# INFORMATION SEARCH & ANALYSIS SKILLS (ISAS)

****

**BINARY SEARCH ALGORITHM**

Group 6:

**Kheyral Sutan Dumas**

**Charisma Bayu Majestyno**

**Naufal Fauzan Wildani**

Faculty:

**Mr. Tri Agus Riyadi, M.T**

Class:

**2CS1**

**CEP CCIT FACULTY OF ENGINEERING**

**UNIVERSITY OF INDONESIA**

**202****4**

# PREFACE

First of all, the author would like to say thanks to Allah SWT. Because of His grace and guidance, our paper is entitled Binary Search Algorithm. Thanks to Mr. Tri Agus Riyadi, M.T and other faculty who always help to finish this task. Also, thanks to other fellow students and fellow workers in CCIT FTUI who supported The Author.

The author wishes that this task can be useful for readers so that it can add knowledge and insight in the future. The author realized this paper was still imperfect in arrangement and content. Therefore, the author really expects all suggestions and criticisms from readers who are constructive for the perfection of this paper and for future tasks.

Finally, hopefully this paper can provide readers and the author with many benefits for both readers and authors.

Depok, 19 March 2024

The Author

# **TABLE OF CONTENT**

[**PREFACE** i](#_Toc161407572)

[**TABLE OF CONTENT** ii](#_Toc161407573)

[**TABLE OF FIGURES** iii](#_Toc161407574)

[**CHAPTER I**](#_Toc161407575) [**INTRODUCTION** 1](#_Toc161407576)

[I.1 Background 1](#_Toc161407577)

[I.2 Writing Objective 1](#_Toc161407578)

[I.3 Problem Domain 1](#_Toc161407579)

[I.4 Writing Methodology 2](#_Toc161407580)

[1.5 Writing Framework 2](#_Toc161407581)

[**CHAPTER II**](#_Toc161407582) [**BASIC THEORY** 3](#_Toc161407583)

[II.1 Algorithm 3](#_Toc161407584)

[II.2 Data Structure 3](#_Toc161407585)

[II.3 Arrays 4](#_Toc161407586)

[II.4 Big O Notation 5](#_Toc161407587)

[**CHAPTER III**](#_Toc161407588) [**PROBLEM ANALYSIS** 7](#_Toc161407589)

[III.1 Binary Search Algorithm 7](#_Toc161407590)

[III.2 How Binary Search Algorithm Works 7](#_Toc161407591)

[III.2.1 Binary Search Analogy 8](#_Toc161407592)

[III.2.2 Binary Search Implementation 9](#_Toc161407593)

[III.3 Pros & Cons 10](#_Toc161407594)

[**CHAPTER IV**](#_Toc161407595) [**CONCLUSION** 12](#_Toc161407596)

[IV.1 Conclusion 12](#_Toc161407597)

[IV.2 Suggestion 12](#_Toc161407598)

[**BLIBIOGRAPHY** 13](#_Toc161407599)

# TABLE OF FIGURES

[**Figure 2. 1 Data Structures** 4](#_Toc161407395)

[**Figure 2. 2 Arrays Indexes** 4](#_Toc161407396)

[**Figure 2. 3 Big O Charts** 5](#_Toc161407397)

[**Figure 3. 1 Big O Charts**…………………………………………………………………………………..7](#_Toc161407400)

[**Figure 3. 2 Dictionary Books** 8](#_Toc161407401)

[**Figure 3. 3 Implementation 1** 9](#_Toc161407402)

[**Figure 3. 4 Implementation 2** 9](#_Toc161407403)

[**Figure 3. 5 Implementation 3** 9](#_Toc161407404)

[**Figure 3. 6 Implementation 4** 10](#_Toc161407405)

[**Figure 3. 7 Result Found** 10](#_Toc161407406)

[**Figure 3. 8 All Picture Above Belong to Programiz** 10](#_Toc161407407)

# CHAPTER I

# INTRODUCTION

## I.1 Background

In-depth discussion of the function and importance of search algorithms in computer science will be provided in this paperwork. In today's digital age, where vast amounts of data are generated every second, the role of search algorithms in computer science cannot be overstated. These algorithms form the backbone of search engines, databases, and various information retrieval systems, enabling users to efficiently locate relevant information from massive datasets.

