Arrays and Array Lists

Saikrishna Arcot
M. Hudachek-Buswell

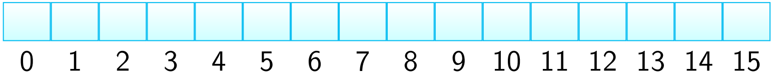
July 18, 2020

Array Definition

- An **array** is a sequenced collection of variables all of the same type. Every **cell** in an array has an **index** denoting its location within the array. The index uniquely refers to the value stored in that cell. The cells of an array, A, are numbered (or indexed) beginning with 0, 1, 2, and so on.

Array Definition

- An **array** is a sequenced collection of variables all of the same type. Every **cell** in an array has an **index** denoting its location within the array. The index uniquely refers to the value stored in that cell. The cells of an array, A, are numbered (or indexed) beginning with 0, 1, 2, and so on.
- Each value stored in an array is often called an **element** of that array.

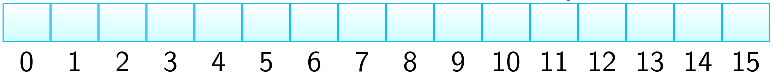


Array Length and Capacity

- Since the length of an array determines the maximum number of items that can be stored in the array, we refer to the length of an array as its **capacity**.The length of an array is established when the array is created, and the length is *fixed*.

Array Length and Capacity

- Since the length of an array determines the maximum number of items that can be stored in the array, we refer to the length of an array as its **capacity**.The length of an array is established when the array is created, and the length is *fixed*.
- In Java, the length of an array named `a` can be accessed using the syntax `a.length`. Thus, the cells of an array, a, are numbered 0, 1, 2, and so on, up through `a.length-1`, and the cell with index `k` can be accessed with syntax `a[k]`.



Array Creation

- There are a couple of ways to declare an array. One can use an array literal or an array declaration. The element type for the array is any Java base type or class type.

Array Creation

- There are a couple of ways to declare an array. One can use an array literal or an array declaration. The element type for the array is any Java base type or class type.
- Given a capacity of $N = 4$,
  Array literal: `int[] myArray = {1, 3, 3, 2}`
  Array declaration: `int[] myArray = new int[4]`

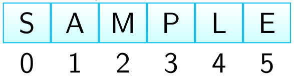
Arrays of Character or Object References

- An array can store primitive elements, such as characters.

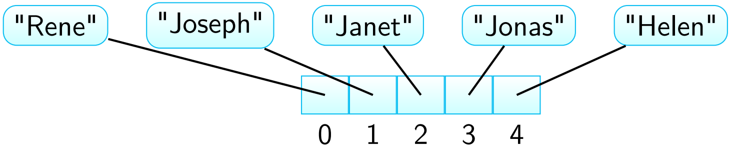
S	A	M	P	L	E
0	1	2	3	4	5

Arrays of Character or Object References

- An array can store primitive elements, such as characters.



- An array can also store references to objects.



Array Lists

- An array can be used as a backing structure for a list. In essence, array lists store objects and not primitive data types. Whereas, arrays can store both, primitives and objects.

Array Lists

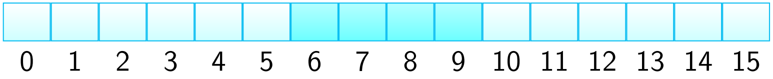
- An array can be used as a backing structure for a list. In essence, array lists store objects and not primitive data types. Whereas, arrays can store both, primitives and objects.
- When an array list is full, it dynamically resizes to a larger array list. Typically, a new, larger backing array is created and the content is copied from the old array to the new array.

Array Lists

- An array can be used as a backing structure for a list. In essence, array lists store objects and not primitive data types. Whereas, arrays can store both, primitives and objects.
- When an array list is full, it dynamically resizes to a larger array list. Typically, a new, larger backing array is created and the content is copied from the old array to the new array.
- The resizing policy depends on the implementation. Java's implementation resizes the backing array to 1.5 times the original size.

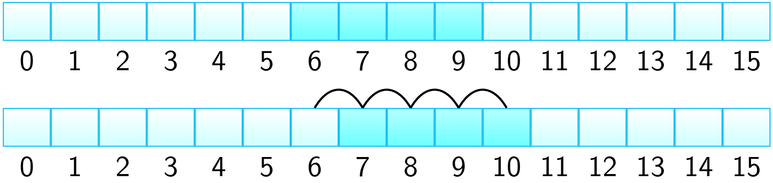
Adding an Entry

- To add an entry `e` into array `myArray` at index `i`, we need to make room for it by shifting forward the $n - i$ entries `myArray[i], ..., myArray[n - 1]`.



