Background

CSCI 232

Running Time – Array Implementation

```
public void addStudent(String name, int studentNumber, double gpa) {
   Student newStudent = new Student(name, studentNumber, gpa);
   Student[] tempDatabase = new Student[database.length + 1];
   for (int i = 0; i < database.length; i++) {
       tempDatabase[i] = database[i];
   }
   tempDatabase[database.length] = newStudent;
   database = tempDatabase;
}
```

Running Time: ??

Running Time – Array Implementation

```
public void addStudent(String name, int studentNumber, double gpa) {
   Student newStudent = new Student(name, studentNumber, gpa);
   Student[] tempDatabase = new Student[database.length + 1];
   for (int i = 0; i < database.length; i++) {
       tempDatabase[i] = database[i];
   }
   tempDatabase[database.length] = newStudent;
   database = tempDatabase;
}
```

Running Time: Number of operations required to complete algorithm.

Running Time – Array Implementation

```
public void addStudent(String name, int studentNumber, double gpa) {
   Student newStudent = new Student(name, studentNumber, gpa);
   Student[] tempDatabase = new Student[database.length + 1];
   for (int i = 0; i < database.length; i++) {
       tempDatabase[i] = database[i];
   }
   tempDatabase[database.length] = newStudent;
   database = tempDatabase;
}
```

Running Time: Number of operations required to complete algorithm.

Big O Notation: Upper bound on asymptotic growth. I.e. Worst case upper bound of a function.

Running Time – Array Implementation

```
public void addStudent(String name, int studentNumber, double gpa) {
   Student newStudent = new Student(name, studentNumber, gpa);
   Student[] tempDatabase = new Student[database.length + 1];
   for (int i = 0; i < database.length; i++) {
       tempDatabase[i] = database[i];
   }
   tempDatabase[database.length] = newStudent;
   database = tempDatabase;
}
```

Running Time: Number of operations required to complete algorithm.

Big O Notation: Upper bound on asymptotic growth. I.e. Worst case upper bound of a function.

$$\mathbf{3n^2 + 2n - 156 \in O\left(n^2\right)}$$

Running Time – Array Implementation

Let n = ??

```
public void addStudent(String name, int studentNumber, double gpa) {
   Student newStudent = new Student(name, studentNumber, gpa);
   Student[] tempDatabase = new Student[database.length + 1];
   for (int i = 0; i < database.length; i++) {
       tempDatabase[i] = database[i];
   }
   tempDatabase[database.length] = newStudent;
   database = tempDatabase;
}
```

Running Time: Number of operations required to complete algorithm.

Big O Notation: Upper bound on asymptotic growth. I.e. Worst case upper bound of a function.

$$3n^2 + 2n - 156 \in O\left(n^2\right)$$

Running Time – Array Implementation

Let n = `database.length`

```
public void addStudent(String name, int studentNumber, double gpa) {
   Student newStudent = new Student(name, studentNumber, gpa);
   Student[] tempDatabase = new Student[database.length + 1];
   for (int i = 0; i < database.length; i++) {
       tempDatabase[i] = database[i];
   }
   tempDatabase[database.length] = newStudent;
   database = tempDatabase;
}
```

Running Time: Number of operations required to complete algorithm.

Big O Notation: Upper bound on asymptotic growth. I.e. Worst case upper bound of a function.

$$\mathbf{3n^2 + 2n - 156 \in O\left(n^2\right)}$$

Running Time – Array Implementation

Let n = database.length

```
   public void addStudent(String name, int studentNumber, double gpa) {
?? →Student newStudent = new Student(name, studentNumber, gpa);
      Student[] tempDatabase = new Student[database.length + 1];
      for (int i = 0; i < database.length; i++) {
          tempDatabase[i] = database[i];
      }
      tempDatabase[database.length] = newStudent;
      database = tempDatabase;
   }
```

Running Time: Number of operations required to complete algorithm.

