Data Structures
COMP242

Ala' Hasheesh
ahashesh@birzeit.edu

Algorithm Analysis & Lists

BIRZEIT UNIVERSITY

Insertion Sort

Given a sorted array insert a new element into it!

Insertion Sort

Given a sorted array insert a new element into it!

Assume we want to insert 9

Insertion Sort

Given a sorted array insert a new element into it!

Assume we want to insert 9

Insertion Sort

Given a sorted array insert a new element into it!

Assume we want to insert 9

9 < 37 ? Yes => Move 37 to the right ($a[i] = a[i-1]$)

$i = 5$

Insertion Sort

Given a sorted array insert a new element into it!

5	10	13	22	37

Assume we want to insert 9

9 < 22 ? Yes => Move 22 to the right ($a[i] = a[i-1]$)

5	10	13	22	37	9
5	10	13	22		37

$i = 4$

Insertion Sort

Given a sorted array insert a new element into it!

5	10	13	22	37

Assume we want to insert 9

9 < 13 ? Yes => Move 13 to the right ($a[i] = a[i-1]$)

5	10	13	22	37	9
5	10	13	22		37
5	10	13		22	37

$i = 3$

Insertion Sort

Given a sorted array insert a new element into it!

5	10	13	22	37

Assume we want to insert 9

9 < 10 ? Yes => Move 10 to the right ($a[i] = a[i-1]$)

5	10	13	22	37	9
5	10	13	22		37
5	10	13		22	37
5	10		13	22	37

$i = 2$

Insertion Sort

Given a sorted array insert a new element into it!

5	10	13	22	37

Assume we want to insert 9

9 < 5 ? No => Stop

5	10	13	22	37	9
5	10	13	22		37
5	10	13		22	37
5	10		13	22	37
5		10	13	22	37

$i = 1$

Insertion Sort

Given a sorted array insert a new element into it!

5	10	13	22	37

Assume we want to insert 9

$$a[i] = 9$$

5	10	13	22	37	9
5	10	13	22		37
5	10	13		22	37
5	10		13	22	37
5	**9**	10	13	22	37

$$i = 1$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

7	5	9	3	8
7				

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$value = 5$

$j = 1$

7	5	9	3	8
7				
7				

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$value = 5$

$7 > 5 \Rightarrow a[1] = 7$

$j = 1$

7	5	9	3	8
7				
7	7			

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

$value = 5$

$j = 0$

7	5	9	3	8
7				
7	7			

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

$value = 5$

$a[0] = 5$

$j = 0$

7	5	9	3	8
7				
5	7			

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$$value = 9$$

$$j = 2$$

7	5	9	3	8
7				
5	7			
5	7			

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$7 > 9$

$value = 9$

$j = 2$

7	5	9	3	8
7				
5	7			
5	7			

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$$value = 9$$

$$j = 2$$

7	5	9	3	8
7				
5	7			
5	7	9		

$$a[2] = 9$$

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$$value = 3$$

$$9 > 3 \Rightarrow a[3] = 9$$

$$j = 3$$

7	5	9	3	8
7				
5	7			
5	7	9		
5	7	9	9	

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$$value = 3$$

$$j = 2$$

7	5	9	3	8
7				
5	7			
5	7	9		
5	7	9	9	

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$$value = 3$$

$$7 > 3 \Rightarrow a[2] = 7$$

$$j = 2$$

7	5	9	3	8
7				
5	7			
5	7	9		
5	7	7	9	

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$value = 3$

$j = 1$

7	5	9	3	8
7				
5	7			
5	7	9		
5	7	7	9	

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$value = 3$

$5 > 3 \Rightarrow a[1] = 5$

$j = 1$

7	5	9	3	8
7				
5	7			
5	7	9		
5	5	7	9	

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$value = 3$

$j = 0$

7	5	9	3	8
7				
5	7			
5	7	9		
5	5	7	9	

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$value = 3$

$j = 0$

7	5	9	3	8
7				
5	7			
5	7	9		
3	5	7	9	

$a[0] = 3$

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$value = 8$

$j = 4$

7	5	9	3	8
7				
5	7			
5	7	9		
3	5	7	9	
3	5	7	9	

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$9 > 8 \Rightarrow a[4] = 9$

$value = 8$

$j = 4$

7	5	9	3	8
7				
5	7			
5	7	9		
3	5	7	9	
3	5	7	9	9

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$value = 8$

$j = 3$

7	5	9	3	8
7				
5	7			
5	7	9		
3	5	7	9	
3	5	7	9	9

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$7 > 8$

$value = 8$

$j = 3$

7	5	9	3	8
7				
5	7			
5	7	9		
3	5	7	9	
3	5	7	9	9

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

$$value = 8$$

$$j = 3$$

7	5	9	3	8
7				
5	7			
5	7	9		
3	5	7	9	
3	5	7	8	9

$$a[3] = 8$$

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

Given the following array what is the time?

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

Given the following array what is the time?

$$T(n) = c_1 + c_2 + c_3 + \cdots + c_n$$

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

Given the following array what is the time?

$$T(n) = c_1 + c_2 + c_3 + \cdots + c_n$$

$$T(n) = c + c + c + \cdots + c$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Given the following array what is the time?

3	7	11	21	23

$$T(n) = c_1 + c_2 + c_3 + \cdots + c_n$$

$$T(n) = c + c + c + \cdots + c$$

$$T(n) = c(n)$$

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

Given the following array what is the time?

3	7	11	21	23

$$T(n) = c_1 + c_2 + c_3 + \cdots + c_n$$

$$T(n) = c + c + c + \cdots + c$$

$$T(n) = c(n)$$

$$\mathbf{O(n)}$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Best case

Array already sorted!

Given the following array what is the time?

3	7	11	21	23

$$T(n) = c_1 + c_2 + c_3 + \cdots + c_n$$

$$T(n) = c + c + c + \cdots + c$$

$$T(n) = c(n)$$

$$\mathbf{O(n)}$$

Insertion Sort

```
void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        int j = i;
        while (j > 0 && arr[j - 1] > value) {
            arr[j] = arr[j - 1];
            j--;
        }
        arr[j] = value;
    }
}
```

Given the following array what is the time?

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Given the following array what is the time?

23	21	11	7	3

$$T(n) = c_1 + 2c_2 + 3c_3 + \cdots + (n-1)c_{n-1}$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Given the following array what is the time?

23	21	11	7	3

$$T(n) = c_1 + 2c_2 + 3c_3 + \cdots + (n-1)c_{n-1}$$

$$T(n) = c + 2c + 3c + \cdots + (n-1)c$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Given the following array what is the time?

23	21	11	7	3

$$T(n) = c_1 + 2c_2 + 3c_3 + \cdots + (n-1)c_{n-1}$$

$$T(n) = c + 2c + 3c + \cdots + (n-1)c$$

$$T(n) = c(1 + 2 + 3 + \cdots + n - 1)$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Rule

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$$1 + 2 + 3 + \cdots + n - 1 = \frac{n(n-1)}{2}$$

Given the following array what is the time?

23	21	11	7	3

$$T(n) = c_1 + 2c_2 + 3c_3 + \cdots + (n-1)c_{n-1}$$

$$T(n) = c + 2c + 3c + \cdots + (n-1)c$$

$$T(n) = c(1 + 2 + 3 + \cdots + n - 1)$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Rule

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$$1 + 2 + 3 + \cdots + n - 1 = \frac{n(n-1)}{2}$$

Given the following array what is the time?

23	21	11	7	3

$$T(n) = c_1 + 2c_2 + 3c_3 + \cdots + (n-1)c_{n-1}$$

$$T(n) = c + 2c + 3c + \cdots + (n-1)c$$

$$T(n) = c(1 + 2 + 3 + \cdots + n - 1)$$

$$T(n) = \frac{cn(n-1)}{2}$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Rule

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$$1 + 2 + 3 + \cdots + n - 1 = \frac{n(n-1)}{2}$$

Given the following array what is the time?

23	21	11	7	3

$$T(n) = c_1 + 2c_2 + 3c_3 + \cdots + (n-1)c_{n-1}$$

$$T(n) = c + 2c + 3c + \cdots + (n-1)c$$

$$T(n) = c(1 + 2 + 3 + \cdots + n - 1)$$

$$T(n) = \frac{cn(n-1)}{2}$$

$$T(n) = \frac{cn^2 - cn}{2}$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Rule

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$$1 + 2 + 3 + \cdots + n - 1 = \frac{n(n-1)}{2}$$

Given the following array what is the time?

23	21	11	7	3

$$T(n) = c_1 + 2c_2 + 3c_3 + \cdots + (n-1)c_{n-1}$$

$$T(n) = c + 2c + 3c + \cdots + (n-1)c$$

$$T(n) = c(1 + 2 + 3 + \cdots + n - 1)$$

$$T(n) = \frac{cn(n-1)}{2}$$

$$T(n) = \frac{cn^2 - cn}{2}$$

$$\mathbf{O(n^2)}$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Worst case

array is sorted in reverse order!

Given the following array what is the time?

