Search and Sort Algorithms for Big Data Structures

Bülent Arslan

Haaga-Helia University of Applied Sciences
Bachelor's Degree Programme in Business Information Technology
Bachelor's Thesis
2022

Abstract

Author
Bülent Arslan

Degree
Bachelor of Business Administration

Thesis Title
Search and Sort Algorithms for Big Data Structures

Number of pages and appendix pages
41 + 1

This work gives an insight of data collections, how they are managed and why it is important to adapt the handling and processing regarding the growth of the information which is saved and needed.

Starting with data collection in general, to give a feeling of the amount of processed information. The target is to show and give a feeling of the continues rise of data and what problems it entails. It elaborates the existing methods to search, sort and analyse the information, what advantages it gives and what might be reached with optimizing and adapting the speed to the rising amount of collected data.

A qualitative practical approach is used which utilizes different resources and tools to collect, compare and refine the needed information. With few questions asked, the focus was kept on those, to not get lost in the wide quantity of information. The empirical part further demonstrats the differences in speed and performance between the chosen algorithms. It is described how the comparisons were approched and how the difficulties were solved.

Key words
Big Data, Search & Sort Algorithms, Information, Processing

Table of contents

Abbreviations

A.I. – Artificial Intelligence, computers and machines have the same problem-solving and decision-making capabilities as human mind.

Bit – is the smallest unit of measurement used to quantify computer data. It contains a single value of 0 or 1.

Byte – a unit of computer information or data-storage capacity that consists of a group of eight bits to represent an alphanumeric character.

Cloud – refers to servers that are accessed over the internet. Users and companies do not have to manage physical server themselves or run software applications on their own premise.

CPU – Central Processing Unit, is the main computer component for processing instructions.

CRM – Customer Relationship Management, gathers customer interactions across all channels in one place. Helps businesses to improve customer experience, satisfaction, retention, and service.

ERP – Enterprise Resource Planning, is the about all core business processes needed to run a compony. Finance, HR, manufacturing, supply chain, services, and others.

GDPR – General Data Protection Regulation, specific data privacy laws for individual citizens in the European Union and the European Economic Area.

HDD – Hard Disk Drive, is a non-volatile, maintains stored data when turned off, data storage device.

IP-address – Internet Protocol, is a unique address that identifies a device on the internet or a local network.

RAM – Random Access Memory, is the physical hardware inside a computer that temporarily stores data, serving as the computer’s working memory.

SSD – Solid State Drive, mass storage device like HDD, but without any moving parts. SSD’s store data using flash memory.

1 Introduction

The idea of this work came up through the courses I took during my studies at Haaga Helia, like Data Management and Databases, Software Development, Software Project, and others. I want to show the importance, for companies of any sizes, what an impact it has, to use the right algorithms and methods for sorting and analysing their data. It is a challenge to find the right methods which can cope with the rising data structures. The already existing algorithms like linear search, binary search, hash search and more are for certain cases. We will go into more detail on those later.

Questions and objectives

The main questions:

- What algorithm is necessary to find and filter out the valuable information?
- What would be the best approach?
- Why is there a need for “better” or more suitable search algorithm?

The idee is to show the different methods and algorithms that exist and for the structures for which they are used. Through optimization of the methods companies can be more efficient, saving time and seeing the result in their revenue, in using the right algorithm.

There is not the one formula for the right algorithm, which can manage all tasks needed. It will be more a combination of algorithms with dependencies, meaning the algorithm will decide which one to use in what case. Nevertheless, to find the right search, sort algorithms, and combine them on a way that they work together is a difficult undertaking.

The best approach to achieve this is to go through the popular algorithms. It will be shown what algorithms are the most applied once and describe their functionality with their advantages and disadvantages. So that the reader will get a good understanding of the matter and can follow along.

The collection of Big Data is rising, therefore the conventional methods used currently will get to their limits if they are not already. The urge to develop new methods or combine existing ones in an effective way is necessary to manage the continuously growing amount of Data. The future

could look something like, that the algorithm which is written by an A.I. (Artificial Intelligence) in combination with machine learning, with the need of the necessary change, will adapt to the situations in seconds to work on the most efficient way possible.

If the speed of managing our sort algorithm would increase, that would mean you get a structured data and can search respectively find the collected information on a more efficient and time saving way. Saving time in business for the same task means more revenue in the end of the day. It is not just time saving but also necessary to increase the speed according to the amount of data collected which is increasing over time.

2 Big Data

2.1 What is Big Data?

Big Data is a new digital data source which can be applied for small as well as for big businesses. Companies have implemented Big Data for processing, analysing, and helping in decision-making, whether for strategic or operational purposes. The evolutions of digital storages which developed from room sized storages over CD's and HDD (Hard Disk Drive) to SSD (Solid State Drive), and into the Cloud's. A greater difficulty that includes all storage possibilities is that the data has not been processed before the storage process. This disadvantage makes it hard to get value from the stored data. There is an advantage of the not processed data which is to make new discoveries from the "source" data. (Fernando Iafrate, 2013)

The characterization of Big Date takes place with the four "V"'s, which are Volume, Variety, Velocity and Value. We now go into more detail and discuss now each of the "V"'s and show what part they play in the picture of Big Data.

2.1.1 Volume

The objects like servers, personal computers, tablets, and smartphones connecting to the internet, generated about eight exabytes (10 to the power of 18) in 2014. Each of those devices has a unique identifier (IP-address, Internet Protocol), which enables communication with its peers. Bits and bytes are necessary to digitalize information. A bit is the basic unit and can have the state of 0 or 1, and a sequence of eight bits are a byte. With one Byte you can digitalize e.g., decimal numbers from 0 to 255 depending on the sequences of the ones and zeros. In Figure 1 below you can see an example of a decimal number in binary code.

0	1	1	1	0	0	1	0

Figure 1. Shows the decimal number 114 in binary form

2.1.2 Variety

A good and structured data, which is used in transaction systems, is put into a so-called decision-support database. This database differs in the way, data is stored and the relationships between the data from other databases.

The advantages of a transaction data model lie in the execution speed of reading, writing and data modification to decrease the transaction duration to a minimum and increase the actions at the same time. E-commerce sites e.g., need to be able to manage thousands of users who are accessing and interacting at the same time. Overall, it enables through the operational data store which implements data tables to a reporting database, to view not the technical but the business data which enables to make reports without to know the physical data model.

The decision data model lies in analysis, modelling, data mining, which needs mostly a large amount of historic information. These makes it hard to use a relational data model, since through the joints and relations between the entities, it has a big effect on the execution time of queries. To overcome that issue the implementation of a denormalized data model was necessary. This model structure is much easier since all the data is stored in one place. It enables access where almost no joints are needed, and which is more sequential.

Businesses dealt earlier with less structured data such as messaging services, blogs, social networks, Web logs, films, photos, … The new data types which appeared need to be managed on a specific way, e.g., classification, MapReduce, to be able to integrate them in the business decision-making solutions.

2.1.3 Velocity

The continues flow of the Internet with its billions of users without any interruptions are generated by software agents like e-commerce sites, blogs, social networks and more. And all this generated data needs to be processed by businesses in real time a term which is still hard to define. This situation makes it for businesses hard to adapt and react on this highly competitive place that is the Internet. The users gain more power through their sea of choices where the businesses and

brands are depending on the users wants and needs, which drives businesses to meet the expectations of their users at any time.

2.1.4 Value

The heart of this, is the value of every data which makes the Big Data. Every piece of date out of the Big Data can be reduced to a part of data that has a cost. Each of the part of data are valued according to their use. Through globalization and digitalization and the rising opportunities makes the competition tougher. The same rules apply also for the Big Data which is seen as an additional source of information that beneficiate businesses and their decision processes. That is where Big Data is transforming into Smart Data. We can see Smart Data the way how different data sources amongst others Big Data are collected, correlated, analysed and further, to help businesses with their decision-making.

Web players like Google, Facebook, and so forth, are another valuable category of cognitive business, which offer their users a specific number of free services. Their ability to analyse and manage the stored data to produce a valuable information which can be sold to economic players.

2.2 Big Data Architecture

Understanding the basics of Big Data architecture, as in Figure 2, will helps to understand how data components fitting together and how data flows through different layers of an organization. The most common layers are Data sources, Data storage, Batch processing, Real-time message ingestion, Stream processing, Analytical data store and Analysis and reporting. (Enterprise Big Data Framework, 7 May 2019)

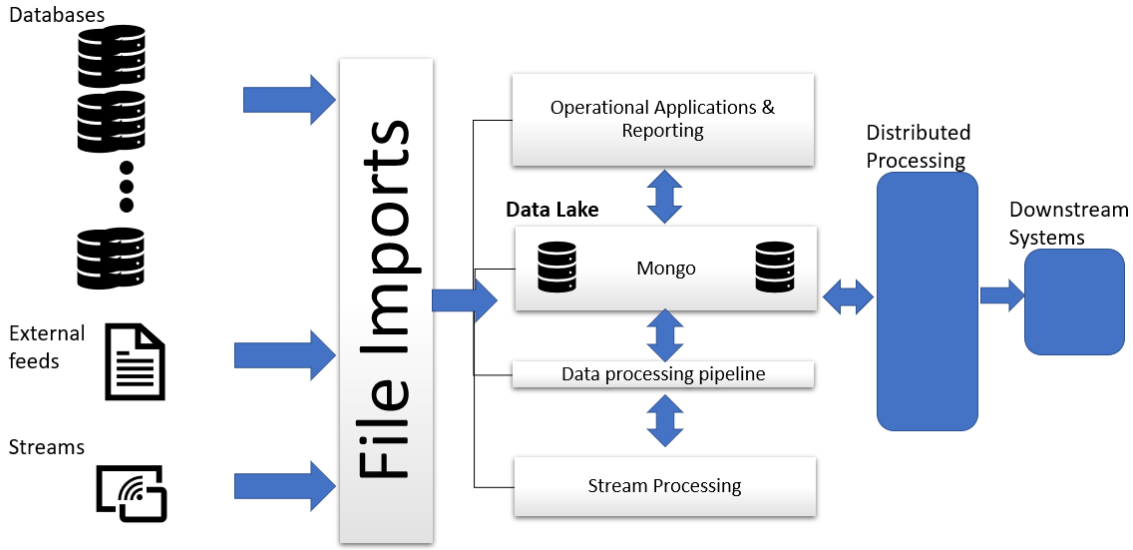


Figure 2. Example of a Big Data Architecture with its components

Data sources: Is a data collection from different inputs in different formats, structured and unstructured data. Including relational databases with ERP (Enterprise Resource Planning) or CRM (Customer Relationship Management) system, data warehouses, social media, email, and real-time streaming data.

