DATA STRUCTURES AND ALGORITHM

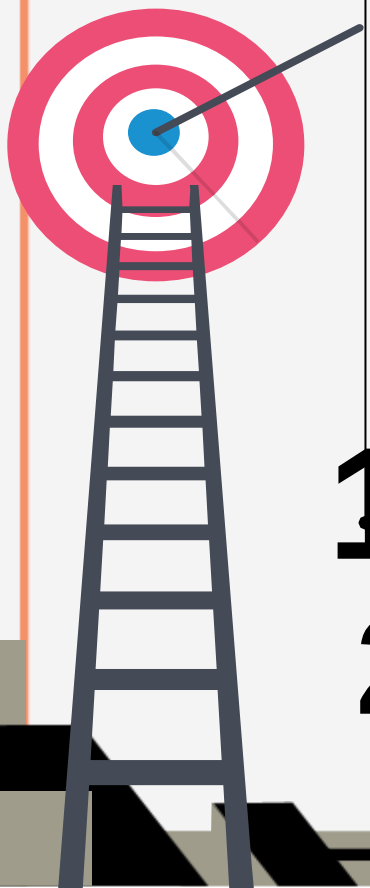
LAB EXERCISE 5

SEARCH ALGORITHM

Submitted to: Prof. Godofredo T. Avena

Submitted by: Castilllo Gregorio, Concepcion Iggo, Cruz Malto, David Nudalo

TABLE OF CONTENT



Optimizing Search Algorithm Performance: A Comprehensive Analysis on Varied Dataset Sizes

ABSTRACT

This project aims to implement and compare various search algorithms in Python, including linear search, binary search, jump search, exponential search, interpolation search, and ternary search. Three different datasets of varying sizes, each containing sorted integers, are generated for testing purposes. The execution times of each search algorithm on these datasets are recorded and analyzed to draw conclusions about their performance.

INTRODUCTION

In this study, we explore the efficiency of different search algorithms when applied to datasets of varying sizes. The algorithms under consideration include linear search, binary search, jump search, exponential search, interpolation search, and ternary search. The motivation is to understand how these algorithms perform in real-world scenarios and how their efficiency is affected by the size of the dataset.

RESEARCH QUESTIONS:

How do different search algorithms perform when applied to datasets of varying sizes, and what factors contribute to their efficiency in locating target elements?

a. Which search algorithm performed the best overall?
b. Did any search algorithms perform better on specific data sets?
c. How did the size of the data set affect the performance of the search algorithms?
d. Write a brief conclusion summarizing your findings

CONCEPTUAL FRAMEWORK

The conceptual framework for this research is built upon the understanding of search algorithms and their fundamental principles. It considers the following key components:

1. Search Algorithms:
- Linear search
- Binary search
- Jump search
- Exponential search
- Interpolation search
- Ternary search

2. Dataset Sizes:
- Small dataset (100 integers)
- Medium dataset (1,000 integers)
- Large dataset (10,000 integers)

3. Performance Metrics:
- Time taken by each search algorithm to find the target element in each dataset.

Variables:
- Independent Variable: Search algorithm type
- **Dependent Variable:** Execution time.

HYPOTHESIS

Based on the research questions, the following null and alternative hypotheses are proposed:

1. Null Hypothesis (H0):
- There is no significant difference in the performance of various search algorithms across different dataset sizes.

2. Alternative Hypothesis (H1):
- The efficiency of search algorithms varies significantly based on dataset size, and certain algorithms may outperform others in specific scenarios.

METHODOLOGY

This detailed methodology ensures a systematic approach to implementing, testing, and analyzing the search algorithms on datasets of varying sizes, leading to robust conclusions about their performance characteristics.

1. Search Algorithm Implementation:
   - Implement each search algorithm in Python: linear search, binary search, jump search, exponential search, interpolation search, and ternary search. Utilize the provided code snippets and explanations for accurate implementation.

2. Generating Test Data:
   - Small dataset: Generate a sorted list of 100 integers. Choose a random target element within this dataset.
   - Medium dataset: Generate a sorted list of 1,000 integers. Randomly select a target element.
   - Large dataset: Generate a sorted list of 10,000 integers. Randomly choose a target element.

3. Testing and Recording:
   - For each search algorithm, execute the algorithm on each dataset.
   - Record the time taken by each algorithm to find the target element in the datasets. Utilize Python's `time` module for accurate time measurements.