23	21	11	7	3

$$T(n) = c_1 + 2c_2 + 3c_3 + \cdots + (n-1)c_{n-1}$$

$$T(n) = c + 2c + 3c + \cdots + (n-1)c$$

$$T(n) = c(1 + 2 + 3 + \cdots + n - 1)$$

$$T(n) = \frac{cn(n-1)}{2}$$

$$T(n) = \frac{cn^2 - cn}{2}$$

$$\mathbf{O(n^2)}$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Average case

On average each element will travel the average of the array

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Average case

On average each element will travel the average of the array

$$T(n) \approx \frac{c_1}{2} + \frac{2c_2}{2} + \frac{3c_3}{2} + \cdots + \frac{(n-1)c_{n-1}}{2}$$

$$T(n) \approx \frac{c}{2} + \frac{2c}{2} + \frac{3c}{2} + \cdots + \frac{(n-1)c}{2}$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Average case

On average each element will travel the average of the array

$$T(n) \approx \frac{c_1}{2} + \frac{2c_2}{2} + \frac{3c_3}{2} + \cdots + \frac{(n-1)c_{n-1}}{2}$$

$$T(n) \approx \frac{c}{2} + \frac{2c}{2} + \frac{3c}{2} + \cdots + \frac{(n-1)c}{2}$$

$$T(n) \approx \frac{1}{2}(1 + 2 + 3 + \cdots + n - 1)$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Average case

On average each element will travel the average of the array

$$T(n) \approx \frac{c_1}{2} + \frac{2c_2}{2} + \frac{3c_3}{2} + \cdots + \frac{(n-1)c_{n-1}}{2}$$

$$T(n) \approx \frac{c}{2} + \frac{2c}{2} + \frac{3c}{2} + \cdots + \frac{(n-1)c}{2}$$

$$T(n) \approx \frac{1}{2}(1 + 2 + 3 + \cdots + n - 1)$$

$$T(n) \approx \frac{1}{2}(\frac{n(n-1)}{2})$$

Insertion Sort

```
void insertionSort(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
      int value = arr[i];
      int j = i;
      while (j > 0 && arr[j - 1] > value) {
         arr[j] = arr[j - 1];
         j--;
      }
      arr[j] = value;
   }
}
```

Average case

On average each element will travel the average of the array

$$T(n) \approx \frac{c_1}{2} + \frac{2c_2}{2} + \frac{3c_3}{2} + \cdots + \frac{(n-1)c_{n-1}}{2}$$

$$T(n) \approx \frac{c}{2} + \frac{2c}{2} + \frac{3c}{2} + \cdots + \frac{(n-1)c}{2}$$

$$T(n) \approx \frac{1}{2}(1 + 2 + 3 + \cdots + n - 1)$$

$$T(n) \approx \frac{1}{2}(\frac{n(n-1)}{2})$$

$$T(n) \approx \frac{n^2 - n}{4}$$

$$\mathbf{O}(n^2)$$

Lists

- A list is a finite, ordered sequence of data items known as elements ("ordered" means that each element has a position in the list)

We say that a_{i+1} follows a_i

Lists

- A List as an Abstract Data Type (ADT)

list is a sequence of items that supports at least the following functionality:

- accessing an item at an arbitrary position in the sequence
- adding an item at an arbitrary position
- removing an item at an arbitrary position
- determining the number of items in the list (the list's length)

Lists

- Abstract Data Type (ADT)

specifies what a list will do, without specifying the implementation.

Operation

- Print List
- Find element
- Add element
- Insert element
- Delete element
- Make null

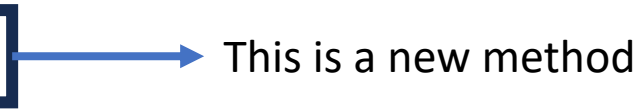
Lists

```
public interface Listable<T> {
   void add(T element);

   void insert(int index, T element);

   T delete(int index);

   int find(T element);

   void clear();

   void print();
}
```

We did this in lab0

Lists

```
public interface Listable<T> {
   void add(T element);

   void insert(int index, T element);

   T delete(int index);

   int find(T element);

   void clear();

   void print();
}
```



Lists

```
public interface Listable<T> {
   void add(T element);

   void insert(int index, T element);

   T delete(int index);

   int find(T element);

   void clear();

   void print();
}
```

$$insert(2, 100)$$

Lists

```
public interface Listable<T> {
   void add(T element);

   void insert(int index, T element);

   T delete(int index);

   int find(T element);

   void clear();

   void print();
}
```

$$insert(2, 100)$$

$$Shift\ elements\ to\ the\ right$$

Lists

```
public interface Listable<T> {
   void add(T element);

   void insert(int index, T element);

   T delete(int index);

   int find(T element);

   void clear();

   void print();
}
```

$$insert(2, 100)$$

5	10	20	20	13	1
0	1	2	3	4	5

$$Shift\ elements\ to\ the\ right$$

Lists

```
public interface Listable<T> {
   void add(T element);

   void insert(int index, T element);

   T delete(int index);

   int find(T element);

   void clear();

   void print();
}
```

$$insert(2, 100)$$

5	10	100	20	13	1
0	1	2	3	4	5

$$a[2] = 100$$

Lists

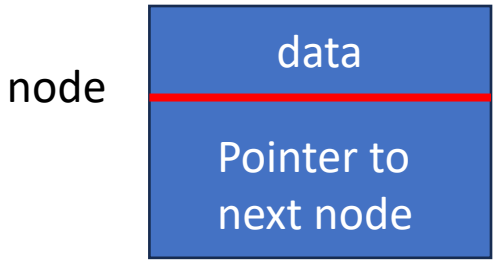
```
public interface Listable<T> {
    void add(T element);

    void insert(int index, T element);

    T delete(int index);

    int find(T element);

    void clear();

    void print();
}
```

```
public void insert(int index, T element) {
    if (count >= data.length) {
        reSize();
    }

    if (index >= count) {
        index = count - 1;
    }

    if (index < 0) {
        index = 0;
    }

    for (int i = this.count; i > index; i--) { // shift
        data[i] = data[i - 1];
    }

    data[index] = element;
    this.count++;
}
```

LinkedList

A linked list is a collection of nodes that together form a linear.

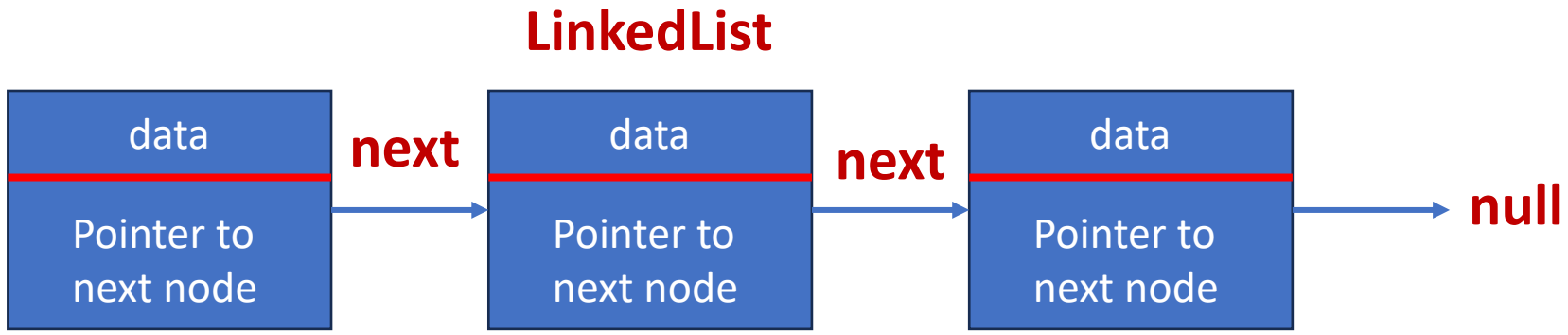
node: A compound object that stores the contents of the node (Element) and a pointer or reference to the next node in the list (next).



LinkedList

A linked list is a collection of nodes that together form a linear.

node: A compound object that stores the contents of the node (Element) and a pointer or reference to the next node in the list (next).



LinkedList

A linked list is a collection of nodes that together form a linear.

node: A compound object that stores the contents of the node (Element) and a pointer or reference to the next node in the list (next).

LinkedList

```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```

LinkedList

In a LinkedList we only need a reference to the first element (head/front)

```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```

LinkedList

In a LinkedList we only need a reference to the first element (head/front)

```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```

How do we reach third element?

LinkedList

In a LinkedList we only need a reference to the first element (head/front)

```
public class Node<T> {
   public T val;
   public Node<T> next;

   public Node(T val) {
      this(val, null);
   }

   public Node(T val, Node<T> next) {
      this.val = val;
      this.next = next;
   }
}
```


```
Node<Integer> n1 = new Node<>(5);
Node<Integer> n2 = new Node<>(10);
Node<Integer> n3 = new Node<>(1);

n1.next = n2;
n2.next = n3;

Node<Integer> head = n1;
Node<Integer> thirdElement = head.next.next;
```

LinkedList

In a LinkedList we only need a reference to the first element (head/front)

```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```

How do we reach 100th element?

LinkedList

In a LinkedList we only need a reference to the first element (head/front)

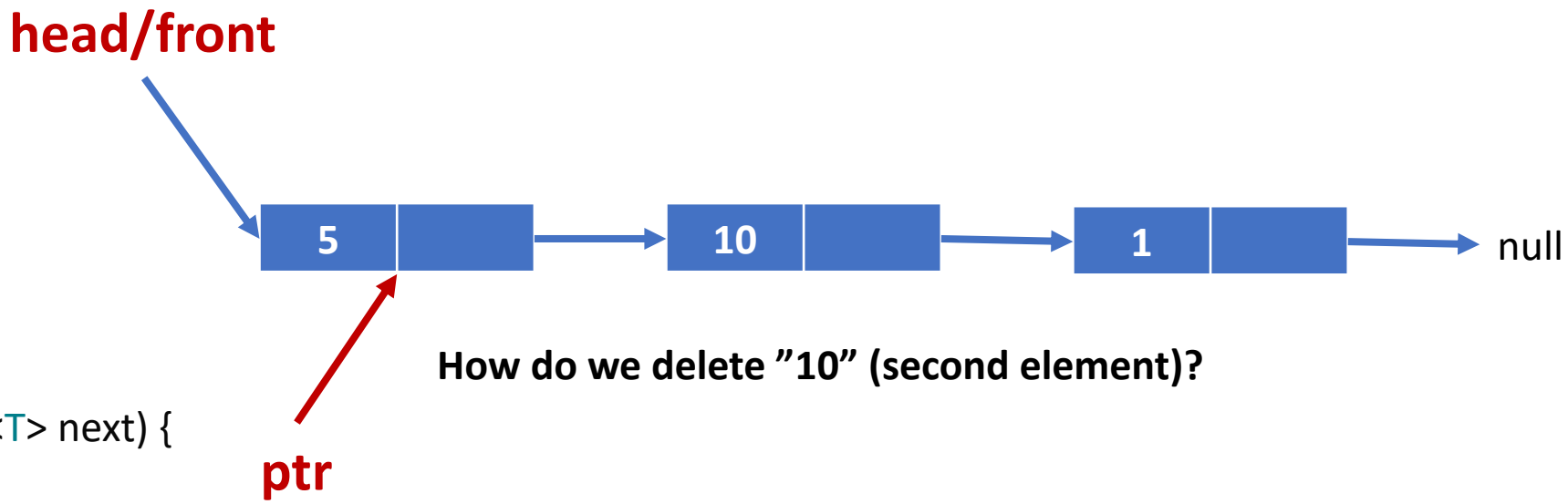
```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```

How do we delete ”10” (second element)?

LinkedList

