ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Applied Discrete Mathematics

Spring 2024 - **Dr. Abdelgwad Elashry**
Abdelgwad.Elashry@sut.edu.eg

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Applied Discrete Mathematics
Lecture 6

Table of Contents

Concept of Algorithms.

Introduction

There are many general classes of problems that arise in discrete mathematics.

For instance: given a sequence of integers, find the largest one; given a set, list all its subsets; given a set of integers, put them in increasing order; given a network, find the shortest path between two vertices.

Concept of Algorithms.

Introduction

When presented with such a problem,

1. The first thing to do is to construct a model that translates the problem into a mathematical context.
2. To complete the solution, a method is needed that will solve the general problem using the model.
3. Ideally, what is required is a procedure that follows a sequence of steps that leads to the desired answer. Such a sequence of steps is called an **algorithm**.

Concept of Algorithms.

Definition 1

An ***algorithm*** is a finite sequence of precise instructions for performing a computation or for solving a problem.

MOHAMMED IBN MUSA AL-KHOWARIZMI

Concept of Algorithms.

EXAMPLE

Describe an algorithm for finding the maximum (largest) value in a finite sequence of integers.

10	5	7	25	2	14
a_1	a_2	a_3	a_4	a_5	a_6

Concept of Algorithms.

EXAMPLE

Describe an algorithm for finding the maximum (largest) value in a finite sequence of integers.

10	5	7	25	2	14
a_1	a_2	a_3	a_4	a_5	a_6

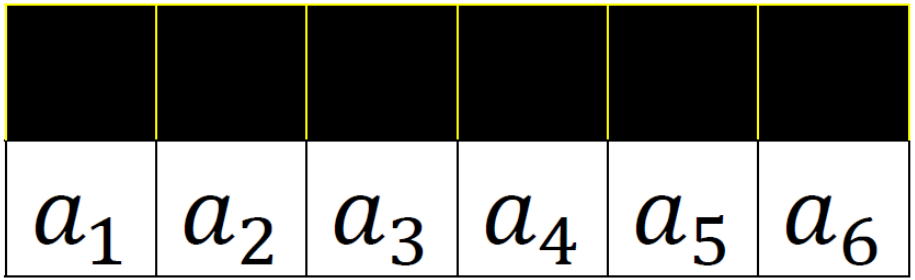
max = 25

return 25

Concept of Algorithms.

EXAMPLE

Describe an algorithm for finding the maximum (largest) value in a finite sequence of integers.



Concept of Algorithms.

Solution: We perform the following steps:

1.Set the *temporary maximum* equal to the first integer in the sequence. (The temporary maximum will be the largest integer examined at any stage of the procedure.)

If you start from the **left**.

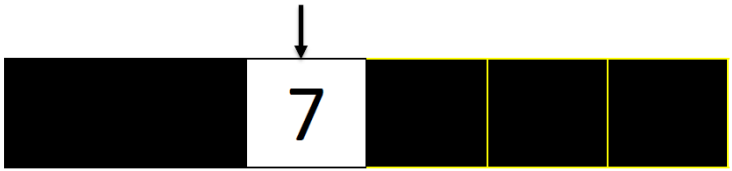
Concept of Algorithms.

Solution: We perform the following steps:

2.Compare the next integer in the sequence to the temporary maximum, and if it is larger than the temporary maximum, set the temporary maximum equal to this integer.

if (*value* > 10) **then** (temporary maximum = *value*)

temporary maximum



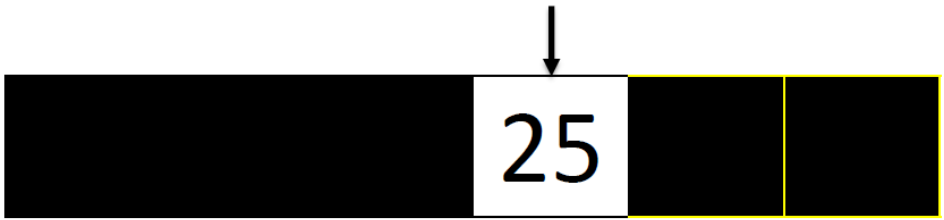
if (*value* > 10) **then** (temporary maximum = *value*)

temporary maximum

Concept of Algorithms.

Solution: We perform the following steps:

3.Repeat the previous step if there are more integers in the sequence.



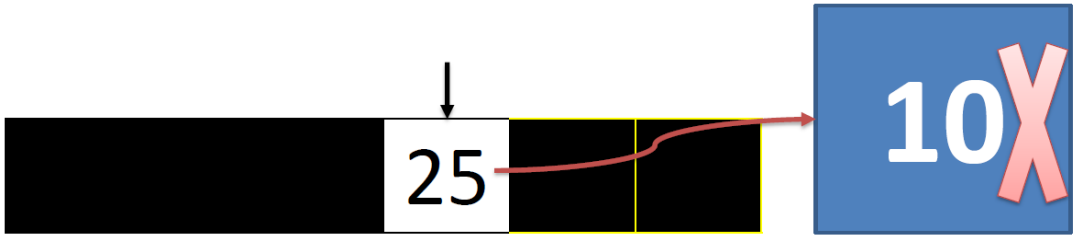
if (*value* > 10) **then** (temporary maximum = *value*)

temporary maximum

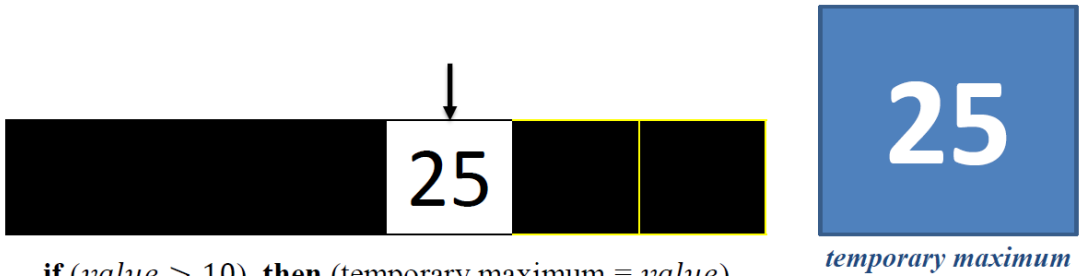
Concept of Algorithms.

Solution: We perform the following steps:

3.Repeat the previous step if there are more integers in the sequence.



if (*value* > 10) **then** (temporary maximum = *value*)



if (*value* > 10) **then** (temporary maximum = *value*)

Concept of Algorithms.

Solution: We perform the following steps:

3.Repeat the previous step if there are more integers in the sequence.

if (*value* > 25) **then** (temporary maximum = *value*)

temporary maximum

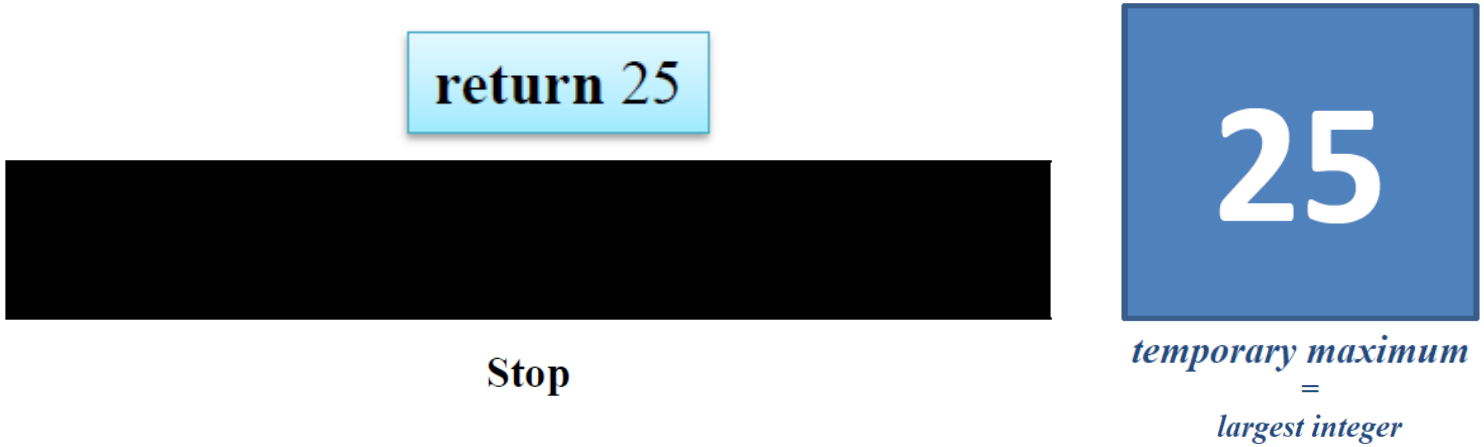
if (*value* > 25) **then** (temporary maximum = *value*)

temporary maximum

Concept of Algorithms.

