**Lab: Searching and Sorting Algorithms**

This document defines the lab for ["Algorithms with Python" course @Software University](https://softuni.bg/opencourses/algorithms-with-python)".

Please submit your solutions (source code) of all below-described problems in [Judge](https://judge.softuni.org/Contests/3461/).

## Binary Search

Implement an algorithm that finds the index of an element in a sorted array of integers in logarithmic time.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 3 4 5  1 | 0 | Index of 1 is 0 |
| -1 0 1 2 4  1 | 2 | Index of 1 is 2 |

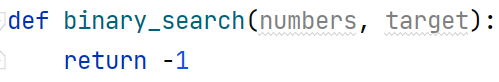
### Hints

First, if you’re not familiar with the concept, read about binary search in [Wikipedia](https://en.wikipedia.org/wiki/Binary_search_algorithm).

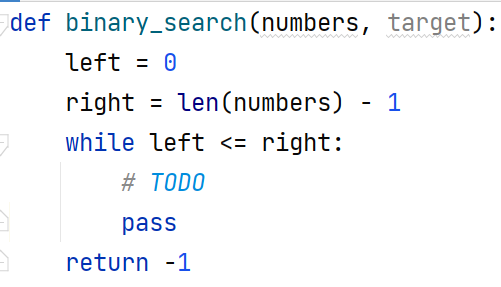
In short, if we have a **sorted collection** of comparable elements, instead of doing a linear search (which takes linear time), we can eliminate half the elements at each step and finish in logarithmic time. Binary search is a **divide-and-conquer** algorithm; we start at the middle of the collection, if we haven’t found the element there, there are three possibilities:

* The element we’re looking for is smaller – then look to the left of the current element, we know all elements to the right are larger
* The element we’re looking for is larger – look to the right of the current element
* The element is not present, traditionally, return -1 in that case

Start by defining a method:



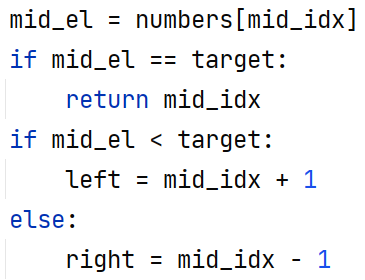
Inside the method, define two variables defining the bounds to be searched and a while loop:



Inside the while loop, we need to find the midpoint:



If the key is to the left of the midpoint, move the right bound. If the key is to the right of the midpoint, move the left bound:



## Selection Sort

Write an implementation of Selection Sort. You should read an array of integers and sort them.

### Output

* You should print out the sorted list in the format described below.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 4 3 2 1 | 1 2 3 4 5 |
| 13 93 37 74 61 65 5 55 17 96 52 70 17 7 89 65 16 38 42 15 86 21 93 10 31 28 36 14 65 7 68 86 97 34 27 32 86 44 51 75 29 64 0 36 33 54 20 40 60 56 51 51 25 77 75 46 47 57 18 12 27 28 29 21 22 37 74 78 34 15 71 75 20 19 76 48 98 36 76 49 83 21 44 12 85 68 24 9 80 41 66 1 54 31 55 33 88 35 32 43 | 0 1 5 7 7 9 10 12 12 13 14 15 15 16 17 17 18 19 20 20 21 21 21 22 24 25 27 27 28 28 29 29 31 31 32 32 33 33 34 34 35 36 36 36 37 37 38 40 41 42 43 44 44 46 47 48 49 51 51 51 52 54 54 55 55 56 57 60 61 64 65 65 65 66 68 68 70 71 74 74 75 75 75 76 76 77 78 80 83 85 86 86 86 88 89 93 93 96 97 98 |

## Bubble Sort

Write an implementation of Bubble Sort. You should read an array of integers and sort them.

### Output

* You should print out the sorted list in the format described below.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 4 3 2 1 | 1 2 3 4 5 |
| 13 93 37 74 61 65 5 55 17 96 52 70 17 7 89 65 16 38 42 15 86 21 93 10 31 28 36 14 65 7 68 86 97 34 27 32 86 44 51 75 29 64 0 36 33 54 20 40 60 56 51 51 25 77 75 46 47 57 18 12 27 28 29 21 22 37 74 78 34 15 71 75 20 19 76 48 98 36 76 49 83 21 44 12 85 68 24 9 80 41 66 1 54 31 55 33 88 35 32 43 | 0 1 5 7 7 9 10 12 12 13 14 15 15 16 17 17 18 19 20 20 21 21 21 22 24 25 27 27 28 28 29 29 31 31 32 32 33 33 34 34 35 36 36 36 37 37 38 40 41 42 43 44 44 46 47 48 49 51 51 51 52 54 54 55 55 56 57 60 61 64 65 65 65 66 68 68 70 71 74 74 75 75 75 76 76 77 78 80 83 85 86 86 86 88 89 93 93 96 97 98 |

## Insertion Sort

Write an implementation of Insertion Sort. You should read an array of integers and sort them.

### Output

* You should print out the sorted list in the format described below.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 4 3 2 1 | 1 2 3 4 5 |
| 13 93 37 74 61 65 5 55 17 96 52 70 17 7 89 65 16 38 42 15 86 21 93 10 31 28 36 14 65 7 68 86 97 34 27 32 86 44 51 75 29 64 0 36 33 54 20 40 60 56 51 51 25 77 75 46 47 57 18 12 27 28 29 21 22 37 74 78 34 15 71 75 20 19 76 48 98 36 76 49 83 21 44 12 85 68 24 9 80 41 66 1 54 31 55 33 88 35 32 43 | 0 1 5 7 7 9 10 12 12 13 14 15 15 16 17 17 18 19 20 20 21 21 21 22 24 25 27 27 28 28 29 29 31 31 32 32 33 33 34 34 35 36 36 36 37 37 38 40 41 42 43 44 44 46 47 48 49 51 51 51 52 54 54 55 55 56 57 60 61 64 65 65 65 66 68 68 70 71 74 74 75 75 75 76 76 77 78 80 83 85 86 86 86 88 89 93 93 96 97 98 |

## Quicksort

Sort an array of elements using the famous quicksort.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 4 3 2 1 | 1 2 3 4 5 |

## Merge Sort

Sort an array of elements using the famous merge sort.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 4 3 2 1 | 1 2 3 4 5 |