RESULTS & DISCUSSION

RESULT

		Linear	Binary	Ternary	Exponential	Interpolation	Jump
Target Set	Search Data	Time in Milliseconds					
100	23	0.006	0.0085	0.009599999	0.020500001	0.0057	0.0079
	34	0.0064	0.005800001	0.002899999	0.0114	0.0049	0.0085
	68	0.014499999	0.004799999	0.0132	0.0123	0.0042	0.0127
	80	0.0117	0.0075	0.048899999	0.0113	0.0044	0.0174
	99	0.0071	0.0043	0.0104	0.008	0.0052	0.0093
1000	12	0.0048	0.0122	0.0064	0.0074	0.0043	0.0143
	34	0.0091	0.010700001	0.0086	0.016	0.0067	0.0157
	566	0.012500001	0.0094	0.0097	0.5428	0.0074	0.0139
	899	0.0514	0.0132	0.0119	0.010700001	0.0073	0.0261
	987	0.4762	0.7469	0.0105	0.014300001	0.006400001	0.0243
10000	100	0.0083	0.00780000027589267	0.012	0.00910000017029233	0.004600000465870835	0.0161
	3000	0.2304	0.0068	0.0089	0.0111	0.0049	0.0187
	6000	0.3661	0.0135	0.0111	0.0118	0.0063	0.0206
	7666	0.3973	0.0067	0.0092	0.0153	0.004499999704421498	0.0357
	9877	0.503399999615794	0.006	0.0126	0.0114	0.0044	0.1082

Figure 1 | Table of Search Data and Time in Milliseconds

AVERAGE						
	Linear	Binary	Ternary	Exponential	Interpolation	Jump
Target Set	Time in Milliseconds					
100	0.01139	0.00417	0.03128	0.00517	0.00443	0.01467
	Linear	Binary	Ternary	Exponential	Interpolation	Jump
Target Set	Time in Milliseconds					
1000	0.1108	0.15848	0.00942	0.11824	0.00642	0.01886
	Linear	Binary	Ternary	Exponential	Interpolation	Jump
Target Set	Time in Milliseconds					
10000	0.250525	0.00825	0.01076	0.0124	0.0052	0.03986