Solution: We perform the following steps:

4. Stop when there are no integers left in the sequence. The temporary maximum at this point is the largest integer in the sequence.



Concept of Algorithms.

Solution: pseudocode

ALGORITHM 1 Finding the Maximum Element in a Finite Sequence.

procedure $max(a_1, a_2, \ldots, a_n$: integers)
$max := a_1$
for $i := 2$ **to** n
    if $max < a_i$ **then** $max := a_i$
return max {max is the largest element}

10	5	7	25	2	14
a_1	a_2	a_3	a_4	a_5	a_6

Searching Algorithms.

- Concept of Algorithms.
- ***Searching Algorithms.***
- Sorting Algorithms.

Searching Algorithms.

Introduction

Locate the ***value*** = **2** or determine that it is not in the list.

10	5	7	25	2	14
a_1	a_2	a_3	a_4	a_5	a_6

Searching Algorithms.

Introduction

Locate the ***value*= 2** or determine that it is not in the list.

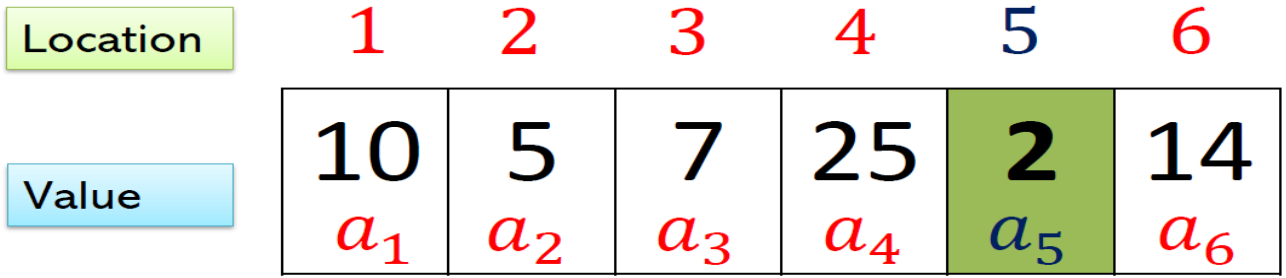
10	5	7	25	**2**	14
a_1	a_2	a_3	a_4	a_5	a_6

The *value* 2 is founded in the location of a_5, namely, 5.

Searching Algorithms.

Introduction

Locate the ***value*= 2** or determine that it is not in the list.



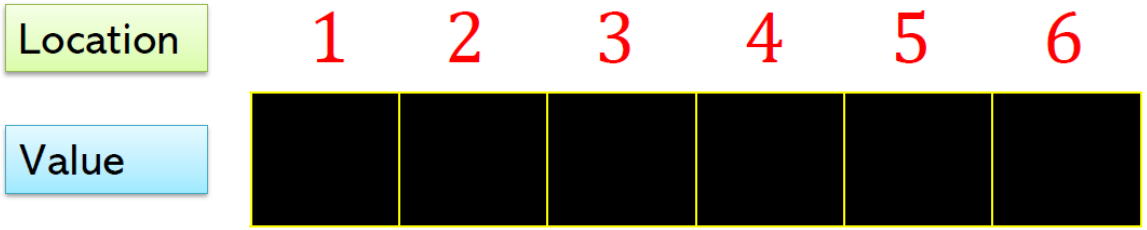
The *value* 2 is founded in the location of a_5, namely, 5.

return 5

Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Locate an element **2** in this list



You can start from the right, left, or middle.

Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Locate an element **2** in this list

Location	1	2	3	4	5	6
Value	10					

Not found in the 1st location

Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Locate an element **2** in this list

Location	1	2	3	4	5	6
Value	10	5	7	25		

Not found in the 4th location

Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Locate an element **2** in this list

Location	1	2	3	4	5	6
Value	10	5	7	25	2	

Founded in the **5th** location

return 5

Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Locate an element 99 in this list

Location	1	2	3	4	5	6
Value	10	5	7	25	2	

Not found in the 5th location

Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Locate an element **99** in this list

Location	1	2	3	4	5	6
Value	10	5	7	25	2	14

Not Founded in all the list

return 0

Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

1.Comparing $x$and $a1$.
When $x=a1$, return the location of $a1$, namely, 1.

2.When $x \neq a1$, compare $x$with $a2$.
If $x=a2$, return the location of $a2$, namely, 2.

3. When $x \neq a2$, compare x with $a3$, and so on.
Continue this process, comparing x successively with each term of the list until a match is found, where the solution is the location of that term, unless no match occurs. If the entire list has been searched without locating x, **return0**

Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	2	3
	10	5	7
	a_1	a_2	a_3

Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

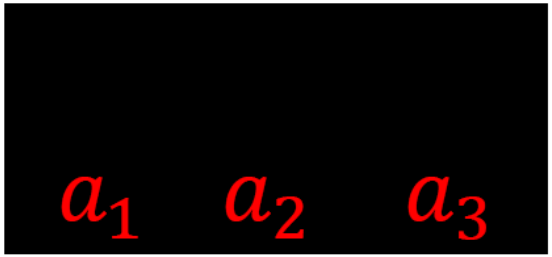
Example 1

ALGORITHM 2 The Linear Search Algorithm.

x	i	n
5		3

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* := i
else *location* := 0
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$ 1 2 3



Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	1	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)

$i := 1$

while ($i \leq n$ and $x \neq a_i$)

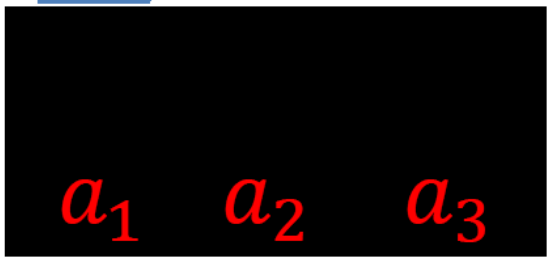
$i := i + 1$

if $i \leq n$ **then** *location* $:= i$

else *location* $:= 0$

return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$ 1 2 3



Searching Algorithms.

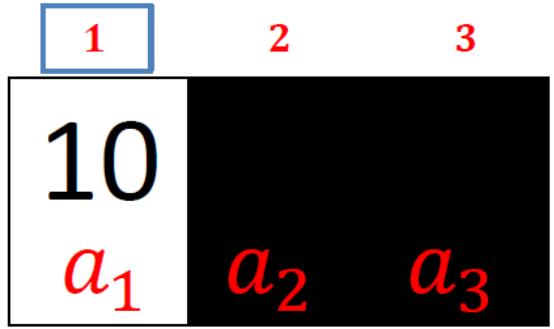
Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	1	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}



Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	1	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)

$i := 1$

while ($i \leq n$ and $x \neq a_i$) — *True*

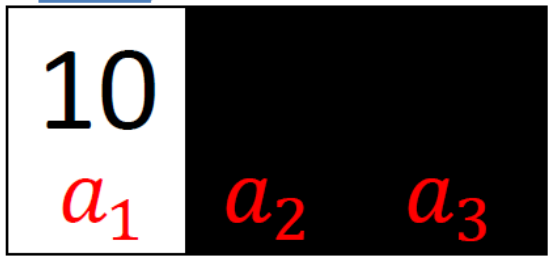
  $i := i + 1$

if $i \leq n$ **then** *location* $:= i$

else *location* $:= 0$

return *location* {*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$ 1 2 3



Searching Algorithms.

