So far, we have written and analysed algorithms using arrays.

An array is a kind of a "Data structure".

Did we make any assumptions about the design of the "array" data structure in analysing the algorithms? If so, what?

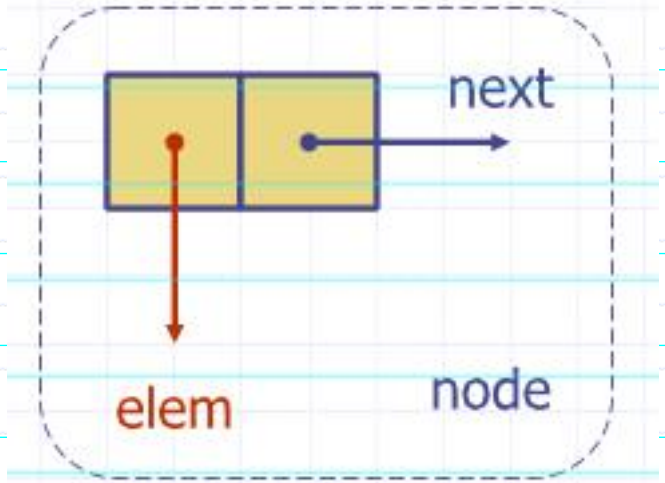
1) Every access takes same time
2) Size is known
3) Address of first element is known and can easily seek the i^{th} element

The Node Class

```
public class Node {
  // Instance variables:
  private Object element;
  private Node next;
  /** Creates a node with null references to its element and next node. */
  public Node() {
    this(null, null);
  }
  /** Creates a node with the given element and next node. */
  public Node(Object e, Node n) {
    element = e;
    next = n;
  }
  // Accessor methods:
  public Object getElement() {
    return element;
  }
  public Node getNext() {
    return next;
  }
  // Modifier methods:
  public void setElement(Object newElem) {
    element = newElem;
  }
  public void setNext(Node newNext) {
    next = newNext;
  }
}
```

- Each node stores
  - element
  - link to the next node

Q: Uses of node?
Q: Modifications of node for more uses?



(1) Write down Qs for clarification.
(2) Write down Questions & Points for exploration (what uses can I make of node)

Abstract Data Types (ADTs)

- An abstract data type (ADT) is an abstraction of a data structure
- An ADT specifies:
  - Data stored
  - Operations on the data
  - Error conditions associated with operations
- Example: ADT modeling a simple stock trading system
  - The data stored are buy/sell orders
  - The operations supported are
    - order buy(stock, shares, price)
    - order sell(stock, shares, price)
    - void cancel(order)
  - Error conditions:
    - Buy/sell a nonexistent stock
    - Cancel a nonexistent order