These libraries and frameworks are really useful in saving time which in turn makes the language even more useful [2].

Since it is not only one of the most popular programming languages, but is also one of the oldest. Hence, it got the time to gather a growing, supportive community of programmers, developers, and coders. The Python language is flexible enough to provide developers with ample time to try new experiments.

Suitability of Programming Languages for Sorting Algorithms

- Java and python are high level languages which are easy to learn with not a very high learning curve hence suitable for general purpose programming.
- Merge sort and heap sort require recursion and both java and python support recursion.
- Java and python simple usage of language construct provides easier processing of large data and implementation of sorting algorithms.
- Java and python are platform independent languages thus enabling us to run sorting code on any operating system increasing portability.

Comparing Sorting Algorithms

Merge and heap sort algorithms are asymptotically optimal algorithms with time complexity O(nlogn). The time required to merge in a merge sort is counterbalanced by the time required to build the heap in heapsort.[3]

Heap Sort is better in terms of space : The Heap Sort sorting algorithm uses O(1) space for the sorting operation while Merge Sort which takes O(n) space

Merge Sort is better in performance

The merge sort is slightly faster than the heap sort for larger sets as heap sort has additional factors than O(nlogn) which also contributes to performance.

Heapsort is not stable because operations on the heap can change the relative order of equal items.

Analysis of Java and Python

1. Lines of Code

One of the important metrics for software measurement is Lines of Code and the lines of code varies from programming language for a particular problem statement. Here we are comparing logical and physical lines of code to implement merge and heap sort. USC Code Count tool is used to calculate lines of code.(https://csse.usc.edu/tools)

Table for physical and logical lines of code

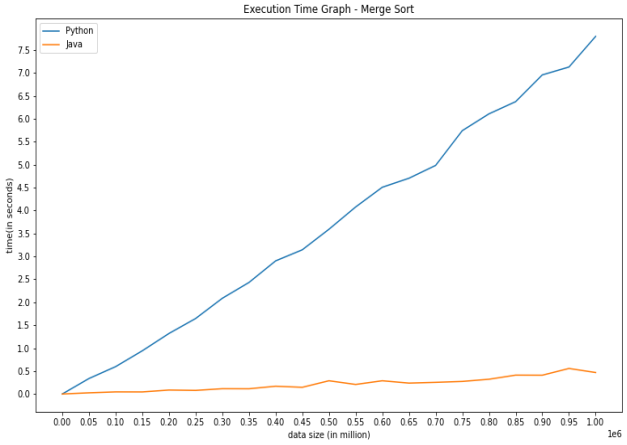
Algorithm	Programming Language	Logical Lines of Code	Physical Lines of Code
Merge Sort	Java	125	150
Merge Sort	Python	98	101
Heap Sort	Java	103	125
Heap Sort	Python	66	66

It can be inferred that python code is concise than java code as python does not use curly braces unlike java and the language construct is simple and easy to write because of the concise nature.

2. Execution Time

Time complexity of the merge sort and heap sort is O(nlogn) and both are considered to be efficient algorithms for sorting data. While processing data of different sizes from 50000 to 1 million we can see that implementing merge and heap sort in java is faster than python. The processing time of python increases significantly as the data size increases. This is due the dynamically typed nature of python where in it interprets everything at run time and hence requires more time for execution.

Reading and writing speed of files in java and python are good. It turns out that java is a bit slower than python probably because java code is reading file line by line and python code is reading all lines using a single function which might be optimally reading file.