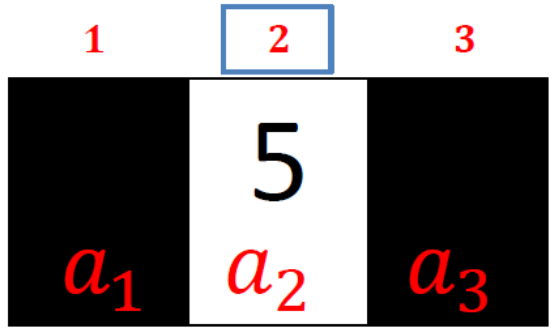
Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}



Searching Algorithms.

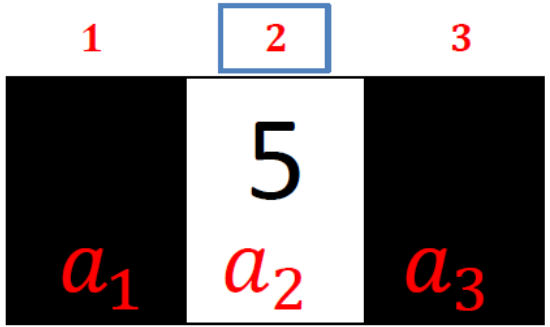
Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}



Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)

$i := 1$

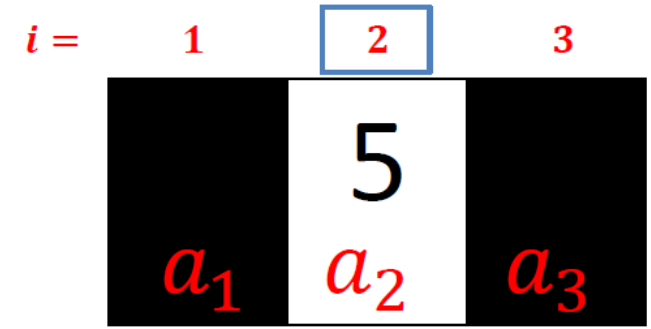
while ($i \leq n$ and $x \neq a_i$)

  $i := i + 1$

if $i \leq n$ **then** *location* := i

else *location* := 0

return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}



Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Example 1

ALGORITHM 2 The Linear Search Algorithm.

x	i	n
5	2	3

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)

$i := 1$

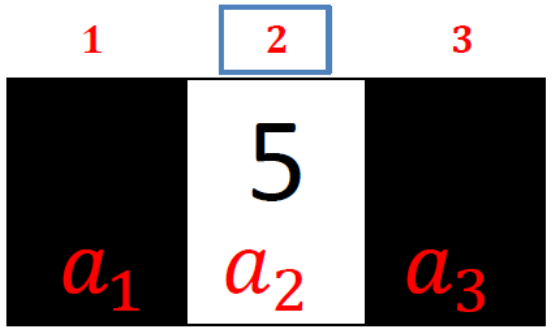
while ($i \leq n$ and $x \neq a_i$)

$i := i + 1$

if $i \leq n$ **then** *location* := i *True*

else *location* := 0

return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}



Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

Example 1

location

x	i	n
5	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)

$i := 1$

while ($i \leq n$ and $x \neq a_i$)

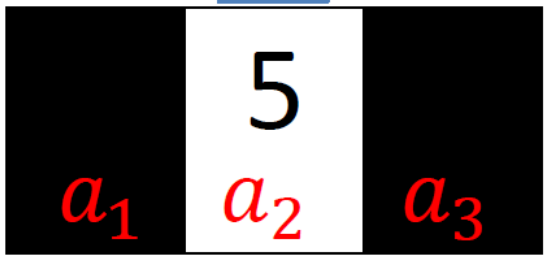
$i := i + 1$

True

if $i \leq n$ **then** *location* := i

~~**else** *location* := 0~~

$i =$ 1 2 3



return *location* {*location* is the subscript of the term that equals x, or is 0 if x is not found}

Searching Algorithms.

Linear Search Algorithm (or Sequential Search)

location

x	i	n
5	2	3

Example 1

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)

$i := 1$

while ($i \leq n$ and $x \neq a_i$)

$i := i + 1$

True

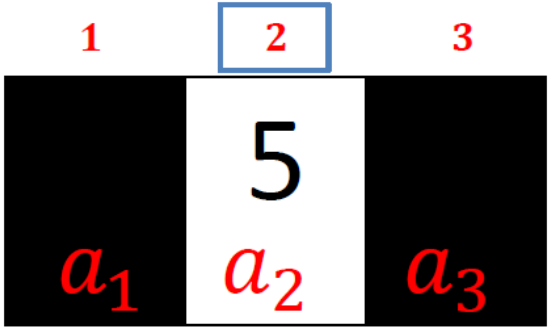
if $i \leq n$ **then** *location* $:= i$

~~**else** *location* $:= 0$~~

return *location* {*location* is the subscript of the term that equals x, or is 0 if x is not found}

2

$i =$ 1 2 3



Searching Algorithms.

Binary Search Algorithm

This algorithm can be used when the list has terms occurring in ***order of increasing size*** (for instance: if the terms are numbers, they are listed from smallest to largest; if they are words, they are listed in alphabetic, order).

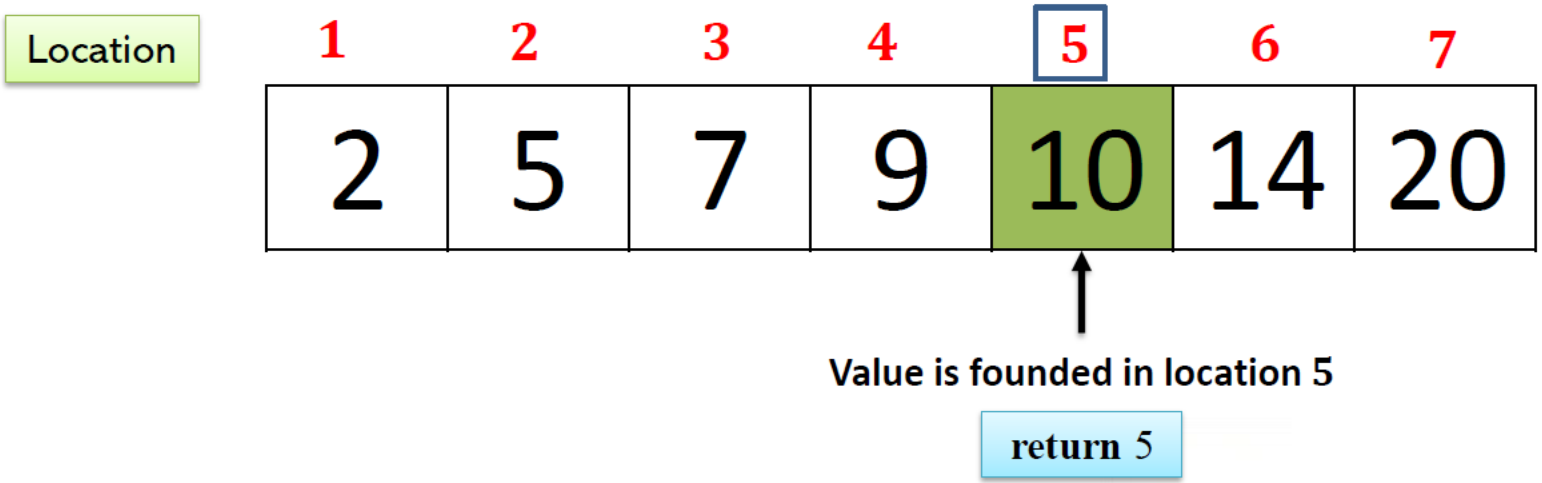
It proceeds by comparing the target element to be located to the middle term of the list.

The list is then split into two smaller sub lists of the same size until to get the target element.

Searching Algorithms.

Example 1

Locate an element **10** in this list



Searching Algorithms.