Data storage: This layer includes the data and converts and stores it in the needed format. Data which is structured is usually stored in a relational database while unstructured is stored in NoSQL (Non-Structured Query Language) databases.

Batch processing: The stack processes should filter, combine, and render data to be able to analyse it afterward. The data is usually read and processed, and the output is stored in new files.

Real-time message ingestion: The data is transitioned into the layers of the environment with this component. With a real-time feature it is necessary to ingest and store the real-time data for streaming processes. For reliable delivery and queuing requirements messages can be dropped into folders or can be in a capture stored which enables scale-out processes.

Stream processing: The real-time messages need to be filtered, aggregated, and prepared for analysis. Included are e.g., Azure Stream, Analytics, Apache Storm and more.

Analytical data store: After the processing stage the data can be presented in a structured format. That happens with querying tools, a known example would be the business intelligence (BI) platform.

Analysis and reporting: The idea of most Big Data platforms is to extract valuable information for the business from the stored data through running analysis and reports on them. Whereas the analyses are straightforward by structured data, it needs more specialized techniques to analyse unstructured data. With a modelling layer like in the BI enables different visualization and modelling techniques.

The benefits of Big Data Architectures are the parallel computing for high performances like processing of large data sets faster. The distributed processing power of several serves are used to perform many calculations simultaneously. The elasticity of scaling horizontally which enables to adjust the environment to the size of the given workload. In addition to the scalability, it is a cost-effective solution which is mostly a cloud-based storage and computing power, adjusted to the needs, you pay what you use. And finally, there is the freedom of choice. Through many different platform solutions which the marketplace offers, enables companies to choose or combine the most fitting solutions for their need. (MongoDB, 2022)

Some downsides of Big Data Architecture are e.g., the security. Though the centralized data storage it needs not be ensured that the data stays safe from intruders. Another issue is the complexity which rises typically though the interlocking moving parts of Big Data architectures. The building-up and maintenance processes demand elevated level of knowledge and skill. That skillset is often very specialized through the atypical and specialized languages which challenges developers and data analysts. (MongoDB, 2022)

2.3 What is a Search Engine?

It is a computer application that enables the user to request information. It gets input data and provides the user with, hopefully, the right information. The task is to narrow the results down to a minimum and on an ideal case, to deliver the one exact information the users is searching for. Most people think about Google if search engine is mentioned, which provides a big amount of information from many sources. Beside the Google engine there are as well some smaller search engines which focuses their contents to a more specific area, those engines are also often called vertical search engines. (Thommaso Teofili, June 2019)

This characteristic of the vertical search engines, to search just in a specific area, provides the user with a more accurate result about their requests. The key functionalities of a search engine include indexing, which inputs and stores data more efficiently, so in case of use it can be fetched faster. Another functionality is the Querying, which enables the end user the ability of a search performance. And Ranking as another key factor, is presenting the request according to the user's satisfaction and needs. A practical point is efficiency as well, since its crucial to keep the user satisfied, otherwise he or she will look for another search engine if the waiting times are rising. (Thommaso Teofili, June 2019)

2.4 Search Engines in Big Data

Finding web pages and being able to manage an astonishing number of requests, and provide the user with the most accurate results, is the core of that algorithm which lies behind that search engines. If we look at some numbers of the most know search engine, Google, which receives about 1.1 billion unique visitors each month, and it manages each single day an estimated amount of data of 20 petabytes (2×10^{16} bytes). With the Google Cloud Dataflow, Google introduced the next step for organizations, which enables companies not to just store but also analyse and process their big data. (Alan Anderson & David Semmelroth, 26 March, 2016)

2.5 Limitations of Big Data

The first limitation which would take place is the storage problem. Even though the storage capacities are rising immensely, there is a limitation. Those storage limitations start with recommendation e.g., how long a confidential data should be stored regarding to its purpose and importance. And

goes till regulations of technical and organisational measurement implementations. In specially the last year, with the start of the COVID-19 pandemic, and the rising amount of remote works, made a big step forward with the GDPR (General Data Protection Regulation). Which demands organizations to change accordingly to those Regulations. I will not go in further detail with the GDPR since it would be out of the scope. (ICO, 2022)

In many cases data is collected through some routines like clinical care in the health sector, or collection of browsing, tracking, and purchasing behaviour with tools like cookies on social media platforms. Information becomes the “new” economy. Big providers of not just storages but whole infrastructures are scalable to not just companies, but also private peoples need. And even those big providers are reaching on some point their limits and they need to increase their capacity, that is how cities of data centres are growing. (ICO, 2022)

There is also the other side of the limitations of data which is e.g., keeping personal data for an extended period, it is no longer needed and will become irrelevant and inaccurate. The deletion of such data will not just make memory space but also hinders to use false or error data regardless of the costs. (ICO, 2022)

2.6 Big Data Challenges

As it is with most implementations in a Business, the changes need to be justified and have a good reason which are going along with the business strategy. After a consolation and cost/benefit analysis, the change needs to be put into weight if it is advisable for the company to make such changes.

Let us go through a few challenges of the big data. The misguiding of companies which are switching the naming of their data analysation processes into big data, because it sounds trendier, with the hope the business will flourish is a fraud. Companies which want to change to a big data project are often underestimating the high complexity and cost it brings with. Before taking that kind of steps it should be fully aware and understood of those changes which comes along.

The complexity of the technology is not ending there it is continuing in the high sets of legal rules. The complexity caused by the big data in special if the data is gathered and processed over international borders. The knowledge of the rules, policies and the joint processes need to be ensured that they are complying with each other. Finding the right person with the right skillset is a common issue for companies, but they need a clear definition of what skills, capabilities, and experiences they are requiring finding a suitable person for that position. (Paul Taylor, 17 June 2020)

Talking from my recently made own experiences, just in a much bigger magnitude, the inaccuracy of gathering and analysing data in such scale brings some pitfalls. In the case of big data, they are e.g., the linking of the data, such as purchases linked to the geographical location, is not done on a proper way, the data being just of mediocre quality and collecting a poor sample size or a misinterpretation of the data through the analysts. To be aware of such inaccuracies and disclaimers should be noted in any analysis process. (Paul Taylor, 17 June 2020)

One more source of error could be the bug in the algorithm and in measuring the wrong values or misinterpretations of the results. Going through the processes with multiple algorithms and watching it through several lenses will minimize the errors and keep it under control. The prediction of the future, which is in a certain way possible, with the change of behaviours can make it in the wrong mindset to a weapon, in controlling and even changing people’s behaviours. (Paul Taylor, 17 June 2020)

3 Databases

3.1 SQL

The Structured Query Language, SQL, is the programming language to manage data in a relational database management system (RDBMS). It is used to store data in so called tables in the database. Those tables are staying in relation to each other and can be cross-referenced. Storage was in the early ages of digital evolvement quite expensive and used with care. Nowadays those databases are much faster and can manage a big amount of data which are organized in linked rows and tables where the data is stored. (Benjamin Anderson, 12 June 2022)

SQL databases can scale vertically, which means to enable more CPU, RAM capacity. That can be costly and limiting to grow. In addition to all that to make SQL dependable and successful in transactions, there are four properties integrated, ACID. (Benjamin Anderson, 12 June 2022)

ACID stands for Atomicity, Consistency, Isolation and Durability. Atomicity means that all transaction needs to be finished on success or failure. It cannot be something in between. Consistency prevents corruption through following rules and validations. Isolation takes care that the transactions are not affecting each other. And durability is that even a system failure cannot roll back a finished transaction. Some know SQL databases are for example, MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server and many more. (Benjamin Anderson, 12 June 2022)

3.2 NoSQL

The non-relational database, NoSQL, can use different structures than the SQL database. They are not organized in rows and columns therefore are more flexible to use. Even though it is NoSQL, it supports still some SQL commands and refers more to “not only SQL”. Through the unstructured data dynamic, it enables the NoSQL database to manage less pre-planned and organized data and simplifies modifications. On the contrary NoSQL scale horizontally, which means it is distributing resources e.g., adding additional servers or nodes to increase the load. (Benjamin Anderson, 12 June 2022)

There are in general four diverse types of structures in NoSQL, Column-oriented, Key-value stores, Document stores and Graph databases. In Column-oriented the data is in unlimited numbers of columns stored instead of rows as in SQL databases. The key-value store is using an array with a key-value pair as their data model. The document store uses documents to hold and encode data in the standard formats like, XML, YAML, JSON and BSON. And the Graph database shows graphically how different datasets are related to each other. (Benjamin Anderson, 12 June 2022)

