



# Data Structures

Nahian Salsabil  
(Slides adapted from Mushtari Sadia)



# Introduction

**Faculty Name:** Nahian Salsabil (NNS)

**Room:** UB80601

**Consultation Hours:**

Sunday: 11:00 AM - 1:30 PM

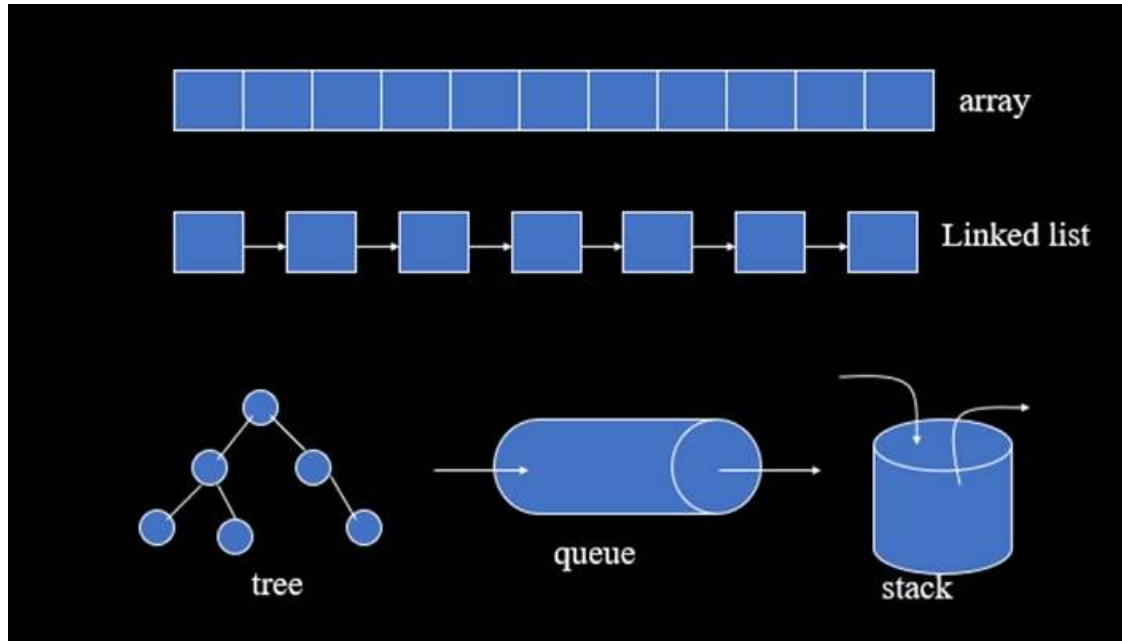
Monday: 9:30 AM - 10:50 PM

Wednesday: 9:30 AM - 10:50 PM

Slide link: <http://tiny.cc/cse220w1c2NNS>

# What is Data Structure?

Data structure is a format for **organizing** and **storing** data



# Reference Books

Sl.	Title	Author(s)	Publication Year	Edition	Publisher	ISBN
1	Algorithms in Java	Robert Sedgewick and Kevin Wayne	2011	4 <sup>th</sup> Edition	Addison-Wesley	ISBN-10: 032157351X ISBN-13: 9780321573513
2	Introduction to Algorithms	Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein	2009	3 <sup>rd</sup> Edition	MIT Press	ISBN-10: 0262033844 ISBN-13: 9780262033848

CLRS Book link:

<https://drive.google.com/file/d/1sthoRL-igfOYY55RZdyMpytPGXypLtX4/view?usp=sharing>