Example 2

Locate an element **25** in this list

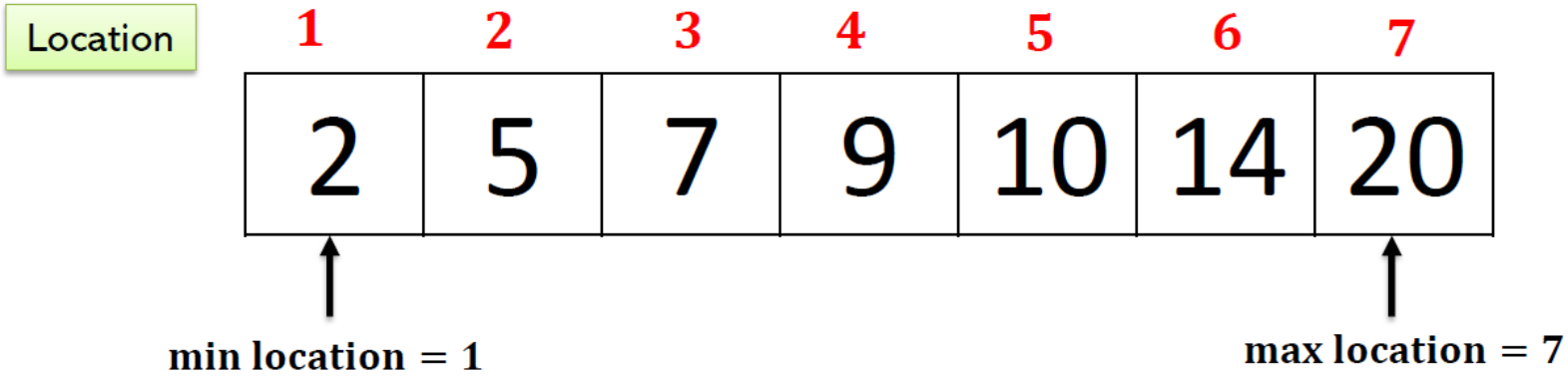
Location

value is not found

return 0

Binary Search Algorithm (1) – Example 1

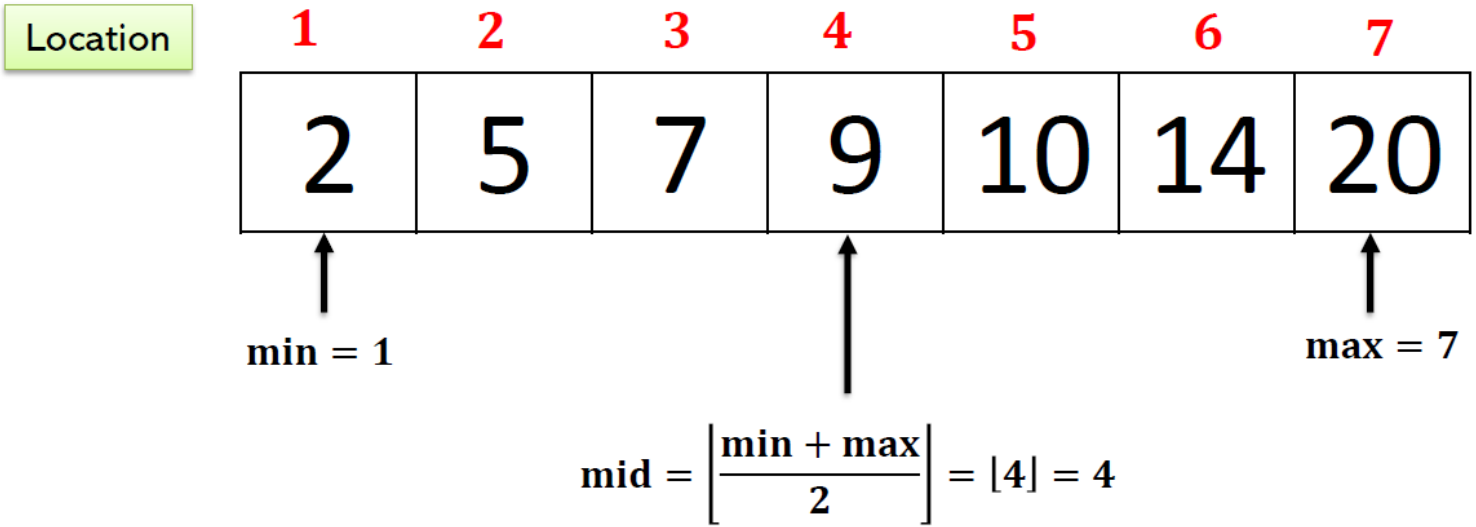
Locate an element **10** in this list



Searching Algorithms.

Binary Search Algorithm (1) – Example 1

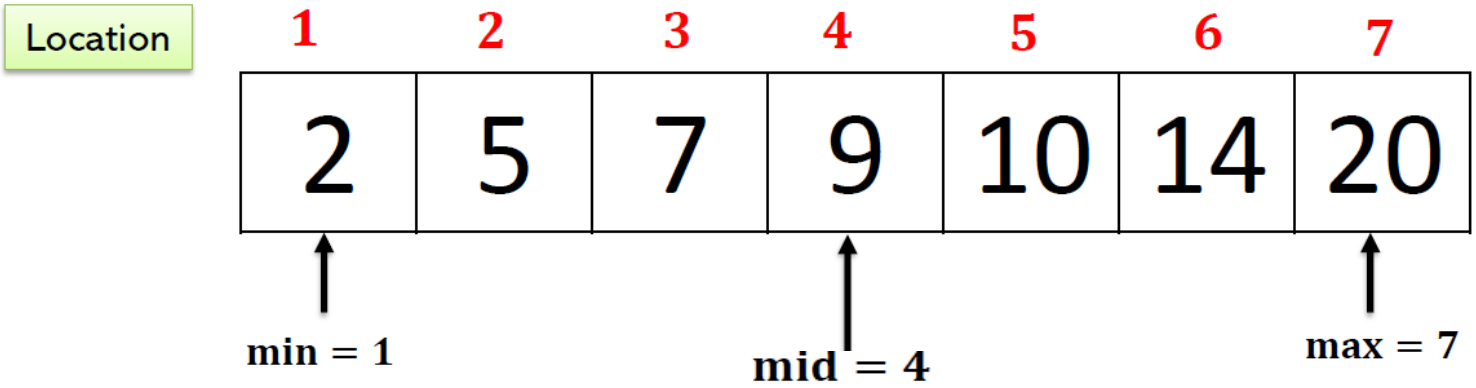
Locate an element **10** in this list



Searching Algorithms.

