ISIT307 - WEB SERVER PROGRAMMING

LECTURE 7.2 – RECURSION AND DATA STRUCTURES IN PHP

LECTURE PLAN

- Recursion
- Data structures

Sources:
Rahman, M., 2017. PHP 7 Data Structures and Algorithms. Packt Publishing Ltd.
Gilberg R., Forouzan B, 2004, Data Structures : A Pseudocode Approach with C. Cengage Learning

RECURSION

- Recursion occurs when something contains, or uses, a similar version of itself
  - That similar version then contains or uses another similar version of itself, and so on…
- The number of repetitions, or "depth" of the recursion, is limited by some sort of end condition
- In programming recursion occurs when a function calls itself
  - end condition - base case
  - reduction step – function call itself

RECURSION

- A good recursive method :
  - begins with a conditional statement with a return – without base case the recursion will run indefinitely
  - recursive calls to sub-problems converge to the base case
  - called sub-problems should not overlap
- *every call creates a new instance of the function

```
function myRecursiveFunction() {
      // (do the required processing...)
  if ( baseCaseReached ) {
      // end the recursion
    return;
  } else {
       // continue the recursion
    myRecursiveFunction();
}
```

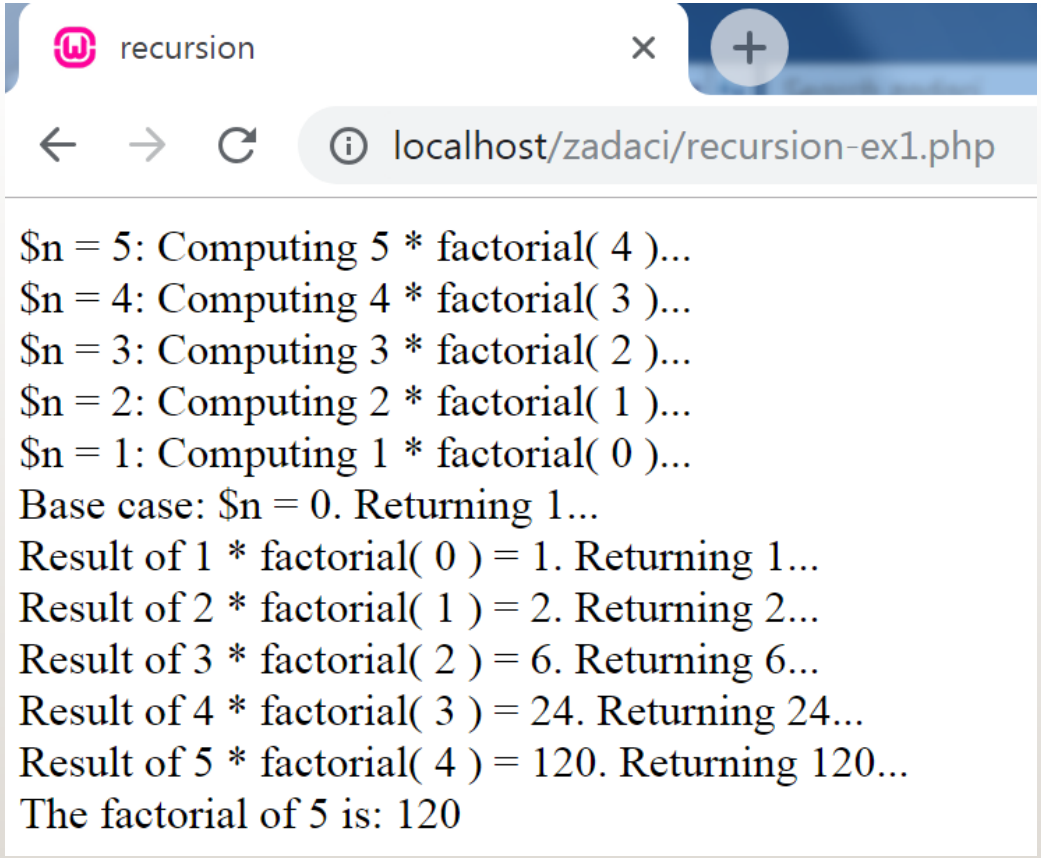
RECURSION

- Some cases when is useful to use recursion:
  - When processing recursively defined data
  - When you have nested structures

