Journal Basic Science and Technology 11 (3) (2022) pp. 88-92

Published by: IOCSCIENCE

Journal Basic Science and Technology

Journal homepage: www.iocscience.org/ejournal/index.php/JBST

Binary search algorithm for natural number targets

Senad Orhani
Department of Education, University of Prishtina "Hasan Prishtina", Kosovo

ARTICLE INFO

Article history:

Received Sep 30, 2022
Revised Okt 17, 2022
Accepted Okt 30, 2022

Keywords:

Binary Search Algorithm
Natural Number
Targets
Python
Pseudocode

ABSTRACT

Targeting the number thought in a list is a common problem in computer science and is commonly implemented with the search algorithm. Numerical math games are a big thing for all of us. One of the really challenging games is targeting the number thought by the opponent. Therefore, the purpose of our study is to find the desired number through the binary search algorithm. For this study, we developed a pseudocode in the Python programming language, which gives the desired result followed after several successful steps. Our study results show that the algorithm was very efficient in solving the problem of finding the intended natural.

Corresponding Author:

Senad Orhani,
Department of Education,
University University of Prishtina "Hasan Prishtina",
George Bush, Nr.31, 10000 Prishtina, Kosovo
Email: senad.orhani@uni-pr.edu

1. INTRODUCTION

Intelligence is one of the main characteristics that distinguish a human being from other living creatures on earth. Basic intelligence covers day-to-day problem-solving and creating strategies to handle various situations that keep arising in everyday life. During the process of solving any problem, one tries to find the necessary steps to be taken in a sequence. In this unit, you will develop your understanding of problem-solving and approaches. In our life, we are forced to solve problems. It can be said that any activity that a human being or machine does to achieve a certain objective is involved in problem-solving (Mahdi, 2013). Therefore, the purpose of our study is to present an algorithm for solving the natural number target problem.

Mathematics is a way of thinking. Does it offer us strategies for organizing, analyzing, and synthesizing data, mainly but not exclusively numerical. That is why mathematics could not be seen as a concrete object to make it real when people talk about it. Making number one thing that people always talk about until now is the object of mathematics. A number in mathematics is an object used to count and measure. In early toddlerhood, the phenomenon of number retention reflects the way children think. Moreover, we will realize the fact of finding the numbers when solving the problem (Reys, Suydam, & Lindquist, 1984).

On the other hand, an algorithm is a sequence of fuzzy instructions for solving a problem, that is, for obtaining a required result for each legitimate input in a limited time (Meghanathan, 2015). To be executed by computers, algorithms must be in the form of a "program". A program is written in a programming language and the activity of expressing an algorithm as a program is called

programming. In algorithms, the steps are expressed in the form of an instruction or statement. Consequently, a computer program contains a series of statements that tell the computer which operation to perform. The programming language used will dictate the nature of the statements in a program (Christodoulou, Szczygiel, & Klapa, 2018).

2. RESEARCH METHOD

Search algorithms can be classified based on their search mechanism into three types of algorithms: linear, binary, and hashing. Linear search algorithms check each record for one that matches a target key in a linear process. Binary, or half-interval, searches consistently target the middle of the search structure and divide the search space in half. Comparison search algorithms improve linear search by sequentially eliminating records based on key comparisons until the target record is found and can be applied to data structures with a defined order. Digital search algorithms work based on the properties of digits in data structures using numeric keys. Finally, hashing directly maps keys to data based on a hash function (Knuth, 1998).

This study to find the imaginary natural number uses the binary search algorithm. This type of searching algorithm is used to find the position of a specific value contained in a sorted array. The binary search algorithm works on the principle of divide and conquer and it is considered the best searching algorithm because it's faster to run (Krishna, 2022). The Binary Search approach compares the target element to the middle element of the array. If the target element is larger than the middle element, then the search continues in the right half. Otherwise, if the target element is less than the middle value, the search continues in the left half. This process is repeated until the middle element is equal to the target element, or the target element is not in the array. If the target element is found, its index is returned, otherwise, -1 is returned (Krishna, 2022).