Binary Search Algorithm (1) – Example 1

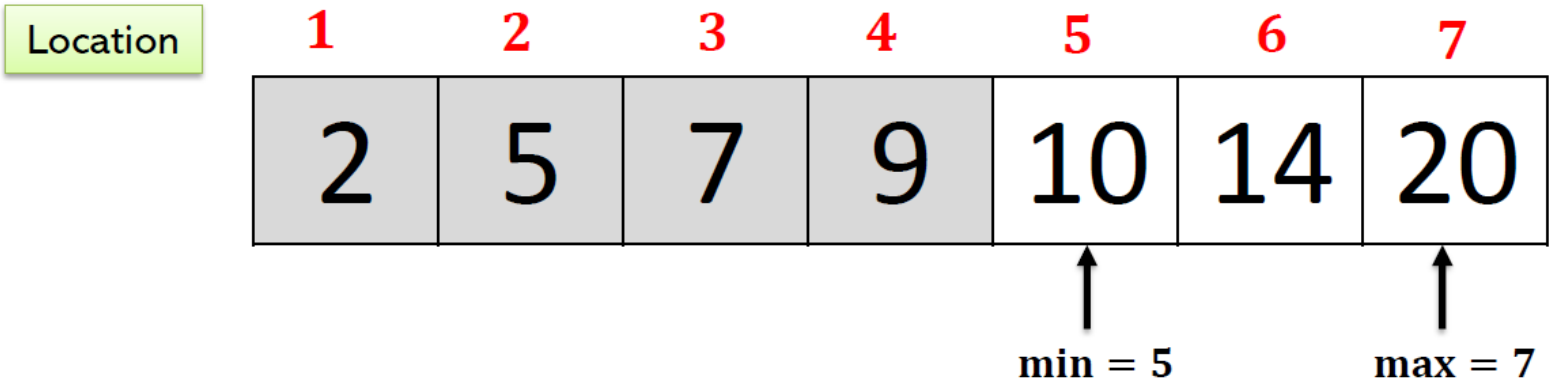
Locate an element **10** in this list



$$\text{Target } (10) > 9$$
$$\therefore \text{min} = \text{mid} + 1 = 5$$

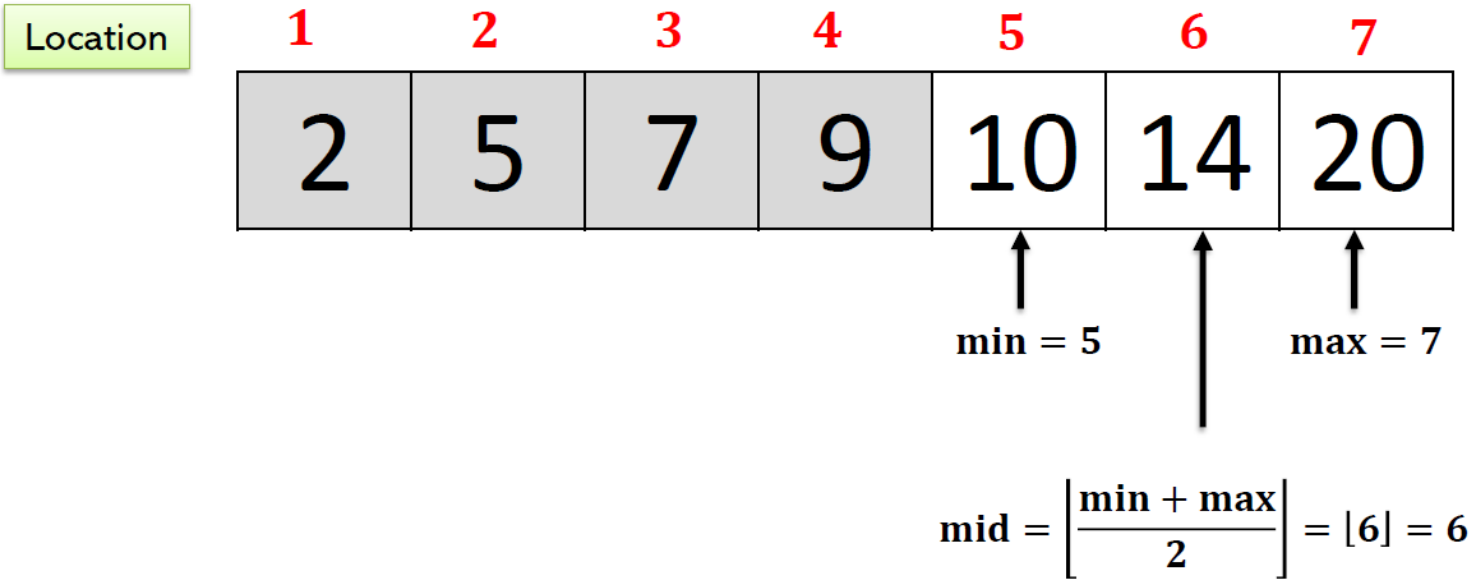
Binary Search Algorithm (1) – Example 1

Locate an element **10** in this list



Binary Search Algorithm (1) – Example 1

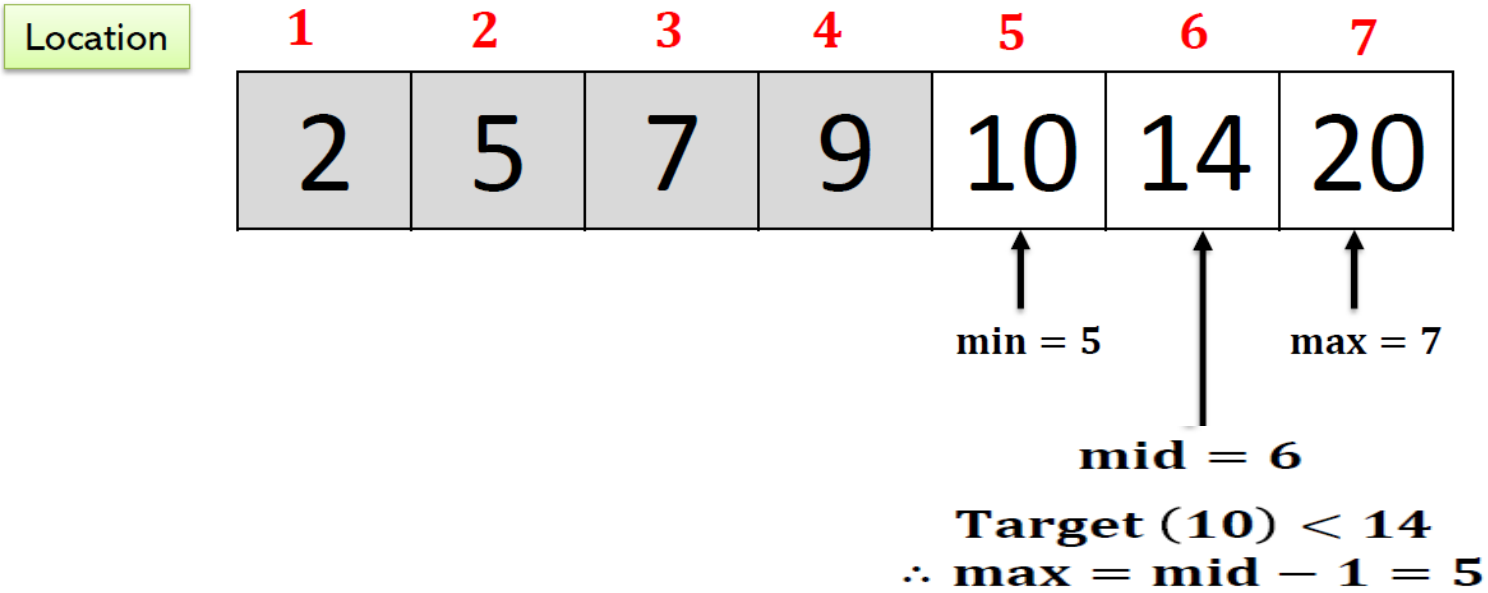
Locate an element **10** in this list



Searching Algorithms.

Binary Search Algorithm (1) – Example 1

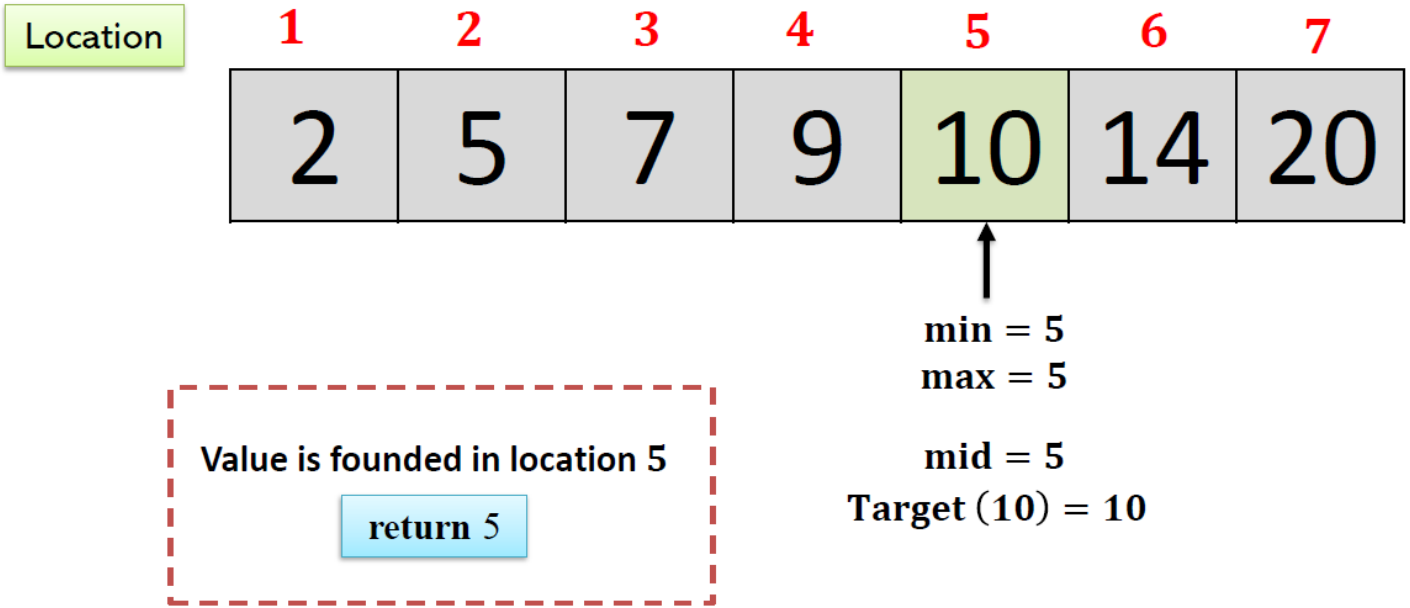
Locate an element **10** in this list



Searching Algorithms.