Even some off the NoSQL databases like, IBM DB2, MongoDB, AWS DynamoDB, are following the ACID rules, the CAP theory is applied in general. The CAP, which stand for Consistency, Availability and Partition tolerance theory, guarantees that only two of the following properties are managed at any given time. Consistency allows to receive either the most recent result or an error. With availability it is watched that every request has a non-error result, that there is no interruption by the occurrence of delay or losses between the operations, is taken care by the partition tolerance. Through the smaller user community there is lesser support for NoSQL user, but the benefits are the quick adaption and the open-source-systems available. (Benjamin Anderson, 12 June 2022)

3.3 Search Engine Types

Search engine databases are text based and semi structured and are made to help users finding information in fast and cost-efficient way. The search engine is optimized in a way to find key words with specialized methods like text search, complex search expressions, and ranking of search results. Inverted indexing enables that the searching in databases is not like a conventional page centric data structured search, instead it is a keyword centric data structure. This results in a faster search performance which is equivalent to search for the keyword in a book by searching the keyword in the index at the end of a book. (Bi-In Sider.com, 14 January 2019)

There are different search functionalities in search engine databases, I will describe the most common below:

- The full text search compares the searching text to the words in a file. It is mostly used to find data from new articles, academic papers, essays, or books.
- Semi structured search has both the fixed structure of a RDBMS like those in a Word or PDF document, which can be converted into XML or JSON format. It has a self-describing structure

which consists of tags and other markers to separate meaning of words and phrases the so-called semantic elements with hierarchies and fields.

- Associating locations to web resources to make location-based queries is done by the geographical search method. The search result is not just related to the topic but also to a physical location associated with the search request.
- The network search offers a relationship-oriented approach, which leads the user to analyse the connections in data within the stored documents. It can link between people, places, preferences, and products which returns information from across the network.
- Navigational search rises other search capabilities to find search results by applying multiple filters based on the search items. It uses a hierarchical structure which enables users to browse information by choosing from a before fixed set of categories.
- The vector search ranks document results based on how close they are to the search request, using a multi-dimensional vector distance model. Vector search is a way to enable fuzzy search, which lists the closest possible results to the keyword. They help to find things if you do not know the exact wording, it lists the neighbour keywords so to say. (Bi-In Sider.com, 14 January 2019)

3.4 Future Databases

Jonathan Ellis, the co-founder, and CTO of DataStax which is a cloud database as a service build on Apache Cassandra, shared his predictions for the involvement of databases for the next five years. His predictions outlined especially five points which will have an impact in the next few years. Those points are Self-optimization, Administration approaches zero, Open source grows in popularity, Data canters become standardized, and Group evolves into a feature. (Jonathan Ellis, 24 February 2020)

- Self-optimization: Currently are most databases quite complex and have many configuration parameters to set, which makes it a challenge for an average administrator to deal with. Jonathan says that databases will be more self-driven, and AI will configure the database to deliver the best performance for the given workload.
- Administration approaches zero: The databases will be cloud native based and it will be self-driven, and administrators will be not needed instead it will optimize the configurations automatically. Fully managed platforms as a service will become the norm according to Jonathan. (Jonathan Ellis, 24 February 2020)

- Open source grows in popularity: Open source will get a renaissance but not for ethical reasons as it was in the nineties. Companies will realize the pragmatic side on open source and its adaptabilities.
- Data centres become standardized: The services will be provided to companies regardless the hardware and will be a standardized product.
- Graph evolves into a feature: According to Jonathan the graph is a feature and not the database. There are pros and cons of the graph, on one side they are good in tree retrieval and pattern matching but on the other hand it does not have a satisfactory performance in logging user activities. Enterprise database providers will combine both worlds of graph and non-graph to get the best out of both. (Jonathan Ellis, 24 February 2020)

4 Basic Sorting Methods

4.1 Insertion Sort

One of the methods to sort for example an array, which is a storage of data collection like for e.g., myArray = [3, 1, 4, 7, 2, ...], in our case we are storing numbers in the array named myArray, and we will apply the insertion sort, to sort myArray from the smallest to the biggest number.

An insertion sort works like that it extracts the array number by number and shifts the remaining numbers and inserts the extracted number in the correct spot. That continues till all numbers are in the right order, in our case that would be myArray = [1, 2, 3, 4, 7, …]. I will try to make it clearer; the algorithm takes the first number and compares it with the second number if the second number is smaller than the first one, it will swich them so that the numbers are in ascending order (smaller one comes first). After the first comparison it will compare the next two numbers till the array is sorted. An illustration of an insertion sort process is shown in the Table 1 with our example myArray = [3, 1, 4, 7, 2]. (GeeksforGeeks, May 2022)

Table 1. The sorting process of insertion sort with myArray= [3, 1, 4, 7, 2]

myArray	3	1	4	7	2
1. Iteration	1	3	4	7	2
2.	1	3	4	7	2
3.	1	3	4	7	2
4.	1	3	4	2	7
5.	1	3	2	4	7
6. Sorted	1	2	3	4	7

4.2 Bubble Sort

As one of the easiest sorting algorithms to understand and code, which is worth to mention since it gives an idea on how sorting algorithms are working is the bubble sort algorithm. Let us go first through the myArray and see on the example how it works.

First it will start comparing the first two elements from an array, myArray = [3, 1, 4, 7, 2], (we went through what an array is previously). If the one of the two elements is smaller than it will swap the small element to the left in our case 1 will be swapped because its smaller than 3 → myArray = [1, 3, 4, 7, 2]. Now it will compare the next two elements 3 and 4, where 3 is smaller than 4 and therefore these are already in order and the algorithm moves on.

The algorithm passes through myArray repeatedly even if myArray is already sorted, because the algorithm does not know till it passes completely through the elements without swapping any values. After that, the sorting is finished. It is far away to be an efficient sorting method but a simple one to put your head around. Let us go through with an example as you can see in Table 2 below. (Freecodecamp, 25 January 2020)

Table 2. The working process of bubble sort

myArray	3	1	4	7	2
1. Iteration	1	3	4	2	7
2.	1	3	2	4	7
3. Sorted	1	2	3	4	7

4.3 Selection Sort

The selection sort algorithm is an in-place comparison-based algorithm where the elements are divided into two parts. The left side as the sorted part which is empty at the beginning and on the right side the unsorted part which holds the entire elements in the beginning. The selection sort is not meant for large data sets because with increasing data the time would proportionally increase as well. (TutorialsPoint, 2022)

The algorithms start to go through the unsorted side to find the element with the lowest value. After finding the lowest element it will put it to the sorted side and the first iteration is done. After each iteration, the algorithm checks the whole unsorted side and picks the smallest valued element left and puts it into the sorted side. On the Table 3 is a graphical demonstration of the select sort algorithm with myArray. (TutorialsPoint, 2022)

Table 3. Select sorting algorithm with its sorted and unsorted elements

Sort

myArray	3	1	4	7	2
1.Iteration	1				
2.	1	2			
3.	1	2	3		
4.	1	2	3	4	
5. Sorted	1	2	3	4	7

Unsorted

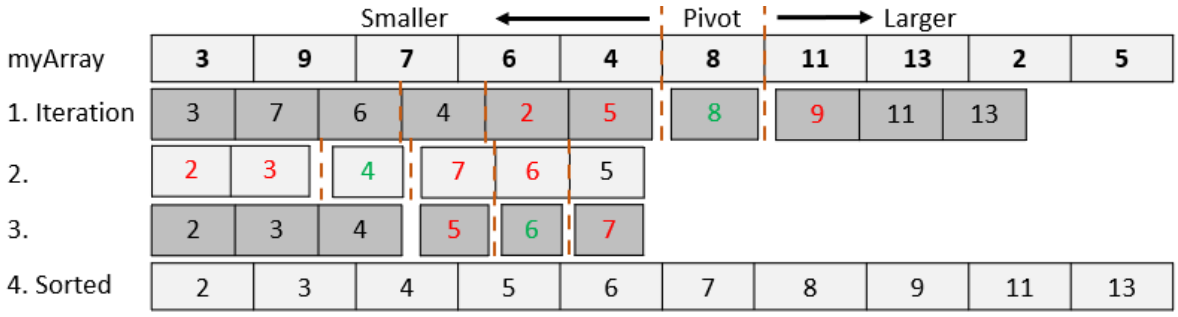
myArray	3	1	4	7	2
1.Iteration	1	3	4	2	7
2.			3	4	7
3.				4	7
4.					7
5.					

4.4 Quicksort

Another simple and widely used sorting algorithm is the quicksort. It is a fast and efficient method which breaks down the algorithm into subproblems and solves those subproblems and combines the results back together, that process is called divide and conquer approach. quicksort uses the first, middle or last element of an array known as pivot. Each element of the array will be compared to the pivot-point and rearranged afterword's on that way that the smaller elements are sorted on the left and the larger ones on the right side. After partitioning both sides, the method recursively runs the same process on each side. The demonstration in the Table 4 will show a clearer understanding of the quicksort algorithm. myArray = [3, 9, 7, 6, 4, 8, 11, 13, 2, 5]. (Joseph Pyram, Feb 11, 2020)

- First step →choose a pivot-point → 8.
- Next step → everything what is smaller than the pivot-point to the left side.
- Everything what is larger than the pivot-point to the right side.
- A big advantage is that the smaller side will not be compared with the larger side.
- Next step → choosing a new pivot-point → 4.
- Next step → repeat step 2.