2.1. Procedures

Let the array be given A by $n$elements with the values $A_0, A_1, A_2, A_3, \dots, A_{n-1}$ ordered in such a way that $A_0 \leq A_1 \leq A_2 \leq A_3 \leq \cdots \leq A_{n-1}$, and with the target value T. Below we present the algorithm (Knuth, 1998):

a. Set it L for $0$and $R$for$n-1$
b. If $L > R$, the search ends as unsuccessful
c. Set m as the position of the middle element in $\frac{L+R}{2}$ which the largest integer is less than or equal to $\frac{L+R}{2}$
d. If $A_m < T$, set L to $m+1$ and go to step 2
e. If $A_m > T$, set R to $m-1$ and go to step 2
f. Now $A_m = T$, the search completes and returns$m$

2.2. Iterative Process

Iteration is the repetition of a process in a computer program. Iterations of functions are common in computer programming since they allow multiple blocks of data to be processed in sequence. This is typically done using a "while loop" or "for loop". These loops will repeat a process until a certain number or case is reached. Simply put, iteration means repeating the same step several times. Iterative algorithms should obey three important principles (Christodoulou, Szczygiel, & Klapa, 2018):

a. An iterative process repeats (iterates) a certain sub-process.
b. Each iteration should change at least one value.
c. There should be some condition under which the iteration

In the iterative approach, the algorithm works in this way (Sugandhi, 2022):

At each step, consider the set between the low and high indices.
Calculate the middle index.
If the element at the middle index is equal to the target value, we return it to the middle.
If the middle element is larger than the target:
Change the high index to middle - 1.
The low value remains the same.

If the middle element is smaller than the target:
Low mid + 1 change.
The high value remains the same.
When the low value is greater than the high value, this indicates that the target does not exist.
Therefore, -1 is returned.

3. RESULTS AND DISCUSSIONS

To see how our result works, we analyze it on the simplest possible basis, which is described for binary search algorithms. To this end, it is important to understand that algorithms are independent of the programming language used, and each algorithm can be expressed in different programming languages and run on different computers. This is why the design of algorithms is a fundamental aspect of computer science. Since there is no general rule, which means that there is no algorithm for designing algorithms. Therefore, the following pseudocode written in the Python programming language represents the approach proposed for this paper:

```
def targetNumber():
print ('Think of a number...')
print ('--------------------------')
maximum=int(input('The maximum number you thought is: ') )
print ('--------------------------')
print ('Is it the number you thought:')
minimum=1
number=round((maximum+minimum)/2.0)
proof=0
answer=1
print ('')
print (number)
print ('--------------------------')
while (answer !='c'):
answer=input('Press "c" for correct, press "h" for a higher number, press "l" for lower number: ')
if (answer=='h'):
proof=proof+1
minimum=number
number=round((maximum+minimum)/2.0)
print ('')
print (number)
print ('')
elif (answer=='l'):
proof=proof+1
maximum=number
number=round((maximum+minimum)/2.0)
print ('')
print (number)
print ('')
elif (answer=='c'):
proof=proof+1
print ('The number you thought is: ')
print ('')
print (number)
print ('')
print ('Congratulations!!! After several attempts, I managed to... ')
print (proof)
```

code by ---------- Senad Orhani---------- ----------

To get the desired result, execute targetNumber(), where then the obtained result will look like the following:

Think of a number...

The maximum number you thought is: 100

Is it the number you thought:

50.0

Press "c" for correct, press "h" for a higher number, press "l" for a lower number: h

75.0

Press "c" for correct, press "h" for a higher number, and press "l" for a lower number: l

62.0

Press "c" for correct, press "h" for a higher number, press "l" for a lower number: c
The number you thought is:

62.0

Congratulations!!! After several attempts, I managed to... 3
How the above pseudocode works:

The user first gives the maximum number and thinks of a number, while the program uses the search algorithm by asking if it is greater to type "h", if it is smaller to type "l" and if it is the required number to type "c".

