

# CSC2111 Computer Science I Lab

## Lab 22

# Objectives

1. Explore how to sort an array using insertion sort algorithm.
2. Learn how to implement binary search algorithm.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	2.78	7.42	0.56	1.12	1.17	0.32	6.21	4.42	3.14	7.71

Iteration 0: step 0.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	2.78	7.42	0.56	1.12	1.17	0.32	6.21	4.42	3.14	7.71


Iteration 1: step 0.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	2.78	0.56	7.42	1.12	1.17	0.32	6.21	4.42	3.14	7.71




Iteration 2: step 0.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	2.78	7.42	1.12	1.17	0.32	6.21	4.42	3.14	7.71



Iteration 2: step 1.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	2.78	7.42	1.12	1.17	0.32	6.21	4.42	3.14	7.71


Iteration 2: step 2.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	2.78	1.12	7.42	1.17	0.32	6.21	4.42	3.14	7.71



Iteration 3: step 0.




# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	1.12	2.78	7.42	1.17	0.32	6.21	4.42	3.14	7.71



Iteration 3: step 1.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	1.12	2.78	7.42	1.17	0.32	6.21	4.42	3.14	7.71


Iteration 3: step 2.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	1.12	2.78	1.17	7.42	0.32	6.21	4.42	3.14	7.71




Iteration 4: step 0.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	1.12	1.17	2.78	7.42	0.32	6.21	4.42	3.14	7.71



Iteration 4: step 1.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	1.12	1.17	2.78	7.42	0.32	6.21	4.42	3.14	7.71


Iteration 4: step 2.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	1.12	1.17	2.78	0.32	7.42	6.21	4.42	3.14	7.71




Iteration 5: step 0.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	1.12	1.17	0.32	2.78	7.42	6.21	4.42	3.14	7.71




Iteration 5: step 1.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	1.12	0.32	1.17	2.78	7.42	6.21	4.42	3.14	7.71



Iteration 5: step 2.




# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.56	0.32	1.12	1.17	2.78	7.42	6.21	4.42	3.14	7.71




Iteration 5: step 3.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	7.42	6.21	4.42	3.14	7.71



Iteration 5: step 4.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	7.42	6.21	4.42	3.14	7.71


Iteration 5: step 5.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	6.21	7.42	4.42	3.14	7.71



Iteration 6: step 0.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	6.21	7.42	4.42	3.14	7.71


Iteration 6: step 1.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	6.21	4.42	7.42	3.14	7.71




Iteration 7: step 0.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	4.42	6.21	7.42	3.14	7.71



Iteration 7: step 1.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	4.42	6.21	7.42	3.14	7.71

Iteration 7: step 2.




# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	4.42	6.21	3.14	7.42	7.71




Iteration 8: step 0.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	4.42	3.14	6.21	7.42	7.71




Iteration 8: step 1.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	3.14	4.42	6.21	7.42	7.71



Iteration 8: step 2.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	3.14	4.42	6.21	7.42	7.71

Iteration 8: step 3.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	3.14	4.42	6.21	7.42	7.71

Iteration 9: step 0.

# Insertion Sort

**Iteration  $i$ .** Repeatedly swap element  $i$  with the one to its left if smaller.

**Property.** After  $i$ th iteration,  $a[0]$  through  $a[i]$  contain first  $i+1$  elements in ascending order.

Array index	0	1	2	3	4	5	6	7	8	9
Value	0.32	0.56	1.12	1.17	2.78	3.14	4.42	6.21	7.42	7.71

Iteration 10: **DONE**.

# Binary Search

**Binary search.** Given `value` and sorted array `a[]`, find index `i` such that `a[i] = value`, or report that no such index exists.

**Invariant.** Algorithm maintains  $a[\text{lo}] \leq \text{value} \leq a[\text{hi}]$ .

Ex. Binary search for 33.

[illegible]

# Binary Search

**Binary search.** Given `value` and sorted array `a[]`, find index `i` such that `a[i] = value`, or report that no such index exists.

**Invariant.** Algorithm maintains  $a[lo] \leq value \leq a[hi]$ .

**Ex.** Binary search for 33.

6	13	14	25	33	43	51	53	64	72	84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
↑							↑							↑
lo							mid							hi