RECURSION – EXAMPLE (1) FACTORIAL

```
<?php
function factorial( $n ) {
  if ( $n == 0 ) {
    echo "Base case: \$n = 0. Returning 1...<br>";
    return 1;
  } else {
     echo "\$n = $n: Computing $n * factorial( " .
                                        ($n-1) . " )...<br>";
     $result = ( $n * factorial( $n-1 ) );
     echo "Result of $n * factorial( " . ($n-1) .
                   " ) = $result. Returning $result...<br>";
     return $result;
   }
}
echo "The factorial of 5 is: " . factorial( 5 );
?>
```

RECURSION – EXAMPLE (1) OUTPUT FACTORIAL



RECURSION – EXAMPLE (2) DISPLAYING TREE OF FILES AND FOLDERS

```
<?php
$folderPath = "C://wamp64/www";
function readFolder( $path ) {
  if ( !( $dir = opendir( $path ) ) )
            die( "Can't open $path" );
  $filenames = array();
  while ( $filename = readdir( $dir ) ) {
    if ( $filename != '.' && $filename != '..' ) {
      if ( is_dir( "$path/$filename" ) )
            $filename .= '/';
      $filenames[] = $filename;
    }
  }
  closedir ( $dir );
```

RECURSION – EXAMPLE (2) DISPLAYING TREE OF FILES AND FOLDERS

```
// Display the filenames, and process any subfolders
  echo "<ul>";
  foreach ( $filenames as $filename ) {
    echo "<li>$filename";

    if ( substr( $filename, -1 ) == '/' )
      readFolder( "$path/" . substr( $filename, 0, -1 ) );
      echo "</li>";
  }
  echo "</ul>";
}
echo "<h2>Contents of '$folderPath':</h2>";
readFolder( $folderPath );
?>
```

RECURSION – EXAMPLE (2) OUTPUT DISPLAYING TREE OF FILES AND FOLDERS

- FileDownloader.php
- PHPCodeBlocks.php
- ViewFiles.php
- backup/
  - Comment.1545915450.4533.txt
  - Comment.1545915465.3897.txt
  - Comment.1545915491.6001.txt
  - Comment.1545954019.5635.txt
  - Comment.1545954031.5897.txt
  - Comment.1545954091.3807.txt
  - Comment.1545954098.7962.txt
  - Comment.1545954276.091.txt
  - Comment.1545955214.9985.txt
  - Comment.1545955221.8813.txt
- comments/
  - Comment.1545915450.4533.txt
  - Comment.1545915465.3897.txt
  - Comment.1545915491.6001.txt
  - Comment.1545954019.5635.txt
  - Comment.1545954031.5897.txt
  - Comment.1545954091.3807.txt
  - Comment.1545954098.7962.txt
  - Comment.1545954276.091.txt
  - Comment.1545955214.9985.txt
  - Comment.1545955221.8813.txt
- createDatabase.php
- createTable.php
- example-ch10/
  - ElectronicsBoutique.css

RECURSION AND STATIC VARIABLES

- A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope
- Static variables provide one way to deal with recursive functions

RECURSION AND STATIC VARIABLES (EXAMPLE)

```
<?php
function test()
{
    static $step= 0;
    if ($step< 10) {
          $step++;
          echo "<p>into function step = $step</p>";
          test();
     }
     else {
          echo "finish";
     }
}
test();
echo "<p>out of function";
?>
```

into function step = 1

into function step = 2

into function step = 3

into function step = 4

into function step = 5

into function step = 6

into function step = 7

into function step = 8

into function step = 9

into function step = 10

finish

out of function

DATA STRUCTURES

- Data structures are very important components for computers and programming languages
- Majority of the data structures are inspired from real life, so can be used to solve a real-life problems or find a solutions
- For example: find the shortest way; diet charting; preparing a family tree; or organization hierarchy

DATA STRUCTURES

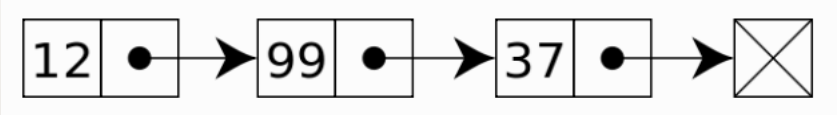
- PHP is a weakly typed language and has eight primitive data types (boolean, integer, float, string, array, object, resource, and null)
- The primitive data types have one particular objective - storing data
- In order to achieve some flexibility in performing operations on those data, the data types can be used to represent particular model and perform some operations