One of the fundamental search algorithms widely used in computer science is the binary search algorithm. This paperwork will provide an in-depth explanation of Binary search, which is well-known for its effectiveness and is essential for locating data in sorted datasets. As the author investigate these methods, it becomes clear why effective search methods are essential, especially when working with large datasets. Readers will understand the fundamentals of optimizing search algorithms for improved computational efficiency and performance through this discussion.

## I.2 Writing Objective

The purposes of writing this ISAS are:

1. What is Algorithm & Binary Search
2. How Binary Search works
3. Binary search Pro & Contra

## I.3 Problem Domain

The problems of writing this ISAS are:

1. What is Algorithm & Binary Search
2. How Binary Search works
3. Binary search Pro & Contra

## I.4 Writing Methodology

This study uses one of the methodologies named qualitative methodology. Observation and description are the most important in this type of methodology. The method works using internet research, disclosure analysis, etc. It provides an explanation using analysis that tends to focus more on the theoretical basic.

## 1.5 Writing Framework

This study focuses on the main subject with a systematics writing as below:

**Chapter I Introduction**

In this chapter, this article discusses about the background of the problem, the writing objective, the problem domain, what writing methodology it uses, and the writing framework.

**Chapter II Basic Theory**

This chapter contains basic knowledge about Algorithm, Data structure, Arrays, and Big O Notation including their definition.

**Chapter III Problem Analysis**

This chapter mainly focuses on identifying Binary Search features and Binary Search’s advantages as well as its disadvantages.

**Chapter IV Conclusion and Suggestions**

The conclusion and suggestions stated in this chapter. The conclusion mostly summarizes the discussion of research that has been done. This chapter also contains suggestions that reflected the presentation of the research finding, the discussion, and the conclusion.

# CHAPTER II

# BASIC THEORY

## II.1 Algorithm

**Algorithm** is a set of steps for accomplishing a task or solving a problem. Typically, algorithms are executed by computers, but it also rely on algorithms in our daily lives. In fact, each time a particular step-by-step process is followed, such as making coffee in the morning or tying shoelaces, an algorithm is being followed.

In the context of computer science, an algorithm is a mathematical process for solving a problem using a infinite number of steps. Algorithms are a key component of any computer program and are the driving force behind various systems and applications, such as navigation systems, search engines, and music streaming services.

More advanced algorithms can use conditionals to divert the code execution through various routes (referred to as automated decision-making) and deduce valid inferences (referred to as automated reasoning), achieving automation eventually. Using human characteristics as descriptors of machines in metaphorical ways was already practiced by Alan Turing with terms such as "memory", "search" and "stimulus".[1]

## II.2 Data Structure

**Data Structure** is a way of organizing and storing data in a computer so that it can be accessed and used efficiently. It refers to the logical or mathematical representation of data, as well as the implementation in a computer program

There is two types of Data Structure:

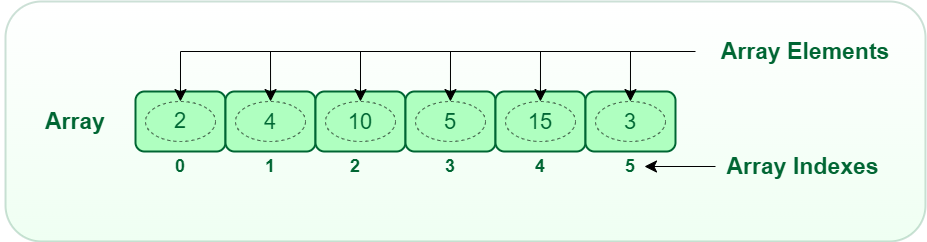


**Figure 2. 1 Data Structures**

***(REF:*** [***https://www.geeksforgeeks.org/data-structure-meaning/***](https://www.geeksforgeeks.org/data-structure-meaning/) ***)***

1. **Linear Data Structure:** A linear data structure is one in which the elements are ordered either sequentially or linearly, with each element being connected to its preceding and subsequent neighboring elements. Arrays, stacks, queues, etc. are some examples.
2. **Non-linear Data Structure:** Non-linear data structures are those in which the arrangement of the data pieces is not linear or sequential. Trees and graphs are two examples.[2]

## II.3 Arrays



***Figure 2. 2 Arrays Indexes***

***(REF:*** [***https://www.geeksforgeeks.org/what-is-array/***](https://www.geeksforgeeks.org/what-is-array/)***)***

**Array** is a group of similar elements or data items of the same type collected at contiguous memory locations. In simple words, we can say that in computer programming, arrays are generally used to organize the same type of data.