Big O Notation: Upper bound on asymptotic growth. I.e. Worst case upper bound of a function.

$$3n^2 + 2n - 156 \in O\left(n^2\right)$$

Running Time – Array Implementation

Let n = `database.length`

```
         public void addStudent(String name, int studentNumber, double gpa) {
   O(1)→Student newStudent = new Student(name, studentNumber, gpa);
    ??  →Student[] tempDatabase = new Student[database.length + 1];
            for (int i = 0; i < database.length; i++) {
                tempDatabase[i] = database[i];
            }
            tempDatabase[database.length] = newStudent;
            database = tempDatabase;
         }
```

Running Time: Number of operations required to complete algorithm.

Big O Notation: Upper bound on asymptotic growth. I.e. Worst case upper bound of a function.

$$3n^2 + 2n - 156 \in O(n^2)$$

Running Time – Array Implementation

Let `n = database.length`

```
      public void addStudent(String name, int studentNumber, double gpa) {
O(1)→    Student newStudent = new Student(name, studentNumber, gpa);
O(n)→    Student[] tempDatabase = new Student[database.length + 1];
        ⎧for (int i = 0; i < database.length; i++) {
 ??     ⎨    tempDatabase[i] = database[i];
        ⎩}
         tempDatabase[database.length] = newStudent;
         database = tempDatabase;
      }
```

Running Time: Number of operations required to complete algorithm.

Big O Notation: Upper bound on asymptotic growth. I.e. Worst case upper bound of a function.

$$3n^2 + 2n - 156 \in O\left(n^2\right)$$

Running Time – Array Implementation

Let n = database.length

```
           public void addStudent(String name, int studentNumber, double gpa) {
    O(1)→Student newStudent = new Student(name, studentNumber, gpa);
    O(n)→Student[] tempDatabase = new Student[database.length + 1];
        ┌for (int i = 0; i < database.length; i++) {
    O(n)┤    tempDatabase[i] = database[i];
        └}
     ??  →tempDatabase[database.length] = newStudent;
         database = tempDatabase;
           }
```

Running Time: Number of operations required to complete algorithm.

Big O Notation: Upper bound on asymptotic growth. I.e. Worst case upper bound of a function.

$$3n^2 + 2n - 156 \in O\left(n^2\right)$$

Running Time – Array Implementation

Let `n = database.length`

```
        public void addStudent(String name, int studentNumber, double gpa) {
O(1) →     Student newStudent = new Student(name, studentNumber, gpa);
O(n) →     Student[] tempDatabase = new Student[database.length + 1];
          ⎧ for (int i = 0; i < database.length; i++) {
O(n)     ⎨     tempDatabase[i] = database[i];
          ⎩ }
O(1) →     tempDatabase[database.length] = newStudent;
 ??  →     database = tempDatabase;
        }
```

Running Time: Number of operations required to complete algorithm.

Big O Notation: Upper bound on asymptotic growth. I.e. Worst case upper bound of a function.

$$3n^2 + 2n - 156 \in O(n^2)$$

Running Time – Array Implementation

Let `n = database.length`

```
         public void addStudent(String name, int studentNumber, double gpa) {
O(1)→ Student newStudent = new Student(name, studentNumber, gpa);
O(n)→ Student[] tempDatabase = new Student[database.length + 1];
     ⎧for (int i = 0; i < database.length; i++) {
O(n)⎨        tempDatabase[i] = database[i];
     ⎩}
O(1)→ tempDatabase[database.length] = newStudent;
O(1)→ database = tempDatabase;
         }
```

Running Time: Number of operations required to complete algorithm.