# Linear Arrays and Circular Arrays

Week 1



# Outline

1. Definition of an Array
2. Properties of an Array
3. Operations on an Array

# Linear Array

a

5	6	10	13	56	76	1	2	4	8
---	---	----	----	----	----	---	---	---	---



b

'a'	'b'	'c'	'd'	'e'
-----	-----	-----	-----	-----

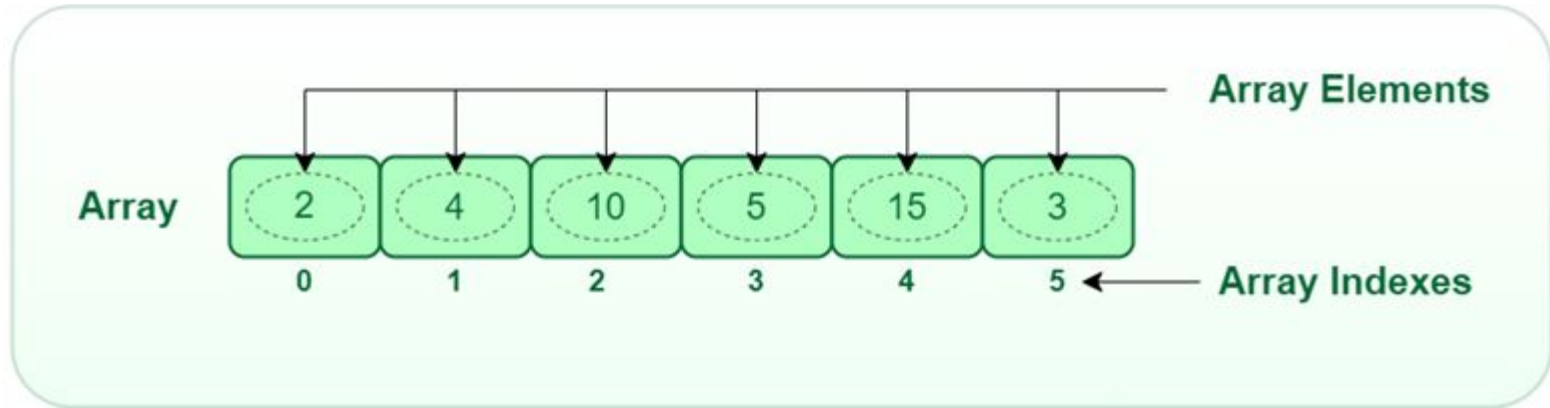


c

'a'	'b'	1	5.6	'e'	34	2	3
-----	-----	---	-----	-----	----	---	---



# Linear Array

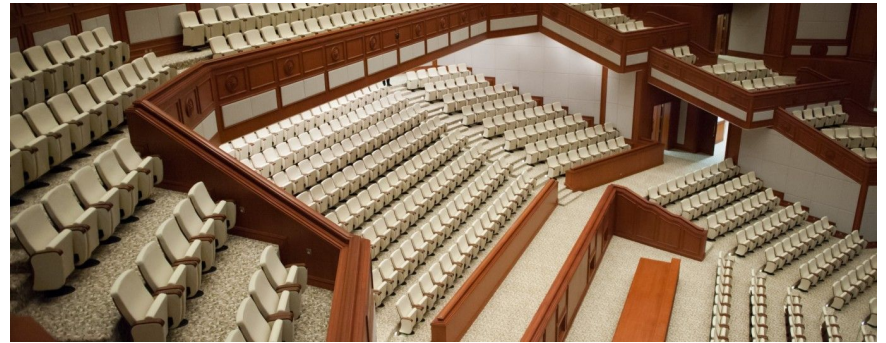




# Linear Array - Properties

1. Dimension
2. Index
3. Length
4. Single Type

# Dimension



# Dimension

1D Array

3	2
---	---

2D Array

1	0	1
3	4	1

3D Array

1	7	9
5	9	3
7	9	9

# Dimension

```
#Let's create a one-dimensional array of five elements  
array_1D = [None]*5
```

```
#Let's create a two-dimensional array of size 4x5  
array_2D = [None]*4  
for i in range(len(array_2D)):  
    array_2D[i] = [None]*5
```

```
#Let's see how to access elements
```

```
print(array_1D[3]) #We're accessing the 3rd element of the array  
print(array_2D[2][3]) #Check the 3rd element of the 2nd row
```

