

Section 8.14
Divide-and-Conquer Algorithms:
Binary Search

Binary Search

- Searching for a value in a sorted sequence
- Works like looking for a word in the dictionary


Binary Search


- Example: Find 6 in the ordered list of 16 integers

$$\frac{x}{6}$$

a

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

 left
1

 right
16

Binary Search

- Example: Find 6 in the ordered list of 16 integers

$$\frac{x}{6}$$

a

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

↑
left
1

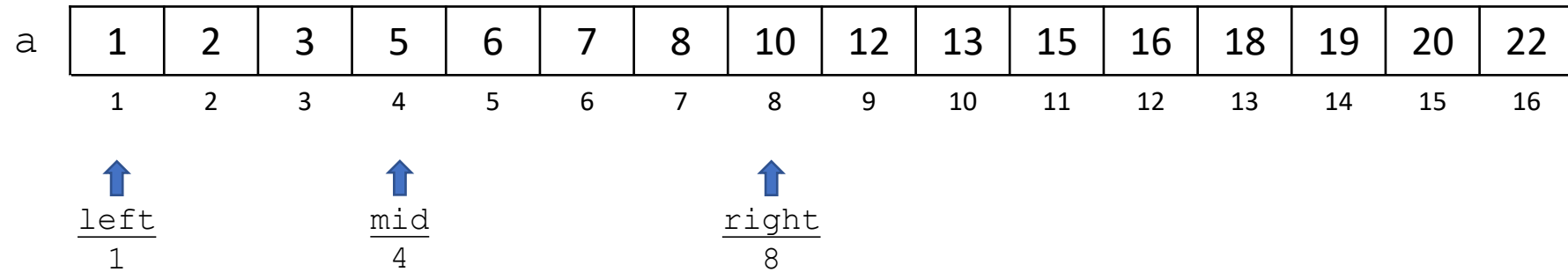
↑
right
8

↑
mid
8

Binary Search

- Example: Find 6 in the ordered list of 16 integers

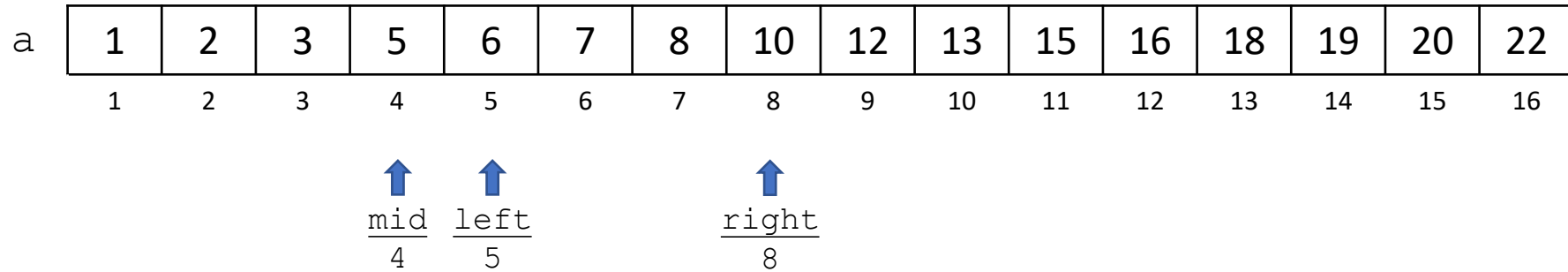
$$\frac{x}{6}$$



Binary Search

- Example: Find 6 in the ordered list of 16 integers

$$\frac{x}{6}$$



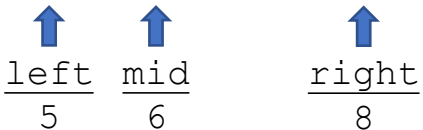
Binary Search

- Example: Find 6 in the ordered list of 16 integers

$$\frac{x}{6}$$

a

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


left mid right
5 6 8

Binary Search

- Example: Find 6 in the ordered list of 16 integers

$\frac{x}{6}$

a

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

↑ ↑
leftright
5 6
↑
mid
6

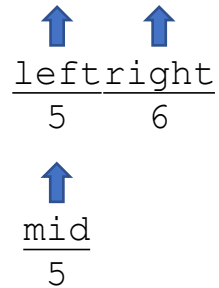
Binary Search

- Example: Find 6 in the ordered list of 16 integers

$$\frac{x}{6}$$

a

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



Binary Search

- Example: Find 6 in the ordered list of 16 integers

$$\frac{x}{6}$$

a

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

↑
left
5

↑
right
5

↑
mid
5

Binary Search

- Example: Find 6 in the ordered list of 16 integers

$$\frac{x}{6}$$

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

↑
 $\frac{\text{left}}{5}$

↑
 $\frac{\text{right}}{5}$

↑
 $\frac{\text{mid}}{5}$

Since $a_{\text{left}} = x$, the location of x is `left`

Binary Search

- Another example: Find 14 in the ordered list of 16 integers

x

14

a

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

↑
left
1

↑
right
16

Binary Search

- Another example: Find 14 in the ordered list of 16 integers

x

14

a

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

↑ ↑
mid left
8 9

↑
right
16

Binary Search

- Another example: Find 14 in the ordered list of 16 integers

x

14

a

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

↑
left
9

↑
mid
12

↑
right
16

Binary Search

- Another example: Find 14 in the ordered list of 16 integers

x

14

a

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

↑
left
9

↑
right
12
↑
mid
12

Binary Search

- Another example: Find 14 in the ordered list of 16 integers

x

14

a

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