Figure 2 | Table of Average Time in Milliseconds per Data Set

RESULT

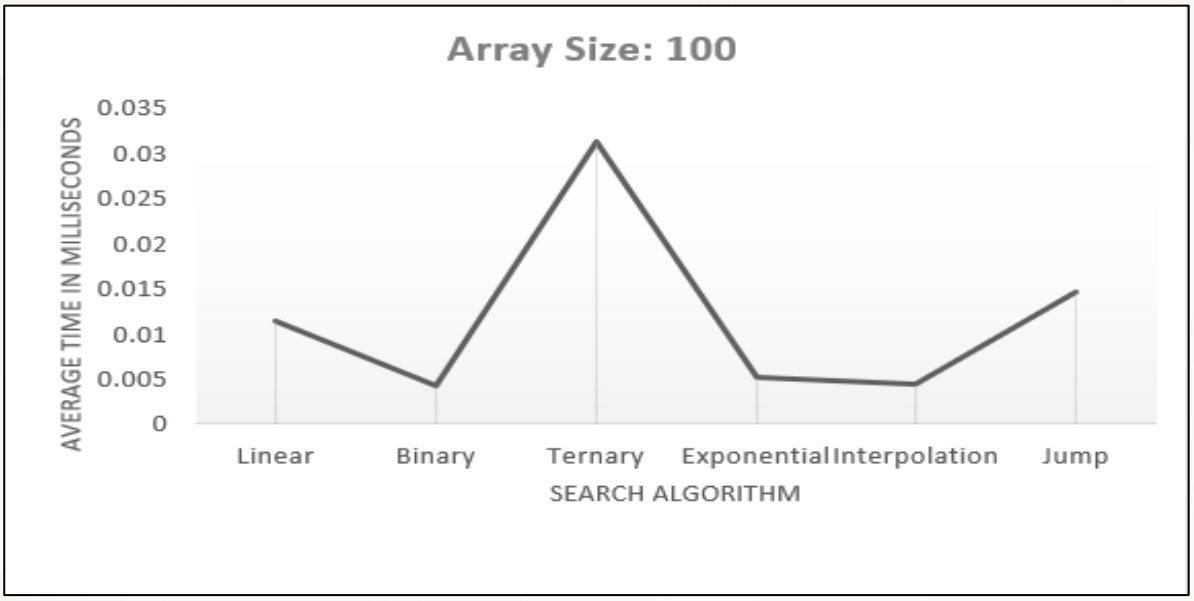


Figure 3 | Graph of Array Size 100 in Ave. Time in Milliseconds

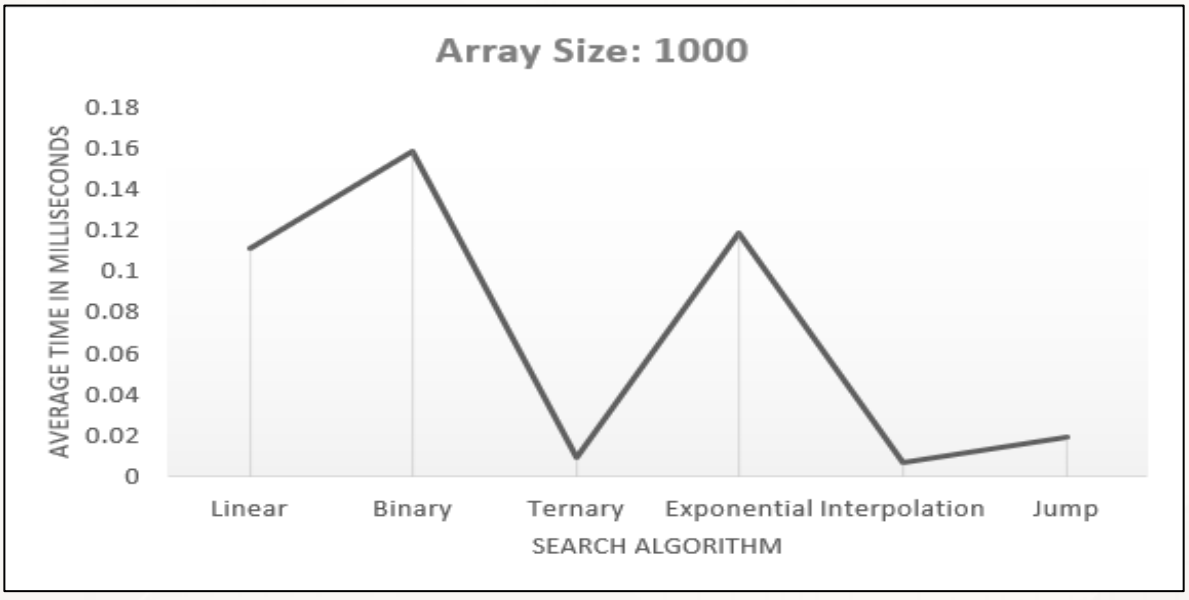


Figure 4 | Graph of Array Size 1000 in Ave. Time in Milliseconds

RESULT

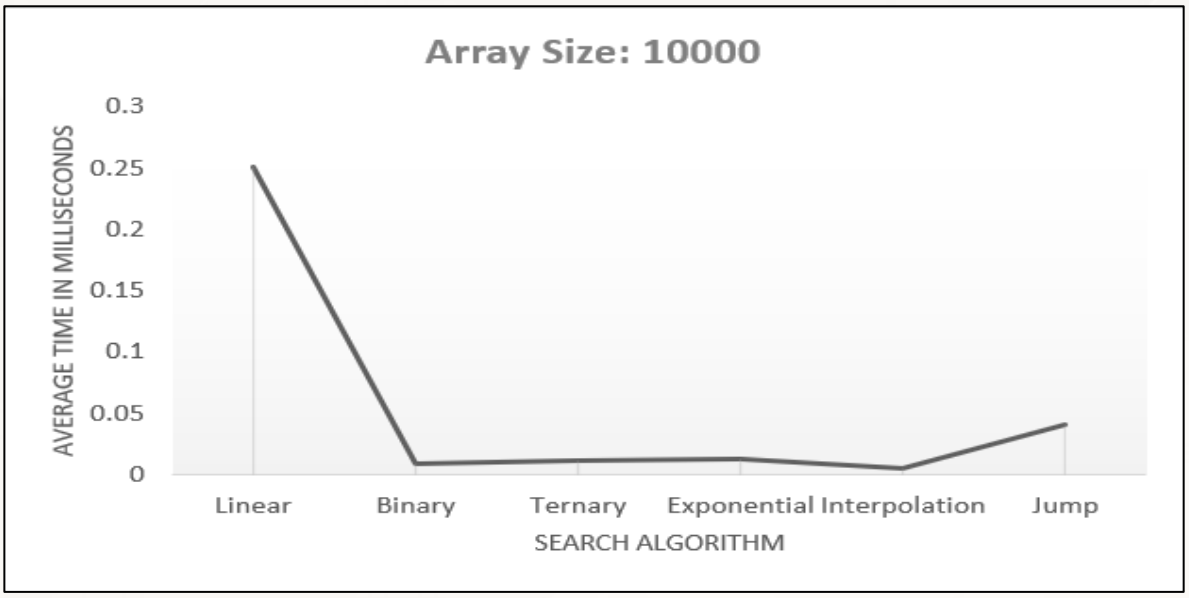


Figure 5 | Graph of Array Size 10000 in Ave. Time in Milliseconds

OVERALL AVERAGE						
	Linear	Binary	Ternary	Exponential	Interpolation	Jump
Time in Milliseconds	0.12423833	0.056966667	0.017153333	0.04527	0.00535	0.024463333

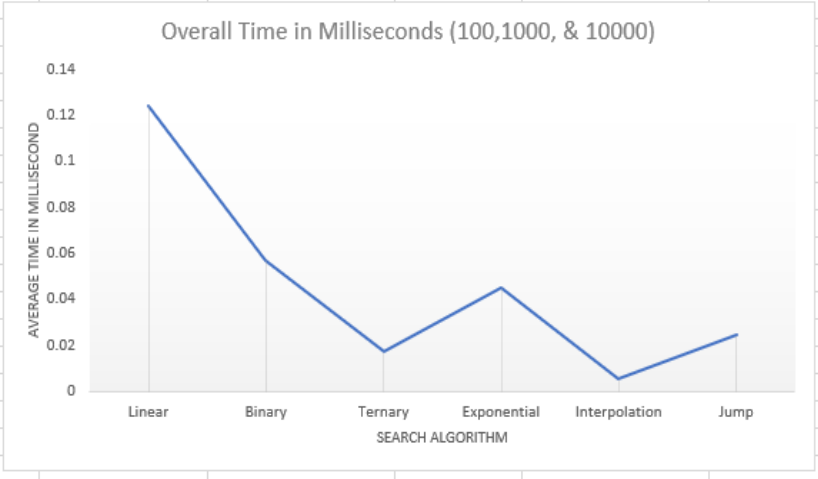


Figure 6 | Table and Graph of Overall Average Time in Milliseconds

DISCUSSION

a. Which search algorithm performed the best overall?

According to Figure 6, which shows the overall results for each algorithm, **Interpolation Search** showed the shortest time in milliseconds (0.00535ms). Through our search algorithm testing site, it consistently demonstrated the most efficient outcomes across the six sizes of datasets we have used. Because of this search algorithm being an improved version of binary search using ordered key values, it firmly established interpolation search as the best performing one.

a. Did any search algorithms perform better on specific data sets?

According to Figure 3, which shows the average time for the small dataset (100), **Binary Search** proved to be the most efficient Python search algorithm with only 0.00417ms. Because binary search algorithm works by dividing a sorted array by half repeatedly, having a smaller data set makes it easier and more efficient to search a specific index. According to Figure 4 and 5, which shows the average time for the medium and big datasets respectively, **Interpolation Search** still proved to be the most efficient Python search algorithm to be used. It took an average of 0.00642ms to