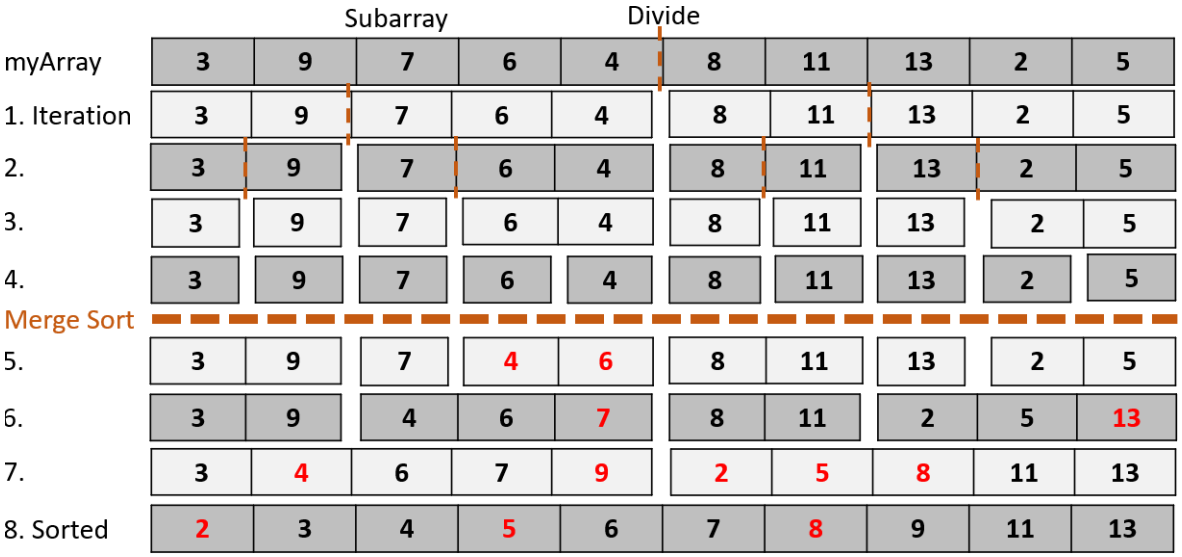
Table 4. Highlight of a quicksort algorithm process



4.5 Merge Sort

As previously with the quicksort, merge sort is also divided into subproblems, using divide and conquer, to sort the elements in an array. Divide and conquer is used in the following way, the array is first divided in half. After we have the subarray, we conquer by recursive sorting the subarray. When the sorting is done, the initially divided array into subarrays, will be merged back into one array. To get a better understanding of the merge sort process, see Table 5 bellow. (Khan Academy, 2022)

Table 5. Merge sort process of myArray= [3, 9, 7, 6, 4, 8, 11, 13, 2, 5]



4.6 Counting Sort

Count sort was discovered in 1954 by Harold Seward. It is a fast and reliable linear sorting algorithm. Count sort is not like the others e.g., merge sort or quicksort, meaning it is not a comparing based algorithm. Instead of comparing it uses the advantage of the array's time insertions and deletions see Figure 3.

Insertion:

myArray = [3, 1, 4, 7, 2]

if we insert between 1 and 4 …

[3, 1, _, 4, 7, 2]

the elements move everything one step to the right.

[3, 1, 5, 4, 7, 2]

and the new value, 5, is inserted to the array

Deletion:

myArray = [3, 1, 4, 7, 2]

and if we remove one element …

[3, 1, _, 7, 2]

In our case four, the hole will be removed.

[3, 1, 7, 2]

Figure 3. Example of array insertion and deletion

Counting sort is used in computer science collecting objects according to their keys which are integer elements. It is finding the positions of the key value in the output sequence by counting the number of objects with distinct key values and applying prefix sum to those counts. It is only suited for direct usage when the number of items is not much more than the variation in keys. Counting

sort is often used as a subroutine in radix sort, which is a more efficient sorting algorithm for larger keys. In Table 6 you see counting sort with an example, myArray = [2, 9, 7, 4, 1, 8, 4].

Table 6. Step by step explanation of the Counting Sort algorithm

myArray	2	9	7	4	1	8	4

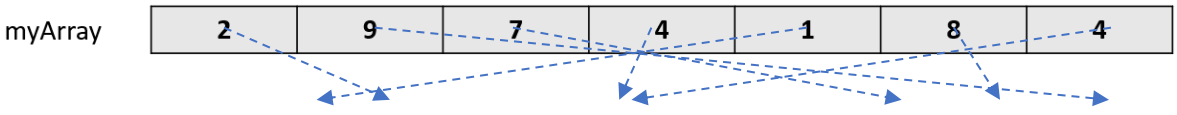
At first, we find the biggest value within the elements. → in our case 9

9	2	7	4	1	8	4

The next step is to initialize an array with 9 + 1 elements with the value zero for all elements. It will be used to store the count of the elements.

Index	0	1	2	3	4	5	6	7	8	9
Count array	0	0	0	0	0	0	0	0	0	0

The count of the array elements will be stored at their corresponding index in the count array.

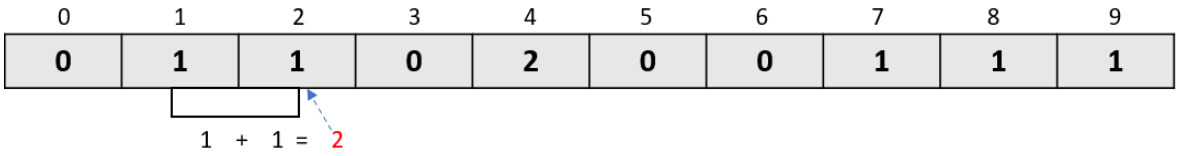


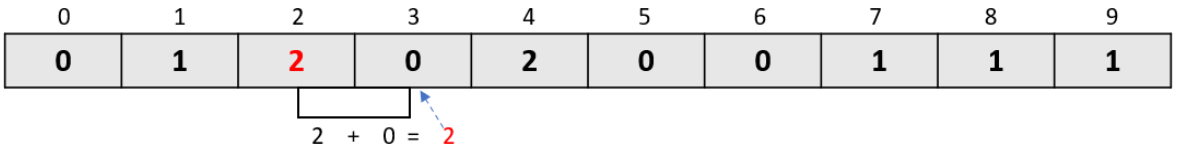
Index	0	1	2	3	4	5	6	7	8	9
Count array	0	1	1	0	2	0	0	1	1	1

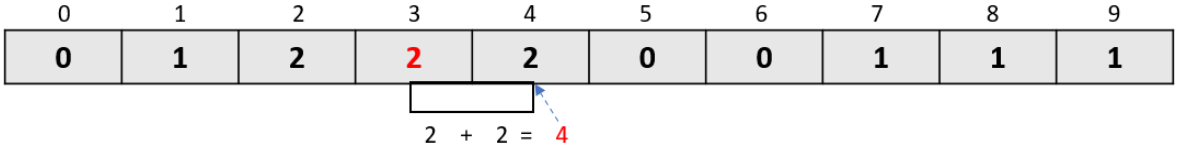
The count of the elements which means we go through the given array and count how often each number is represented in the given array. After the counting we fill into the count array on the index which matches with the element of the given array and put it into the count array. Let us take the

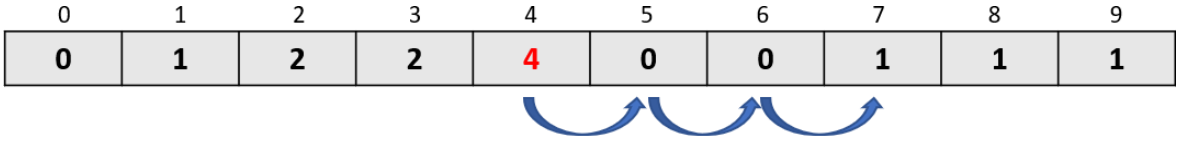
first element in the given array, 2, two is once in the given array therefore we put on the index two in the Count array a one. Next, we can watch the element four in the given array, which appears twice. On the index four in the counting array, we put a two. For the indexes which are not in the given array we write a zero.