3rd element of the array

3rd element of 2nd row

# Index

```
#Let's create a one-dimensional array of five elements  
array_1D = [None]*5
```

**Index of an array  
starts from 0;  
So from now on,  
always start counting  
from 0!**

```
#Let's create a two-dimensional array of size 4x5  
array_2D = [None]*4  
for i in range(len(array_2D)):  
    array_2D[i] = [None]*5
```

```
#Let's see how to access elements
```

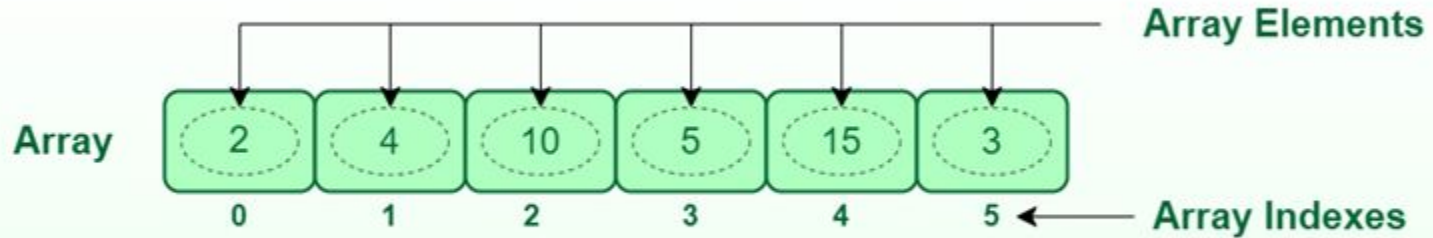
```
print(array_1D[3])  
print(array_2D[2][3])
```

3rd element of the array

3rd element of 2nd row

# Index

**Index of an array starts from 0;  
So from now on,  
always start counting  
from 0!**



# Length

When you create an array in the program, you have to specify the length of each of its dimensions that **you cannot change later**

# Length

```
1  # Let's look at the length property clearly
2  array = [None]*10
3  #We just declared an array of 10 length. Let's fill it up with something.
4
5  Names = ['Emon','Nusrat','Joy','Abrar','Ifty','Ashik','Raisa','Suba','Sazid','Jawad']
6  for i in range(10):
7      |   array[i] = Names[i]
8  print(array)
```

```
['Emon', 'Nusrat', 'Joy', 'Abrar', 'Ifty', 'Ashik', 'Raisa', 'Suba', 'Sazid', 'Jawad']
```



# Length

```
[1] 1 print(array[1])
```

Nusrat

```
[13] 1 print(array[11])
```

```
-----  
IndexError                                Traceback (most recent call last)  
<ipython-input-13-92562eb7d277> in <cell line: 1>()  
----> 1 print(array[11])
```

IndexError: list index out of range

SEARCH STACK OVERFLOW

```
[14] 1 print(array[0])
```

Emon

```
1 print(array[10])
```

```
-----  
IndexError                                Traceback (most recent call last)  
<ipython-input-15-a0db80d8cb8e> in <cell line: 1>()  
----> 1 print(array[10])
```

# Operations On Array - Read and Write

## Operations on Array

The most common operations that you do on an array are writing/reading a data item to/from an index. Suppose you have a 1D array named *student\_names*. If you want get the 5<sup>th</sup> student from the array then you write:

```
fifth_student = student_names[5]
```

To change the value at 5<sup>th</sup> index and set it to 'John Doe', you write:

```
student_names[5] = 'John Doe'
```

# Single Type

- An array can hold elements of only a single type. That is, you cannot mix strings with numbers in an array – not even mix integer numbers with fractional numbers.