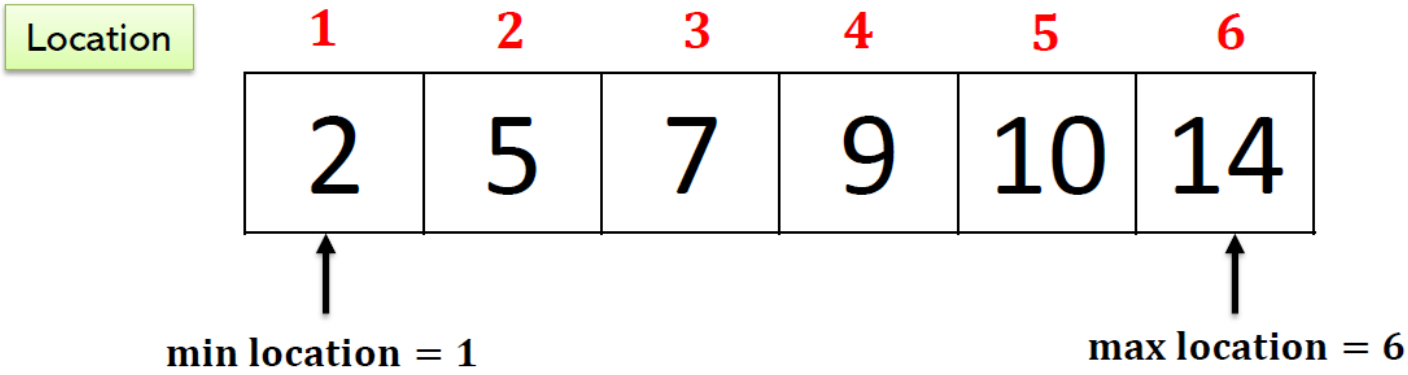
Binary Search Algorithm (1) – Example 1

Locate an element **10** in this list



Binary Search Algorithm (1) – Example 2

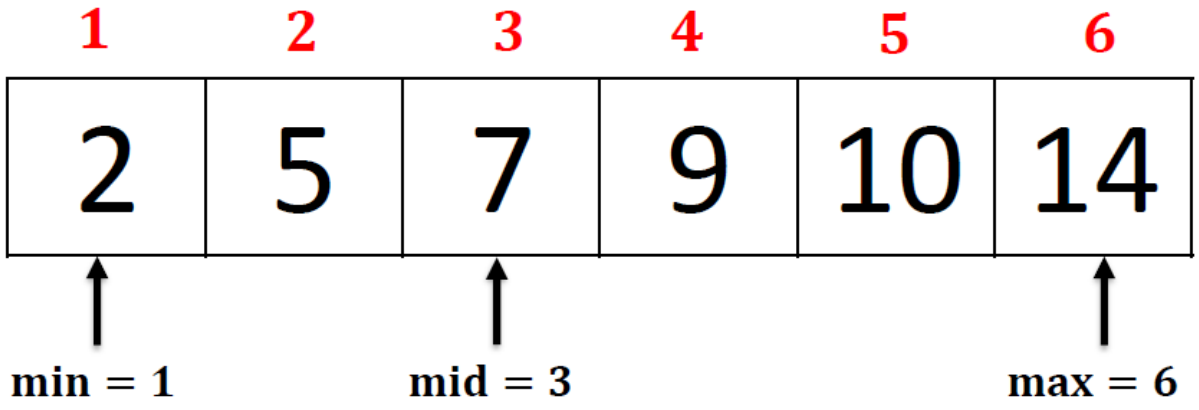
Locate an element **4** in this list



Binary Search Algorithm (1) – Example 2

Locate an element **4** in this list

Location



Target (4) < 7

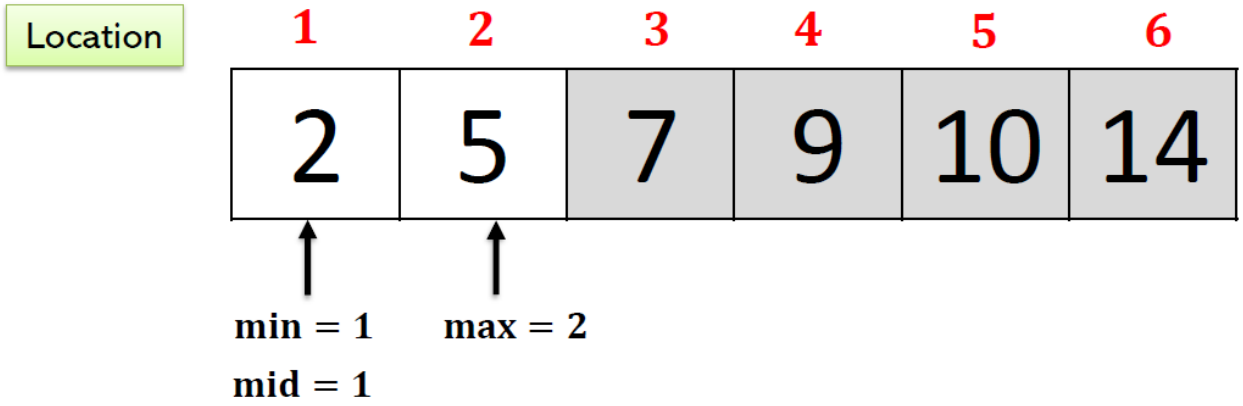
∴ max = mid − 1 = 2

Searching Algorithms.

Binary Search Algorithm (1) – Example 2

Continue

Locate an element **4** in this list



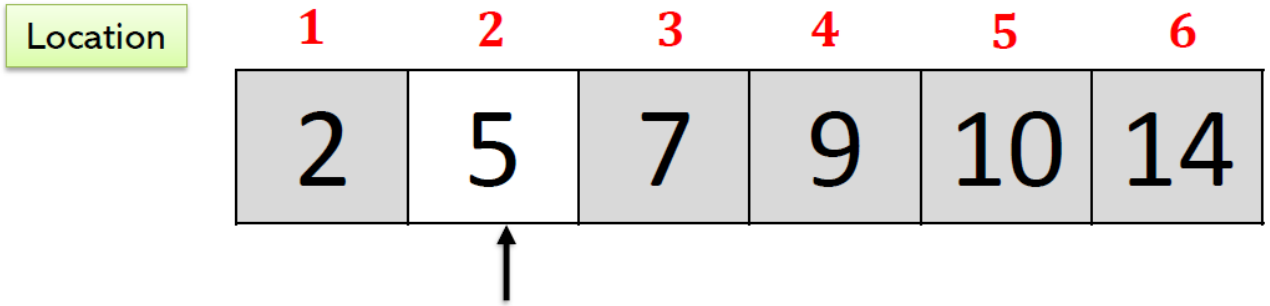
Target $(4) > 2$

$\therefore$ **min** $=$ **mid** $+ 1 = 2$

Searching Algorithms.