DATA STRUCTURES

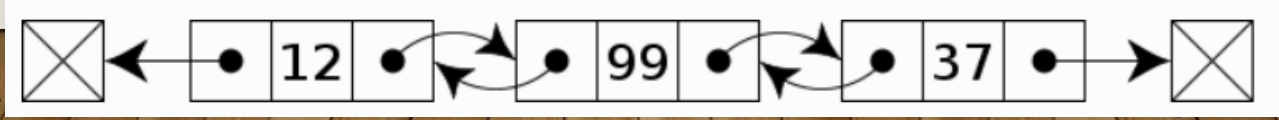
- This particular way of handling data through a conceptual model is known as Abstract Data Type (ADT)
- Data structures are concrete representations
- Some common ADT:
  - List
  - Map
  - Set
  - Stack
  - Queue
  - Priority queue
  - Graph
  - Tree

DATA STRUCTURES

- A linked list is a linear data structure which is a collection of data elements also known as nodes
- Listed items are connected through a pointer which is known as a link - linked list

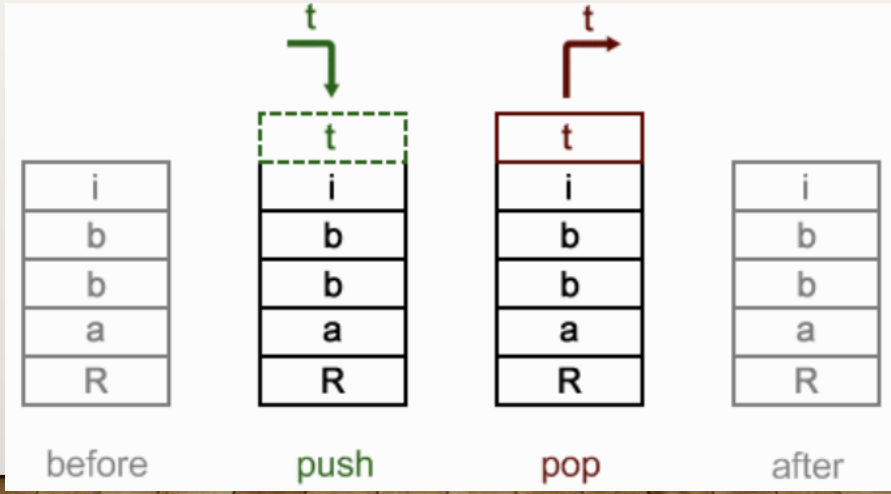


- A doubly linked list is a special type of linked list where not only previous node is connected to the next node, but reverse link apply as well
- As a result, it can move forward and backward within the list



DATA STRUCTURES

- A stack is a linear data structure with the LIFO principle
- Stacks have only one end to add a new item or remove an item
- It is one of the oldest and most used data structures in computer technology

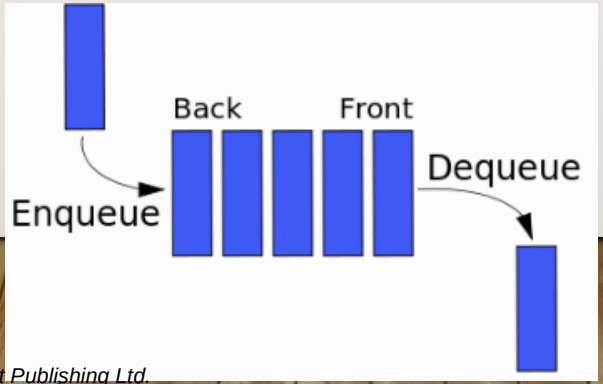


STACK - SAMPLE IMPLEMENTATION

```
class Stack
{
    protected $stack;
    protected $top;

    public function __construct() {
       $this->stack = array();
       $this->top = -1;
    }

    public function push($item) { . . . }

    public function pop() { . . . }

    public function top() { . . . }

    public function isEmpty() { . . . }
}
```