# Single Type

a

5	6	10	13	56	76	1	2	4	8
---	---	----	----	----	----	---	---	---	---

b

'a'	'b'	'c'	'd'	'e'
-----	-----	-----	-----	-----

c

'a'	'b'	1	5.6	'e'	34	2	3
-----	-----	---	-----	-----	----	---	---

# Single Type

a

5	6	10	13	56	76	1	2	4	8
---	---	----	----	----	----	---	---	---	---



b

'a'	'b'	'c'	'd'	'e'
-----	-----	-----	-----	-----



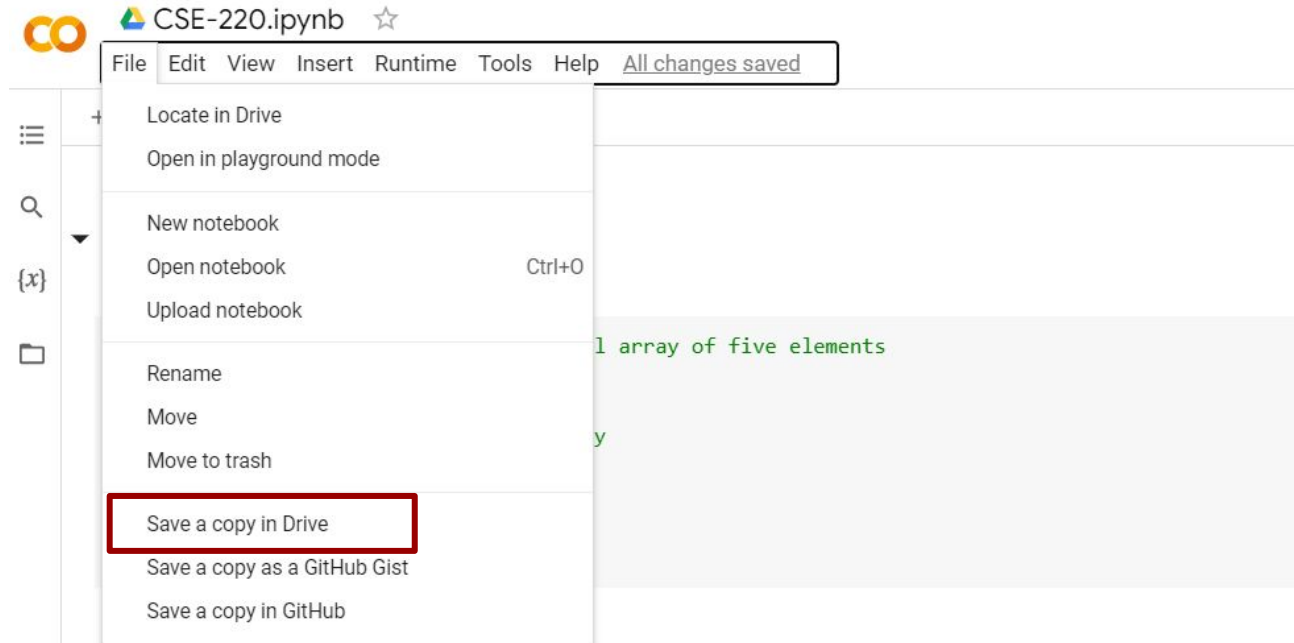
c

'a'	'b'	1	5.6	'e'	34	2	3
-----	-----	---	-----	-----	----	---	---



# Practice

**Notebook Link:** <http://tiny.cc/cse220nb>



# Functions on array

**Iteration:** Iteration refers to checking all the values index by index. The main idea is to go to that memory location and check all the values index by index.

```
1 def iteration(source):  
2     for i in range(len(source)):  
3         print(source[i])  
4  
5 def reverseIteration(source):  
6     for i in range(len(source) - 1, -1, -1):  
7         print(source[i])
```