```
public class Node<T> {
   public T val;
   public Node<T> next;

   public Node(T val) {
      this(val, null);
   }

   public Node(T val, Node<T> next) {
      this.val = val;
      this.next = next;
   }
}
```

In a LinkedList we only need a reference to the first element (head/front)

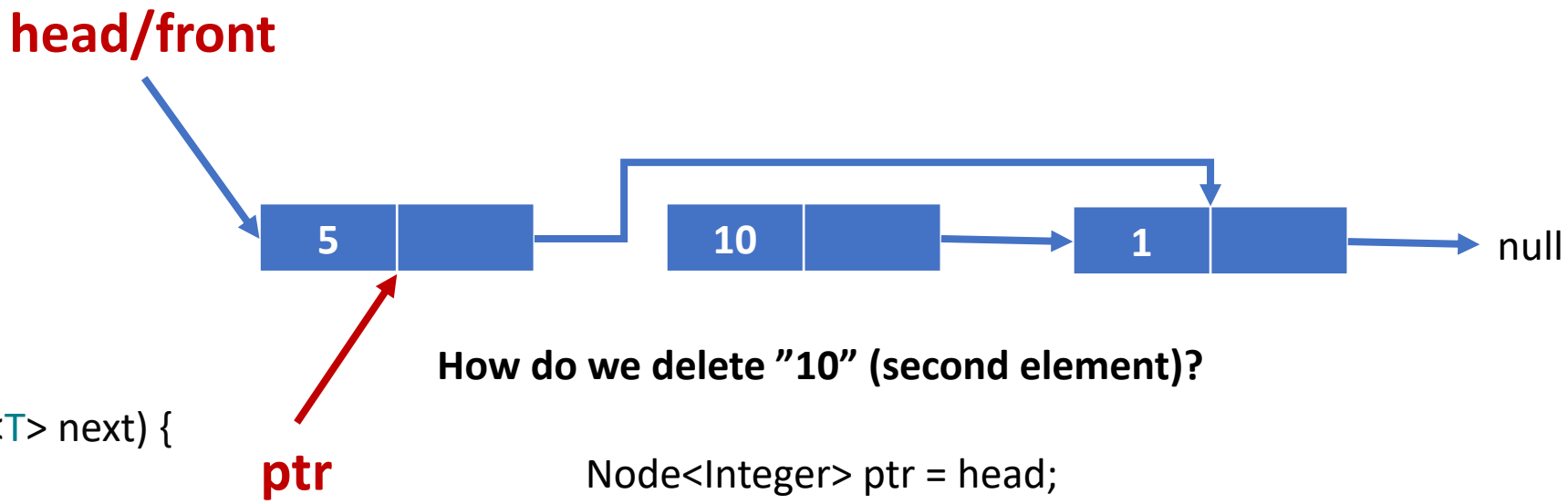


How do we delete ”10” (second element)?

LinkedList

In a LinkedList we only need a reference to the first element (head/front)

```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```



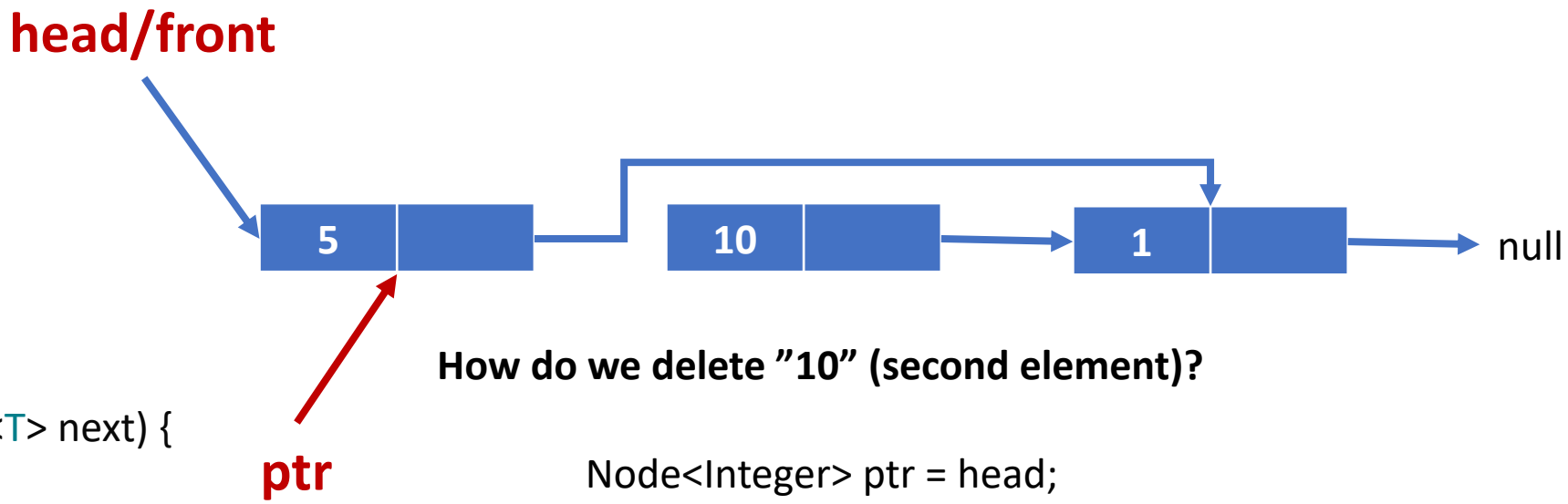
How do we delete ”10” (second element)?

```
Node<Integer> ptr = head;
ptr.next = ptr.next.next;
```

LinkedList

In a LinkedList we only need a reference to the first element (head/front)

```
public class Node<T> {
   public T val;
   public Node<T> next;

   public Node(T val) {
      this(val, null);
   }

   public Node(T val, Node<T> next) {
      this.val = val;
      this.next = next;
   }
}
```



How do we delete ”10” (second element)?

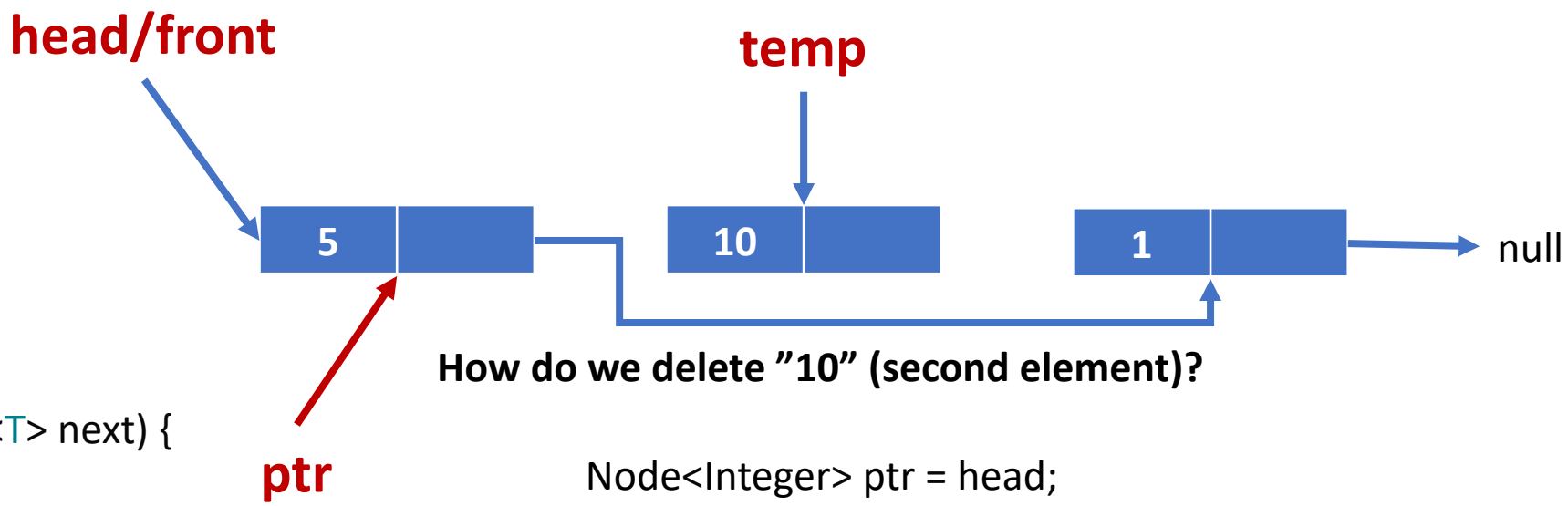
```
Node<Integer> ptr = head;
ptr.next = ptr.next.next;
```

We didn’t need to shift anything!

LinkedList

In a LinkedList we only need a reference to the first element (head/front)

```
public class Node<T> {
   public T val;
   public Node<T> next;

   public Node(T val) {
      this(val, null);
   }

   public Node(T val, Node<T> next) {
      this.val = val;
      this.next = next;
   }
}
```



How do we delete ”10” (second element)?

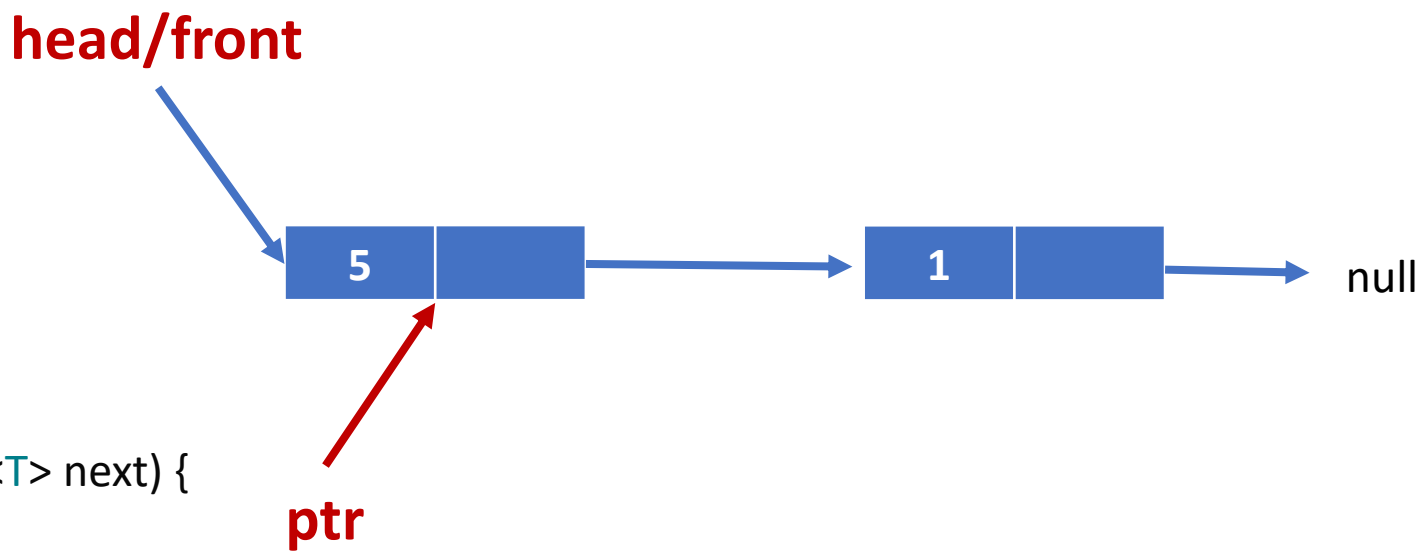
```
Node<Integer> ptr = head;
Node<Integer> temp = ptr.next;
ptr.next = temp.next;
temp.next = null;
```

We didn’t need to shift anything!