Big O Notation: Upper bound on asymptotic growth. I.e. Worst case upper bound of a function.

$$3n^2 + 2n - 156 \in O\left(n^2\right)$$

Running Time – Array Implementation

Let n = database.length

```
        public void addStudent(String name, int studentNumber, double gpa) {
O(1)→   Student newStudent = new Student(name, studentNumber, gpa);
O(n)→   Studen                           = new Student[database.length + 1];
     ⎧  for (i                      tabase.length; i++) {
O(n) ⎨      te                      database[i];
     ⎩  }
O(1)→   tempDatabase[database.length] = newStudent;
O(1)→   database = tempDatabase;
        }
```

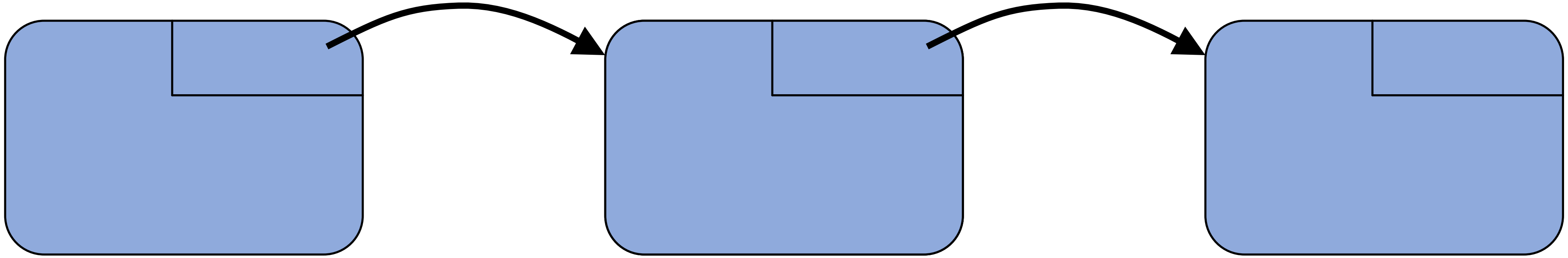
$$\boxed{O(n)}$$

Running Time: Number of operations required to complete algorithm.

Big O Notation: Upper bound on asymptotic growth. I.e. Worst case upper bound of a function.