Binary Search Algorithm (1) – Example 2

Locate an element **4** in this list

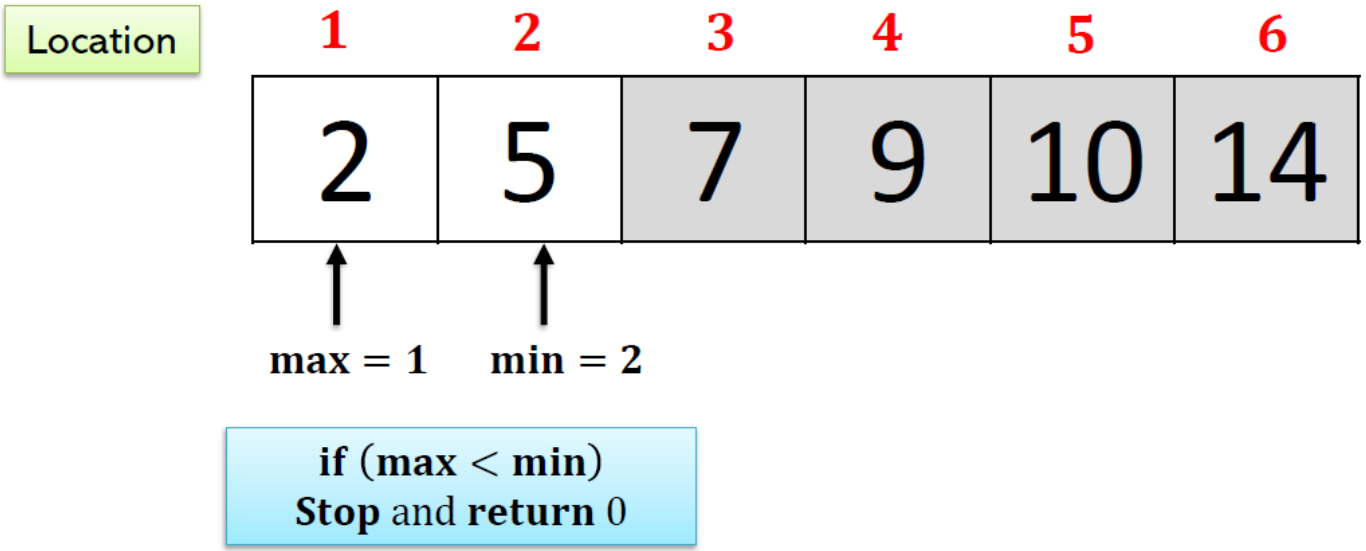


max = 2
min = 2
mid = 2

Target (4) < 5
∴ max = mid − 1 = 1

Binary Search Algorithm (1) – Example 2

Locate an element **4** in this list



if (**max** < **min**)
Stop and **return** 0

Searching Algorithms.

Binary Search Algorithm (1)

1. Let $min = 1$ and $max = n$.
2. If $max < min$, then **stop**: *target* is not present in array. **Return** 0.
3. Compute *guess* as the average of *max* and *min*, rounded down (so that it is an integer).
4. If array[*guess*] equals *target*, then stop. **Return** *guess*.
5. If the *guess* was too low, that is, array[*guess*] < *target*, then set $min = guess + 1$.
6. Otherwise, the *guess* was too high. Set $max = guess - 1$.
7. Go back to step 2.

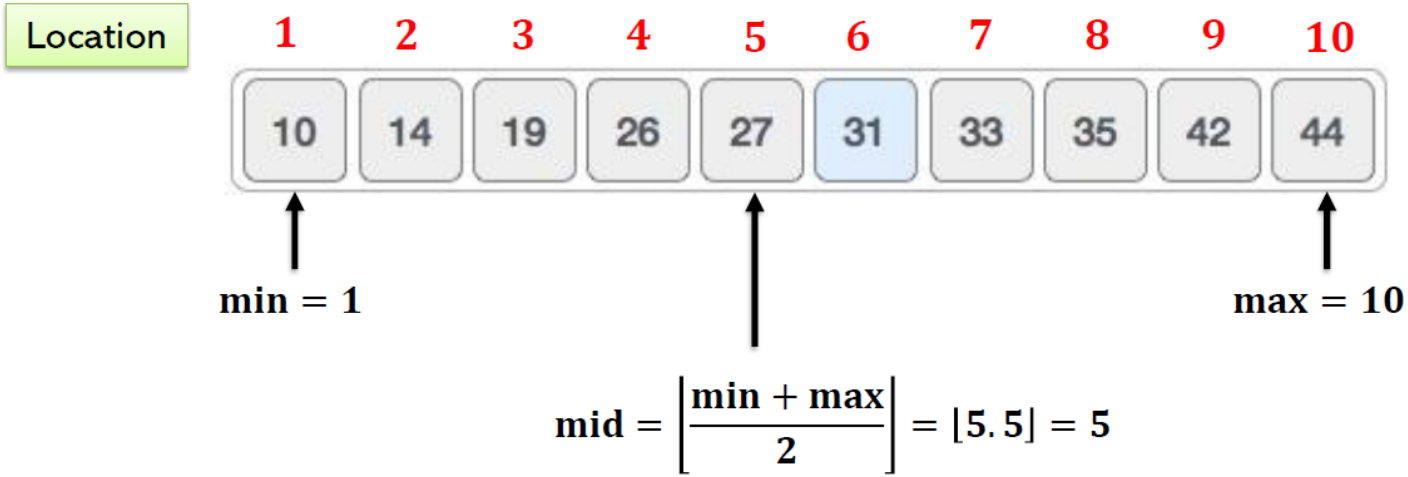
Searching Algorithms.

Binary Search Algorithm (1) – Example 3

Locate an element **31** in this list

Binary Search Algorithm (1) – Example 3

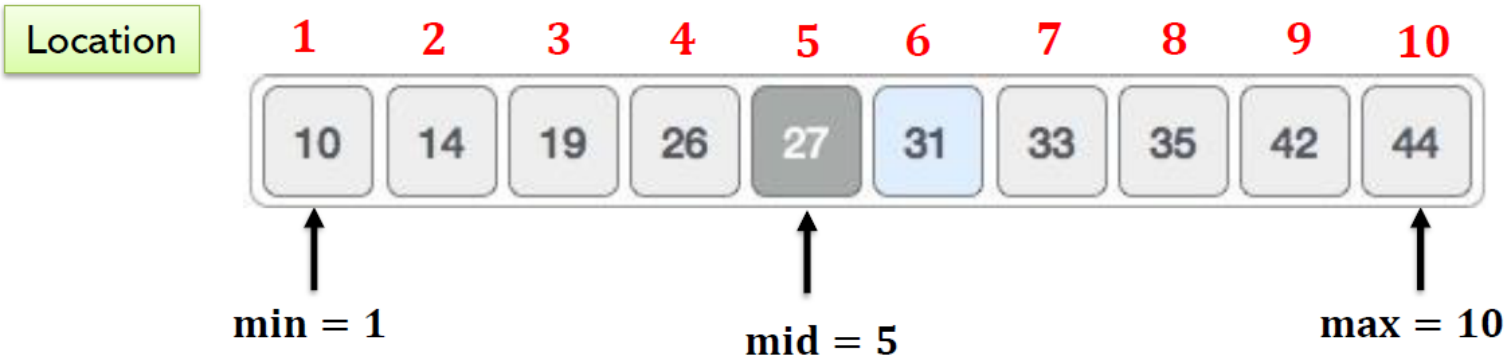
Locate an element **31** in this list



Searching Algorithms.

Binary Search Algorithm (1) – Example 3

Locate an element **31** in this list



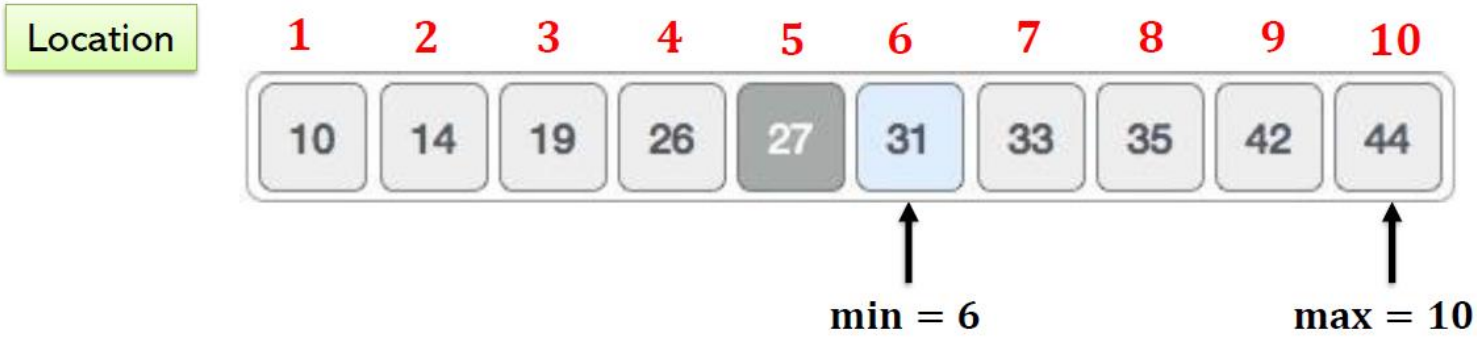
Target (31) > 27

∴ min = mid + 1 = 6

Searching Algorithms.