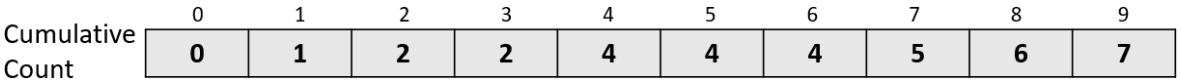
The last part is to store the cumulative sum of count array elements which will help to place the elements at the right index of the sorted array. (Simpl;learn, 19 October 2021)



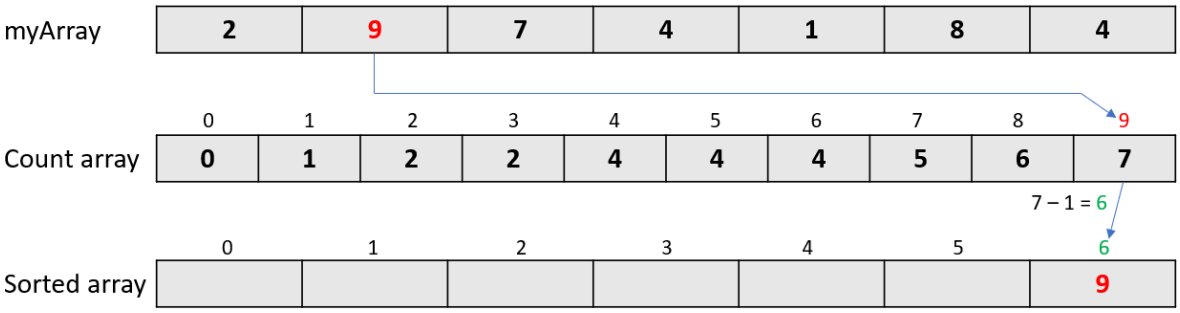




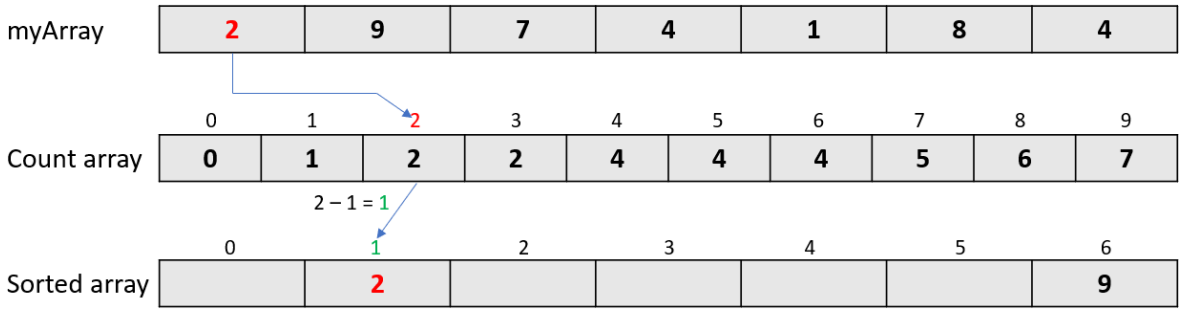


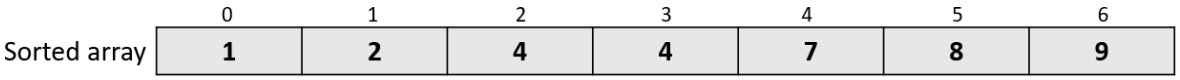


Next finding the index of each element of the original array


myArray
2 9 7 4 1 8 4
0 1 2 3 4 5 6 7 8 9
Count array
0 1 2 2 4 4 4 5 6 7
7 – 1 = 6
0 1 2 3 4 5 6
Sorted array
9

And next …


myArray
2 9 7 4 1 8 4
0 1 2 3 4 5 6 7 8 9
Count array
0 1 2 2 4 4 4 5 6 7
2 – 1 = 1
0 1 2 3 4 5 6
Sorted array
2 9


0 1 2 3 4 5 6
Sorted array
1 2 4 4 7 8 9

5 Advanced Methods

5.1 Sort and Search Algorithms

Moore's Law predicted in the sixties, that the number of transistors will double every two years for the next 10 years and will than end. But as we know it shell continue for another 50 years, but his prediction is proven right. The physical size is reaching its limits and might be replaced by a quantum sized transistor. The only issue respectively will be that it is not working properly with the current technology. (Rod Stephens, 2013)

Another way to increase computing power would be to use multiple processors instead of increasing the transistors in one processor, which is the case in nowadays computers. The challenge is to have an algorithm which is made for a multicore CPU, which means able to work in parallel. The processing of multiple sets of instructions at the same time is what is called parallel computing. (Rod Stephens, 2013)

The widest used and known algorithm nowadays is the ranking of pages off Google's search engine. The name of this algorithm is called PageRank. The main task was to rank pages depending on their importance. It is still used in some other parts at Google's algorithms. The future of Google is all about artificial intelligence, which will deliver the information before you know you will need it.

5.2 Neural Search

The term neural search appeared the first time at the SIGIR (Special Interest Group on Information Retrieval) conference in 2016. SIGIR is focused on the retrieval and distribution of non-numeric information like natural language to highly structured data bases. We talk from neural search if a deep artificial neural network is helping to solve problems in the search category. I will introduce how neural networks are composed, how they are working and used in real live with the focus lying in search engines. In Figure 4 you can see the core figures in a deep artificial neural network, which is subdivided in Machine Learning and Deep Learning.

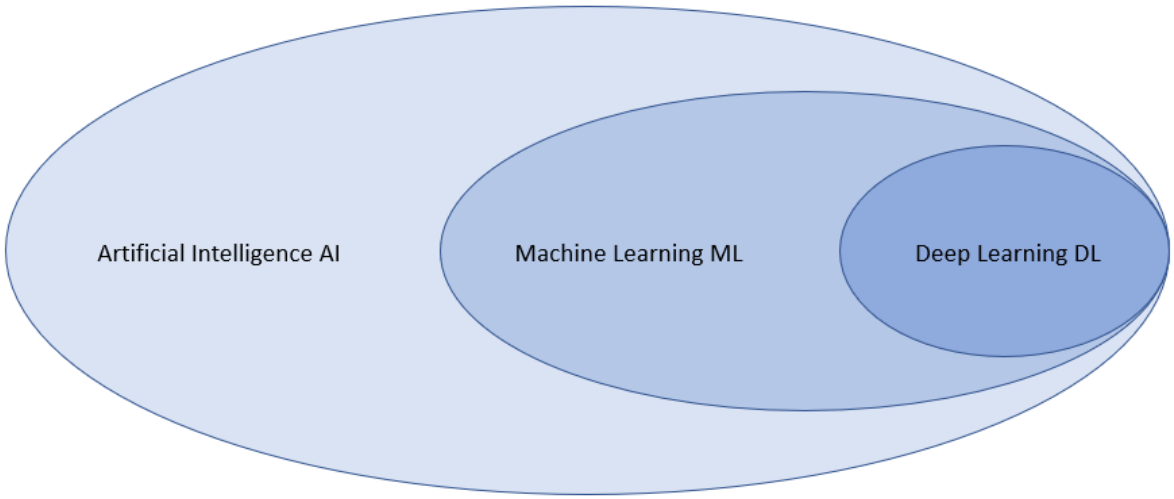


Figure 4. The different layers of Artificial Intelligence AI

Usually, we find mostly everything we are searching with google in most of the cases. So why there is a need of a neural search? What makes neural search so valuable and why is it worth to take it seriously. To mention some key points of deep neural networks e.g.:

- Word and documents are captured semantically which enables a machine to tell which word and documents are semantically similar.
- Meaningful text generation on the right context.
- Images which pertain to their composing objects allow to develop highly functioning face/object-recognition systems.
- Enabling machine translations more efficient.
- Under unusual circumstances it makes approximations of functions possible. In theory there is no limitation to the tasks of deep neural networks.

Let us put that information about deep neural networks into a bit more tangible examples, on how engineers and users are benefiting from those. The major pain points when using search engines which you might be familiar with are:

- Not getting satisfactory results, instead just related stuff which was not searched.
- Time consuming search attempts which are ending on giving up the search.

- Going through a lot of pre-text, reading yourself through the data jungle till the actual data is found.
- Main and most comprehensive results are in the English language, which might be not for everyone easy to go through.
- A desired image once glimpsed on a website, is not able to be located.

Now you might know the one or the other case. But the good news is that if, the deep neural networks are adjusted properly, it can resolve all the mentioned issues above. The deep learning (DL) algorithm enables the search engines to:

- Deliver more useable results for the users which increases the usability and the user satisfaction.
- Being able to search for pictures i.e., binary content, the same way as for texts.
- Serving content in different languages reaching a wider user group.
- An increased sensitivity of served data which lead to more result than no result.

The deep learning will adapt to your data through customization, the solutions are based on your data and not on some general algorithms. The quality of search results is essential for the user. The one task in which search engines should shine, is to find the best and most useful results specific tailored for the user's information and needs. This leads to not just more but also quicker search results which is the reason the emphasis lies on the topic of relevant results. According to a published Forbs article, "By providing better search results, Netflix estimates that it is avoiding cancelled subscriptions that would reduce its revenue by $1B annually." (Thommaso Teofili, June 2019)

What was earlier typed by human as metadata into e.g., pictures describing content, what enabled user to find those pictures, is nowadays the deep neural network which can abstract a representation of an image that captures what is in there, where no human interaction is needed anymore. An exciting possibility would be to change the way search engines are returning information back to the user. Usually, the search engine gives a list of results back to the user according to the search criteria. DL techniques can be used to let the information returned from the search engine, with exactly what the user was searching for, e.g., a single line of text or an image. That would save the

user a lot of headaches, like reading the results to find the one line or picture the user needs. (Thommaso Teofili, June 2019)

Neural networks are already a few years around, but just recently become effective which makes them appealing to use. Researchers discovered a more efficient way of using the neural networks, so they gain more interest which opens more possibilities. One key factor was in the early 2000's, to be able to use more powerful computers. (Thommaso Teofili, June 2019)

6 Research Methodology

The idea in this research part is to show a practical experimental setup, which shows the different approaches of algorithms and their results. Beforehand I want to introduce you to the concept of classifying algorithms in their performance, with the so called “Big O Notation”. The Big O Notation describes how complex an algorithm, in the worst-case scenario, is. In the Figure 5 you see the different notations from extremely deficient performance till excellent performance. If we look at the terrible area, which is the upper part and in red, the performance of the O (n!), O (2^n) and O (n^2) is extremely poor. Let us take for example the O (n^2) notation. The n represents the input size and the function inside the big O notation, in our case (n^2), shows the complexity of the algorithm.