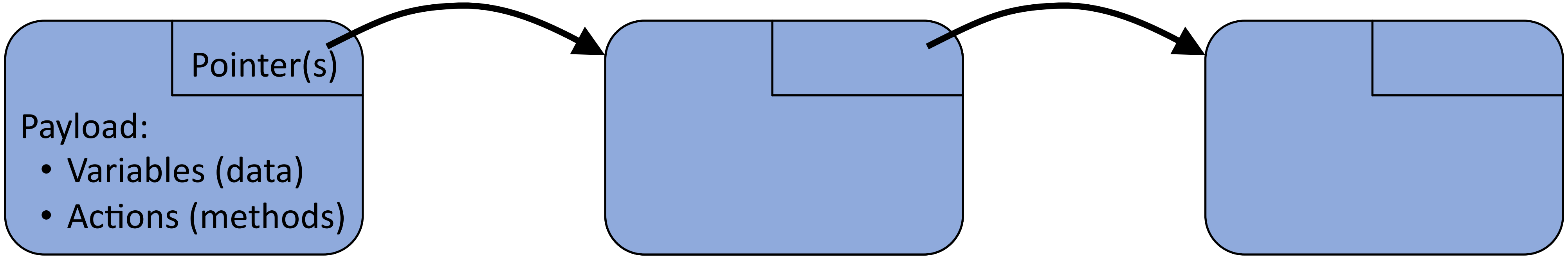
$$3n^2 + 2n - 156 \in O(n^2)$$

Linked List



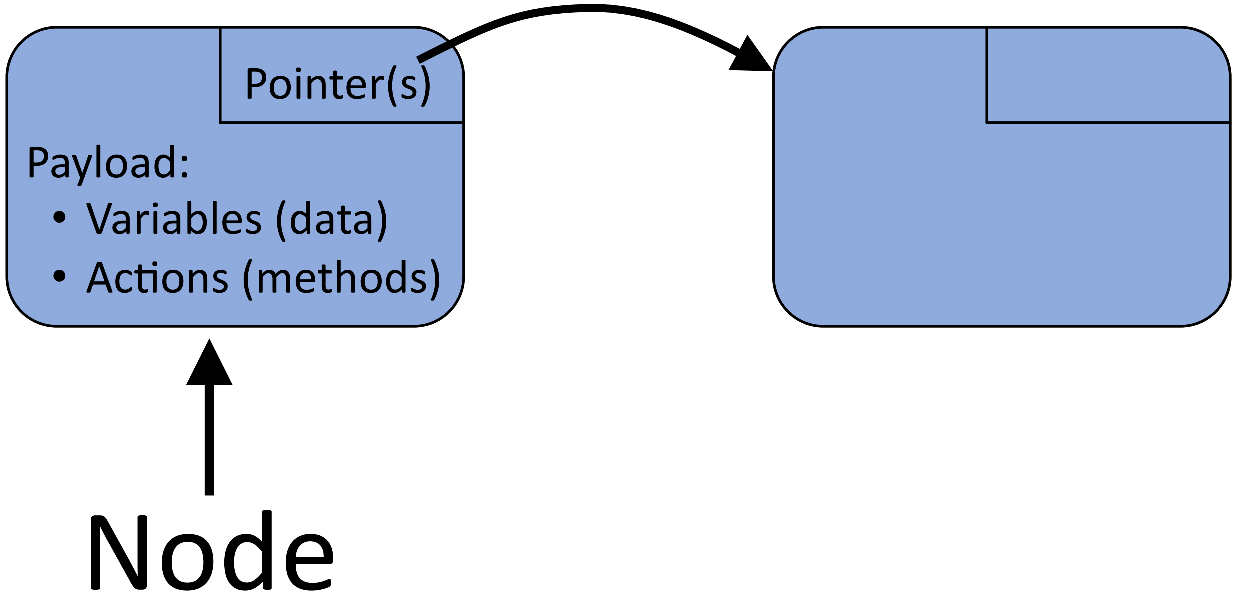
Linked List: Linear data structure of ordered objects.

Linked List



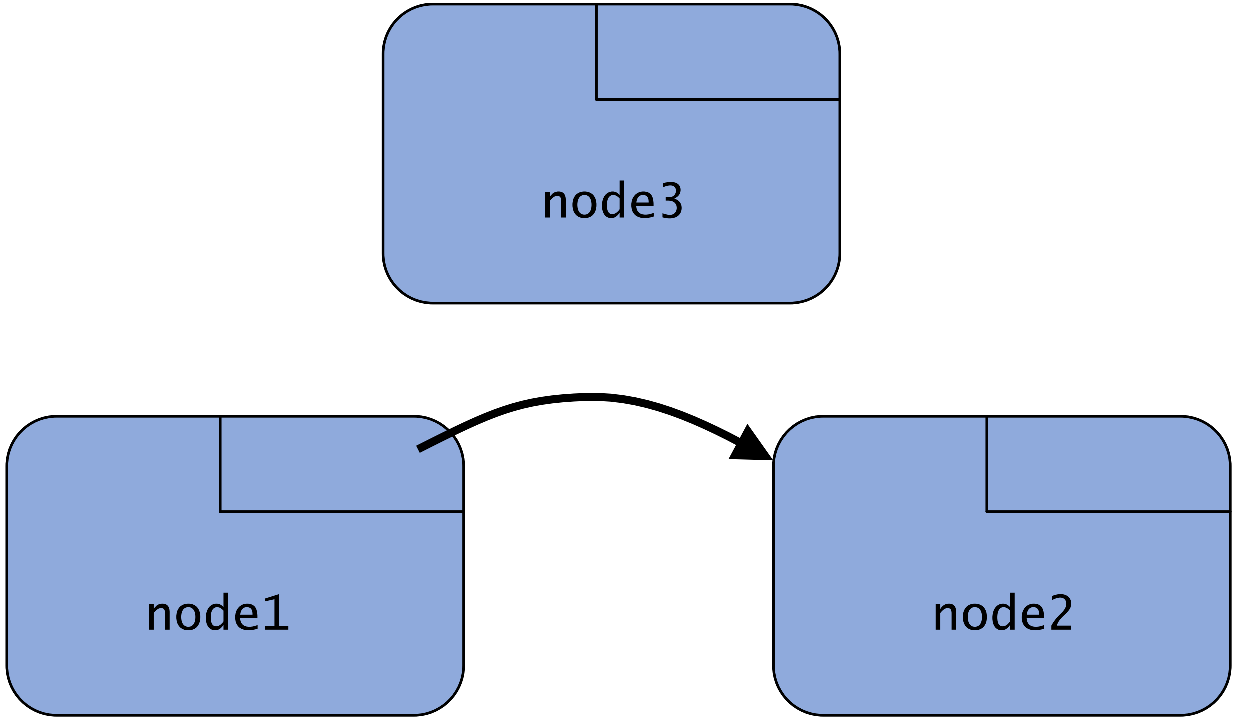
Linked List: Linear data structure of ordered objects.

