DATA STRUCTURES

By

Dr. Yasser Abdelhamid

الجامعة المصرية
للتعلم الإلكتروني الأهلية

EELU
THE EGYPTIAN E-LEARNING UNIVERSITY

OUTLINE

- ADT
- Lists
- Searching algorithms

RESOURCES

- http://geeksforgeeks.com/

ABSTRACT DATA TYPES (ADT)

- **Definition:**
  - Abstract Data type (ADT) is a type (or class) of objects whose state and behavior is defined by a **set of values** and a **set of operations**.
  - ADT only mentions what operations are to be performed **but not how these operations will be implemented**.
  - It **does not specify how data will be organized in memory** and what **algorithms** will be used for implementing the operations.
  - It is called "**abstract**" because it gives an **implementation-independent view**.

LISTS

- List is a data structure that stores elements in an ordered and sequential manner.
- "Ordered" in this definition means that each element has a position in the list.
- It does not mean that the list elements are sorted by value.
- A list can store repetitive elements which means a single element can occur more than once in a list.

LIST OPERATIONS

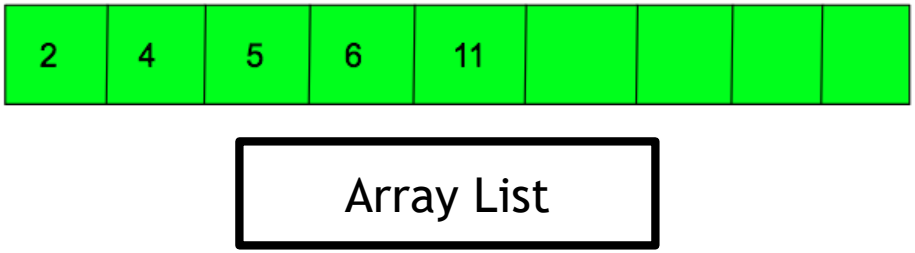
- List defines a member of functions that any list implementation inheriting from it must support, along with their parameters and return types.

LIST ADT

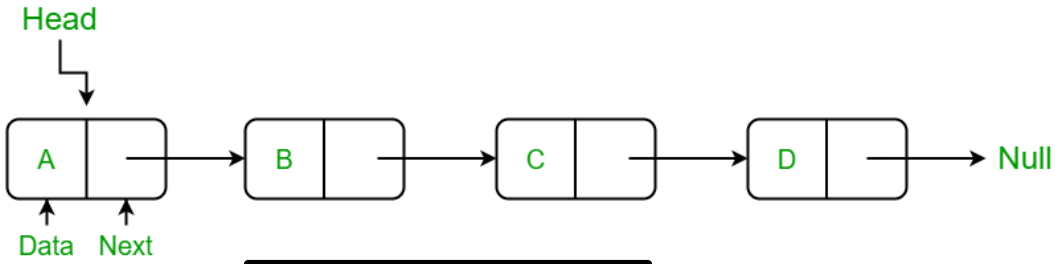
```
// List class ADT. Generalize by using "Object" for the element type.
public interface List { // List class ADT
  // Remove all contents from the list, so it is once again empty
  public void clear();

  // Insert "it" at the current location
  // The client must ensure that the list's capacity is not exceeded
  public boolean insert(Object it);

  // Append "it" at the end of the list
  // The client must ensure that the list's capacity is not exceeded
  public boolean append(Object it);

  // Remove and return the current element
  public Object remove() throws NoSuchElementException;

  // Set the current position to the start of the list
  public void moveToStart();

  // Set the current position to the end of the list
  public void moveToEnd();

  // Move the current position one step left, no change if already at beginning
  public void prev();

  // Move the current position one step right, no change if already at end
  public void next();

  // Return the number of elements in the list
  public int length();

  // Return the position of the current element
  public int currPos();

  // Set the current position to "pos“ (a specific position)
  public boolean moveToPos(int pos);

  // Return true if current position is at end of the list
  public boolean isAtEnd();

  // Return the current element
  public Object getValue() throws NoSuchElementException;

  public boolean isEmpty();
}
```

LIST REPRESENTATIONS

- There are two main representations of lists.
- Array lists.
- Linked lists.



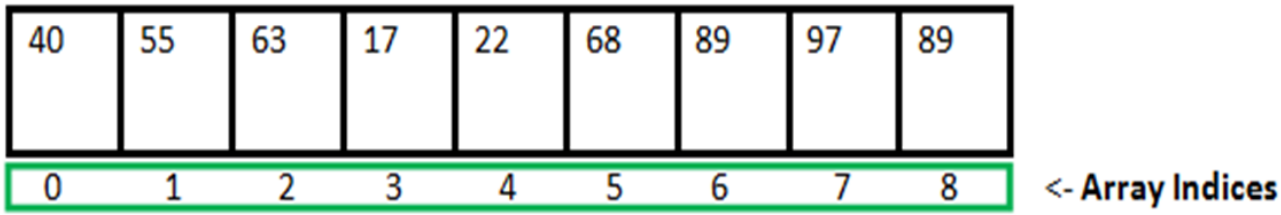
Array List



Linked List

ARRAY LISTS

- Array lists are a collection of similar data items stored at contiguous memory locations and elements can be accessed randomly using indices of an array.



Array Length = 9
First Index = 0
Last Index = 8

ADVANTAGES OF ARRAY LISTS

- Random access of list elements using array index.
- Simple declaration and manipulation.
- Traversal through the array is easy using a single loop.
- Sorting is easy as it can be implemented by less lines of code.

DISADVANTAGES OF ARRAY LISTS

- List size is fixed, and is defined at the time of declaration.
- The whole size of the list is reserved whether it is used or not.
- Insertion and deletion operations cost more time as it needs multiple shift operations.