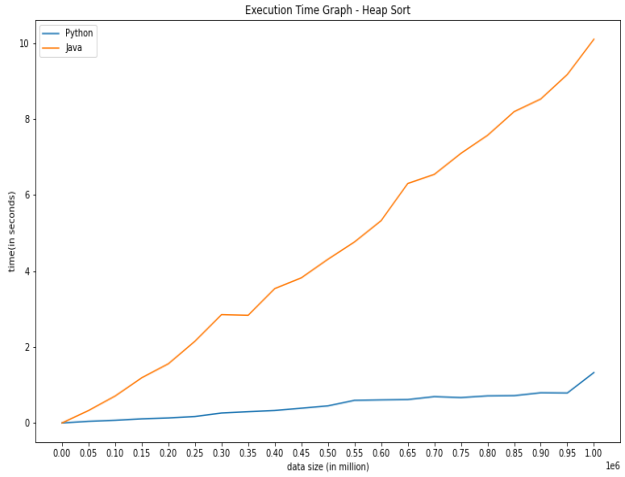
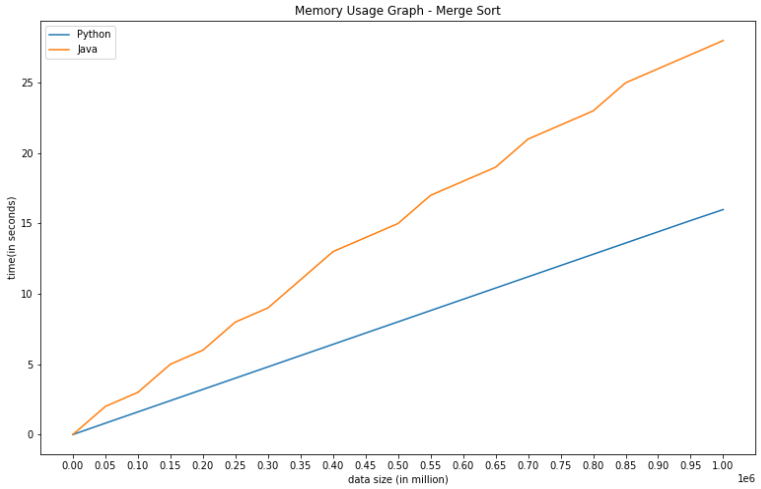
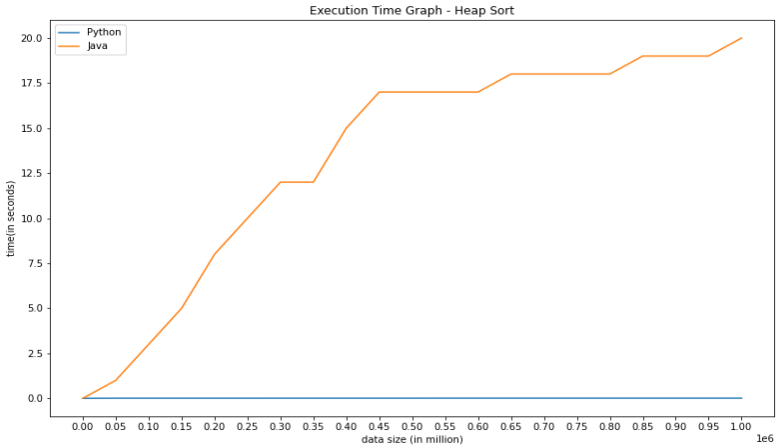


Table for comparison of performance for sorting algorithm in java and python

Sorting Algorithm (1 million records)	Programming Language	Execution Time (in seconds)	Memory Usage (in MB)	Reading Time in seconds (1 million Records)	Writing Time in seconds (1 million records)
Merge Sort	Python	7.79	15.99	0.2513	0.40404
Merge Sort	Java	0.468	28	1.8234436	0.1950
Heap Sort	Python	10.09	0.010	0.239096	0.120798
Heap Sort	Java	1.324	20	1.6102163	0.3031682

3. Memory Usage





It can be seen in the graph above that the curve of usage of memory is more in Java than in python with increase in data size. Usage of excessive memory is one of the major issues for enterprise applications and Java is often responsible for excessive memory usage that can result in memory leaks. Once the object is no longer referenced by the application, Java's garbage collector removes it and clears the memory so that it is available for usage by the application. So simply in Java, the more objects you use, the more memory is consumed by the application.

Whereas Python manages objects by using reference-counting. This means that the management of the objects is done by keeping tracking of the references to each object in the program. This is done by the python memory manager. Once the object is no longer being used, the memory is freed from that particular object by the garbage collector.

Table for comparison for memory usage in Java and Python

Sorting Algorithm	Programming Language	Memory Usage(in MB)
Merge Sort	Java	28
Merge Sort	Python	15.99
Heap Sort	Java	20
Heap Sort	Python	0.010173

4. Programming Paradigm

Python is a multi paradigm programming language which supports procedural, functional, object oriented and concurrent programming styles. Java too supports all of these except procedural style as java was initially designed to be object oriented and functional programming was introduced in java 8. Merge sort and heap sort can be implemented using procedural and object oriented programming language. The two sorting algorithms are implemented in procedural style in python as there are no classes created and only functions and main method to execute the program is used to implement the sorting algorithms. In java each program is inside a class hence the object oriented nature of java where everything is treated as objects [11,12].