↑ ↑ ↑
left mid right
9 10 12

Binary Search

- Another example: Find 14 in the ordered list of 16 integers

x

14

a

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

↑ ↑ ↑
mid left right
10 11 12

Binary Search

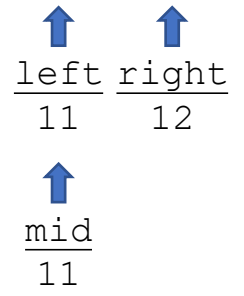
- Another example: Find 14 in the ordered list of 16 integers

x

14

a

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



Binary Search

- Another example: Find 14 in the ordered list of 16 integers

x

14

a

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

↑
left
11

↑
right
11

↑
mid
12

Binary Search

- Another example: Find 14 in the ordered list of 16 integers

$$\frac{x}{14}$$

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

Since $a_{\text{left}} \neq x$, x is not in the list

↑
left
11

↑
right
11

↑
mid
12

Binary Search

- The search repeats the following steps
 1. $mid := \lfloor (left+right) / 2 \rfloor$
 2. If $x \leq a_{mid}$ then $right := a_{mid}$
 3. Otherwise, $left := mid+1$
- The search ends when $left = right$
- At which point, if $x = a_{left}$ then x was found at index $left$

Binary Search

BinarySearch(A, x, left, right)

Input: a sorted sequence of numbers: $A=a_1\dots a_n$,

a number to search for: x

the boundaries within which to search: left and right

Output: The index of x in A if it is in A, otherwise -1

```
if (left = right)
```

```
  if (x = aleft)
```

```
    return left
```

```
  else
```

```
    return -1
```

```
  end-if
```

```
end-if
```

```
mid := [(left+right)/2]
```

```
if (x ≤ amid)
```

```
  right := mid
```

```
else
```

```
  left := mid+1
```

```
end-if
```

```
return binarysearch(A, x, left, right)
```


Binary Search Analysis

- If the size of the sequence of numbers, A , is a power of two, i.e. 2^n , then how many recursive calls to `binarysearch` are made?
- Each recursive call to `BinarySearch` reduces the portion of sequence A that is to be searched by half.
- Recursive calls are made until `left=right`, i.e., the size of the region of A to be searched is 1
- How many times can 2^n be divided by 2 until the result is 1?
 - n times
 - $\log_2(2^n) = n$

Binary Search Analysis

- Another way to analyze the BinarySearch algorithm is to describe the number of steps required to compute binarysearch when the size of the region of A to be searched is n
- $T(1) = 3$
- $T(n) = 9 + T(n/2)$

```
if (left = right)
  if (x = aleft)
    return left
  else
    return -1
  end-if
end-if

mid := [(left+right)/2]
if (x ≤ amid)
  right := mid
else
  left := mid+1
end-if
return binarysearch(A, x, left, right)
```