LinkedList

In a LinkedList we only need a reference to the first element (head/front)

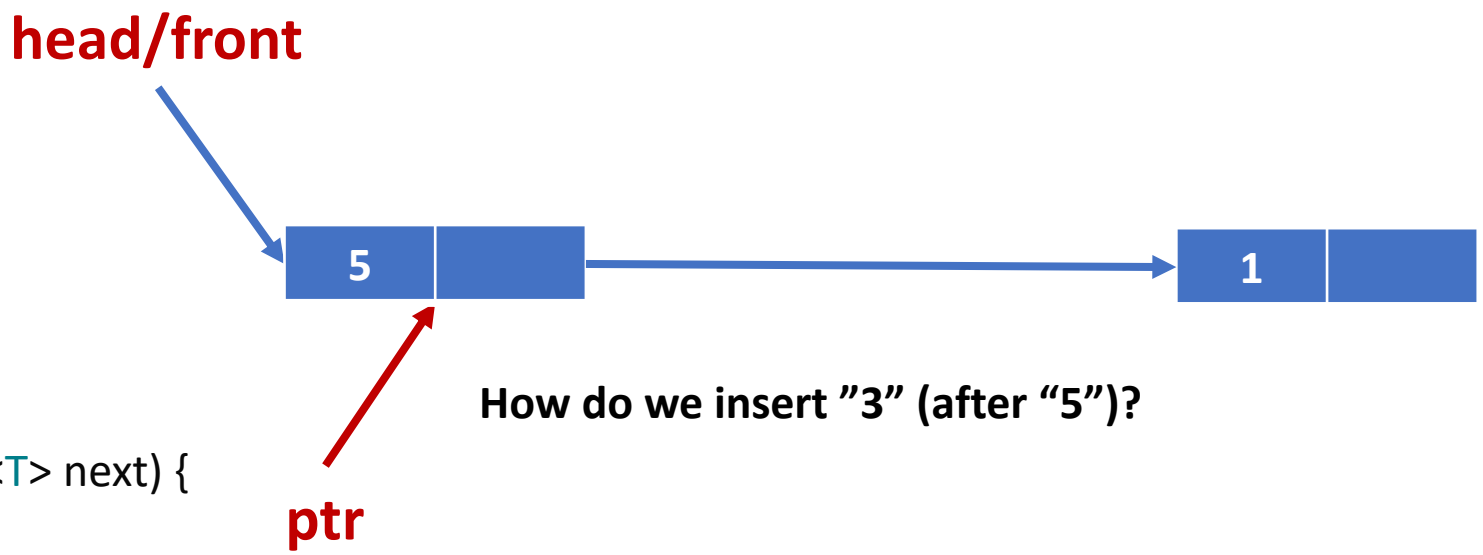
```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```



LinkedList

In a LinkedList we only need a reference to the first element (head/front)

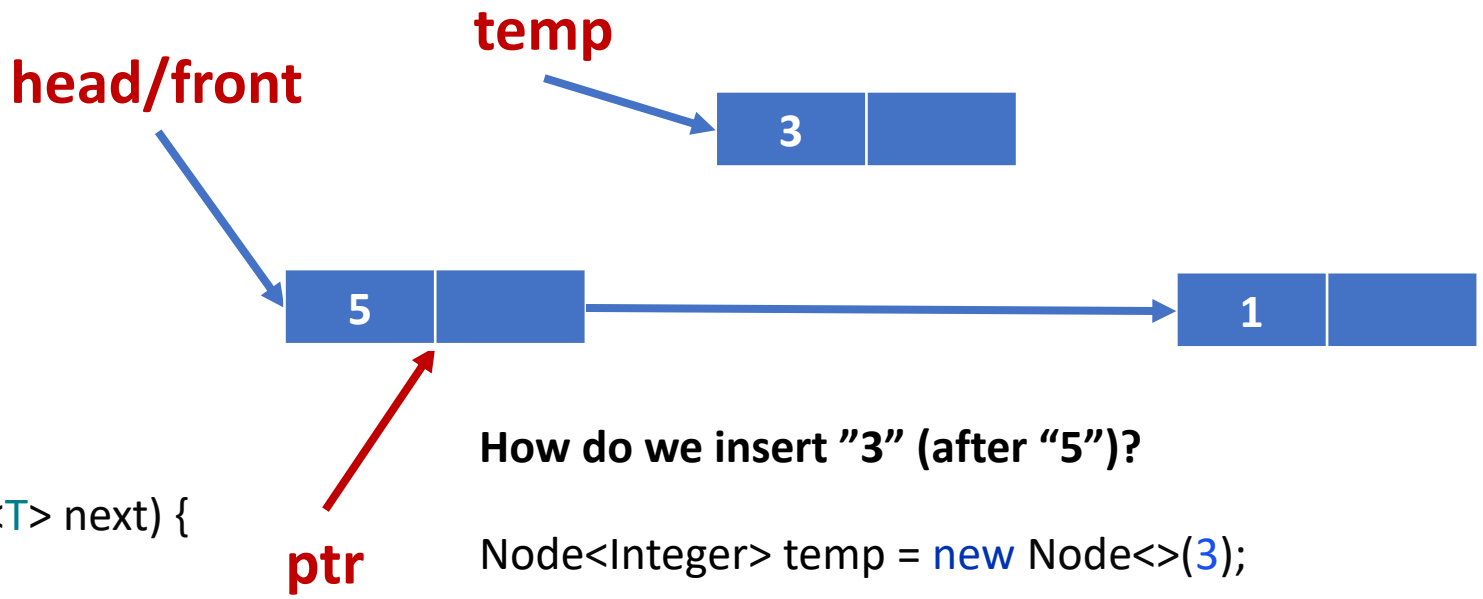
```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```



LinkedList

In a LinkedList we only need a reference to the first element (head/front)

```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```



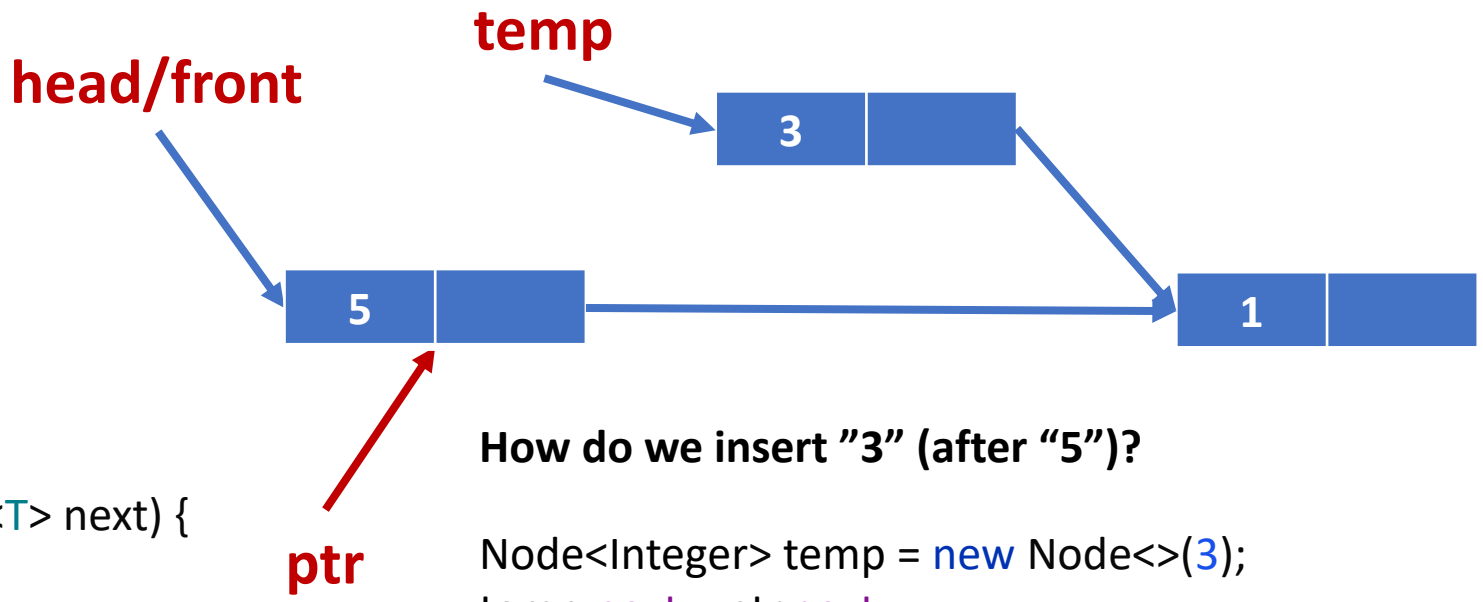
How do we insert ”3” (after “5”)?