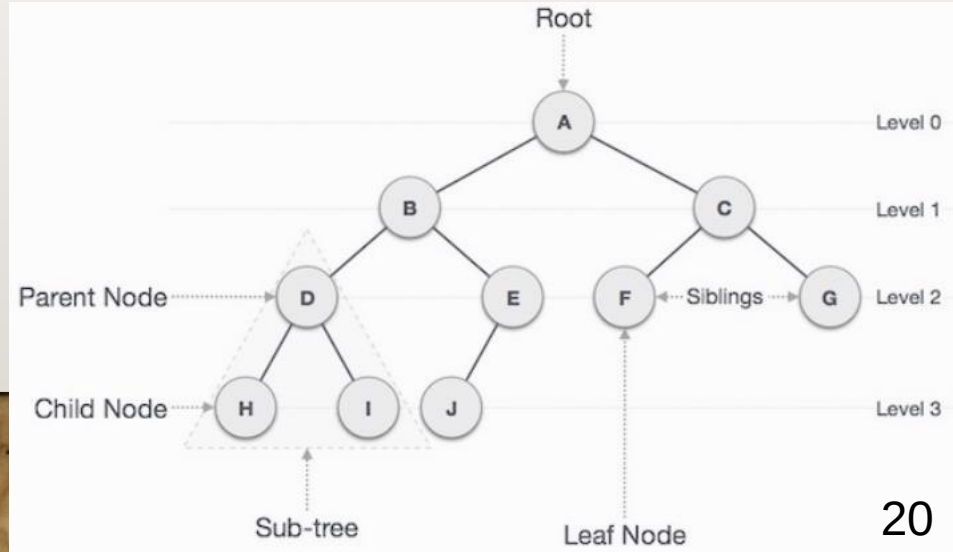
DATA STRUCTURES

- A queue is another linear data structure which follows the FIFO principle
- A queue allows two basic operations on the collection
  - **enqueue** - allows us to add an item to the back of the queue
  - **dequeue** - allows us to remove an item from the front of the queue
- A queue is another of the most used data structures in computer technology



DATA STRUCTURES

- A tree is the most widely used nonlinear data structure in the computing world
  - It is highly used for hierarchical data structures
- A tree consists of nodes and there is a special node which is known as the root of the tree which starts the tree structure
  - Other nodes descend from the root node
- Tree data structure is recursive which means a tree can contain many subtrees
- Nodes are connected with each other through edges



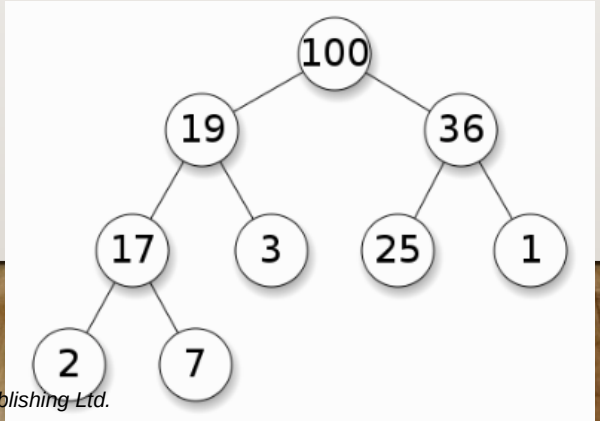
BINARY TREE – SAMPLE IMPLEMENTATION

```
class BinaryNode
{
    public $value;
    public $left;
    public $right;

    public function __construct($item) {
        $this->value = $item;
        $this->left = null;  // new nodes are leaf nodes
        $this->right = null;
    }

    public function addChildren($left, $right) {
        $this->left = $left;
        $this->right = $right;
    }
}
- - -
class BinaryTree
{
    . . .
}
```

Source: https://www.sitepoint.com/data-structures-2/

DATA STRUCTURES

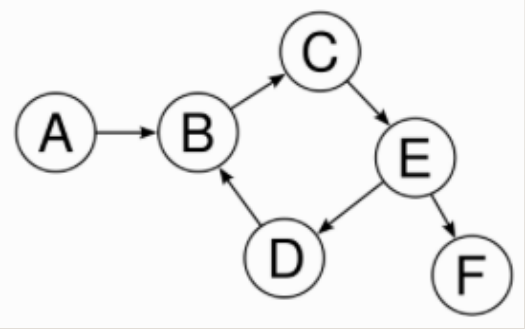
- A heap is a special tree-based data structure which satisfies the heap properties
  - If the largest key is the root and smaller keys are leaves – it is known as max heap
  - If the smallest key is the root and larger keys are leaves – it is known as min heap
- Though the root of a heap structure is either the largest or smallest key of the tree, it is not necessarily a sorted structure



