Programming Refresher Workshop

Session 3 Exercises

Learning objectives:

- Using array
- Adapting binary search
- Problem solving

Exercise 7 (ex7): Efficient search on a sorted, rotated array

You have seen how binary search can be used to efficiently search for the index of an element within a given sorted array. Suppose the sorted array is now rotated some elements to the right with wrap-around. As an example, consider the sorted array below.

4	8	9	10	11	12	15	27
0	1	2	3	4	5	6	7

Rotating the sorted array three elements towards the right gives

12	15	27	4	8	9	10	11
0	1	2	3	4	5	6	7

To perform a search on this array, a simple linear search can be performed: searching for the element 9 returns the index 5; searching for 15 returns 1; searching for 13 returns -1. However, the worst-case performance for a linear search of n elements is n comparisons.

On the other hand, binary search has been shown to be a much more efficient search routine; however, it can only be performed on a sorted array. Your task is to adapt the binary search routine to a sorted and rotated array.

<u>Input</u>

The first line of input consists of a sequence of unique integers separated by spaces.

The second line of input contains the key value to be searched.

You may assume that there are at most 100 values in the sequence.

<u>Output</u>

The output is the index of the element within the array where the key value is located; otherwise the program outputs "Not found".

Sample run

```
Enter data:
12 15 27 4 8 9 10 11
12
0
```

```
Enter data:
12 15 27 4 8 9 10 11
10
6
```

```
Enter data:
12 15 27 4 8 9 10 11
13
Not found
```

Algorithm template

Input

How to accept the input for the array and key value?

Processing

How to adapt binary search to a sorted, rotated array?

How to decide which half of the array to discard?

Output

How to output the result?