Why is Binary Search preferred over Ternary Search?

The following is a simple recursive **Binary Search** function in C++ taken from here.

```
// A recursive binary search function. It returns location of x in
// given array arr[l..r] is present, otherwise -1
int binarySearch(int arr[], int l, int r, int x)
{
   if (r >= l)
   {
        int mid = l + (r - l)/2;

        // If the element is present at the middle itself
        if (arr[mid] == x)  return mid;
```

```
        // If element is smaller than mid, then it can only be present
        // in left subarray
        if (arr[mid] > x) return binarySearch(arr, l, mid-1, x);

        // Else the element can only be present in right subarray
        return binarySearch(arr, mid+1, r, x);
   }

   // We reach here when element is not present in array
   return -1;
}
```

The following is a simple recursive **Ternary Search** function in C++.

```
// A recursive ternary search function. It returns location of x in
// given array arr[l..r] is present, otherwise -1
int ternarySearch(int arr[], int l, int r, int x)
{
   if (r >= l)
   {
        int mid1 = l + (r - l)/3;
        int mid2 = mid1 + (r - l)/3;

        // If x is present at the mid1
        if (arr[mid1] == x)  return mid1;

        // If x is present at the mid2
        if (arr[mid2] == x)  return mid2;

        // If x is present in left one-third
        if (arr[mid1] > x) return ternarySearch(arr, l, mid1-1, x);

        // If x is present in right one-third
        if (arr[mid2] < x) return ternarySearch(arr, mid2+1, r, x);

        // If x is present in middle one-third
        return ternarySearch(arr, mid1+1, mid2-1, x);
   }
   // We reach here when element is not present in array
   return -1;
}
```

Which of the above two does less comparisons in worst case?
From the first look, it seems the ternary search does less number of comparisons as it makes [Tex]Log_3 n[/Tex] recursive calls, but binary search makes [Tex]Log_2 n[/Tex] recursive calls. Let us take a closer look.
The following is recursive formula for counting comparisons in worst case of Binary Search.

```
   T(n) = T(n/2) + 2,  T(1) = 1
```

The following is recursive formula for counting comparisons in worst case of Ternary Search.

```
   T(n) = T(n/3) + 4, T(1) = 1
```

In binary search, there are [Tex]2log_2n[/Tex] + 1 comparisons in worst case. In ternary search, there are [Tex]4log_3n[/Tex] + 1 comparisons in worst case.

Therefore, the comparison of Ternary and Binary Searches boils down the comparison of expressions [Tex]2log_3n[/Tex] and [Tex]log_2n[/Tex]. The value of [Tex]2log_3n[/Tex] can be written as [Tex](2/Log_2 3) * Log_2 n[/Tex]. Since the value of [Tex]2/Log_2 3 [/Tex] is more than one, Ternary Search does more comparisons than Binary Search in worst case.

Exercise:
Why Merge Sort divides input array in two halves, why not in three or more parts?

This article is contributed by **Anmol**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Related Topics:

- Find Union and Intersection of two unsorted arrays
- Pythagorean Triplet in an array
- Maximum profit by buying and selling a share at most twice
- Design a data structure that supports insert, delete, search and getRandom in constant time
- Print missing elements that lie in range 0 – 99
- Iterative Merge Sort
- Group multiple occurrence of array elements ordered by first occurrence
- Given a sorted and rotated array, find if there is a pair with a given sum

Writing code in comment? Please use **ideone.com** and share the link here.

Times Square • 5 months ago

Doesn't Binary Search require 2 comparisons per recursive call ? First with the mid element and if its not equal then compare again to find which half to recurse next
if (arr[mid] == x) return mid;
if (arr[mid] > x)

1 • Reply • Share ›

GeeksforGeeks Mod → Times Square • 5 months ago

Thanks for pointing this out. We have updated the post.

• Reply • Share ›

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string

  - Memory Layout of C Programs
  - Understanding “extern” keyword in C
  - Median of two sorted arrays
  - Tree traversal without recursion and without stack!
  - Structure Member Alignment, Padding and Data Packing
  - Intersection point of two Linked Lists
  - Lowest Common Ancestor in a BST.
  - Check if a binary tree is BST or not
  - Sorted Linked List to Balanced BST
- Follow @GeeksforGeeks

- ## Recent Comments

  - lt_k

    i need help for coding this function in java...

    Java Programming Language · 1 hour ago

  - Piyush

    What is the purpose of else if (recStack[*i])...

    Detect Cycle in a Directed Graph · 1 hour ago

  - Andy Toh

    My compile-time solution, which agrees with the...

    Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

  - lucy

    because we first fill zero in first col and...

    Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

  - lucy

    @GeeksforGeeks i don't n know what is this long...

    Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 2 hours ago

  - manish

    Because TAN is not a subsequence of RANT. ANT...

    Given two strings, find if first string is a subsequence of second · 2 hours ago

-