The program is written in the Python programming language by copying the written pseudocode. Below, the mathematical formula round ((maximum + minimum)/ 2, 0) is used with the entire search algorithm.

Example: If the user gives the maximum value 100 and thinks the number 62?
The program asks if it is greater than 50, less than 50, or the number 50 itself, and prompts the user to type "h", “l" or "c". After pressing the "h" key, the algorithm continues by eliminating the numbers less than 50. Then, it continues by taking the numbers from 50 to 100 and finding the average, and thus asks the second question whether the required number is less than 75, greater than 75, or is the number 75.

The program also asks if the number you thought is 62, or greater than 62, or less than 62. And the user presses the "c" key for the correct answer, which is the number 62.
And at the end, the program calculates after how many attempts the number 62 was shot, after the calculation it is printed that after 3 attempts the required number was shot.

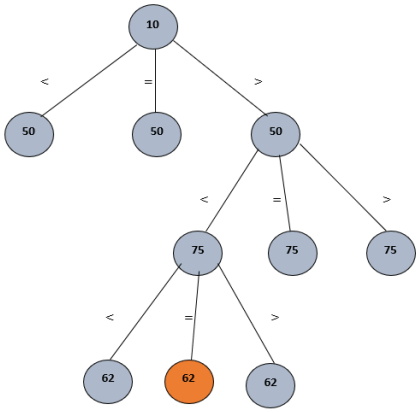

Figure 1. Search Algorithm for Natural Number Targets

4. CONCLUSION

In this paper, Binary Search Algorithm is addressed to find the target natural number. Finding a particular element in the list is the process of searching. The method is considered successful and returns the location of the element if the element is present in the list. If not, the search is considered fruitless. Binary search is justified by the assumption that a key exists. The value to be searched is stored in this key. The sum of the two highest and lowest values is divided by two. The highest and lowest values of the array, as well as its first and last elements. The key is then compared to the midpoint value. If the middle is the same as the key, we get the output immediately. The operation is repeated on the condensed array if the key is greater than the middle, in which case the middle +1 becomes the lowest value. Otherwise, the middle -1 becomes the largest value and the process repeats on the reduced set if the main value is less than the middle. An error message is displayed if it cannot be found anywhere (Sugandhi, 2022).

In other words, every day we encounter many problems and find one or more than one solutions to that particular problem. Some solutions may be more efficient than others and some solutions may be less efficient. In general, we tend to use the most efficient solution. Therefore, from the results of our study, we can say that the search algorithm was efficient in solving the problem of finding the target natural number. This code treated in this study with different modifications can also be implemented in calculation problems or solving other problems.

REFERENCES

Christodoulou, M., Szczygiel, E., & Klapa, Ł. (2018). Algorithmic and Programming. *PTEA Wszechnica Sp. z o. o*

Knuth, D. (1998). Sorting and Searching. The Art of Computer Programming, 3(2).

Krishna, A. (2022, 01 11). Search Algorithms – Linear Search and Binary Search Code Implementation and Complexity Analysis. Retrieved from FreeCodeCamp: https://www.freecodecamp.org/news/search-algorithms-linear-and-binary-search-explained/

Mahdi, A. Y. (2013). Algorithm Flow Charts. Retrieved from Faradars: https://faradars.org/wp-content/uploads/2015/07/algorithm-and-flow-chart.pdf

Meghanathan, N. (2015). Module 1: Analyzing the Efficiency of Algorithms. Jackson State University.

Reys, RE, Suydam, MN, & Lindquist, MM (1984). Helping Children Learn Mathematics. London: *Prentice-Hall International, Inc.*

Sugandhi, A. (2022). Binary Search Algorithm with Example Code. Retrieved from Knowledge Hut: https://www.knowledgehut.com/blog/programming/binary-search-algorithm