Binary Search Algorithm (1) – Example 3

Locate an element **31** in this list

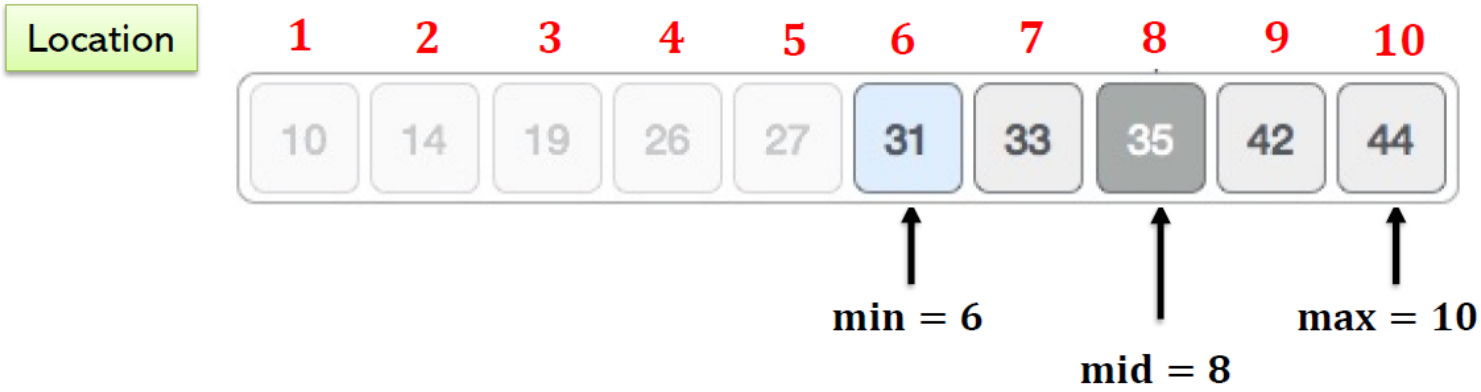


$$mid = \left\lfloor \frac{min + max}{2} \right\rfloor = \lfloor 8 \rfloor = 8$$

Searching Algorithms.

Binary Search Algorithm (1) – Example 3

Locate an element **31** in this list

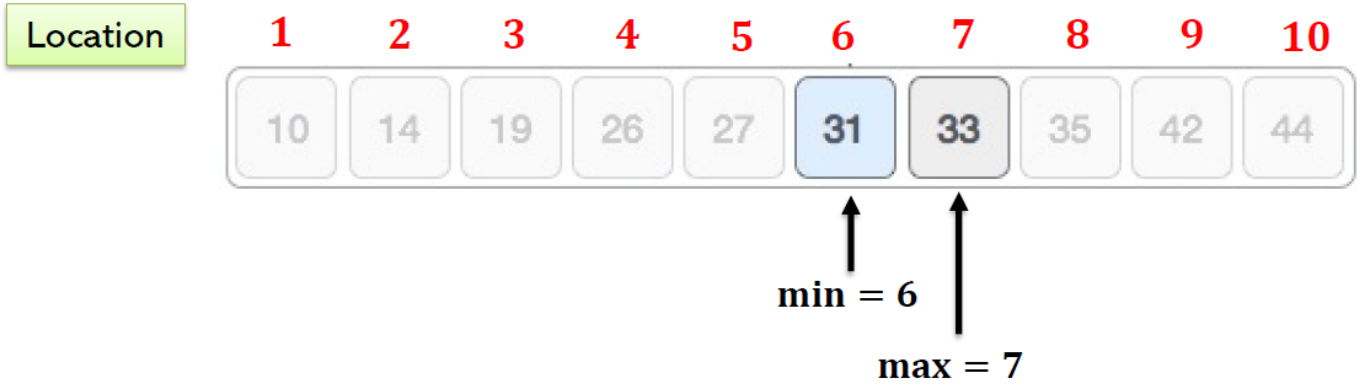


Target (31) < 35

∴ max = mid − 1 = 7

Binary Search Algorithm (1) – Example 3

Locate an element **31** in this list

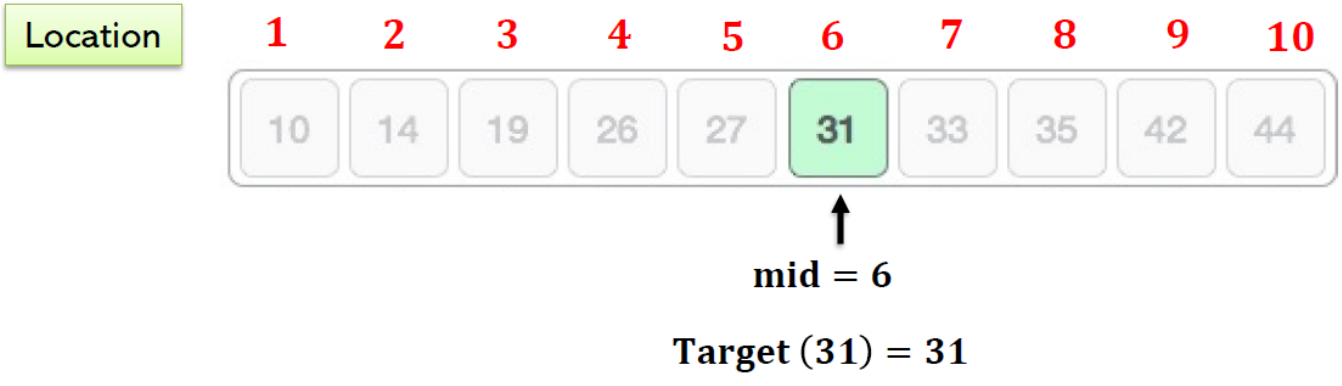


$$\text{mid} = \left\lfloor \frac{\text{min} + \text{max}}{2} \right\rfloor = \lfloor 6.5 \rfloor = 6$$

Searching Algorithms.

Binary Search Algorithm (1) – Example 3

Locate an element **31** in this list



Value is founded in location 6

return 6

Searching Algorithms.

Binary Search Algorithm (2)

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19			

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	1		

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	1	16	

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	1	16	8

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	1	16	8

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	**10**	12	13	15	16	18	19	20	22

Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	9	16	8

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	**10**	12	13	15	16	18	19	20	22

Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	9	16	8

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	9	16	12

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

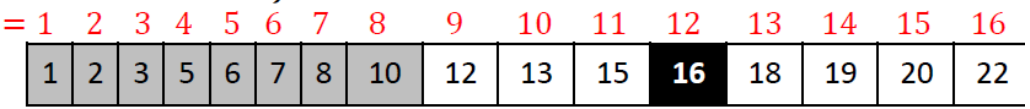
Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	9	16	12

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}



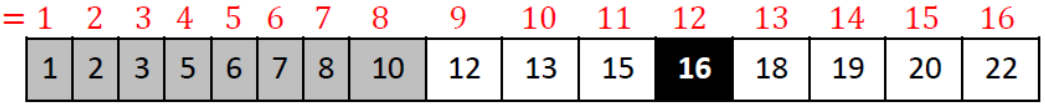
Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	13	16	12

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}



Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	13	16	12

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	14	14	13

Continue

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	10	12	13	15	16	18	**19**	20	22

Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	14	14	13

location

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* := i
else *location* := 0
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	10	12	13	15	16	18	**19**	20	22

Searching Algorithms.

Binary Search Algorithm (2)

n	x	i	j	m
16	19	14	14	13

location

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$ {i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location* {*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

index =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	5	6	7	8	10	12	13	15	16	18	**19**	20	22

14

Thank you !