ASSIGNMENT

- Implement the List interface methods using arrays in Java programming language.

SEARCHING ALGORITHMS

SEARCH ALGORITHMS

- Given a list of records.
- Each record has an associated key.
- We are searching for a record containing a particular key.
- Efficiency is quantified in terms of average time analysis (number of comparisons) to retrieve an item.

SEARCH ALGORITHM

- **Parameters**
  - Array containing the list
  - Length of the list
  - Item that we are searching for

- **Result**
  - If the item is found, report “success”, return location in array
  - If the item is not found, report “not found” or “failure”

LINEAR SEARCH ALGORITHM

- Start at first element of the list.
- Compare the current value to the (key) value which we are searching for.
- If the two values match, return current position with success. Otherwise, continue to the next element of the array.
- If the end of the list is reached, return -1 with failure.

LINEAR SEARCH

```
public static int linearSearch(int[] arr, int key){
    for(int i=0;i<arr.length;i++){
        if(arr[i] == key){
            return i;
        }
    }
    return -1;
}
```

PROVE THAT LINEAR SEARCH IS O(N)

According to the definition of bog-O:

T(n) is O(g(n)) if there exist two constants c, n_0 where T(n) <= c.g(n) for all values of n >= n_0

T(n) = 3n+3

To prove that T(n) is O(n) we must find two constants c, and n_0 , where c >0 and n_0 >= 0

and T(n) >= c.n for all values of n >= n_0

let c = 4

we need to calculate the value of n_0 that makes the two lines T(n), 4n intersect

that is 3n+3 = 4n.

That makes $n_0 = 3$

Assignment:

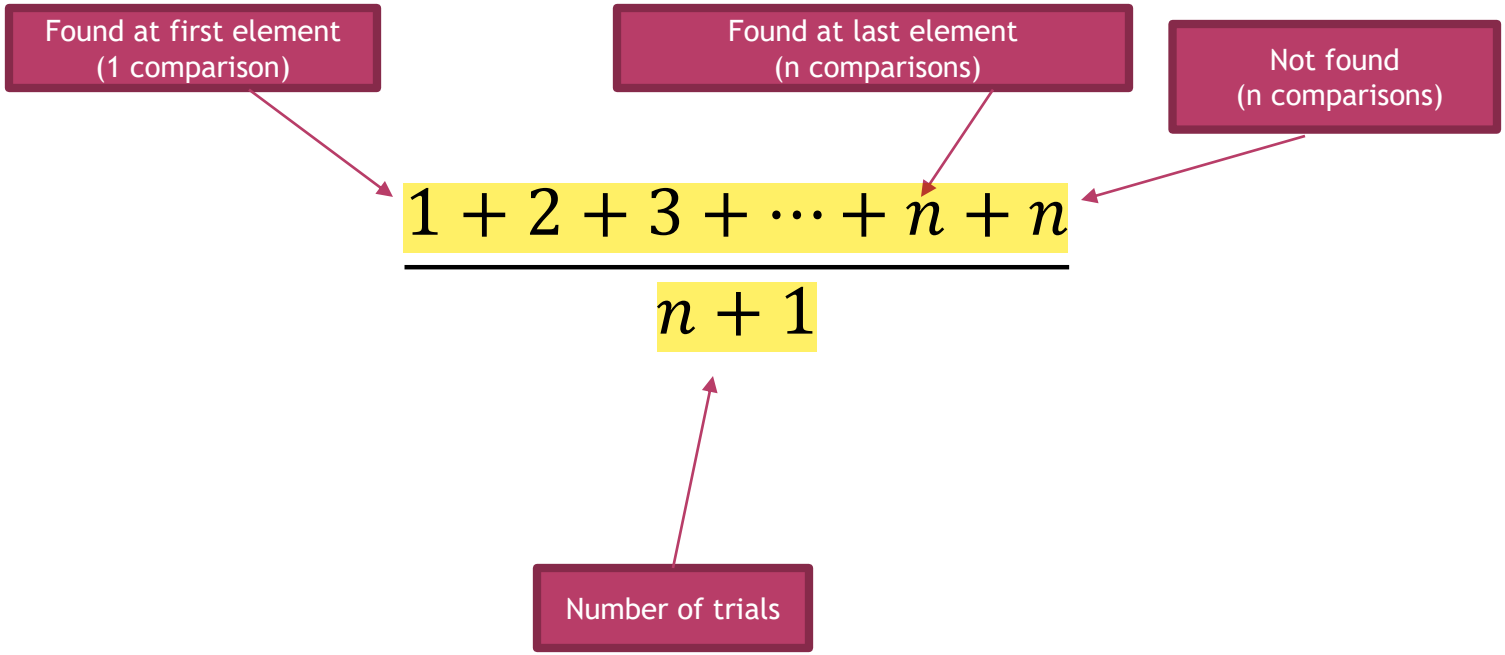
Draw the two functions T(n), c.g(n) indicating n_0

AVERAGE PERFORMANCE ANALYSIS

- Suppose that the first element in the array list contains the variable key, then we have performed one comparison to find the key.
- •Suppose that the second element in the array list contains the variable key, then we have performed two comparisons to find the key.
- Continue the analysis till the key is found in the last element of the array list. In this case, we have performed N comparisons (N is the size of the array list) to find the key.
- Finally if the key is NOT in the array list, then we would have performed N comparisons and the key is NOT found and we would return -1.

AVERAGE PERFORMANCE ANALYSIS

- **Average number of comparisons is:**



Therefore, the average case time complexity for Linear or sequential search is O(N).

THANK YOU