```
Node<Integer> temp = new Node<>(3);
```

LinkedList

```
public class Node<T> {
   public T val;
   public Node<T> next;

   public Node(T val) {
      this(val, null);
   }

   public Node(T val, Node<T> next) {
      this.val = val;
      this.next = next;
   }
}
```

In a LinkedList we only need a reference to the first element (head/front)



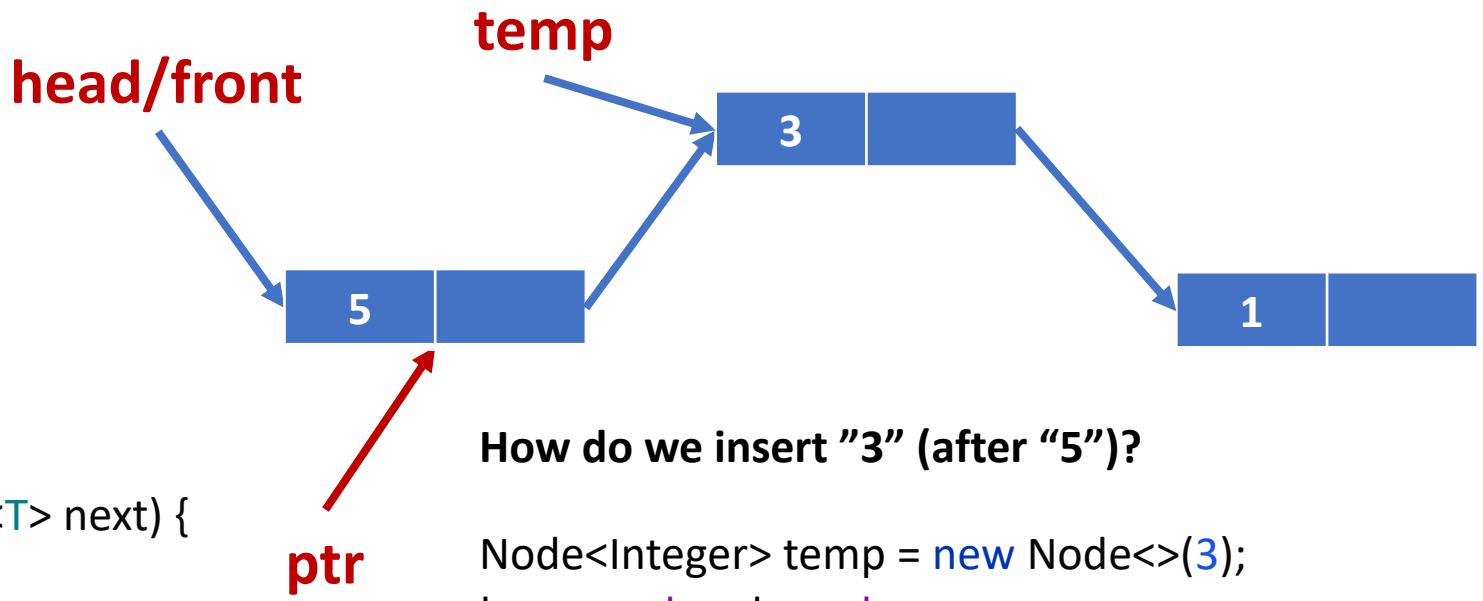
How do we insert ”3” (after “5”)?

```
Node<Integer> temp = new Node<>(3);
temp.next = ptr.next;
```

LinkedList

```
public class Node<T> {
   public T val;
   public Node<T> next;

   public Node(T val) {
      this(val, null);
   }

   public Node(T val, Node<T> next) {
      this.val = val;
      this.next = next;
   }
}
```

In a LinkedList we only need a reference to the first element (head/front)



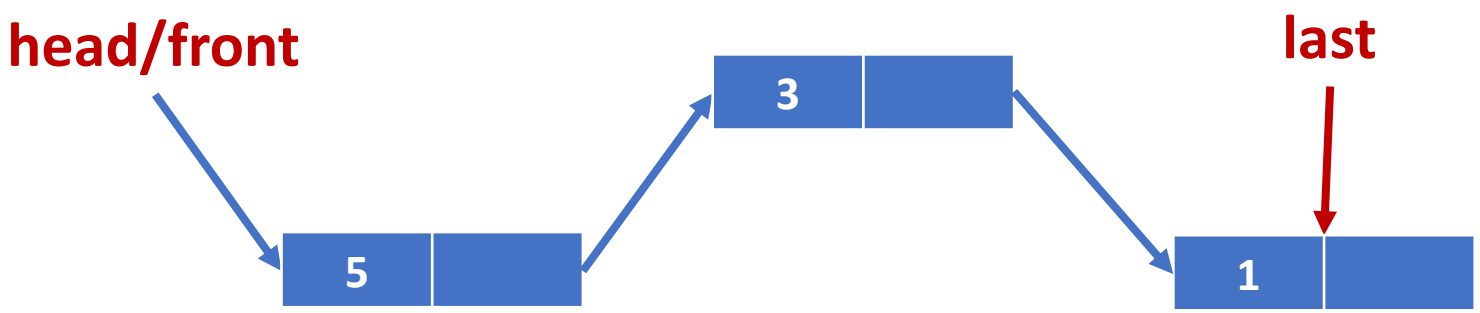
How do we insert ”3” (after “5”)?

```
Node<Integer> temp = new Node<>(3);
temp.next = ptr.next;
ptr.next = temp;
```

We didn’t need to shift anything!

LinkedList

In a LinkedList we only need a reference to the first element (head/front)



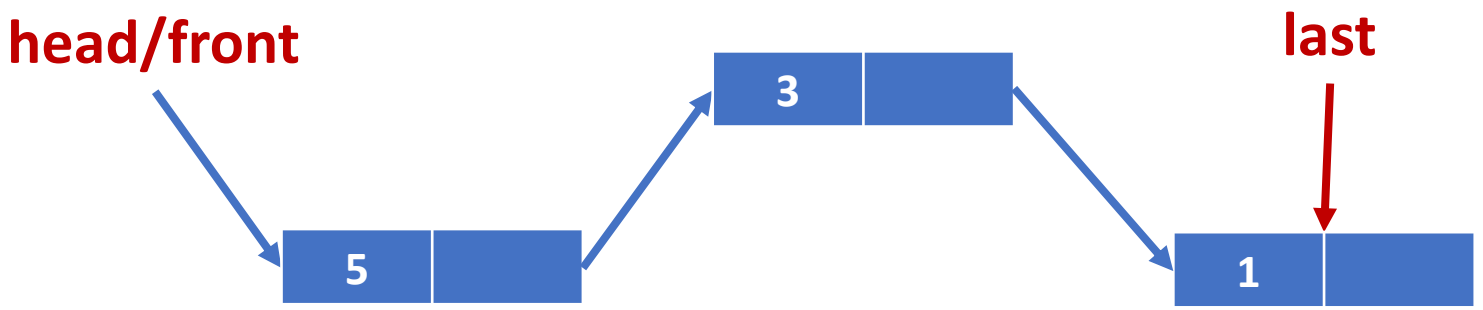
How do we insert ”10” (after “1”)?

```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```

LinkedList

```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```

In a LinkedList we only need a reference to the first element (head/front)



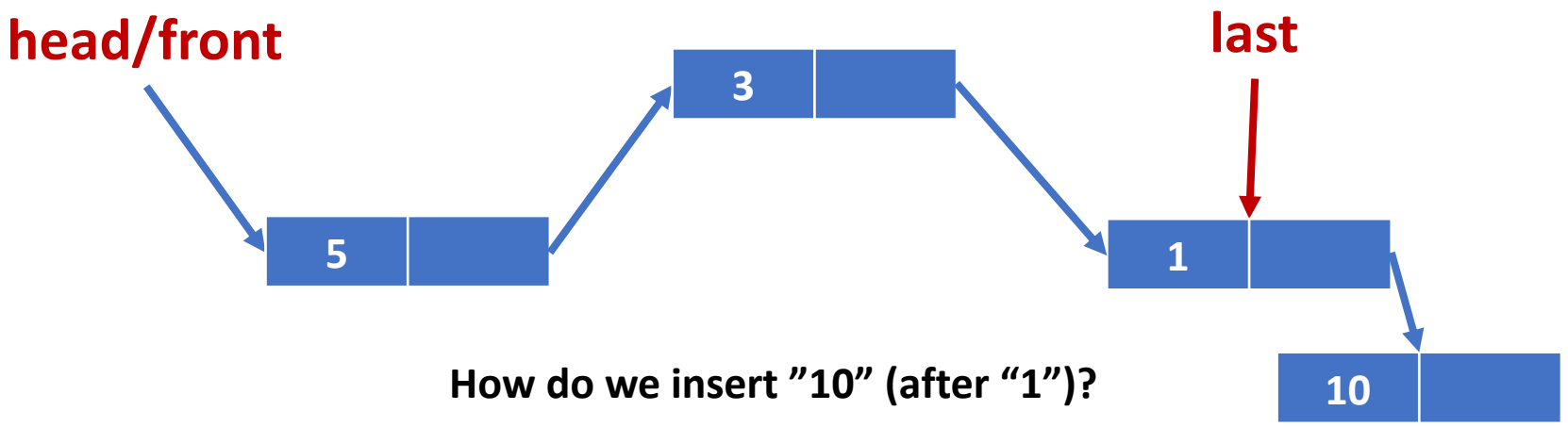
How do we insert ”10” (after “1”)?


```
Node<Integer> temp = new Node<>(10);
```

LinkedList

```
public class Node<T> {
    public T val;
    public Node<T> next;

    public Node(T val) {
        this(val, null);
    }

    public Node(T val, Node<T> next) {
        this.val = val;
        this.next = next;
    }
}
```

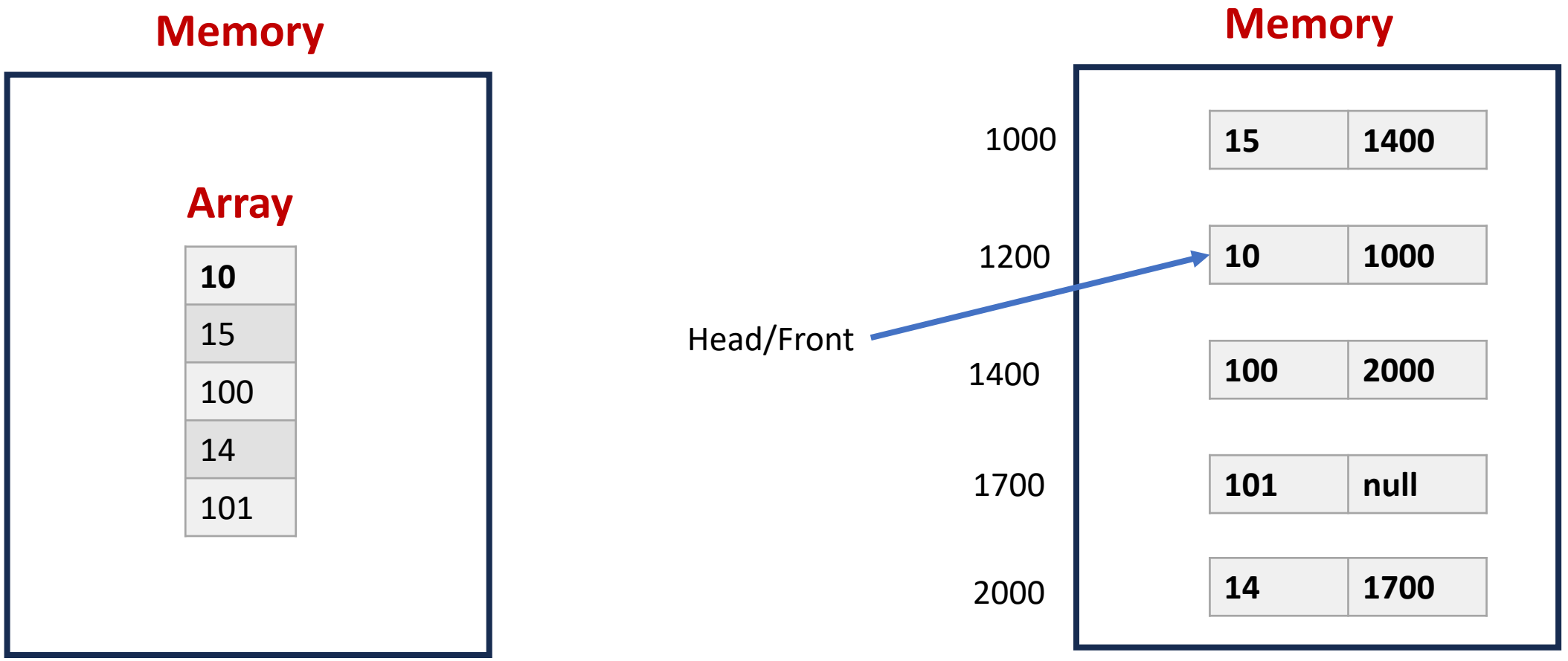
In a LinkedList we only need a reference to the first element (head/front)



How do we insert ”10” (after “1”)?