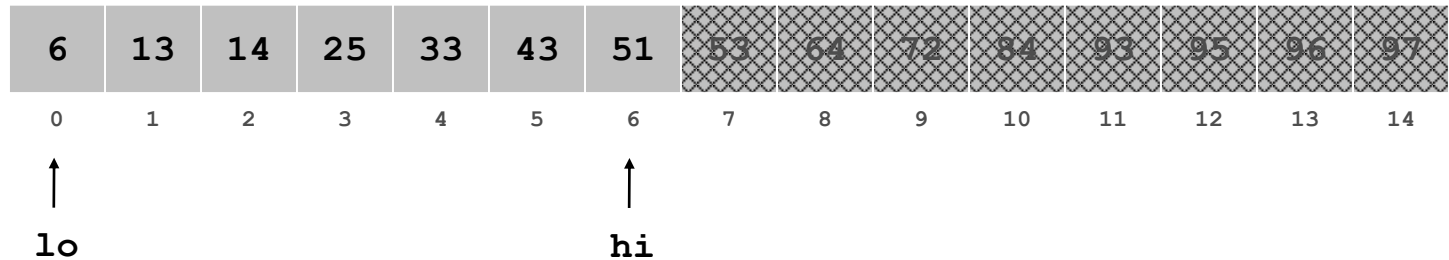


# Binary Search

**Binary search.** Given `value` and sorted array `a[]`, find index `i` such that `a[i] = value`, or report that no such index exists.

**Invariant.** Algorithm maintains  $a[lo] \leq value \leq a[hi]$ .

**Ex.** Binary search for 33.

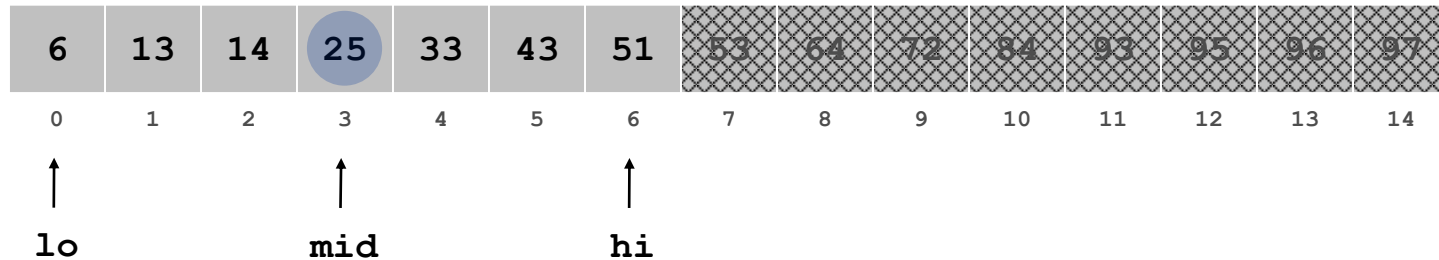


# Binary Search

**Binary search.** Given `value` and sorted array `a[]`, find index `i` such that `a[i] = value`, or report that no such index exists.

**Invariant.** Algorithm maintains  $a[\text{lo}] \leq \text{value} \leq a[\text{hi}]$ .

Ex. Binary search for 33.

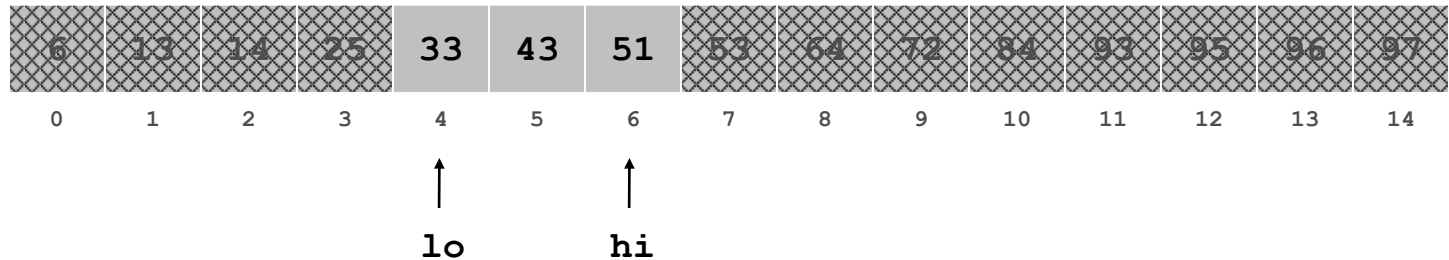


# Binary Search

**Binary search.** Given `value` and sorted array `a[]`, find index `i` such that `a[i] = value`, or report that no such index exists.

**Invariant.** Algorithm maintains  $a[\text{lo}] \leq \text{value} \leq a[\text{hi}]$ .

Ex. Binary search for 33.

