**Title: Binary Search**

**Abstract:**

Binary search is a more specialized algorithm than sequential search as it takes advantage of data that has been sorted. The underlying idea of binary search is to divide the sorted data into two halves and to examine the data at the point of the split. Since the data is sorted, we can easily ignore one half or the other depending on where the data we're looking for lies in comparison to the data at the split. This makes for a much more efficient search than linear search.

**Methodology:** At first We open our Code-Blocks Editor ,then File, New , Empty file, save As. Then file Name : Insertion ,save . finally coding start . At first We Declaration header file ,then define p of print,and s of scanf , then Declaration main method .

**Discussion:**

* **Binary search** works on sorted arrays.
* **Binary search** begins by comparing the middle element of the array with the target value.
* The target value matches the middle element, its position in the array is returned.
* The target value is less than the middle element, the **search** continues in the lower half of the array.
* **Worst-case space complexity**‎: ‎[O(1)](https://en.wikipedia.org/wiki/Big_O_notation#Orders_of_common_functions)
* **Worst-case performance**‎: ‎[O(log n)](https://en.wikipedia.org/wiki/Big_O_notation#Orders_of_common_functions)
* **Best-case performance**‎: ‎[O(1)](https://en.wikipedia.org/wiki/Big_O_notation#Orders_of_common_functions)
* **Average performance**‎: ‎[O(log n)](https://en.wikipedia.org/wiki/Big_O_notation#Orders_of_common_functions)
* There are many cases in Binary search proves to be a better alternative to linear search.
* The reason it is used mostly is that binary search has a time complexity of **O(log(n))** for each search on a list of **n items** provided items are in **sorted order.**
* [The Ubiquitous Binary Search](https://www.geeksforgeeks.org/the-ubiquitous-binary-search-set-1/)
* [Interpolation search vs Binary search](https://www.geeksforgeeks.org/g-fact-84/)
* [Find the minimum element in a sorted and rotated array](https://www.geeksforgeeks.org/find-minimum-element-in-a-sorted-and-rotated-array/)
* [Count the number of occurrences in a sorted array](https://www.geeksforgeeks.org/count-number-of-occurrences-in-a-sorted-array/)
* [Median of two sorted arrays](https://www.geeksforgeeks.org/median-of-two-sorted-arrays/)
* [Floor and Ceiling in a sorted array](https://www.geeksforgeeks.org/search-floor-and-ceil-in-a-sorted-array/)
* [Find the maximum element in an array which is first increasing and then decreasing](https://www.geeksforgeeks.org/find-the-maximum-element-in-an-array-which-is-first-increasing-and-then-decreasing/)
* **Binary Search** and **Applications** in Swift.

Source code:

#include <stdio.h>

int main()

{

int c, first, last, middle, n, search, array[100];

printf("Enter number of elements\n");

scanf("%d",&n);

printf("Enter %d integers\n", n);

for (c = 0; c < n; c++)

scanf("%d",&array[c]);

printf("Enter value to find\n");

scanf("%d", &search);

first = 0;

last = n - 1;

middle = (first+last)/2;

while (first <= last) {

if (array[middle] < search)

first = middle + 1;

else if (array[middle] == search) {

printf("%d found at location %d.\n", search, middle+1);

break;

}

else

last = middle - 1;

middle = (first + last)/2;

}

if (first > last)

printf("Not found! %d isn't present in the list.\n", search);

return 0;

}