To simplify matters, an array can be envisioned as a set of stairs, with a value (let's say, a friend) positioned on each step. Here, the whereabouts of any pals can be determined simply by knowing the number of steps they have taken.

As a result, it is now easier to determine each element's position by just adding an offset to a base value, or the memory location of the array's first element, which is typically indicated by the array name. Index 0 is the base value, while the offset is the difference between the two indexes.[3]

## II.4 Big O Notation

**Big O notation** is one of the most fundamental tools for computer scientists to analyze the cost of an algorithm. Algorithms can take on several forms depending on who developed them. However, the problem is **which algorithm is more effective and efficient?**



***Figure 2. 3 Big O Charts***

**(REF:** [**https://www.geeksforgeeks.org/analysis-algorithms-big-o-analysis/**](https://www.geeksforgeeks.org/analysis-algorithms-big-o-analysis/)**)**

The complexity of an algorithm is divided into two parts: time complexity and space complexity. The term **"time complexity"** refers to the amount of time needed to launch an algorithm. On the other hand, **"space complexity"** refers to the amount of large memory that used when launching an algorithm.

1. **Time Complexity**

Time complexity describes how the runtime of an algorithm grows as the size of the input increases. It measures the maximum amount of time an algorithm takes to execute relative to the size of its input. Time complexity is often expressed using Big O notation.

For example:

1. **O(1):** Constant time complexity. The algorithm's runtime remains the same regardless of the input size.
2. **O(log N):** Logarithmic time complexity. The runtime grows logarithmically as the input size increases.
3. **O(N):** Linear time complexity. The runtime grows linearly with the input size.
4. **O(N^2):** Quadratic time complexity. The runtime grows quadratically with the input size.
5. **Space Complexity**

Space complexity refers to how much memory an algorithm requires relative to the size of its input. It measures the maximum amount of memory used by an algorithm as a function of the input size. Like time complexity, space complexity is often expressed using Big O notation.

For example:

* 1. **O(1):** Constant space complexity. The amount of memory used remains the same regardless of the input size.
  2. **O(log N):** Logarithmic space complexity. The memory used grows logarithmically with the input size.
  3. **O(N):** Linear space complexity. The memory used grows linearly with the input size.
  4. **O(N^2):** Quadratic space complexity. The memory used grows quadratically with the input size.[4]

# CHAPTER III

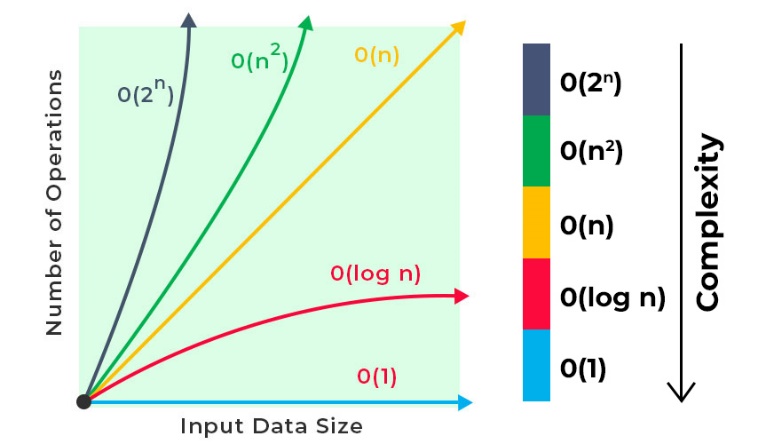
# PROBLEM ANALYSIS

## III.1 Binary Search Algorithm

Binary Search, also known as Half-Interval Search, Logarithmic Search, or Binary Chop, is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array. If they are not equal, the half in which the target is eliminated and the search continues on the remaining half, again taking the middle element to compare to the target value, and repeating this until the target value is found. If the search ends with the remaining half being empty, the target is not in the array.[5]