look for a number in the dataset of 1000, while in the dataset of 10000, it took for it to search the dataset in only an average of 0.0052ms. Since interpolation uses the interpolation formula, which searches for the estimated location and separated the list. This proves to be the most efficient search algorithm for larger datasets because it will only focus on the numbers surrounding the estimated location from the formula, reducing the time it takes to search for a number.

a. How did the size of the data set affect the performance of the search algorithms?

An important portion of this project focused on understanding the influence of the size of the dataset on the time it will take to search these datasets. The results have clearly shown that there will be a variation of results depending on not only the search algorithm used, but as well as the dataset size. This shows that it is important that programmers consider the size of dataset when choosing which search algorithm to use in their works.

CONCLUSION

After testing it with datasets of varying sizes. After this project, our goal was to know which is the overall best-performing search algorithm, how search algorithms perform under varying dataset sizes, and how the size of datasets affects the performance of each search algorithm. Through extensive testing and analysis of the results, we can conclude the following: ***Interpolation Search*** consistently demonstrated the most efficient outcomes across the three sizes of datasets we used, as indicated by Figure 6 showing the overall results for each algorithm. It exhibited the briefest average time duration in milliseconds of only 0.00535ms. Due to its refined approach, derived from an improved version of binary search using ordered key values, Interpolation Search firmly established itself as the best-performing algorithm in our study.

