Data Structures

lecture 5

28-9-2022

Unit 1: Basics of Data Structures

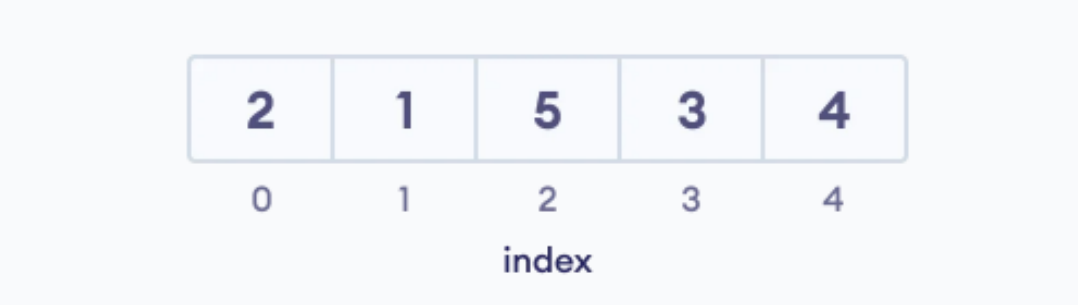
Types of Data Structures

- Basically, data structures are divided into two categories:
  - Linear data structure
  - Non-linear data structure

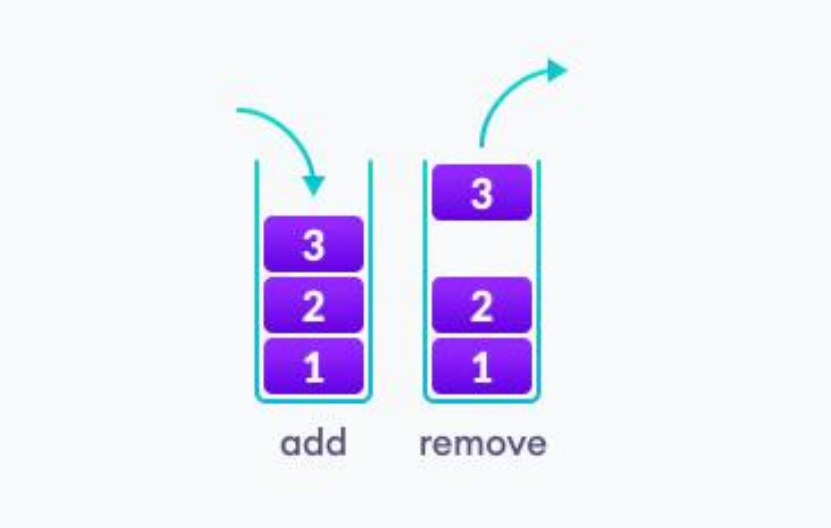
Linear Data Structures

- Elements are arranged sequentially
- every element is attached to its previous and next adjacent element
- Example:
  - Array
  - Stack
  - Queue
  - Linked List