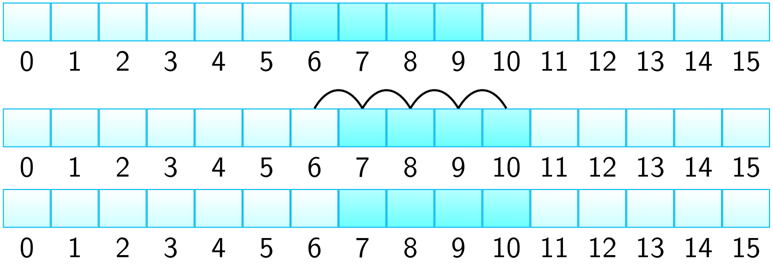
Adding an Entry

- To add an entry `e` into array `myArray` at index `i`, we need to make room for it by shifting forward the $n - i$ entries `myArray[i], ..., myArray[n - 1]`.



Adding an Entry

- To add an entry `e` into array `myArray` at index `i`, we need to make room for it by shifting forward the $n - i$ entries `myArray[i], ..., myArray[n - 1]`.



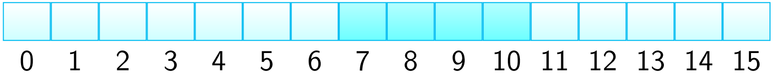
In this example, the capacity of myArray is 16 and size of myArray is n = 10. So adding at i = 6, shifts myArray elements 6-9 to elements 7-10.

Adding an Entry

```
procedure ADD(i,e)
    if $size >= arr.len$ then
        Regrow the array
    end if
    for $j \leftarrow size - 1, i$ do
        $arr[j+1] \leftarrow arr[j]$
    end for
    $arr[i] \leftarrow e$
    $size \leftarrow size + 1$
end procedure
```

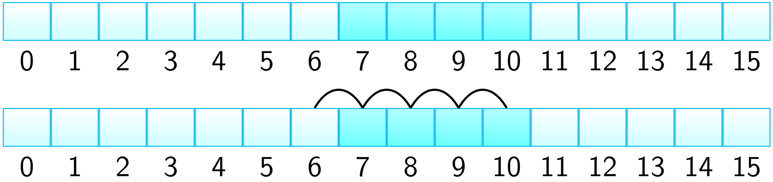
Removing an Entry

- To remove an entry `e` at index `t`, we need to fill the hole left by `e` by shifting backward the $n - i - 1$ elements `board[i + 1], ..., board[n - 1]`.



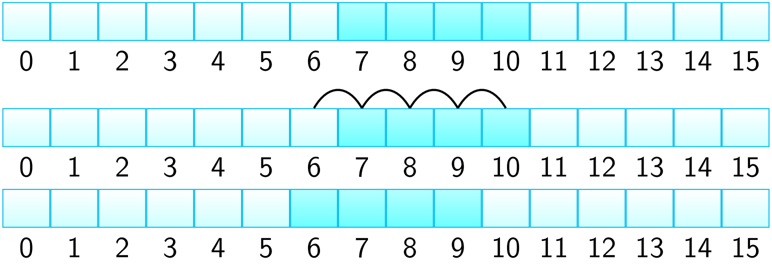
Removing an Entry

- To remove an entry `e` at index `t`, we need to fill the hole left by `e` by shifting backward the $n - i - 1$ elements `board[i + 1], ..., board[n - 1]`.



Removing an Entry

- To remove an entry `e` at index `t`, we need to fill the hole left by `e` by shifting backward the $n - i - 1$ elements `board[i + 1], ..., board[n - 1]`.



In this example, the capacity of myArray is 16 and size of myArray is n = 11. So removing at i = 6, shifts myArray elements 7-10 to elements 6-9.

Removing an Entry

```
procedure REMOVE(i)
    $item \leftarrow arr[i]$
    $arr[i] \leftarrow$ NULL
    for $j \leftarrow i, size - 2$ do
        $arr[j] \leftarrow arr[j + 1]$
    end for
    $size \leftarrow size - 1$
    return $item$
end procedure
```

Array List Summary and Complexity

- Array lists are dynamic and store objects.

Array List Summary and Complexity

- Array lists are dynamic and store objects.
- Accessing elements is a cost of O(1), constant time

Array List Summary and Complexity

- Array lists are dynamic and store objects.
- Accessing elements is a cost of O(1), constant time
- Inserting, searching or removing from anywhere other than the back of the array list is a cost of O(n), linear time

Array List Summary and Complexity

- Array lists are dynamic and store objects.
- Accessing elements is a cost of O(1), constant time
- Inserting, searching or removing from anywhere other than the back of the array list is a cost of O(n), linear time
- Array lists are used in tracking characters in an online game map