# Functions on array

**Copy Array:** Copy array means you initialize a new array with the same length as the given array to copy and then copy the old array's value by value. As the variable where we store the array only stores the memory location, only copying the value is not enough for array copying. For example, if you have an array titled `arr = [1, 2, 3, 4]` and write `a2 = arr`. It does not mean you have copied `arr` to `a2`.

```
1 def copyArray(source):  
2     newArray = [None] * len(source)  
3     for i in range(len(source)):  
4         newArray[i] = source[i]  
5     return newArray
```



# Functions on array

**Resize Array:** We can not resize an array because it has a fixed memory location. However, if we ever need to resize an array, we need to create a new array with a new length and then copy the values from the original array. For example, if we have an array [10, 20, 30, 40, 50] whose length is 5 and want to resize the array with length 8. The new array will be [10, 20, 30, 40, 50, None, None, None].

```
1 def resizeArray(oldArray, newCapacity):  
2     newArray = [None] * newCapacity  
3     for i in range(len(oldArray)):  
4         newArray[i] = oldArray[i]  
5     return newArray
```

# Functions on array

**Reversing an Array:** Reversing an array can be implemented in two ways. First, we will create a new array with the same size as the original array and then copy the values in reverse order. The method is called out-of-the-place operation.

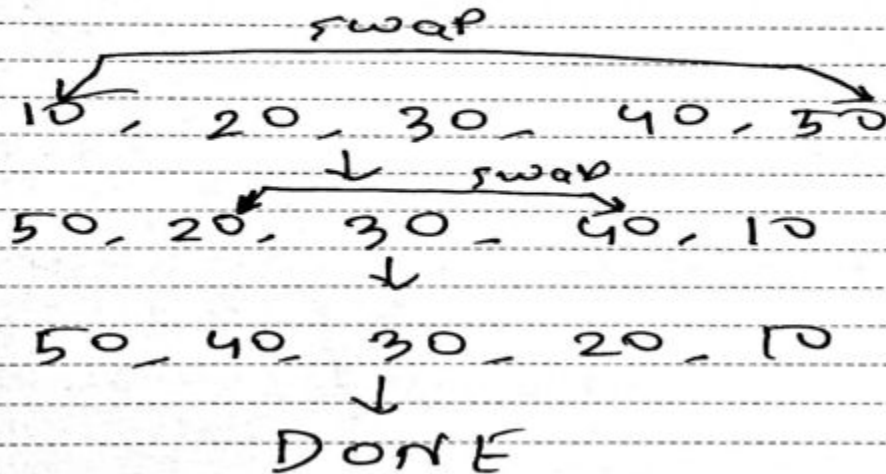
```
1 def reverseArrayOutOfPlace(arr):
2     revArr = [None] * len(arr)
3     i = 0
4     j = len(arr) - 1
5     while i < len(arr):
6         revArr[i] = arr[j]
7         i += 1
8         j -= 1
9     return revArr
```

# Functions on array

However, an efficient approach might be to reverse the array in the original array. By this, we will not need to allocate extra spaces. This is known as an in-place operation. To do so we need to start swapping values from the beginning position to the end position. The idea is to swap starting value with the end value, then the second value with the second last value, and so on.

# Functions on array

In place Reverse:



# Functions on array

```
1 def revArrInPlace(arr):  
2     i = 0  
3     j = len(arr) - 1  
4     while i < j:  
5         temp = arr[i]  
6         arr[i] = arr[j]  
7         arr[j] = temp  
8         i += 1  
9         j -= 1
```

# Functions on array

**Shifting an Array Left:** Shifting an entire array left moves each element one (or more, depending how the shift amount) position to the left. Obviously, the first element in the array will fall off at the beginning and be lost forever. The last slot of the array before the shift (ie. the slot where the last element was until the shift) is now unused (we can put a None there to signify that). The size of the array remains the same however because the assumption is that you would something in the now-unused slot. For example, shifting the array [5, 3, 9, 13, 2] left by one position will result in the array [3, 9, 13, 2, None]. Note how the array[0] element with the value of 5 is now lost, and there is an empty slot at the end.