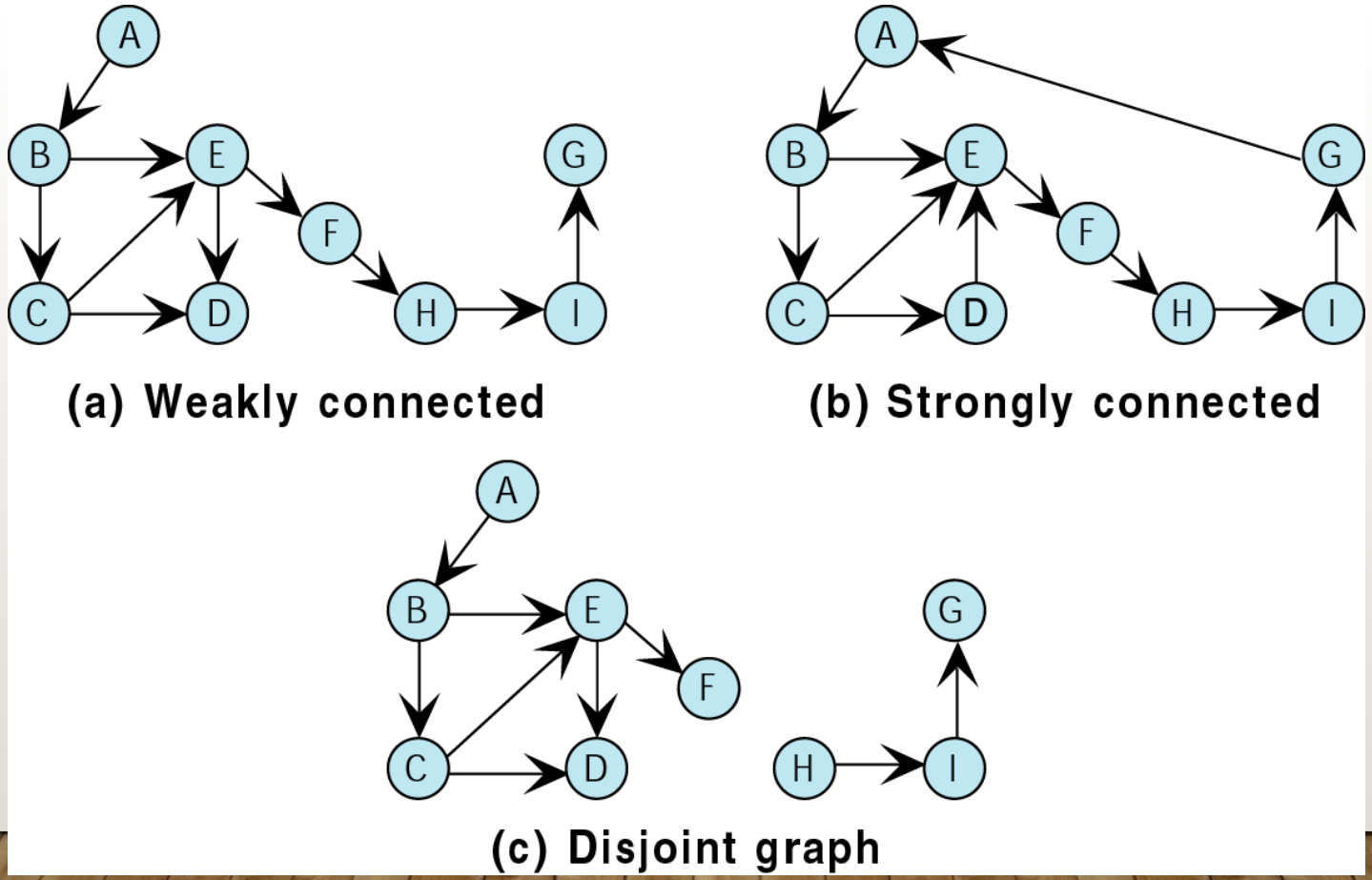
PHP 7 Data Structures and Algorithms, Mizanur Rahman, 2017, Packt Publishing Ltd.

