

**Faculty of Computing & Information Technology**

**Computer Science Department**

CPCS 223 Project

**Empirical Analysis Between Binary Search & Interpolation Search**

Analysis and Design of Algorithms

Student Name

**Omar Abdul-Aziz Hassan Al-Qurashi**

**ID: 1742589**

**Section: DB**

Supervisor Name

**Mr. Mohammed Shuaib Qureshi**

2019/1440H

**Algorithms' Pseudocode:**

**Binary Search Pseudocode:**

**ALGORITHM** *BinarySearch(A*[0*..n* − 1]*, K)*

//Implements nonrecursive binary search

//Input: An array *A*[0*..n* − 1] sorted in ascending order and

// a search key *K*

//Output: An index of the array's element that is equal to *K*

// or −1 if there is no such element

*L* ← 0; *r* ← *n* − 1

**while** *l* ≤ *r* **do**

*m* ←

**if** *K* = *A*[*m*] **return** *m*

**else if** *K < A*[*m*] *r* ← *m* − 1

**else** *l* ← *m* + 1

**return** −1

**Interpolation Search Pseudocode:**

**ALGORITHM** *InterpolationSearch(A*[0*..n* − 1]*, v)*

//Implements nonrecursive Interpolation Search

//Input: An array *A*[0*..n* − 1] sorted in ascending order and

// a search value *v*

//Output: An index of the array's element that is equal to *v*

// or −1 if there is no such element

*L* ← 0; *r* ← *n* − 1

**while** *l* ≤ *r and v ≥ A[l] and v ≤ A[r]***do**

*x* ←

**if** *v* = *A*[*x*] **return** *x*

**else if** *v < A*[*x*] *r* ← *x* − 1

**else** *l* ← *x* + 1

**return** −1

**Study Design:**

**Inputs:**

* Random **key value** generated in range 0-2000.
* Array with size {0,100000,200000,300000,400000,500000}, and random elements' value generated in range 0-1000.

**Procedures:**

By using the previews inputs in our source code (JAVA code is in the **appendix**) and run in **NetBeans IDE**, the output of the program is the following:

|  |
| --- |
| For n = 0:  Binary Search | Interpolation Search  Total Time: 2051 | Total Time: 3692 (Trial: 1)  Total Time: 410 | Total Time: 820 (Trial: 2)  Total Time: 411 | Total Time: 410 (Trial: 3)  ----------------- Cases ------------------  Best case: 410 | Best case: 410  Worst case: 2051 | Worst case: 3692  AVG case: 957 | AVG case: 1641  For n = 100000:  Binary Search | Interpolation Search  Total Time: 2052 | Total Time: 820 (Trial: 1)  Total Time: 1641 | Total Time: 820 (Trial: 2)  Total Time: 2051 | Total Time: 821 (Trial: 3)  ----------------- Cases ------------------  Best case: 1641 | Best case: 820  Worst case: 2052 | Worst case: 821  AVG case: 1915 | AVG case: 820  For n = 200000:  Binary Search | Interpolation Search  Total Time: 10257 | Total Time: 5743 (Trial: 1)  Total Time: 1231 | Total Time: 410 (Trial: 2)  Total Time: 1641 | Total Time: 1231 (Trial: 3)  ----------------- Cases ------------------  Best case: 1231 | Best case: 410  Worst case: 10257 | Worst case: 5743  AVG case: 4376 | AVG case: 2461  For n = 300000:  Binary Search | Interpolation Search  Total Time: 4103 | Total Time: 821 (Trial: 1)  Total Time: 1231 | Total Time: 1230 (Trial: 2)  Total Time: 1641 | Total Time: 820 (Trial: 3)  ----------------- Cases ------------------  Best case: 1231 | Best case: 820  Worst case: 4103 | Worst case: 1230  AVG case: 2325 | AVG case: 957  For n = 400000:  Binary Search | Interpolation Search  Total Time: 17641 | Total Time: 15179 (Trial: 1)  Total Time: 6974 | Total Time: 410 (Trial: 2)  Total Time: 5743 | Total Time: 1641 (Trial: 3)  ----------------- Cases ------------------  Best case: 5743 | Best case: 410  Worst case: 17641 | Worst case: 15179  AVG case: 10119 | AVG case: 5743  For n = 500000:  Binary Search | Interpolation Search  Total Time: 20102 | Total Time: 3692 (Trial: 1)  Total Time: 1231 | Total Time: 821 (Trial: 2)  Total Time: 2052 | Total Time: 410 (Trial: 3)  ----------------- Cases ------------------  Best case: 1231 | Best case: 410  Worst case: 20102 | Worst case: 3692  AVG case: 7795 | AVG case: 1641 |