```
Node<Integer> temp = new Node<>(10);
last.next = temp;
```

Memory (Array)



Lists (Arrays vs LinkedList)

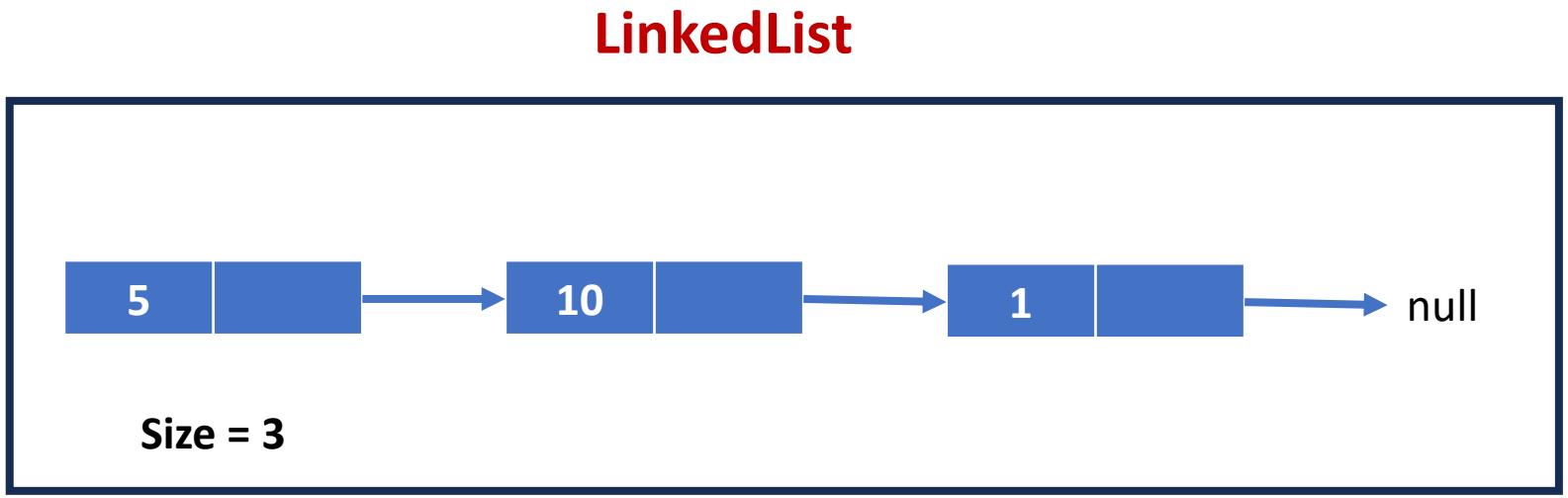
Function	Array	LinkedList
Insert at beginning	O(n)	O(1)
Insert at end	O(1)	**O(n)**
Insert at middle	O(n)	O(n)
Delete at beginning	O(n)	O(1)
Delete at end	O(1)	**O(n)**
Delete at middle	O(n)	O(n)
Find	O(n)	O(n)
Get Element at index	O(1)	O(n)
Print	O(n)	O(n)
Size	O(1)	**O(n)**

Lists (Arrays vs LinkedList)

Function	Array	LinkedList
Insert at beginning	O(n)	O(1)
Insert at end	O(1)	**O(n)**
Insert at middle	O(n)	O(n)
Delete at beginning	O(n)	O(1)
Delete at end	O(1)	**O(n)**
Delete at middle	O(n)	O(n)
Find	O(n)	O(n)
Get Element at index	O(1)	O(n)
Print	O(n)	O(n)
Size	O(1)	**O(n)**

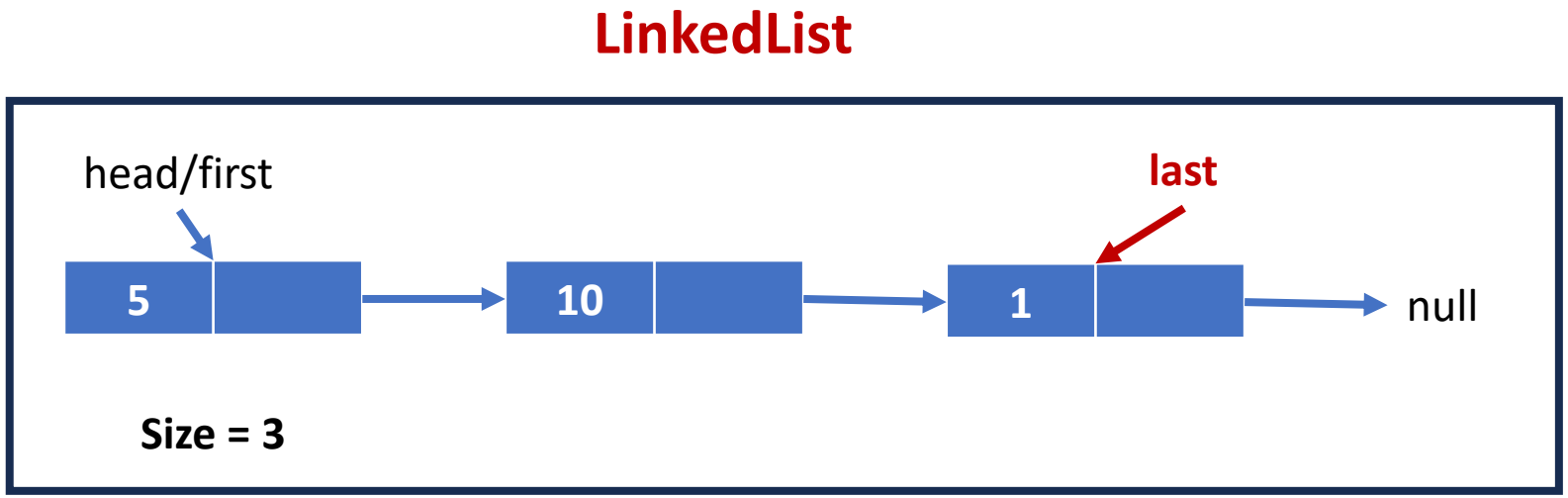
Can we improve time complexity for items bolded in **red**?

Lists (Arrays vs LinkedList)



Add size field and update it every time an element is added/deleted

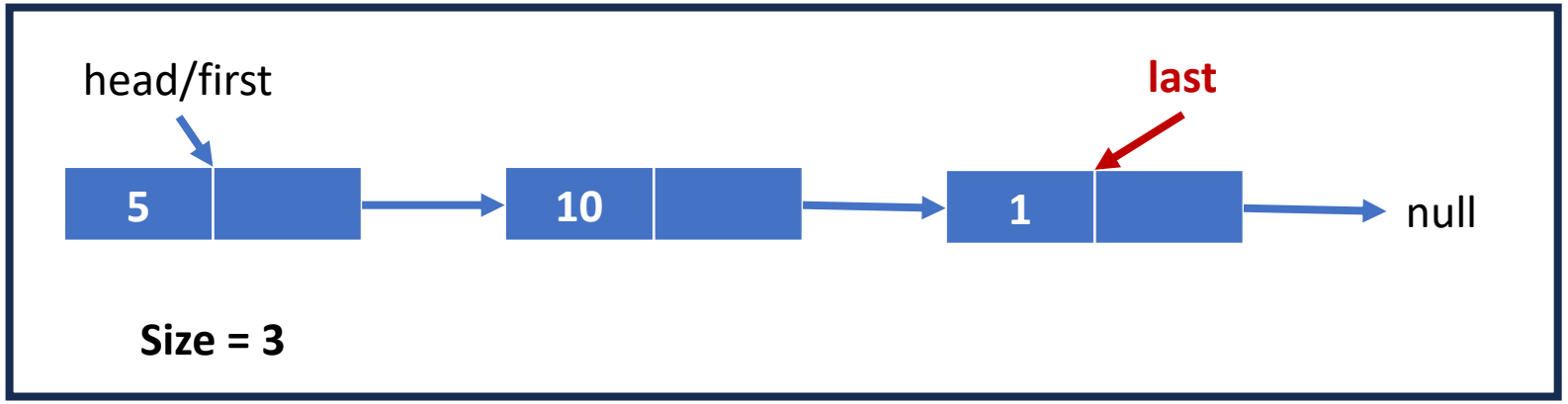
Lists (Arrays vs LinkedList)



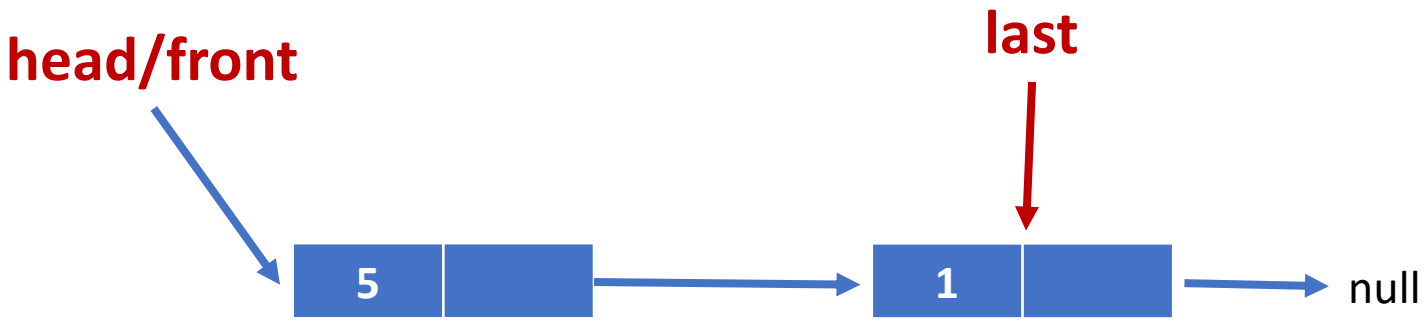
1. Add size field and update it every time an element is added/deleted.
2. Add last pointer that points to the last element!
    1. Now we can add a new node to the end in O(1)
    2. We can delete an element from the end in O(1)

LinkedList (Implementation)

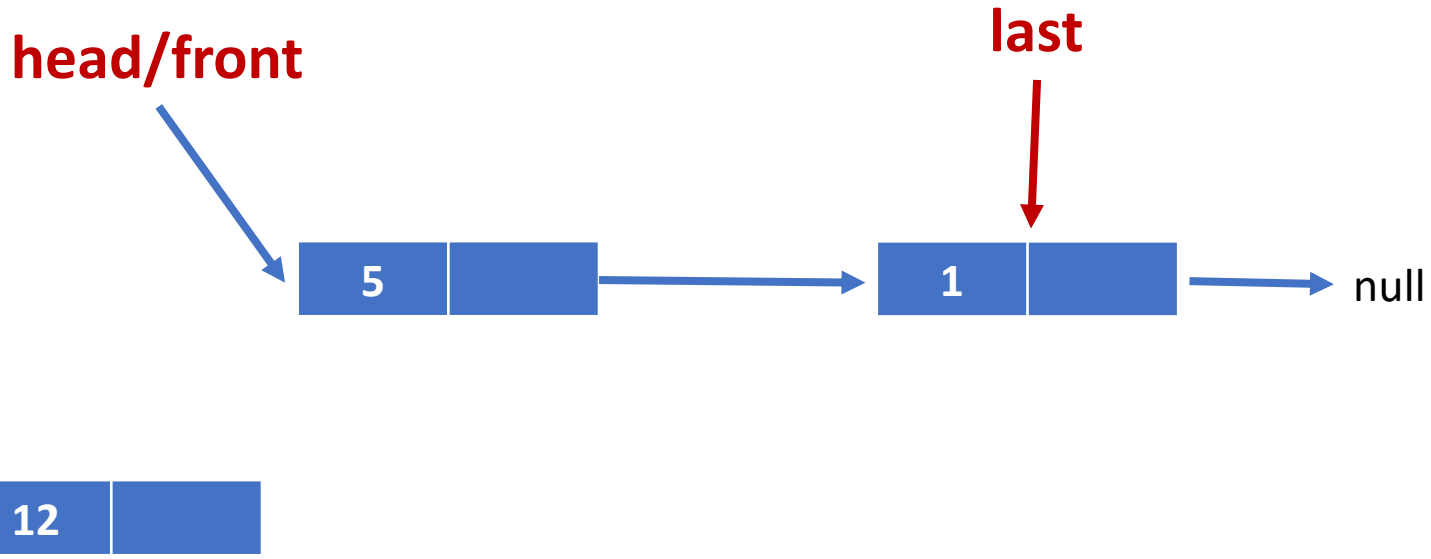
```
public class LinkedList<T> {
    Node<T> first;
    Node<T> last;
    int count;