# Functions on array

```
1 def shiftLeft(arr):  
2     for i in range(1, len(arr)):  
3         arr[i-1] = arr[i]  
4     arr[len(arr) - 1] = None  
5     return arr
```



# Functions on array

**Shifting an Array Right:** Shifting an entire array right moves each element one (or more, depending how the shift amount) position to the right. Obviously, the last element in the array will fall off at the end and be lost forever. The first slot of the array before the shift (ie., the slot where the first element was until the shift) is now unused (we can put a None there to signify that). The size of the array remains the same however because the assumption is that you would something in the now-unused slot. For example, shifting the array [5, 3, 9, 13, 2] right by one position will result in the array [None, 5, 3, 9, 13]. Note how the `array[4]` element with the value of 2 is now lost, and there is an empty slot at the beginning.



# Functions on array

```
1 def shiftRight(arr):  
2     for i in range(len(arr) - 1, 0, -1):  
3         arr[i] = arr[i - 1]  
4     arr[0] = None  
5     return arr
```

# Functions on array

**Rotating an Array Left:** Rotating an array left is equivalent to shifting a circular or cyclic array left where the 1st element will not be lost, but rather move to the last slot. Rotating the array [5, 3, 9, 13, 2] left by one position will result in the array [3, 9, 13, 2, 5].

# Functions on array

```
1 def rotateLeft(arr):  
2     temp = arr[0]  
3     for i in range(1, len(arr)):  
4         arr[i-1] = arr[i]  
5     arr[len(arr) - 1] = temp  
6     return arr
```

# Functions on array

**Rotating an Array Right:** Rotating an array right is equivalent to shifting a circular or cyclic array right where the last element will not be lost, but rather move to the 1st slot. Rotating the array [5, 3, 9, 13, 2] right by one position will result in the array [2, 5, 3, 9, 13].

# Functions on array

```
1 def rotateRight(arr):  
2     temp = arr[len(arr) - 1]  
3     for i in range(len(arr) - 1, 0, -1):  
4         arr[i] = arr[i - 1]  
5     arr[0] = temp  
6     return arr
```

# Functions on array

**Inserting an element into an Array:** To insert an element into the array we need to make an empty slot first and then insert the value in the array. To make an empty slot first we need to check if any slots are available or not. If slots are not available then we need to resize the array using the concept of array resizing mentioned above. After that, we will use the idea of the right shift so that we can have an empty slot. The initialization point of the right shift would be the index where we want to insert the value. After the right shift operation, we will insert the value in that corresponding index.

For example, we have an array consisting of values [5, 7, 3, 6, None, None] (None represents the empty slots). Now if we want to insert 10 at index 1 we will need to use the right shift starting from index 1. So our array will be like [5, None, 7, 3, 6, None]. Here we can use the index 1 and insert the value 10 and the final array would be [5, 10, 7, 3, 6, None]

# Functions on array

```
1 def insertElement(arr, size, elem, index):
2     # Practice how to throw exception if there is no empty space
3     if size == len(arr):
4         print("No space left. Insertion failed")
5     else:
6         for i in range(size, index, -1):
7             arr[i] = arr[i - 1] #Shifting right till the index
8         arr[index] = elem #Inserting element
9         return arr
```



# Functions on array

**Removing an element from the Array:** We need to use the concept of left shift to remove an element from the array. The idea is to start the left shift from the index which value we want to remove.

For example, we have an array consisting of values [10, 6, 8, 11, 15, 20] and want to remove value 8 which is at index 2, we need to do the left shift operation starting from index 2. So the resulting array would be [10, 6, 11, 15, 20, None].



# Functions on array

```
1 def removeElement(arr, index, size):  
2     for i in range(index + 1, size):  
3         arr[i - 1] = arr[i] #Shifting left from removing index  
4     arr[size - 1] = None #Making last space empty
```