**INTERPOLATION SEARCH**

1. **Introduce:**

*Example:* For example, we have a problem that does not include N integer elements (characters, ...) that have been sorted in ascending or descending order. We are asked to find an element X in it.

* Solution: With the above problem, we have many ways to do it such as using Linear Search, Binary Search, ... But the above ways will lead to stack overflow with N=1000000000000. But with the Interpolation Search algorithm, it will help us deal with large N (of course it also has a certain limit) because it has a complexity of O(). Example is N=1000000, Binary Search has O(log(1000000))=20, Linear Search has O(1000000) and Interpolation Search has O()=4,3 => The Interpolation Search algorithm is very efficient when N is large.

1. **Definition of Interpolation Search algorithm:**

-Interpolation Search is an improved variant of Binary Search. For this search algorithm to work correctly, the data set must be sorted. For example, in the case of the phone book, if we want to find the phone number of Craysis for example. In this case, Linear Search and also Binary Search can be slow to perform the search, since we can directly jump to the part of memory space whose name starts with C is stored.

1. **Locate in Interpolation Search:**

- Starting from the formula to find the middle part of the set according to Binary Search, we have:

pos = left + (right - left)/2 = (left + right)/2

- We will improve by replacing the value 1/2 with the following expression:

(X – T[left])/(T[right] – T[left]). So:

pos = left + (X- T[left]) \* (right – left) / (T[right] – T[left])

Inside:

+ X – T[left] : Number of elements from element X to the left element.

+ right – left : Number of elements in position from left to right.

+ (X- T[left]) / (T[right] – T[left]): In which region is the percentage of element X located.

1. **Implemention:**

bool Interpolation\_Sort(int arr[], int n,int x)

{

left <- 0;

right <- n - 1;

while (left <= right and x >= arr[left] and x <= arr[right])

{

val1 <- (x - arr[left]) / (arr[right] - arr[left]);

val2 <- (right - left);

pos <- left + val1 \* val2;

if (arr[pos] == x)

return pos;

if (arr[pos] < x)

left <- pos + 1;

else

right <- pos - 1;

}

return -1;

}

1. **Illustration:**