5. Compilation and Portability

Although Java is a compiled language and Python is an interpreted language, this does not mean that Java does not have an interpretation step or Python does not have a compilation step. Both these languages do compile at a specific step and interpret also. Java has a separate step of compilation that converts the source code into bytecode before actual execution, which is platform independent. At the execution time, this bytecode is converted into machine code by JVM, so any computer or mobile device that can run the Java virtual machine can run a Java application. However, In Python, the code is compiled, but it is not performed as a separate step, and it is hidden from the developer that it appears the source code is directly executed, which is not true. Python code is also compiled into bytecode and then executed; the only thing is both these tasks are done during run time/execution, so the programs need an interpreter installed on the target machine to translate Python code. Hence, compared to Java, Python is less portable, while both are cross-platform languages since they compile bytecode and run it on virtual machines[7][8].

6. Program Execution

Different computers have different ways of executing instruction sets. This could lead to integer overflow or the same program could give different results on different computers. Java is different in this aspect as Java uses JVM(Java Virtual Machine) which using its JIT(Just in time) compiler converts the java code to java bytecode which can be interpreted by the interpreter [13]. There are two steps:

1. Compilation : convert java source code to byte code
2. Interpretation: reads and executes the byte code

On the other hand python is an interpreted language. The translation of source code to machine level code happens at runtime simultaneously, done by an integrated learning environment. Therefore the interpreter does a lot of parallel work behind the scenes. This is one of the reasons that java code runs faster than python [14].

7. Language Construct

Python and java have different language constructs and they differ in many ways. Python does not allow programmers to declare **variable** data type; it determines the data type during runtime whereas java needs to specify the data type of variable in order to make use of it. Python uses **indentation** for encapsulating code whereas java uses curly braces. **Methods** have a different implementation as java needs to specify the return type of method whereas python does not. **Loops** are also implemented differently, in java loops are implemented in standard format wherein you specify the initial value followed by condition and incrementer, in python the loops are equivalent to enhanced for loops, it creates an item and iterates through the data structure[4].

9. References

[1] https://www.geeksforgeeks.org/introduction-to-java/

[2] https://www.tutorialspoint.com/python/index.htm

[3]https://courses.cs.washington.edu/courses/cse373/17su/lectures/Lecture%2019%20-%20Comparison%20Sorting.pdf

[4] Khoirom, Mrs. Selina, Moirangthem Sonia, Borishphia Laikhuram, Jaeson Laishram and Tekcham Davidson Singh. “Comparative Analysis of Python and Java for Beginners.” (2020).

[5] “Merge sort,” *Wikipedia*, 04-May-2022. [Online]. Available: https://en.wikipedia.org/wiki/Merge_sort. [Accessed: 12-May-2022].

[6] Soileau, Gina, et al. "An Analysis of the Pragmatic Differences in How the Insertion Sort, Merge Sort, and Quicksort Algorithms Comparison Sort Numbers Stored in the Common Java-language Array Data Structure." (2008).

[7] Jim Hugunin,“Python and Java: The Best of Both Worlds”. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.117.6454&rep=rep1&type=pdf

[8] Amit Mathur, “JAVA VS PYTHON : COMPILED OR INTERPRETED”. Available: https://mytechdevice.com/java-vs-python-compiled-or-interpreted

[9] “HeapSort”. Available:https://en.wikipedia.org/wiki/Heapsort

[10] “Heap Sort” Available: https://www.geeksforgeeks.org/heap-sort/
[11] “Perceiving Python programming Paradigm”. Available: https://opensource.com/article/19/10/python-programming-paradigms
[12] “Functional Programming and Programming Paradigms in Java”. Available: https://dev.to/ggorantala/functional-programming-and-programming-paradigms-in-java-323f
[13] “How Java Works” Available: https://www.cs.cornell.edu/courses/JavaAndDS/files/howJavaWorks.pdf
[14] “The python interpreter”. Available: https://courses.cs.washington.edu/courses/cse140/13wi/lectures/02-interpreter.pdf