## III.2 How Binary Search Algorithm Works

Binary search runs in logarithmic time in good case, making **O(log n)** comparisons, where **n** is the number of elements in the array. Binary search is faster than linear search except for small arrays. However, the array must be sorted first to be able to apply binary search. There are specialized data structures designed for fast searching, such as hash tables, that can be searched more efficiently than binary search. However, binary search can be used to solve a wider range of problems, such as finding the next-smallest or next-largest element in the array relative to the target even if it is absent from the array.[6]



**Figure 3. 1 Big O Charts**

**(REF:** [**https://www.geeksforgeeks.org/what-is-logarithmic-time-complexity/**](https://www.geeksforgeeks.org/what-is-logarithmic-time-complexity/)**)**

### III.2.1 Binary Search Analogy



**Figure 3. 2 Dictionary Books**

***(REF:*** [***https://www.dataversity.net/what-is-a-data-dictionary/***](https://www.dataversity.net/what-is-a-data-dictionary/)***)***

Finding a term in a dictionary is similar to using binary search. Suppose the task of locating the word "elephant" in a large dictionary is assigned. Here's an analogy of how binary search functions:"

* 1. **Defined the array:** Assume a sorted array of words in a dictionary.
  2. **Choose the target:** The word "elephant" is the selected target
  3. **Start in the Middle:** The dictionary is opened roughly in the middle, not at the beginning or the end. Providing a word to compare with "elephant".
  4. **Comparison:** Supposed the word found is "lion". Now, a decision needs to be made regarding whether "elephant" would surpass or behind "lion" in the dictionary.
  5. **Halving the Search Space:** Given that dictionaries are sorted, it is understood that if "elephant" is behind "lion," there is no need to search the first half of the dictionary any longer. Half of the dictionary has been effectively eliminated.
  6. **Repeat the Process:** The process is repeated with the remaining half of the dictionary. It is again opened roughly in the middle, and the word found is compared with "elephant."
  7. **Continue Until Found**: Keep halving the search space and comparing until the word of "elephant" is found.

### III.2.2 Binary Search Implementation

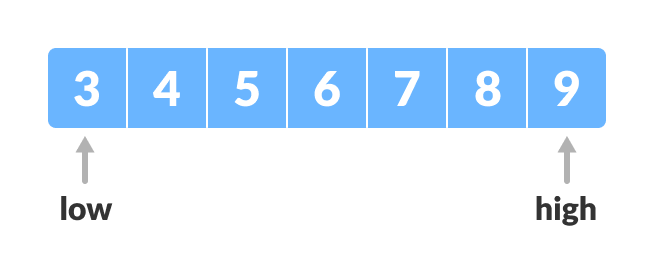
Binary Search Algorithm can be implemented which are discussed below.

1. The array in which is to be performed is:



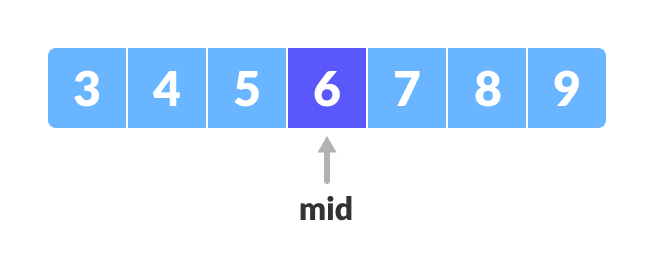
**Figure 3. 3 Implementation 1**

1. Let’s say the target is number 4, **x == 4**
2. Set two pointers low and high at the lowest and the highest positions respectively



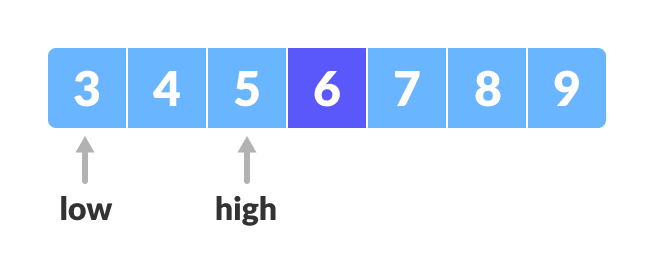
**Figure 3. 4 Implementation 2**

1. Find the middle element mid of the array, **arr[index(low + high)/2] = 6**



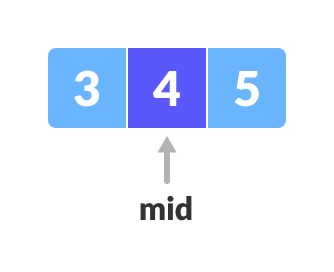
**Figure 3. 5 Implementation 3**

1. If the element at mid is equal to 4, its done.
2. If 4 is greater than mid, focus on the right half of the array. set **low = mid + 1**.
3. If 4 is less than mid, eliminate the right side from mid and then set **high = mid – 1**, now **5** is the new high point.