**Findings:**

The total times shows the amount of time that **Binary Search** and **Interpolation Search** took in nanosecond. Also, the output shows three important info:

* Best Case
* Worst Case
* Average Case

From the previews information that we saw about the three cases (Best, Worst and Average) of **Binary Search** and **Interpolation Search**, we can see that the **Interpolation Search** usually is faster than **Binary Search**.

(Scatter Plots show the difference between **Binary Search** and **Interpolation Search** in term of Time efficiency)

**Conclusion:**

Although that **Binary Search** algorithm is slower than **Interpolation Search** algorithm, **Binary Search** is still useful if we use with small datasets because the difference between it and **Interpolation Search** algorithm is not that big, actually, it is sometimes better to use **Binary Search** algorithm rather than **Interpolation Search** algorithm because it is more reliable (The **Interpolation Search** algorithm's formula can cause error if )

|  |
| --- |
| if(totalTime < bestCaseI)  bestCaseI = totalTime;  if(totalTime > worstCaseI)  worstCaseI = totalTime;  } // end of the inner "for loop"  System.out.printf(" ----------------- Cases ------------------\n"  + " Best case: %6d | Best case: %6d\n"  + " Worst case:%6d | Worst case:%6d\n"  + " AVG case: %6.0f | AVG case: %6.0f\n\n",  bestCaseB,bestCaseI,worstCaseB,worstCaseI, (double)totalTimeB/3,(double)totalTimeI/3);  } // end of the outer "for loop"  }  public static int binarySearch(int A[],int k){  int l = 0; // Lower bound  int r = A.length - 1; // Upper bound  while(l <= r ){  int m = ( l + r ) / 2; // midpoint  if (A[m] == k)  return m; // found  else if (A[m] > k)  r = m - 1;  else  l = m + 1;  }  return -1;// if not found...  } // end of binary search  public static int InterpolationSearch(int A[],int v){  int l = 0; // Lower bound  int r = A.length - 1; // Upper bound  while(l <= r && v >= A[l] && v <= A[r]){  int x = l + ((v-A[l])\*(r-l)/(A[r]-A[l]));  if (A[x] == v)  return x; // found  else if (A[x] > v)  r = x - 1;  else  l = x + 1;  }  return -1;// if not found...  } // end of Interpolation Search  } |
| //Name: Omar Abdulziz Alqurashi, 1742589, Section: DB  import java.util.Arrays;  public class CPCS223\_Project {  public static void main(String[] args) {  long startTime, endTime,totalTimeB, bestCaseB,worstCaseB,  totalTimeI,bestCaseI,worstCaseI,totalTime;  int keyValue;  for(int n = 0; n <= 500000; n = n + 100000){  int array[]=new int[n]; // size = n  System.out.println("For n = "+(n)+":\n"  +" Binary Search | Interpolation Search");  // initialization of the following:  totalTimeB = 0;  bestCaseB = 2000000000; worstCaseB = -1;  totalTimeI = 0;  bestCaseI = 2000000000; worstCaseI = -1;  for(int i = 0; i < n; i++)  array[i]= (int)(Math.random()\*1001);  for(int t = 0; t < 3; t++){ // trials  keyValue = (int)(Math.random()\*2001);  Arrays.sort(array);//sort the array  startTime = System.nanoTime();  binarySearch(array, keyValue);  endTime = System.nanoTime();  totalTime = endTime - startTime;  System.out.printf(" Total Time: %6d | "  , totalTime);//nanoSecond  totalTimeB += totalTime;  if(totalTime < bestCaseB)  bestCaseB = totalTime;  if(totalTime > worstCaseB)  worstCaseB = totalTime;    startTime = System.nanoTime();  InterpolationSearch(array, keyValue);  endTime = System.nanoTime();  totalTime = endTime - startTime;  System.out.printf("Total Time: %6d (Trial: "  +(t+1)+")\n", totalTime);//nanoSecond  totalTimeI += totalTime; |