In Figure 3***, Binary Search*** stands out as the most efficient Python search algorithm for the small dataset (100), with a rapid 0.00417ms search time. This efficiency is attributed to its strategy of dividing a sorted array by half, making it particularly effective for smaller datasets, while Ternary Search took the longest average search time (0.03128ms) because of its 4 comparisons, resulting in a longer time complexity. Figures 4 and 5 highlight Interpolation Search as the optimal algorithm for medium and large datasets. It demonstrated an average search time of 0.00642ms for a dataset of 1000 and an even more efficient 0.0052ms for a dataset of 10000.

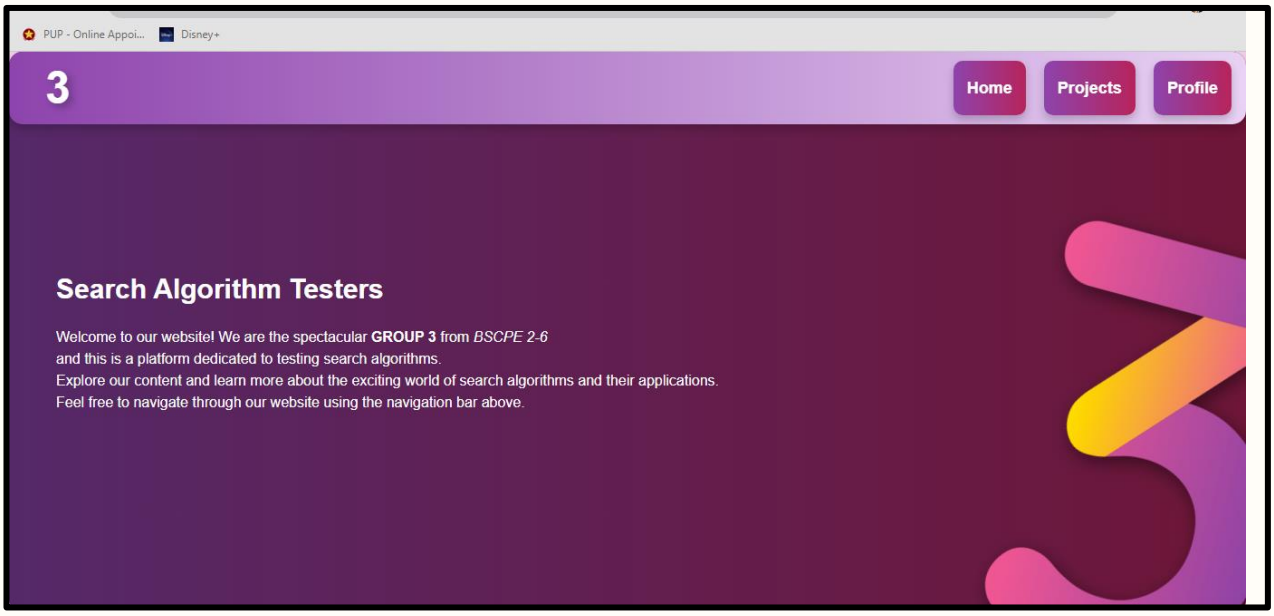
The effectiveness of Interpolation Search for larger datasets lies in its use of the interpolation formula, which estimates the location and strategically divides the list, significantly reducing search time. While binary search took the longest time for the 1000 dataset size (0.015848ms). Because of the much larger dataset size, it will have a harder time comparing halves of the array, while linear search took the longest time for the 10000 dataset size (0.250525ms) because as the size of the dataset increases, the time taken to search for a number also linearly increases.