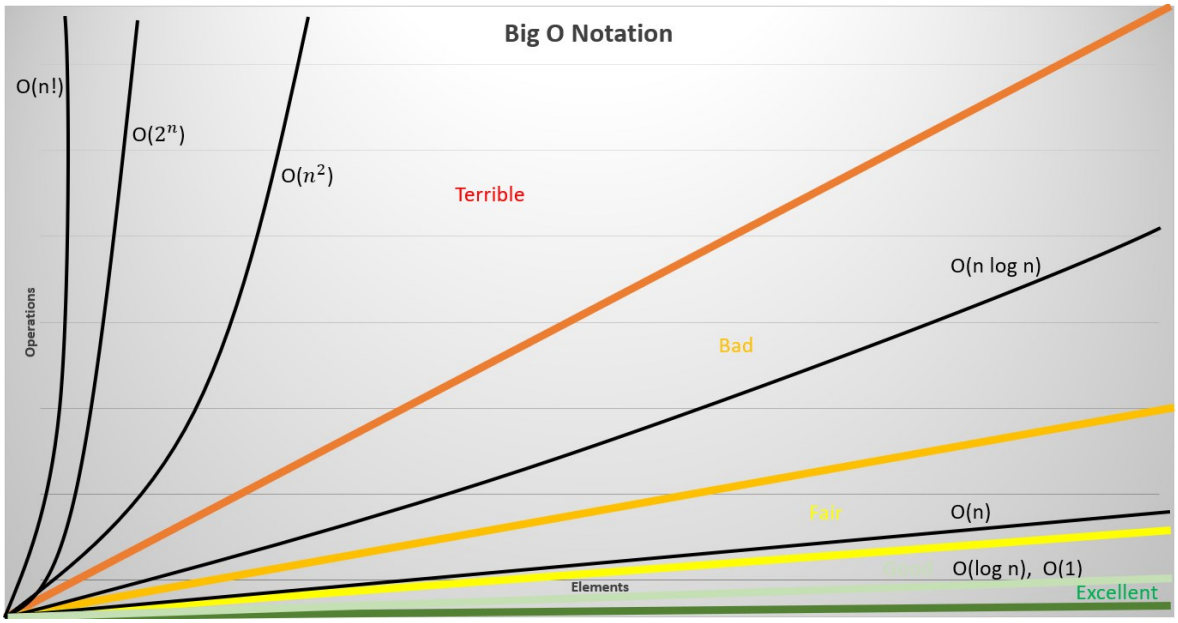


Figure 5. Graphical overview of the Big O Notations

An example for an O (n^2) is the selection sort algorithm. The selection sort algorithm iterates through all elements to compare the first element with the rest of the elements to find the smallest part (brute force). On the next step it jumps to the next position and repeats the procedure till the list of elements is sorted. The number of elements in the list is represented by the size n. If we take the first element n and compare it with the rest 1 + 2 + 3 … + n, we will compare it n^{th} times, which takes one loop, till we have the smallest element in our case. But since we need to compare each

element against each other we will need another loop it is n^2 to sort the list. That results in our Big O Notation, O (n^2) for the selection sort algorithm.

6.1 Bubble, Quick and Merge Sort

The chosen algorithms which are compared with each other were in this case bubble sort, quick sort, and merge sort. All algorithms task was to sort an array of integer numbers. The amount of the numbers starting in the Table 9. The arrays were randomly generated, and the same arrays were used by all sorting algorithms, bubble, quick and merge sort. The complexity of the bubble sort algorithm is O (n^2), of quick sort O (n log n) and merge sort O (n log n). That means according to Figure 5, that the quick and merge sort performance should get better with the rising amount of the elements in the array whereas by the bubble sort algorithm the time should rise exponential. The objective was to show differences in sorting speed and performance.

After spending a while on finding a tool which would fulfil my needs in comparing algorithms without a success, I decided to code my own tool. I used a simple HTML page where I implemented the algorithms with JavaScript and tried to make it functional for the purpose of showing a difference between the different algorithms. I will not go into more details of the code, but if you wish to see and try for yourself, I will leave my GitHub link as well as the essential parts of the code in the appendix.

The tool has pages to access bubble sort, quick sort, and merge sort, measuring the time during the algorithm sorts the given random array. The starting point is the bubble sort site, where you choose first how many numbers your array should have e.g., 1 000. Now after pressing start, a random array with the range between 1-100 will fill these 1 000 items of the array. That means you have 1 000 numbers, and each number is randomly chosen in the range of 1 to 100.

The random generated array is visible on the bubble sort site. That is how I get the test example array and now the actual task is to sort these 1 000 numbers in order, from the lowest number to the highest. This will take some time and I am measuring the time differences between, clicking the Start Sorting button and the finished sorted array. After the first part is done, I switched to the quick

sort site with the same array which is already visible there as well. We can press here as well the start sorting button and wait till it is done. When finished it will show the time it took with the quick sort algorithm to sort the previous array in ascending order, meaning from lowest to highest. The last part is to go through the same process to measure the merge sort algorithm.

6.2 The Environment

The idea was to run the same code in a Windows 11 and Linux Ubuntu 20.04.5 LTS environment with closing all unnecessary tasks to save CPU power which might influence the results. In the Table 7 and 8 are the specifications for both Operating Systems.

Table 7. Windows OS Specifications

Operating System	**Microsoft Windows 11 Home**
Version	21H2
OS Type	64-bit
PC	Dell XPS 13, Notebook
Processor	11th Gen Intel® Core™ i7-1185G7 @ 3.00 GHz
RAM	32.0 GB
System Type	x64-based processor

Table 8. Linux OS Specifications

Operating System	**Ubuntu 20.04.5 LTS**
Version	3.36.8
OS Type	64-bit
PC	Dell XPS 13, Notebook
Processor	Intel® Core™ i5-5200U CPU @ 2.20 GHz x 4
RAM	8.0 GB
System Type	64-bit

The browser used on both OS, the Windows, and the Linux system, is Google Chrome. I started with an array of 100 elements, randomly generated, let them sort with bubble sort, wrote down the time it took and let the same array sort five times and took the average speed time. The same with the quick sort but taking the same array previously generated, sorting it and took the speed time. And lastly with the merge sort algorithm. That was the first round, next was the same process but with an array of 1 000 elements. The target was to reach an array which took significantly long and were the computer started to struggle and heat up.

6.3 Results

6.3.1 Windows 11 Environment

I started with the Windows environment. The first step was to close all unnecessary applications, where I used the task manager, since it gives an overview of all actions. In the next step I opened all applications I needed to measure and document my test. The used applications were, Excel to document the findings, Windows Task Manager to have an overview of the CPU process loads, Core Temp which is a nice application which gives an inside of the core temperature and the power consumption of the CPU and other information in real time and Google Chrome to use the self-made tool.

After setting up everything, I proceeded to choose the number of elements my array should have, starting with 100, and the bubble sort algorithm to sort the array and measuring the time. Without changing anything i.e., with the same array I switched to the quick sort algorithm and started the sorting there as well. And finally measuring the merge sort algorithm. For each chosen number of elements which was in our example 100, I repeated the process five times and took the average of the measured speed. In the Tables 9, 10 and 11 you see the measured data collected in the Windows 11 environment.

Table 9. Bubble sort measurement in Windows environment

Size	1.	2.	3.	4.	5.	Avg.	
100	0	0	0	0	1	**0**	**ms**
1 000	2	1	0	1	1	**1**	**ms**
2 000	3	3	3	4	2	**3**	**ms**
5 000	5	9	4	5	4	**6**	**ms**
7 000	27	36	35	43	37	**35**	**ms**
10 000	68	76	77	69	68	**73**	**ms**
15 000	231	741	733	702	350	**602**	**ms**
25 000	681	822	696	671	656	**718**	**ms**
50 000	2 946	3 696	2 995	3 700	3 726	**3 334**	**ms**
75 000	6 778	8 865	7 527	6 782	6 844	**7 488**	**ms**
100 000	13 621	13 326	12 273	14 231	13 266	**13 363**	**ms**
150 000	30 267	29 633	29 096	33 282	35 659	**30 570**	**ms**
200 000	62 459	64 491	63 329	52 407	62 125	**60 672**	**ms**
250 000	99 017	86 092	89 101	105 888	104 767	**95 025**	**ms**
300 000	143 387	154 384	152 977	135 279	149 968	**146 507**	**ms**
400 000	267 801	235 210	232 263	237 658	245 857	**243 233**	**ms**
500 000	458 739	437 134	347 185	355 827	368 358	**399 721**	**ms**

Table 10. Quick sort measurement in Windows environment

Size	1.	2.	3.	4.	5.	Avg.	
100	0	0	0	0	0	**0**	**ms**
1 000	0	1	0	1	1	**1**	**ms**
2 000	0	1	1	0	0	**1**	**ms**
5 000	1	0	0	0	1	**0**	**ms**
7 000	0	0	0	0	0	**0**	**ms**
10 000	1	1	1	1	1	**1**	**ms**
15 000	1	1	1	1	1	**1**	**ms**
25 000	1	2	2	1	2	**2**	**ms**
50 000	3	4	3	5	5	**4**	**ms**
75 000	4	5	6	7	5	**6**	**ms**
100 000	6	6	9	7	7	**7**	**ms**
150 000	1	13	12	1	14	**11**	**ms**
200 000	16	14	15	16	19	**15**	**ms**
250 000	2	29	21	17	16	**22**	**ms**
300 000	2	26	19	26	21	**23**	**ms**
400 000	31	33	37	26	39	**32**	**ms**
500 000	28	79	32	35	33	**44**	**ms**