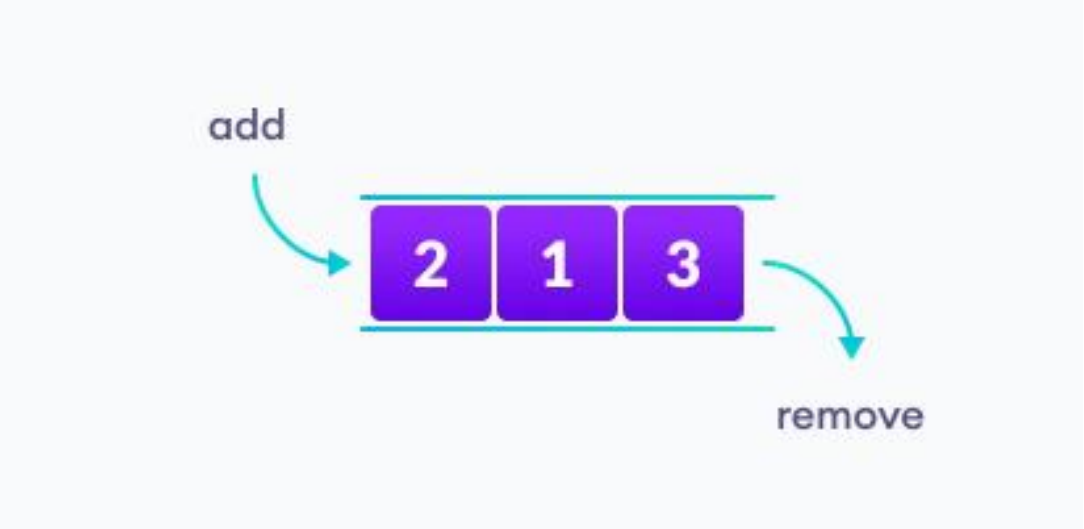
Array



Stack



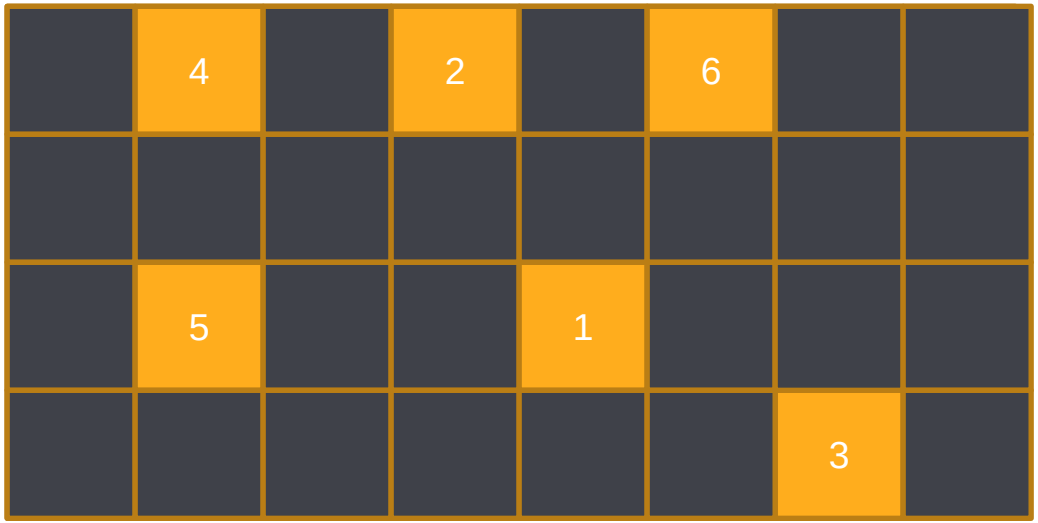
Queue



Linked List

Need of Linked List

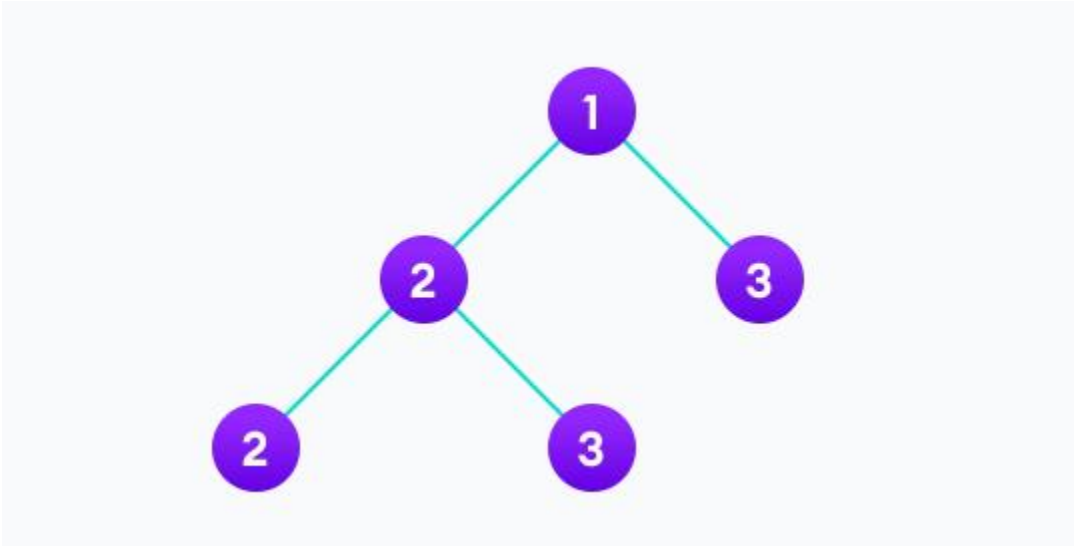
- Data: 4,2,6,5,1,3



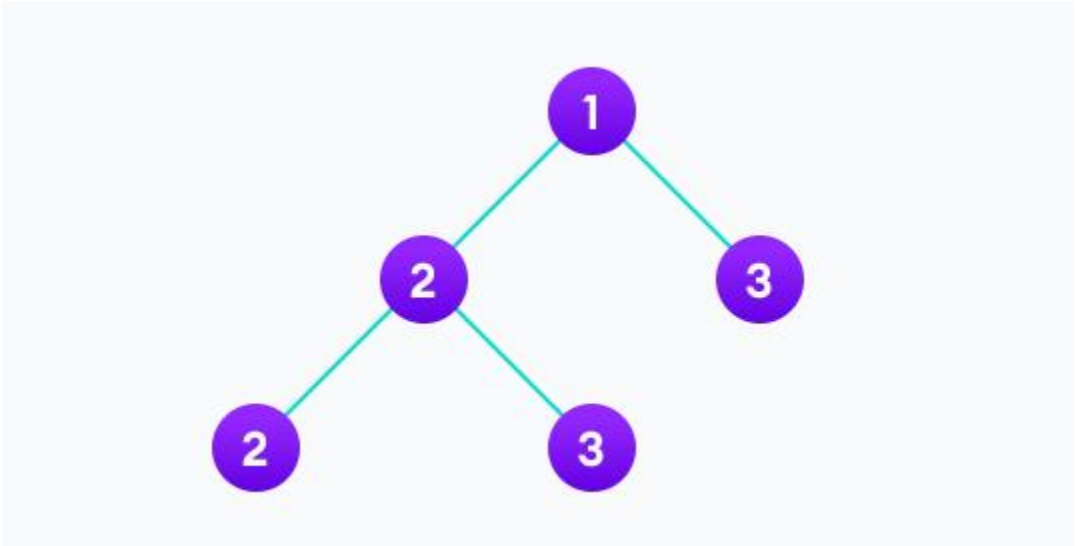
Non-Linear Data Structures

- Elements are not arranged sequentially.
- Every element is attached to more than one other elements.
- Example:
  - Tree
  - Graph

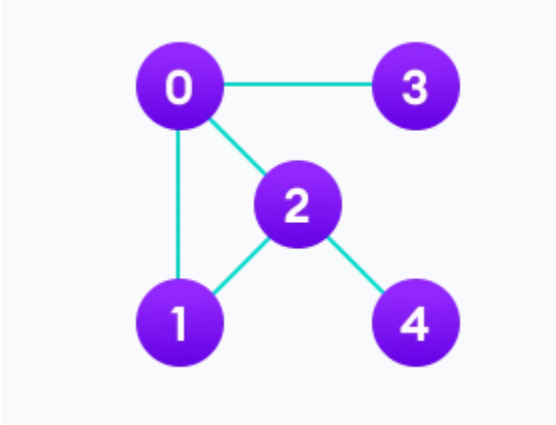
Tree



Tree



Graph



Quick Compare

Linear Data Structures	Non Linear Data Structures
arranged in sequential order, one after the other.	arranged in non-sequential order (hierarchical manner).
It can be traversed on a single run. That is, if we start from the first element, we can traverse all the elements sequentially in a single pass.	It requires multiple runs. That is, if we start from the first element it might not be possible to traverse all the elements in a single pass.
The memory utilization is not efficient.	Different structures utilize memory in different efficient ways depending on the need.
The time complexity increase with the data size.	Time complexity remains the same.
Example: Arrays, Stack, Queue etc.	Example: Tree, Graph

Algorithm efficiency.

- There are more than one solutions (algorithms) to any problem
- We need to choose most efficient one.
- Major factor affecting efficiency is repetitive operations
  - Recursion / Loops

- Therefore Loops are very important in efficiency calculation

Efficiency

- Efficiency => mathematical function of the number of elements to be processed in loop

  - f(n) = efficiency

Linear Loops

- How many times following code will execute

```
for (i = 0; i < 1000; i++)
    {
        // Some code
    }
```

Linear Loops

- Answer: 1000
- The number of iterations is directly proportional to the loop factor, 1000