Linked List



```
public class Node {
     private Node next;
     private String data1;
     private int data2;
     public Node(???) {
          ???
     }
     public Node getNext() {
          return next;
     }
     public void setNext(Node next) {
          this.next = next;
     }
}
```

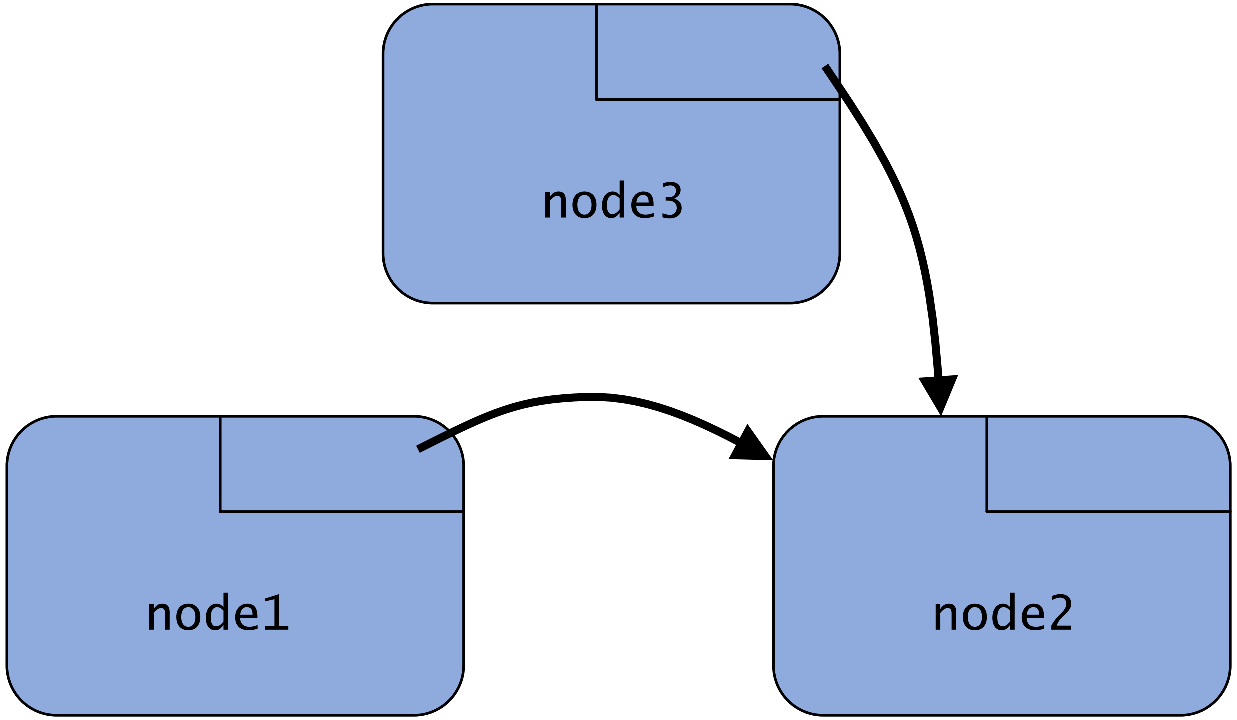
Linked List



```
// Linked List insertion??
```

```
public class Node {
     private Node next;
     private String data1;
     private int data2;
     public Node(???) {
          ???
     }
     public Node getNext() {
          return next;
     }
     public void setNext(Node next) {
          this.next = next;
     }
}
```

