BASE-8 SEARCH : OCTARY SEARCH

Rishabh Johri

19BDS0021

1st yr , B Tech,CSE(BDS)

rishabh.johri2019@vitstudent.ac.in

ABSTRACT:

This research paper explains the details of working of octary search, its time complexity analysis and its comparison with binary search.

INTRODUCTION:

In computer science, there are many ways to search the position of the required input value in an array. There are algorithms such as binary search and linear search. In this research paper, I will describe another new search algorithm which has been written to make the search even faster. This algorithm has been named as octary search as it divides the array into eight parts. After dividing the array, seven elements are obtained which are compared to the specified input value. If the specified value matches any of the elements, the result is displayed. If not, then this process is repeated between any of the eight parts of the array depending upon the specified value. This process is repeated until the result is found.

LIMITATION OF BINARY SEARCH that I endeavor to improve through this new search algorithm:

1>Elements need to be sorted.

2>Time complexity : O(log2N)

Improvement:

1>the time complexity of base 8 search is : O(log8N)

Hence it is faster.(which will be proved later in this document)

………………………………………………………………………………………………………………………………………………………………………

Working of other Search Algorithms

A search algorithm finds the position of a specified input value within an array.

Linear Search:

Linear Search checks every element in the array sequentially until the desired element is found.

Binary Search:

A binary search locates an item in a sorted array by repeatedly dividing the search interval in half.

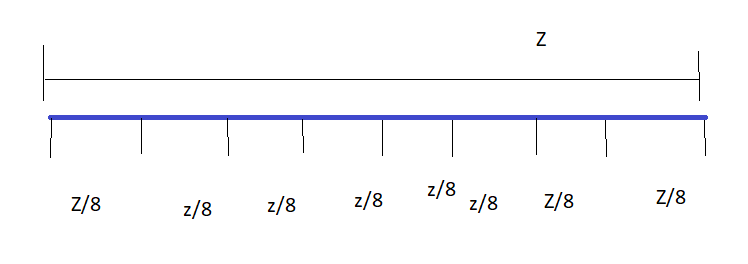
**Detailed Working of Octary Search :**

This algorithm works by dividing the array into eight equal parts. Hence, seven elements are obtained. The specified element is then compared to those seven elements. If any of them match with the required element, the position of that element in the array is obtained.

We need a general formula for dividing the array into eight equal parts. The formula is obtained as follows:

Aim: To find a general formula for dividing an array into eight equal parts.

Diagram:



Proof: Let the largest and lowest element of an array be ‘h’ and ‘l’ respectively. Let the middle element be ‘m’. Let the element at 1/8 distance from ‘l’ be ‘a’. Let the element at 1/8 distance from ‘h’ be ‘b’.

M=(l+h)/2

A=(l+M)/4

A=(l+(l+h)/2)/4

A=l/4 + l/8+ h/8

A=(3l+h)/8

Similarly:

B=(3h+l)/8

**Proof that octary search is faster than binary search**

(when not considering the time taken by if-else statement)

It is nearly impossible to consider the time taken by if else statements.

R.H.S = O(log2n)

L.H.S = O(log8n)

Aim:

To prove that L.H.S > R.H.S for any positive value of n.

Let O(log2n) = x

O(n) = 2x

L.H.S Let O(log8n) = y

O(n) = 8y

8 x> 2x for any positive value of x.

|  |  |  |
| --- | --- | --- |
| N | base 2 | base 8 |
| 100 | 6.64 | 2.21 |
| 1000 | 9.96 | 3.321 |
| 10000 | 13.28 | 4.429 |
| 100000 | 16.609 | 5.536 |
|  |  |  |
|  |  |  |
|  |  |  |



ALGORITHM :

1. If the array bounds are improper, return “error” and terminate the process.
2. Else compute the following numbers:

m=(l+h)/2

a1=(3l+h)/8

a2=2(3l+h)/8

a3=3(3l+h)/8

b1= 6(3h+l)/8

b2=5(3h+l)/8

b3=4(3h+l)/8

1. compare the target(element to be found)(ele) with each of the seven values:

if ele ==a1 or a2 or a3 or b1 or b2 or b3:

return (respective index) +1

1. else :

if l<ele< a1:

h=a1

repeat the process

elif a1<ele<a2:

l=a1 ,h=a2

repeat the process

elif a2<ele<a3:

l=a2,h=a3

repeat the process

elif a3<ele<b3:

l= a3,h=b3

repeat the process

elif b3<ele<b2:

l=b3,h=b2

repeat the process

elif b2<ele<b1:

l=b2,h=b1

repeat the process

APPLICATIONS:

Octary search is a robust and effiecient algorithm to search large datasets and thus it will have a wide application in data management

and IT sector.

LIMITATION:

1) The inputs should be in sorted order.

References:

1)geeksforgeeks.com

2)Data Structures in C by mark allen Weiss

Acknowledgement:

I would like to thank my DSA faculty Seetha R for guiding me.

Link for code explaination:

https://youtu.be/I4fx5VSgLjA