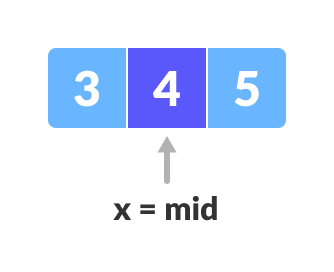


**Figure 3. 6 Implementation 4**

1. Repeat steps 2 until low meets high, eventually 4 will be found in the array.[7]



**Figure 3. 7 Result Found**

****

**Figure 3. 8 All Picture Above Belong to Programiz**

***(REF:*** [***https://www.programiz.com/dsa/binary-search#google\_vignette***](https://www.programiz.com/dsa/binary-search#google_vignette)***)***

## III.3 Pros & Cons

**Pros :**

**1. Faster than most of search methods**: Binary search's time complexity of O(logN) outperforms for larger datasets. It reduces search time logarithmically.

**2. Efficient search space reduction:** It halves the search space with each iteration, significantly speeding up the search process.

**3. Versatile:** Binary search can handle various data types and structures as long as the dataset is sorted. This makes it suitable for a wide range of applications.

**Cons :**

1. **Preprocessing overhead:** Binary search requires the dataset to be sorted beforehand, which adds additional time and memory overhead.
2. **Not suitable for dynamic data:** Binary search performs best on static or unchanging datasets. If the data frequently changes, maintaining the sorted order becomes a challenge.[8]

# CHAPTER IV

# CONCLUSION

## IV.1 Conclusion

This paperwork serves as a comprehensive exploration of search algorithms, focusing notably on the binary search algorithm. It elucidates fundamental concepts, methodologies, and practical implementations, aiming to enhance understanding and application in computational contexts. Additionally, it provides insights into the advantages and disadvantages of binary search, aiding in informed decision-making regarding its utilization in various scenarios.

## IV.2 Suggestion

As the learning journey continues, exploring more about different types of search algorithms such as hash table (which is considered the fastest search algorithm) or linear search, and how they're used in real-life situations, is encouraged. Practicing by writing code and working on projects to observe how these algorithms operate in action is also recommended. By staying curious and practicing regularly, confidence in the ability to understand and apply these important concepts in computer science will be gained.

# BLIBIOGRAPHY

1. Gillis, A. S. (2023, July 31). *What is an algorithm?: TechTarget*. WhatIs. Available at: https://www.techtarget.com/whatis/definition/algorithm (Accessed: 19 March 2024)
2. GfG (2023) *What is data structure?*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/data-structure-meaning/ (Accessed: 13 March 2024)
3. GfG (2023a) *What is array?*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/what-is-array/ (Accessed: 15 March 2024)
4. Ismail, A.M. (2018) *Penjelasan Sederhana Tentang time complexity Dan Big-O notation*, *Medium*. Available at: https://medium.com/bee-solution-partners/penjelasan-sederhana-tentang-time-complexity-dan-big-o-notation-4337ba275cfe (Accessed: 15 March 2024)
5. *Binary search algorithm* (2024) *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Binary\_search\_algorithm (Accessed: 12 March 2024)
6. Adimike, C.S. (2022) *Binary search and logarithmic search*, *Medium*. Available at: https://medium.com/@nicsylvia15f/binary-search-and-logarithmic-search-e0429c70d683 (Accessed: 15 March 2024)
7. *Binary search* (no date) *Programiz*. Available at: https://www.programiz.com/dsa/binary-search (Accessed: 12 March 2024)
8. Shah, D. (2023) *Binary search: Finding the perfect match in a Sea of Numbers*, *LinkedIn*. Available at: https://www.linkedin.com/pulse/binary-search-finding-perfect-match-sea-numbers-devansh-shah (Accessed: 15 March 2024)