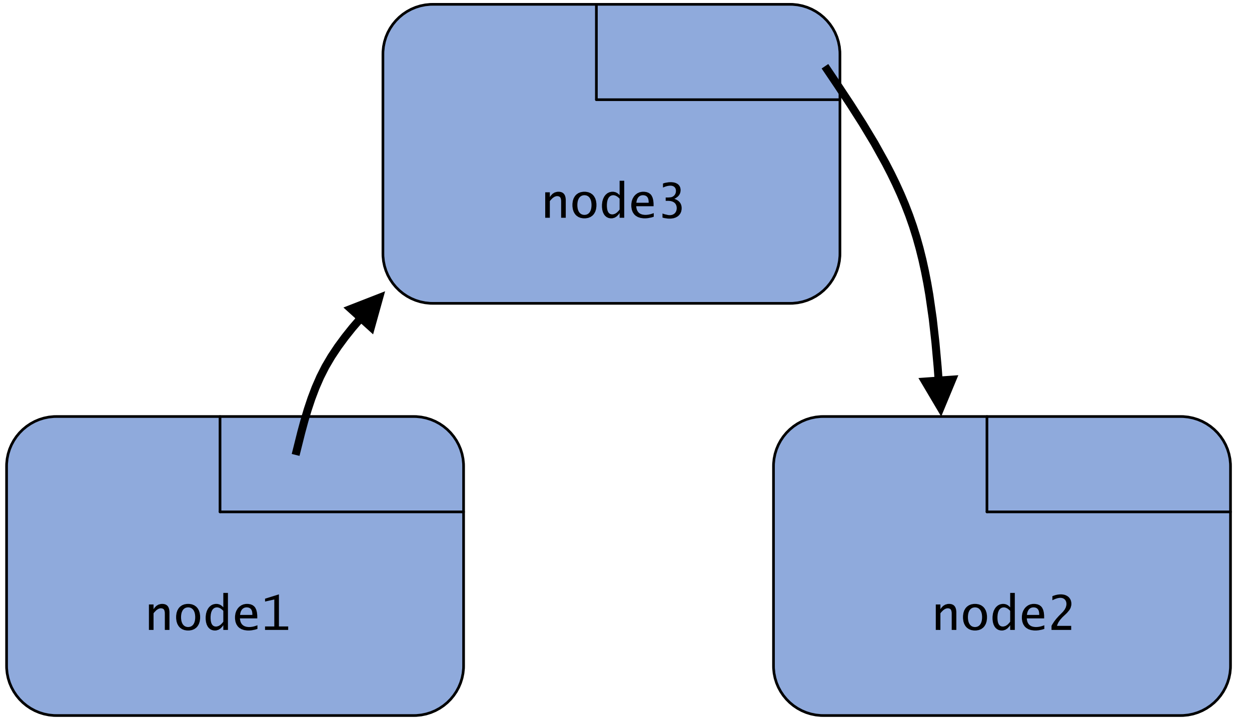
Linked List



```
// Linked List insertion??
node1.setNext(node3);
node3.setNext(node2);
```

```
public class Node {
     private Node next;
     private String data1;
     private int data2;
     public Node(???) {
          ???
     }
     public Node getNext() {
          return next;
     }
     public void setNext(Node next) {
          this.next = next;
     }
}
```