    public LinkedList() {
        first = last = null;
        count = 0;
    }
}
```



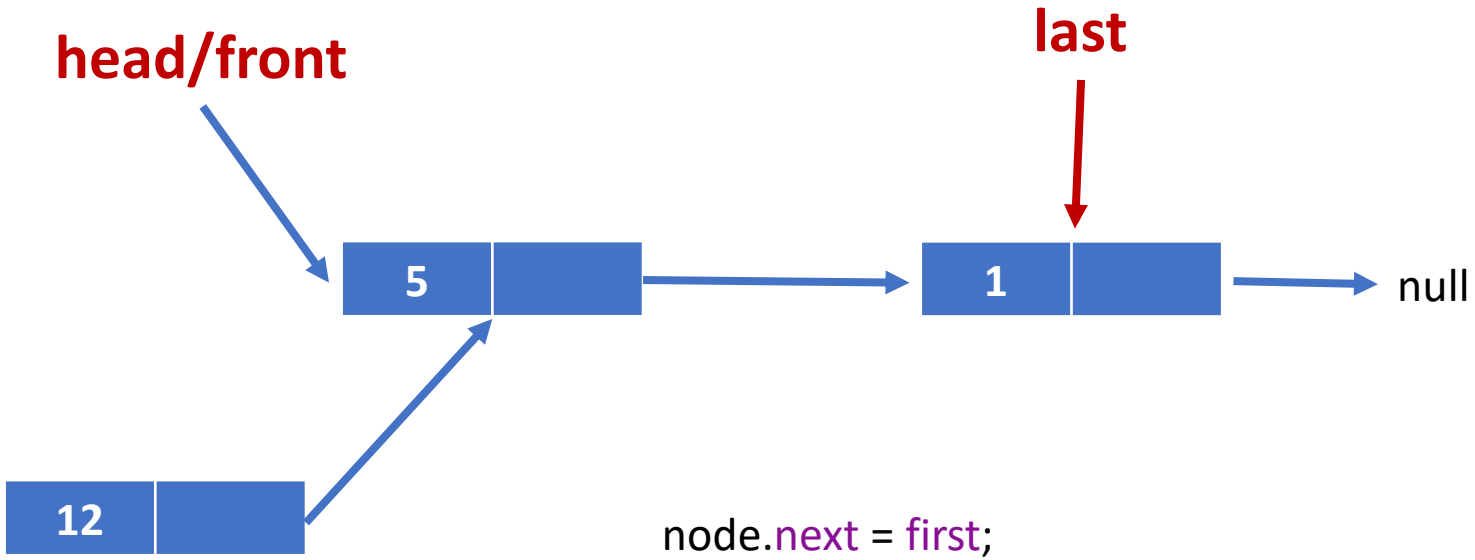
LinkedList (addFirst)



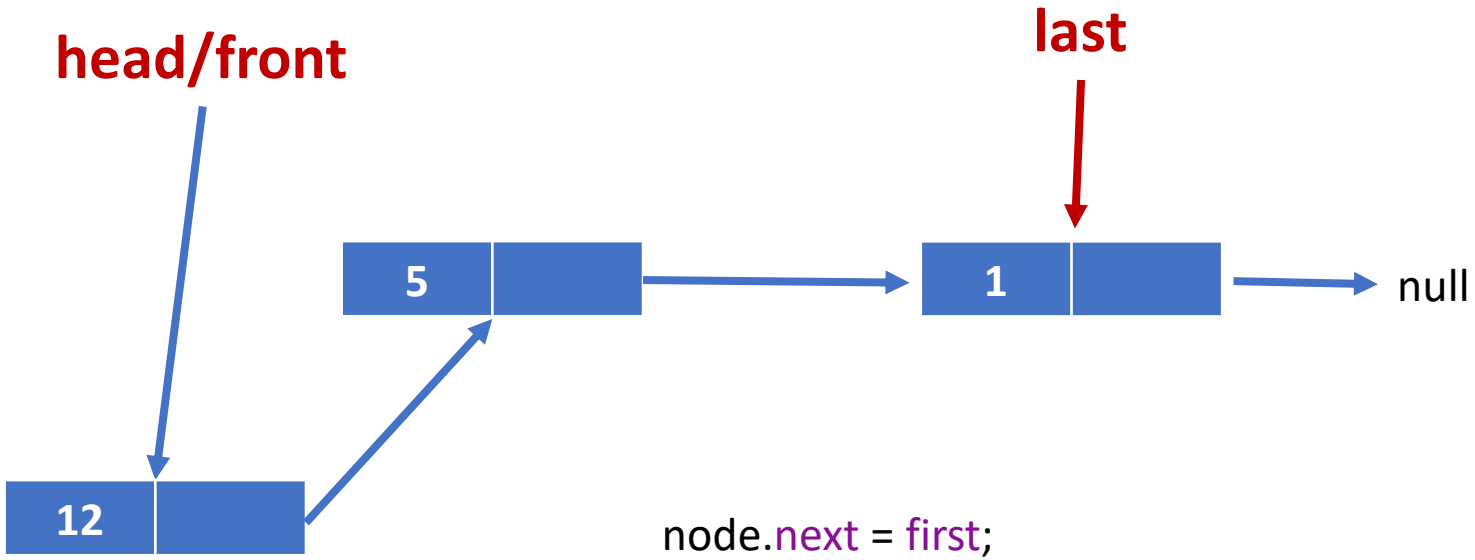
LinkedList (addFirst)



LinkedList (addFirst)



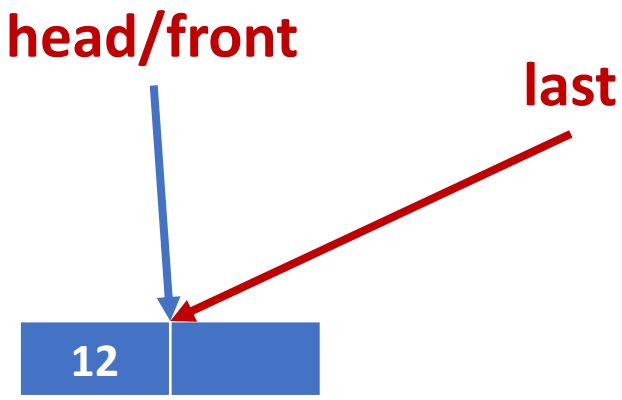
LinkedList (addFirst)