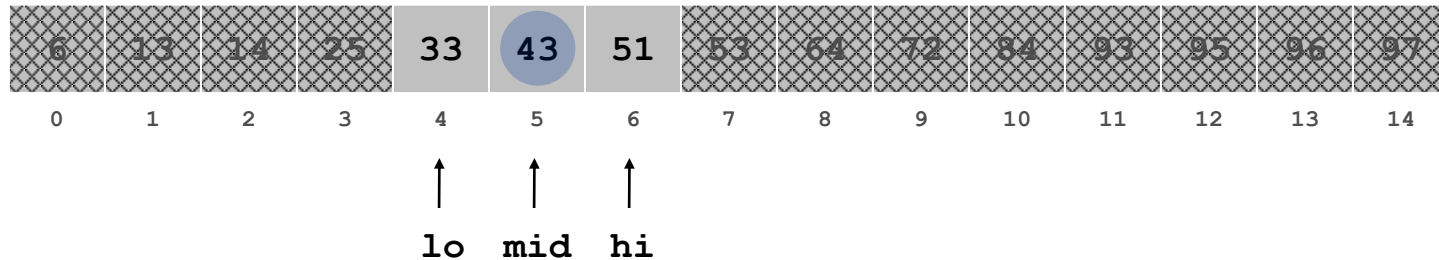


# Binary Search

**Binary search.** Given `value` and sorted array `a[]`, find index `i` such that `a[i] = value`, or report that no such index exists.

**Invariant.** Algorithm maintains  $a[\text{lo}] \leq \text{value} \leq a[\text{hi}]$ .

Ex. Binary search for 33.

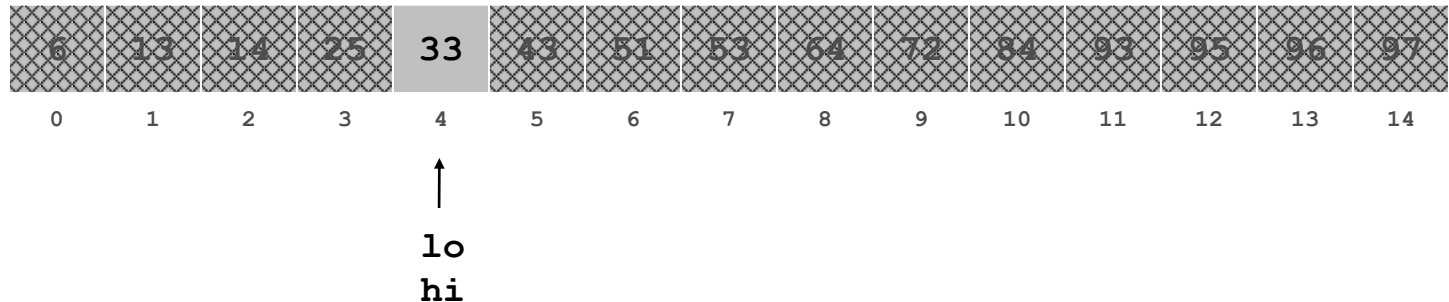


# Binary Search

**Binary search.** Given `value` and sorted array `a[]`, find index `i` such that `a[i] = value`, or report that no such index exists.

**Invariant.** Algorithm maintains  $a[lo] \leq value \leq a[hi]$ .

**Ex.** Binary search for 33.

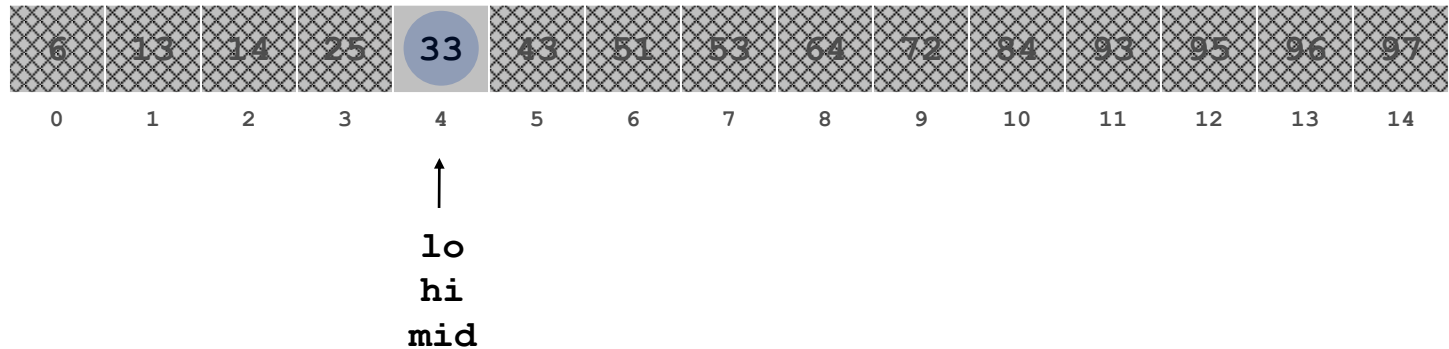


# Binary Search

**Binary search.** Given `value` and sorted array `a[]`, find index `i` such that `a[i] = value`, or report that no such index exists.

**Invariant.** Algorithm maintains  $a[lo] \leq value \leq a[hi]$ .

**Ex.** Binary search for 33.



# Binary Search

**Binary search.** Given `value` and sorted array `a[]`, find index `i` such that `a[i] = value`, or report that no such index exists.

**Invariant.** Algorithm maintains  $a[lo] \leq value \leq a[hi]$ .

**Ex.** Binary search for 33.