DATA STRUCTURES

- A graph data structure is a special type of nonlinear data structure which consists of a finite number of vertices or nodes, and edges or arcs
- A graph can be directed or undirected
  - a directed graph clearly indicates the direction of the edges
  - an undirected graph mentions the edges, not the direction



DATA STRUCTURES - GRAPH

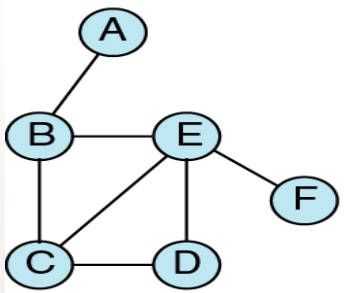


Gilberg R., Forouzan B, 2004, Data Structures : A Pseudocode Approach with C. Cengage Learning

DATA STRUCTURES - GRAPH

- To represent a graph, we need to store two sets
  - the first set represents the vertices (nodes) of the graph and
  - the second set represents the edges (arcs)
- Two methods can be used:
  - Adjacency matrix
  - Adjacency list

DATA STRUCTURES - GRAPH

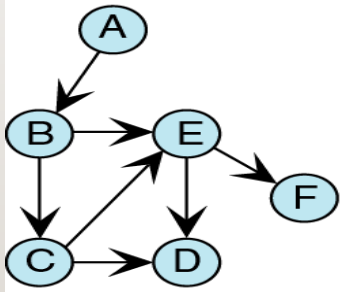


Vertex vector
A
B
C
D
E
F

Adjacency matrix

	A	B	C	D	E	F
A	0	1	0	0	0	0
B	1	0	1	0	1	0
C	0	1	0	1	1	0
D	0	0	1	0	1	0
E	0	1	1	1	0	1
F	0	0	0	0	1	0

Adjacency matrix for non-directed graph



Vertex vector
A
B
C
D
E
F

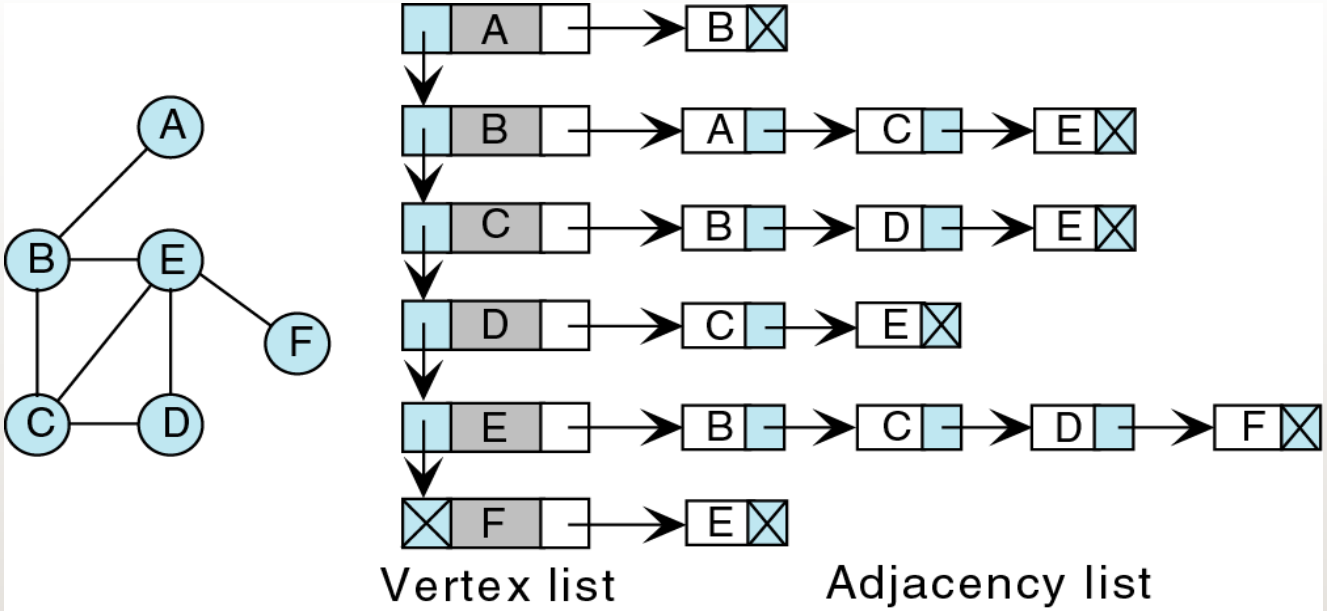
Adjacency matrix

	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	1	0	1	0
C	0	0	0	1	1	0
D	0	0	0	0	0	0
E	0	0	0	1	0	1
F	0	0	0	0	0	0