Linked List

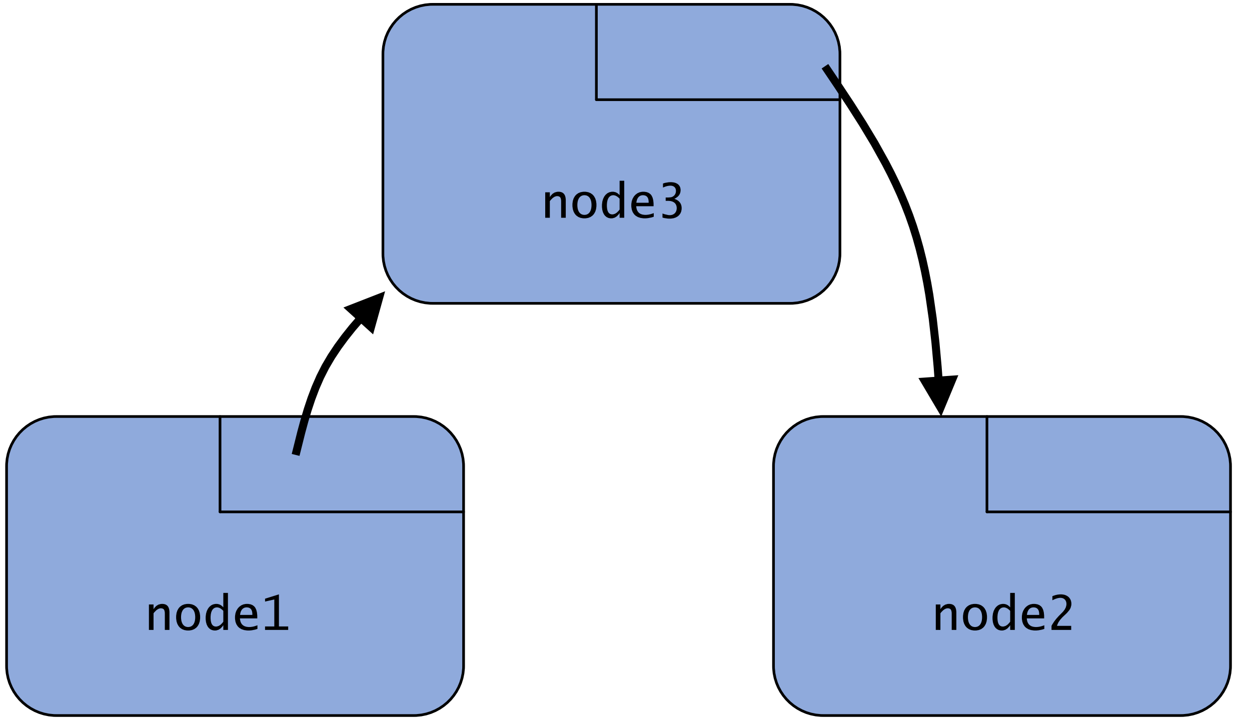


```
// Linked List insertion??
node1.setNext(node3);
node3.setNext(node2);
```

Time complexity?

```
public class Node {
    private Node next;

    private String data1;
    private int data2;

    public Node(???) {
        ???
    }

    public Node getNext() {
        return next;
    }

    public void setNext(Node next) {
        this.next = next;
    }
}
```

Linked List



```
public class Node {
    private Node next;
    private String data1;
    private int data2;
    public Node(???) {
        ???
    }
    public Node getNext() {
        return next;
    }
    public void setNext(Node next) {
        this.next = next;
    }
}
```

```
// Linked List insertion??
node1.setNext(node3);
node3.setNext(node2);
```

Time complexity? $O(1)$