- The higher the factor, the higher the number of loops.
- Because the efficiency is directly proportional to the number of iterations
  - $f(n) = n$

Linear Loops

- Some times it is not straight forward

```
for (i = 0; i < 1000; i+=2)
    {
        // Some code
    }
```

```
for (i = 1000; i>=0; i--)
    {
        // Some code
    }
```

```
for (i = 1000; i>=0; i-=2)
    {
        // Some code
    }
```

Logarithmic Loops

- How many times following code will execute

```
for (i = 0; i < 1000; i*=2)
    {
        // Some code
    }
```

Multiply Loops

```
for (i = 1000; i >= 1; i/=2)
    {
        // Some code
    }
```

divide Loops

multiply $2^{Iterations} < 1000$

divide $1000 / 2^{Iterations} >= 1$

Logarithmic Loops

Multiply		Divide	
Iteration	Value of `i`	Iteration	Value of `i`
1	1	1	1000
2	2	2	500
3	4	3	250
4	8	4	125
5	16	5	62
6	32	6	31
7	64	7	15
8	128	8	7
9	256	9	3
10	512	10	1
(exit)	1024	(exit)	0

Logarithmic Loops

- $f(n) = \log n$

Nested Loops : Quadratic Loop

- How many times loop will execute

```
for (i = 0; i < 5; i++)
    {
        for (j = 0; j < 5; j++)
            {
                //application code
            }
    }
```

Nested Loop: Quadratic

$$f(n) = n^2$$

Nested Loops

- Generalization

```
for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j *= 2)
        {
            //application code
        }
    }
```

m * n

Nested Loop: Linear Logarithmic

```
for (i = 0; i < 10; i++)
    {
        for (j = 0; j < 10; j *= 2)
        {
            //application code
        }
    }
```

10log10

Efficiency	Big-O	Iterations	Estimated Time
Logarithmic	$O(\log n)$	14	microseconds
Linear	$O(n)$	10,000	seconds
Linear logarithmic	$O(n(\log n))$	140,000	seconds
Quadratic	$O(n^2)$	$10{,}000^2$	minutes
Polynomial	$O(n^k)$	$10{,}000^k$	hours