Adjacency matrix for directed graph

Gilberg R., Forouzan B, 2004, Data Structures : A Pseudocode Approach with C. Cengage Learning

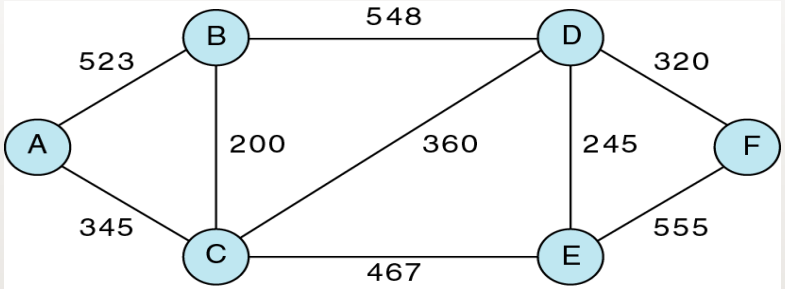
DATA STRUCTURES - GRAPH



Gilberg R., Forouzan B, 2004, Data Structures : A Pseudocode Approach with C. Cengage Learning

DATA STRUCTURES

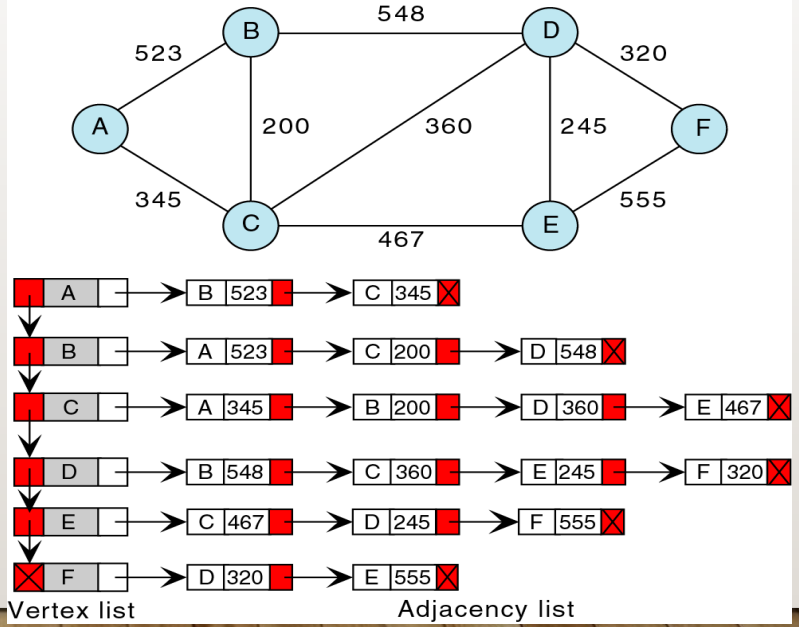
- A network is a graph whose lines are weighted - weighted graph
- The meaning of the weights depends on the application



Vertex vector
A
B
C
D
E
F

	A	B	C	D	E	F
A	0	523	345	0	0	0
B	523	0	200	548	0	0
C	345	200	0	360	467	0
D	0	548	360	0	245	320
E	0	0	467	245	0	555
F	0	0	0	320	555	0

Adjacency matrix



Gilberg R., Forouzan B, 2004, Data Structures : A Pseudocode Approach with C. Cengage Learning

STANDARD PHP LIBRARY (SPL) AND DATA STRUCTURES

- SPL was created to solve common problems which were lacking in PHP as its support of data structures
- SPL comes with core PHP installations and does not require any extension or change in configurations to enable it

STANDARD PHP LIBRARY (SPL) AND DATA STRUCTURES

- SPL provides a set of standard data structures through Object-Oriented Programming in PHP
- The supported data structures are:
  - Doubly linked lists: It is implemented in SplDoublyLinkedList
  - Stack: It is implemented in SplStack by using SplDoublyLinkedList
  - Queue: It is implemented in SplQueue by using SplDoublyLinkedList
  - Heaps: It is implemented in SplHeap; it also supports max heap in SplMaxHeap and min heap in SplMinHeap
  - Priority queue: It is implemented in SplPriorityQueue by using SplHeap
  - Arrays: It is implemented in SplFixedArray for a fixed size array
  - Map: It is implemented in SplObjectStorage