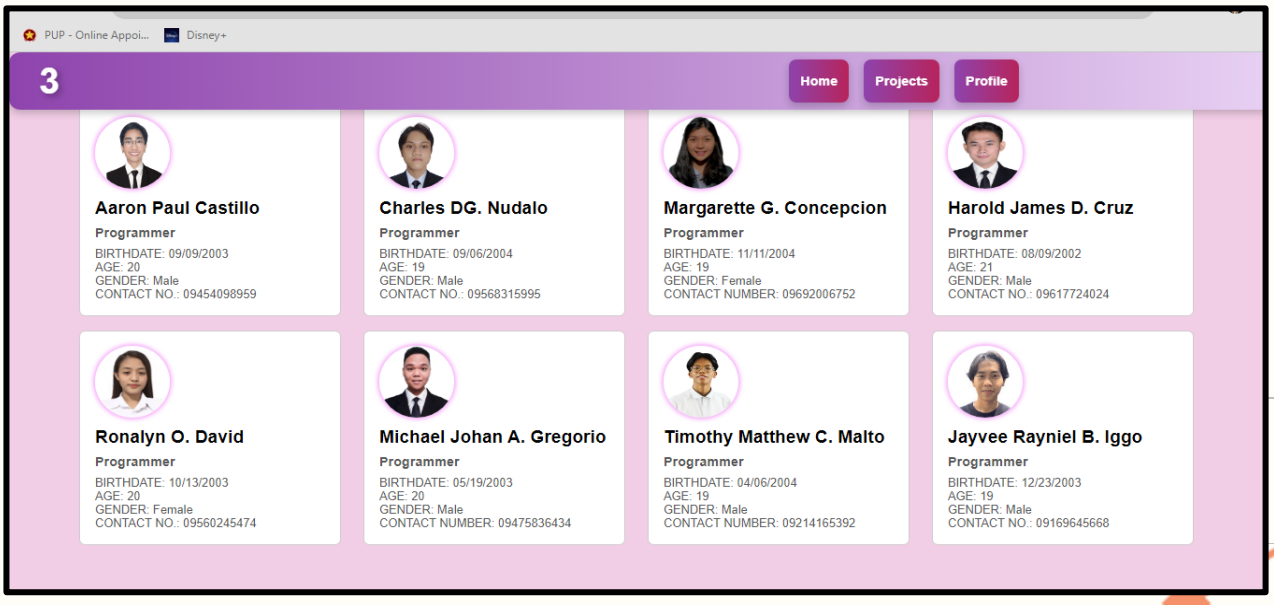
A key aspect of this project was examining how dataset size impacts search time. The results unequivocally highlight variations not only based on the chosen search algorithm but also on dataset size. This underscores the significance of programmers taking dataset size into account when selecting an appropriate search algorithm for their projects.

With this project, we have gained a deeper understanding into the intricate factors affecting the performance of Python search algorithms. The insights derived from our research project hold significant value for both academic and real-life applications. It is vital to recognize that as the field of search algorithms evolves, further investigation and refinement of these findings will surely play a crucial role in the development of these algorithms across a wide range of applications and scenarios.

APPENDICES

SEARCH ALGO


PUP - Online Appoi...
Disney+
3
Home
Projects
Profile
Search Algorithm Testers
Welcome to our website! We are the spectacular GROUP 3 from BSCPE 2-6
and this is a platform dedicated to testing search algorithms.
Explore our content and learn more about the exciting world of search algorithms and their applications.
Feel free to navigate through our website using the navigation bar above.


PUP - Online Appoi...
Disney+
3
Home
Projects
Profile
Aaron Paul Castillo
Programmer
BIRTHDATE: 09/09/2003
AGE: 20
GENDER: Male
CONTACT NO.: 09454098959
Charles DG. Nudalo
Programmer
BIRTHDATE: 09/06/2004
AGE: 19
GENDER: Male
CONTACT NO.: 09568315995
Margarette G. Concepcion
Programmer
BIRTHDATE: 11/11/2004
AGE: 19
GENDER: Female
CONTACT NUMBER: 09692006752
Harold James D. Cruz
Programmer
BIRTHDATE: 08/09/2002
AGE: 21
GENDER: Male
CONTACT NO.: 09617724024
Ronalyn O. David
Programmer
BIRTHDATE: 10/13/2003
AGE: 20
GENDER: Female
CONTACT NO.: 09560245474
Michael Johan A. Gregorio
Programmer
BIRTHDATE: 05/19/2003
AGE: 20
GENDER: Male
CONTACT NUMBER: 09475836434
Timothy Matthew C. Malto
Programmer
BIRTHDATE: 04/06/2004
AGE: 19
GENDER: Male
CONTACT NUMBER: 09214165392
Jayvee Rayniel B. Iggo
Programmer
BIRTHDATE: 12/23/2003
AGE: 19
GENDER: Male
CONTACT NO.: 09169645668

SEARCH ALGO

Search Algorithms Comparison

Array

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100

Target

Enter the target integer

Search Algorithm

--Select--

Data Size

--Select--

Search

Target found at index 22 using binary method in 0.008500000149069 82 milliseconds.

Clear