Table 11. Merge sort measurement in Windows environment

Size	1.	2.	3.	4.	5.	Avg.	
100	0	0	1	0	0	**0**	**ms**
1 000	2	1	1	1	1	**1**	**ms**
2 000	4	2	1	2	3	**2**	**ms**
5 000	8	5	5	5	5	**6**	**ms**
7 000	1	6	7	7	7	**8**	**ms**
10 000	18	11	12	11	1	**13**	**ms**
15 000	48	63	44	33	54	**47**	**ms**
25 000	53	56	52	41	46	**170**	**ms**
50 000	778	772	571	573	788	**674**	**ms**
75 000	1 541	1 406	1 666	1 707	1 773	**1 580**	**ms**
100 000	3 442	4 030	4 300	3 521	2 297	**3 823**	**ms**
150 000	7 707	8 593	7 757	7 754	7 718	**7 953**	**ms**
200 000	18 512	16 737	15 840	18 366	16 562	**17 364**	**ms**
250 000	29 730	25 478	23 887	24 583	24 601	**25 920**	**ms**
300 000	44 211	36 381	35 425	29 638	41 415	**36 414**	**ms**
400 000	71 124	85 095	69 493	86 850	74 544	**78 141**	**ms**
500 000	183 177	112 690	109 570	114 224	110 726	**129 915**	**ms**

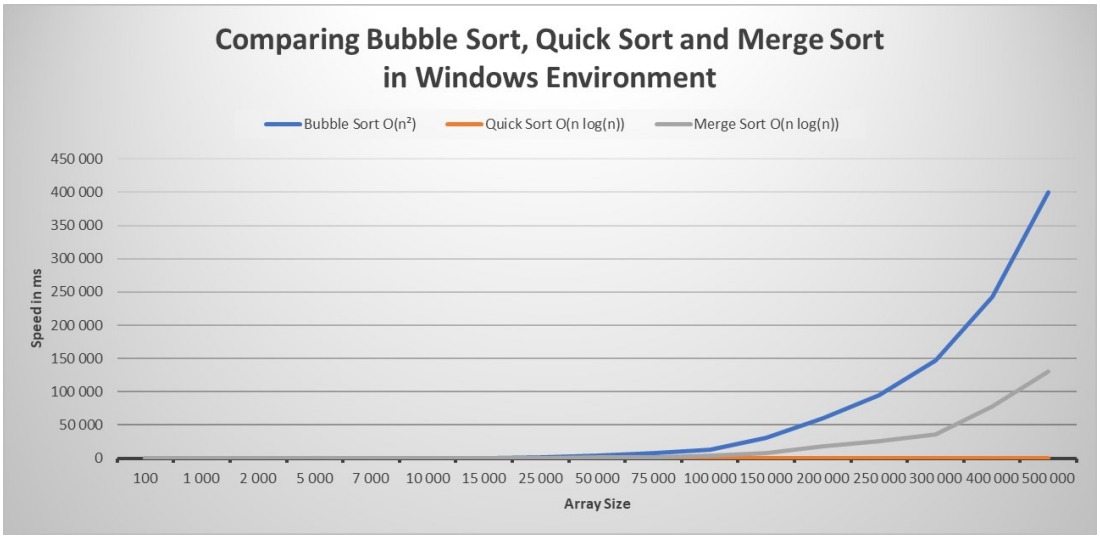


Figure 6. Shows the differences between the bubble, quick and merge sort algorithm

In Figure 6 you see a comparison of the bubble sort, quick sort, and the merge sort algorithm. The x axis represents here the rising number of elements on each point starting with 100 elements as minimum to 500 000 elements as the maximum. The y axis is showing the time it took to sort the

respective algorithm in milliseconds. The outcome is as expected, with the bubble sort taking longer time with the rising amount of the elements whereas the quick sort and merge sort algorithms remain constant regardless the rising amount of the elements.

All three algorithms remain constant till the amount of 50 000 elements and go apart with further elements added. After that point, the struggle of the bubble sort algorithm rises exponentially as we see in the Big O Notation in Figure 5. As expected, with the rising number of elements in the array the bubble sort algorithm got slower, which is here proven to be correct.

6.3.2 Linux Ubuntu 20.04 Environment

As in the Windows environment I used the same code which I uploaded first on GitHub and cloned it in my Linux Ubuntu environment. I took the same procedures as described previously in the Windows environment. I closed all unnecessary applications and installed tools like ‘sensors’ in the Terminal to measure the core temperature of the PCU and the power consumption, the system monitor, which is an equivalent to the task manager in Windows, to measure the CPU performance, and time.

The real difference though was in special the time measurements for bubble, quick and merge sort algorithm as shown in the Tables 12, 13 and 14. The slight difference between the Windows and the Linux measurements respectively the graphs is due to the weaker specs of the Linux environment.

Table 12. Bubble sort measurements in Linux Ubuntu environment

Size	1.	2.	3.	4.	5.	Avg.	
100	3	1	0	1	1	**1**	**ms**
1 000	12	13	4	4	2	**8**	**ms**
2 000	13	11	12	12	13	**12**	**ms**
5 000	41	40	42	4	4	**41**	**ms**
7 000	83	82	76	77	75	**80**	**ms**
10 000	173	172	170	170	169	**171**	**ms**
15 000	394	391	0402	404	400	**398**	**ms**
25 000	1 141	1 136	1 124	1 130	1 128	**1 133**	**ms**
50 000	4 743	4 754	4 833	4 832	4 835	**4 791**	**ms**
75 000	10 823	10 792	10 768	10 809	10 875	**10 798**	**ms**
100 000	19 345	19 335	19 296	19 383	19 350	**19 340**	**ms**
150 000	43 700	43 727	43 633	43 601	44 036	**43 665**	**ms**
200 000	77 437	77 500	78 641	78 458	79 437	**78 009**	**ms**
250 000	121 353	121 145	121 143	121 128	121 266	**121 192**	**ms**
300 000	174 570	174 353	177 129	176 876	177 002	**175 732**	**ms**
400 000	310 012	309 154	311 251	309 917	310 997	**310 084**	**ms**
500 000	483 654	485 247	492 213	492 740	491 668	**489 214**	**ms**

Table 13. Quick sort measurements in Linux Ubuntu environment

Size	1.	2.	3.	4.	5.	Avg.	
100	0	1	1	1	0	**1**	**ms**
1 000	0	1	1	1	1	**1**	**ms**
2 000	1	1	1	1	1	**1**	**ms**
5 000	2	2	2	2	2	**2**	**ms**
7 000	3	2	2	2	2	**3**	**ms**
10 000	4	3	6	4	3	**4**	**ms**
15 000	5	5	6	5	5	**5**	**ms**
25 000	9	8	7	9	8	**8**	**ms**
50 000	14	13	13	14	15	**14**	**ms**
75 000	18	19	17	19	16	**18**	**ms**
100 000	21	20	22	21	20	**21**	**ms**
150 000	28	29	30	27	30	**29**	**ms**
200 000	34	35	36	35	31	**35**	**ms**
250 000	43	41	39	40	44	**41**	**ms**
300 000	51	47	48	44	47	**48**	**ms**
400 000	62	60	62	61	59	**61**	**ms**
500 000	68	68	70	71	70	**69**	**ms**

Table 14. Merge sort measurements in Linux Ubuntu environment

Size	1.	2.	3.	4.	5.	Avg.	
100	1	2	1	2	1	**2**	**ms**
1 000	13	5	4	4	3	**7**	**ms**
2 000	18	9	7	6	7	**10**	**ms**
5 000	22	15	15	15	15	**17**	**ms**
7 000	26	16	19	16	17	**19**	**ms**
10 000	35	22	21	21	22	**25**	**ms**
15 000	45	27	26	26	28	**31**	**ms**
25 000	57	46	38	40	44	**45**	**ms**
50 000	185	167	170	167	167	**172**	**ms**
75 000	348	335	340	340	340	**341**	**ms**
100 000	744	741	738	737	744	**740**	**ms**
150 000	1 659	1 601	1 659	1 662	1 632	**1 645**	**ms**
200 000	3 310	3 294	3 276	3 305	3 315	**3 296**	**ms**
250 000	5 251	5 288	5 256	5 275	5 242	**5 268**	**ms**
300 000	7 713	7 695	7 761	7 743	7 788	**7 728**	**ms**
400 000	14 765	14 794	14 774	14 642	14 830	**14 744**	**ms**
500 000	23 582	23 530	23 391	23 612	23 558	**23 529**	**ms**

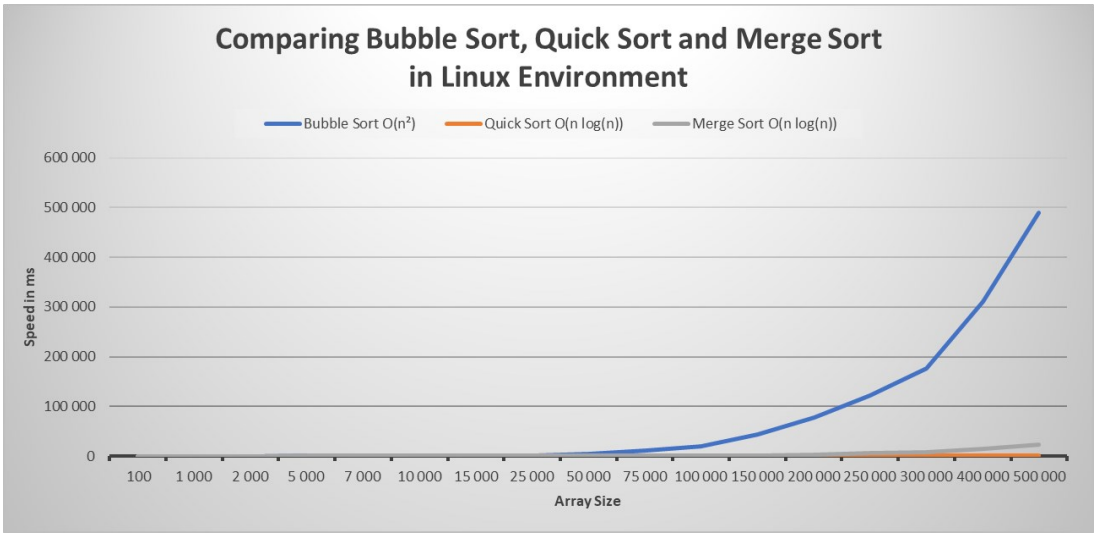


Figure 7. Differences between bubble, quick and merge sort in Linux environment

7 Discussion

Starting with the theoretical part which was necessary to get a got base and understand of fundamentals of the algorithms, big data structures and databases. The connection between those parts is crucial, to be able to make a useful analysis in a decent time, of the stored information in the database. Going through the theoretical part I realised that there is need of a practical part as well to show the functionality respectively to make it tangible.

