Algorithms

Solve any problem to achieve a rank

View Leaderboard

Topics: Ternary Search

Ternary Search

TUTORIAL PROBLEMS

Like linear search and binary search, ternary search is a searching technique that is used to determine the position of a specific value in an array. In binary search, the sorted array is divided into two parts while in ternary search, it is divided into 3 parts and then you determine in which part the element exists.

Ternary search, like binary search, is a divide-and-conquer algorithm. It is mandatory for the array (in which you will search for an element) to be sorted before you begin the search. In this search, after each iteration it neglects ⅓ part of the array and repeats the same operations on the remaining ⅔.

Implementation

```
int ternary_search(int l,int r, int x)
{
    if(r>=l)
    {
        int mid1 = l + (r-l)/3;
        int mid2 = r -  (r-l)/3;
        if(ar[mid1] == x)
            return mid1;
        if(ar[mid2] == x)
            return mid2;
        if(x<ar[mid1])
```

```
            return ternary_search(l,mid1-1,x);
        else if(x>ar[mid2])
            return ternary_search(mid2+1,r,x);
        else
            return ternary_search(mid1+1,mid2-1,x);

    }
    return -1;
}
```

Let us consider the following example to understand the code.

Let the sorted array be $ar[]$=$\{2, 3, 5, 6, 8, 9, 12, 13, 14\}$ with indices from 0 to 8. You are required to find the position of $x = 13$ in this array. Divide the sorted array into the following 3 parts by evaluating the values of $mid1$ and $mid2$:

- $\{2, 3, 5\}$
- $\{6, 8, 9\}$
- $\{12, 13, 14\}$

Here $ar[mid1] = 5$ and $ar[mid2] = 12$. As 13 is not equal to $ar[mid1]$ and $ar[mid2]$ and it is also not smaller than $ar[mid1]$, you can safely assume that it lies in the 3^{rd} part of the array as it is greater than $ar[mid2]$.

Run the ternary search again with $l = 7$ and $r = 8$.

Now, $ar[mid1] = ar[7] = 13$ and $ar[mid2] = ar[8] = 14$.

As $ar[mid1] = x$, $mid1$ is the required answer.

If the value is not in the array, it returns -1 as the answer.

Complexity
$O(log_3 N)$, where N is the size of the array

Use of ternary search:

This concept is used in unimodal functions to determine the maximum or minimum value of that function. Unimodal functions are functions that, have a single highest value.

Let us consider a function $func$ in the interval $[a, b]$, and you are required to determine the x for which $func(x)$ is maximized. The function $func$ is unimodal in nature, i.e. it strictly increases in the interval $[a, x]$ and strictly decreases in the interval $[x, b]$.

This can be done by various other methods like double differentiation or by using a modified binary search. In the case when the function cannot be differentiated easily, ternary search is useful. It is less prone to errors and easy to implement when:

- Dealing with floating point integers

- Required maximum value is reached at the end of the interval.

Implementation

```
double func(double x)
{
return -1*1*x*x + 2*x +3;
}


double ts(double start, double end)
{
    double l = start, r = end;

    for(int i=0; i<200; i++) {
      double l1 = (l*2+r)/3;
      double l2 = (l+2*r)/3;
      //cout<<l1<<" "<<l2<<endl;
      if(func(l1) > func(l2)) r = l2; else l = l1;
      }

     double x = l;
    return func(x);
}
```

The code is making 200 iterations because at each step the interval $[a, b]$ is reduced to ⅔ of its previous size. After 200 iterations, the answer has an error of at most $⅔^{200}$ of the original interval, which is a good precision! You can modify this based on your requirements.

Let's understand the code.

You are first dividing the interval into the following 3 parts:

- $(start\ to\ l_1)$
- $(l_1 + 1\ to\ l_2 - 1)$
- (l_2, end)

At each iteration you search for the part in which the maximum lies and ignore ⅓ part of the current interval.

If $fun(l_1) > func(l_2)$, you can observe that the maximum value does not lie in the last interval. Therefore, ignore it by assigning $r = l_2$, otherwise the maximum value does not lie in the first part.

Consider a unimodal function, $-ax^2 + bx + 3$ where $a = 1, b = 2\ and\ c = 3$. The maximum value of $func(x)$ between the interval $start = 0$ and $end = 1$ will be 4.

Contributed by: Prateek Garg