```
node.next = first;
first = node;
```

LinkedList (addFirst)

What if it's empty?



first = last = node;

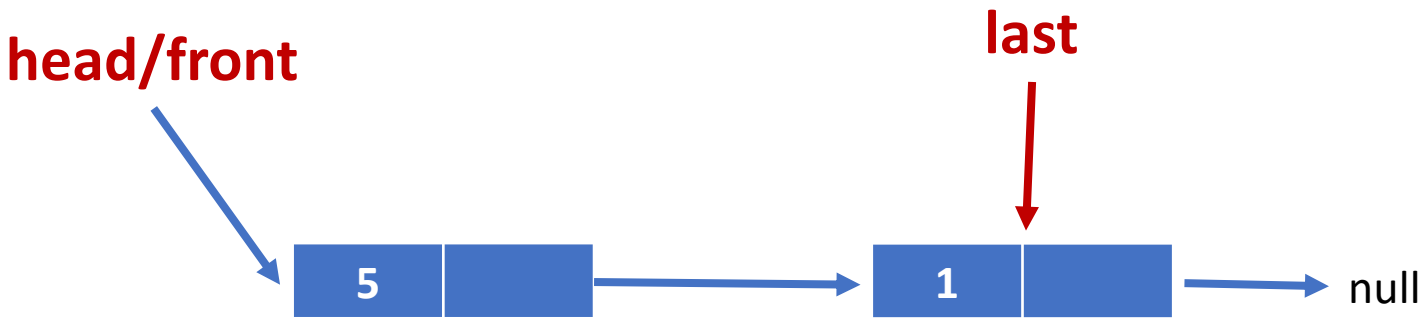
LinkedList (addFirst)

```
public void addFirst(T object) {
    Node<T> node = new Node<>(object);
    if (count == 0) {
        first = last = node;
    } else {
        node.next = first;
        first = node;
    }

    count++;
}
```

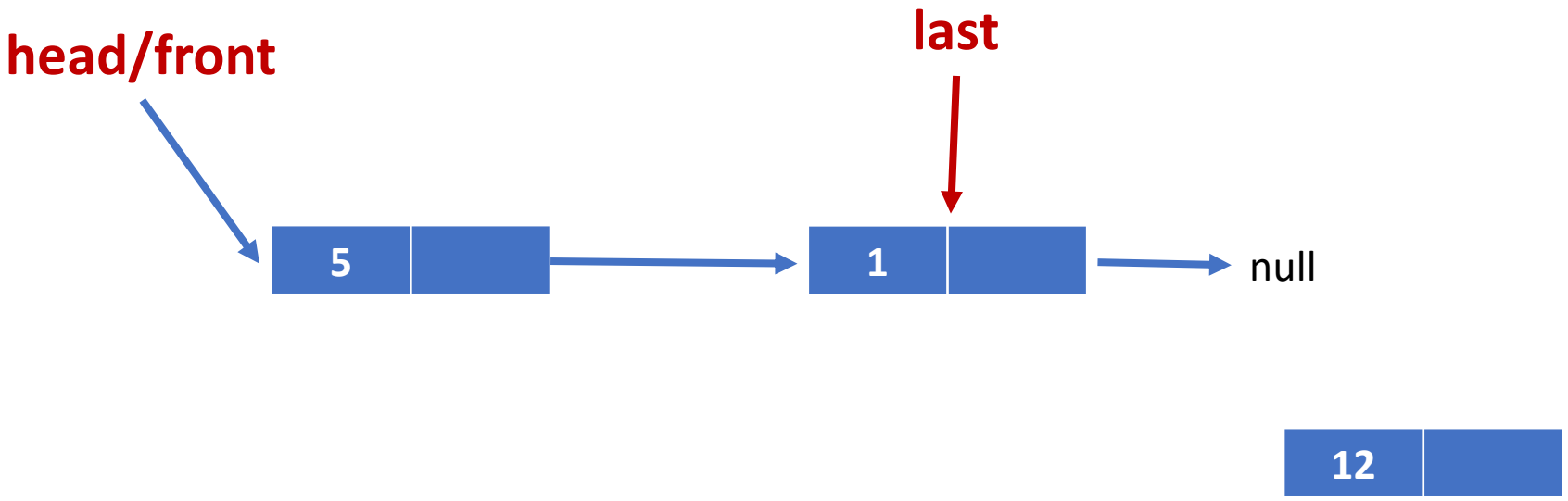
LinkedList (getFirst)

```
public T getFirst() {
    if (count == 0) {
        return null;
    }

    return first.val;
}
```

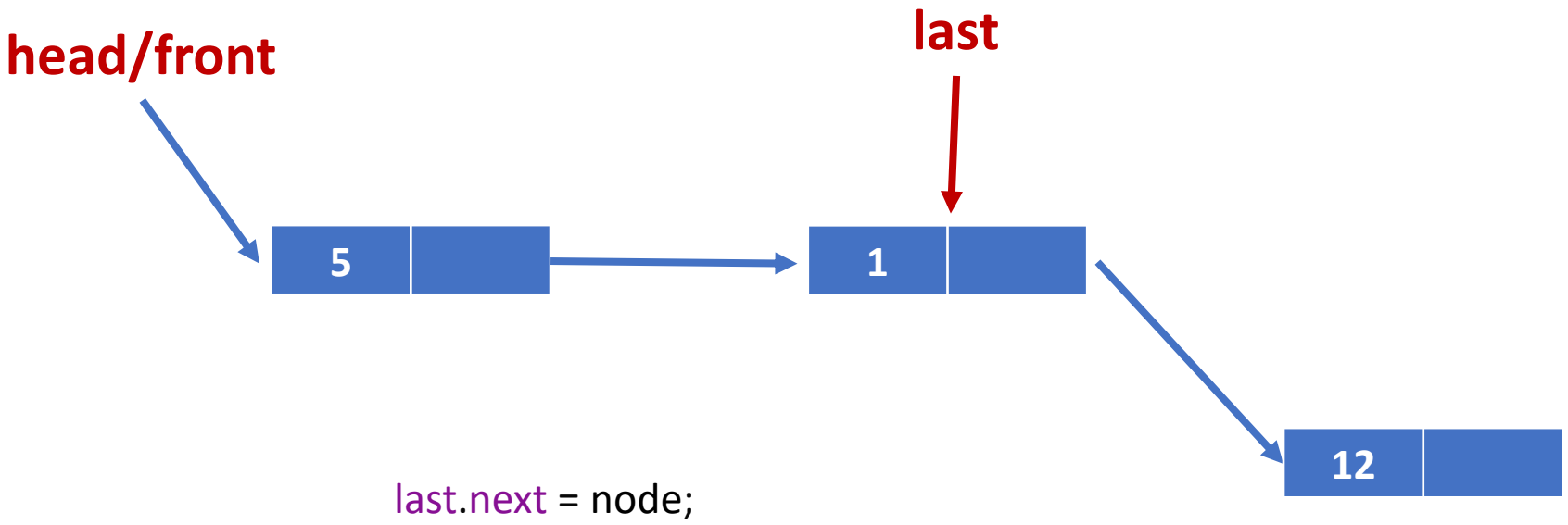
LinkedList (addLast)



LinkedList (addLast)

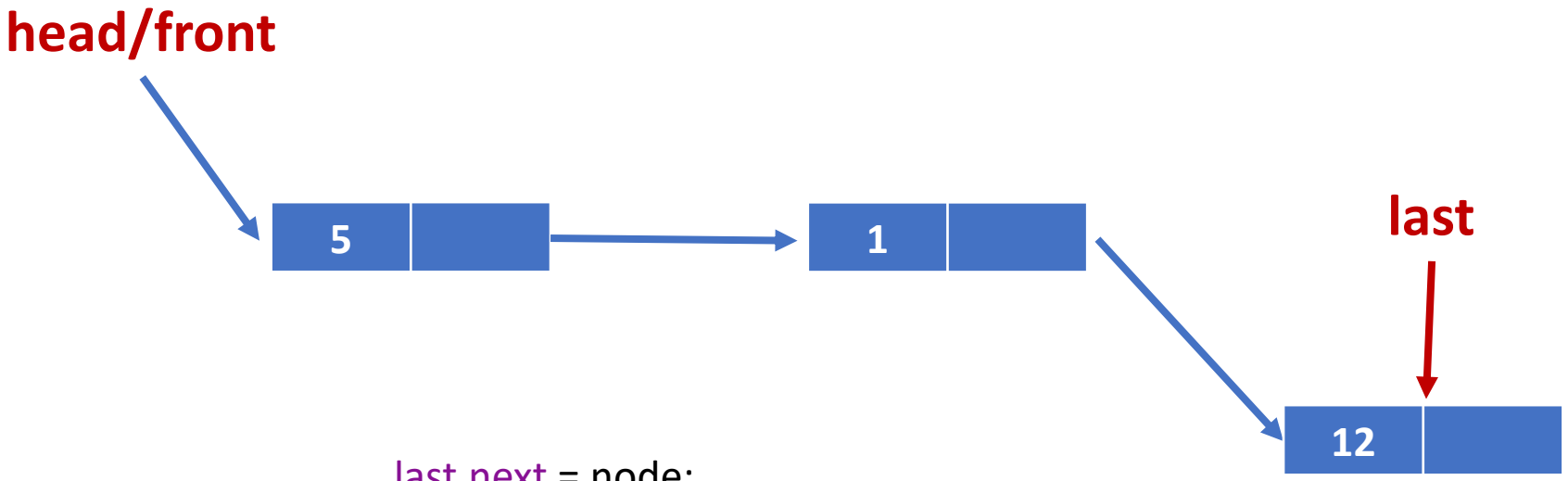


LinkedList (addLast)



last.next = node;

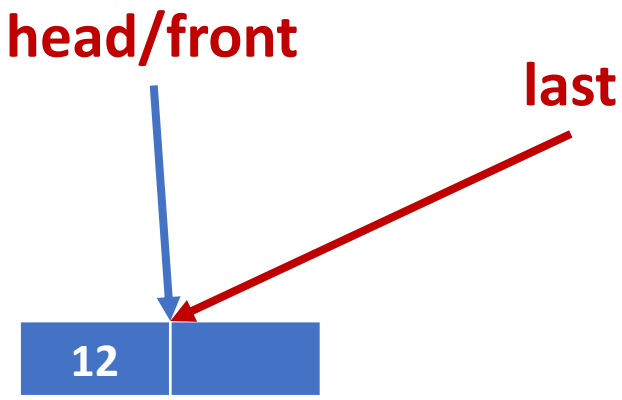
LinkedList (addLast)




```
last.next = node;
last = last.next;
```

LinkedList (addLast)

What if it's empty?



LinkedList (addLast)

```
public void addLast(T object) {
    Node<T> node = new Node<>(object);
    if (count == 0) {
        first = last = node;
    } else {
        last.next = node;
        last = last.next;
    }

    count++;
}
```

LinkedList (getLast)

```
public T getLast() {
    if (count == 0) {
        return null;
    }

    return last.val;
}
```