The idea took place, developed during the search for a fitting ready tool, for the purpose of having different algorithms and the possibility to measure the performance. With the doubts rising to spend too much time on the search for finding the right tool, I decided to build my own tool, perfectly fitting for my needs. The design aspect of the tool has room to improve, which was not my focus point. The functional part of my tool however, helped me to present the differences of the chosen algorithms in a reasonable way.

The first take of measurements in the Windows as well as the Linux environment came with unexpected results. The measure points and graphs of all algorithms were almost identical. Going through the code to bugfix my errors, I started first to replace each respectively algorithm with different once without any drastic changes. My next approach was to take out every part which do not belong to the actual sorting algorithm and time measurement. Taking a step back and just go through the code, helped me finally to realize that the issue was the placement of the stopping time in the end of the sorting algorithm were the visualization part of the array elements with a bar chart was considered. That means basically that the building up time of the graph was included which falsified the results. So, I started all over.

The second time the measurements were successful and going in line with the expectations according to the theory part. The Big O Notation as shown in Figure 5, is the object of comparison which are shown in both environments in Figure 6 and 7. Spending quite a time with the algorithms and comparing the results, in specially of the quick sort algorithm, was a little satisfactory, it was amazing to see the speed differences in your own build tool. While the bubble sort algorithm rises “exponentially,” the bigger the elements become, the quick and merge sort algorithm in contrast remained constant.

The quick sort algorithm further is an algorithm which has a wide range of practical use. The divide and conquer strategy of quick sort performs like an advanced algorithm which I verified during the practical part. The measured speed regardless of the number of elements was quite impressive as you can notice in Table 10 and 13. The time changes, resulted in a difference of just 10 ms (milli-seconds, 10^{-6}) for each 100 000 elements. Out of curiosity I tried the Quick Sort algorithm with 1 000 000 elements, and it took just 89 ms, compared to the test with 500 000 elements which resulted in 44 ms (Windows). By doubling the number of elements, the speed doubled as well. That performance is quite impressive.

8 Conclusion

This works task was to show if there is a suited algorithm which can fulfil all necessary task regardless the amount of data which needs to be dealt with. Although there are quite capable and immensely powerful algorithms, they are also limited to certain conditions. The almost unlimited amount of different data varied sizes makes it a big challenge to find the fitting algorithm to manage the data in a proper way. Before choosing the right algorithm, the data needs to be analysed to find out what algorithm would be a fitting one.

The research part helped me to understand the basics of big data, databases, and the sorting algorithms. Going through the algorithms gives a good understanding of the basics and the functionality of the algorithms. The big O notation shows how the algorithms are working with different data loads. That all comes together in the practical part to verify respectively falsify the previously gained knowledge.

In the practical part the outcome was like expected. Whereas in the beginning of the rising data collections, in form of arrays of numbers, all applied algorithms responded the same, it started changing when the number of elements was rising. The idea is to sort first the given data which makes it easier and faster to analyse or search for the desired information. That is in especially important if there is an excessively big data. The desired result is to find exactly what the user is searching for in a decent time. That can be done with a deep learning algorithm which learns the users search habits, with increased information collected which makes the results more accurate.

I am looking exited into the future, what it will bring and change. The steps were quite constant in the last fifty years in the digital area, there is the time and technology to make now a big one.

Sources

Benjamin Anderson, IBM, 12 June 2022. SQL vs. NoSQL Databases: What's the Difference? URL: https://www.ibm.com/cloud/blog/sql-vs-nosql. Accessed 26.08.2022.

Bi-In Sider.com, 14 January 2019. Search Engine NoSQL Database. URL: https://bi-in-sider.com/posts/search-engine-nosql-database/. Accessed 20.08.2022.

Enterprise Big Data Framework, 7 May 2019. What is Big Data Architecture? URL: https://www.bigdataframework.org/big-data-architecture/. Accessed 08.09.2022.

Fernando Iafrate, 2013. From Big Data to Smart Data. Volume 1.

Freecodecamp, 25 January 2020. Bubble Sort Explained. URL: https://www.freeco-decamp.org/news/bubble-sort/. Accessed 06.08.2022.

GeeksforGeeks, 11th May 2022. Insertion Sort. URL: https://www.geeksforgeeks.org/insertion-sort/. Accessed 01.06.2022.

ICO., Information Commissioner's Office, 2022. Principle (e): Storage limitation. URL: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/principles/storage-limitation. Accessed 07.11.2022.

Jonathan Ellis, DataStax, 24 February 2020. The Future of Databases. URL: https://www.datastax.com/blog/future-databases. Accessed 30.08.2022.

Joseph Pyram, Feb 11, 2020. Quick Sort Explained in Under 5 minutes. URL: https://betterprogramming.pub/quicksort-explained-in-5-minutes-d32cf430a592. Accessed 09.08.2022.

Khan Academy, 2022. Overview of merge sort. URL: https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/overview-of-merge-sort. Accessed 13.08.2022.

MongoDB, 2022. What is Big Data Architecture? URL: https://www.mongodb.com/big-data-explained/architecture. Accessed 08.09.2022.

Paul Taylor, 17 June 2020. The challenges of big data. URL: https://www.bcs.org/articles-opinion-and-research/the-challenges-of-big-data/. Accessed 12.11.2022.

Rod Stephens, 2013. Essential Algorithms, A Practical Approach to Computer Algorithms.

Simpl;learn, 19 October 2021. Counting Sort Algorithm: Overview, Time Complexity, Implementation in C & More. URL: https://www.simplilearn.com/tutorials/data-structure-tutorial/counting-sort-algorithm. Accessed 16.08.2022.

Thommaso Teofili, June 2019. Deep Learning for Search.

TutorialsPoint, 2022. Data Structure and Algorithms Selection Sort. URL: https://www.tutorialspoint.com/data_structures_algorithms/selection_sort_algorithm.htm. Accessed 06.08.2022.

Appendices

Appendix 1. The core part of the Sort algorithms

For further interest, the complete code is available on the following link:

https://github.com/Bulent2019/Sorting-Algorithms

Bubble Sort

```
for (let i = 0; i < backArray.length - 1; i++) {
        let swapCheck = false;
        for (let j = 0; j < backArray.length - i - 1; j++) {
            let a = backArray[j];
            let b = backArray[j + 1];
            if (a > b) {
                let temp = backArray[j];
                backArray[j] = backArray[j + 1];
                backArray[j + 1] = temp;
                swapCheck = true;
            }
        }
        if (!swapCheck) {
            break;
        }
}
```

Quick Sort

```
function swap (arr, left, right){
        var temp = arr[left];
        arr[left] = arr[right];
        arr[right] = temp;
    }

    function partition (arr, left, right) {
        let pivot   = arr[Math.floor((right + left) / 2)], //middle element
            i       = left, //left pointer
            j       = right; //right pointer
        while (i <= j) {
            while (arr[i] < pivot) {
                i++;
            }
            while (arr[j] > pivot) {
                j--;
            }
            if (i <= j) {
                swap(arr, i, j); //swap two elements
                i++;
                j--;
            }
        }
return i;
}



function quickSor t(arr, left, right) {
```

```
        let index;
        if (arr.length > 1) {
            index = partition(arr, left, right); //index returned from partition
            if (left < index - 1) { //more elements on the left side of the pivot
                quicksort (arr, left, index - 1);
            }
            if (index < right) { //more elements on the right side of the pivot
                quicksort (arr, index, right);
            }
        }
return arr;
}
```

Merge Sort

```
 function mergeSort (arr) {
        if (arr.length <= 1) {
            return arr;
        }
        let midIdx = Math.floor(arr.length / 2);
        let left = arr.slice(0, midIdx);
        let right = arr.slice(midIdx);

        let leftSorted = mergeSort(left);
        let rightSorted = mergeSort(right);
return merge(leftSorted, rightSorted);
};

function merge (arr1, arr2) {
```

```
    let merged = [];

    while (arr1.length && arr2.length) {
      if (arr1[0] < arr2[0]) {
        merged.push(arr1.shift());
      } else {
        merged.push(arr2.shift());
      }
    }
return [...merged, ...arr1, ...arr2];
};
```