Computational Thinking

Prof. Madhavan Mukund
Department of Computer Science
Chennai Mathematical Institute

Prof. G. Venkatesh
Indian Institute of Technology Madras

Mr. Omkar Joshi
Course Instructor
IITM Online Degree Programme

Computational Thinking

Tutorial on pseudocode for recursion and Depth First Search (DFS)

Content

Recursion

- Recursion is a process in which a procedure calls itself.
- All problems can not be solved using recursion.
- Only those problems which have a base condition can be solved using recursion.
- Recursion simplifies the pseudocode.

Pseudocode to find the factorial of a number

```
Procedure factorial (n)
    fact = 1
    while (n > 0) {
        fact = fact * n
        n = n - 1
    }
    return (fact)
End factorial
```

Pseudocode to find the factorial of a number

```
Procedure factorial (n)
    fact = 1
    while (n > 0) {
        fact = fact * n
        n = n - 1
    }
    return (fact)
End factorial
```

```
Procedure factorial (n)
    if (n == 0) {
        return (1)
    }
    else {
        return (n * factorial (n - 1))
    }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

```
Procedure factorial (n)
    if (n == 0) {
        return (1)
    }
    else {
        return (n * factorial (n - 1))
    }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

```
Procedure factorial (5)
    if (5 == 0) {
        return (1)
    }
    else {
        return (5 * factorial (5 - 1))
    }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

```
Procedure factorial (5)
        if (5 == 0) {
                return (1)
        }
        else {
                return (5 * factorial (5 - 1))
        }
End factorial
```

```
Procedure factorial (4)
        if (4 == 0) {
                return (1)
        }
        else {
                return (4 * factorial (4 - 1))
        }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

```
Procedure factorial (5)
    if (5 == 0) {
        return (1)
    }
    else {
        return (5 * factorial (5 - 1))
    }
End factorial
```

```
Procedure factorial (4)
    if (4 == 0) {
        return (1)
    }
    else {
        return (4 * factorial (4 - 1))
    }
End factorial
```

```
Procedure factorial (3)
    if (3 == 0) {
        return (1)
    }
    else {
        return (3 * factorial (3 - 1))
    }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

```
Procedure factorial (5)
    if (5 == 0) {
        return (1)
    }
    else {
        return (5 * factorial (5 - 1))
    }
End factorial
```

```
Procedure factorial (4)
    if (4 == 0) {
        return (1)
    }
    else {
        return (4 * factorial (4 - 1))
    }
End factorial
```

```
Procedure factorial (3)
    if (3 == 0) {
        return (1)
    }
    else {
        return (3 * factorial (3 - 1))
    }
End factorial
```

```
Procedure factorial (2)
    if (2 == 0) {
        return (1)
    }
    else {
        return (2 * factorial (2 - 1))
    }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

```
Procedure factorial (5)
    if (5 == 0) {
        return (1)
    }
    else {
        return (5 * factorial (5 - 1))
    }
End factorial
```

```
Procedure factorial (4)
    if (4 == 0) {
        return (1)
    }
    else {
        return (4 * factorial (4 - 1))
    }
End factorial
```

```
Procedure factorial (3)
    if (3 == 0) {
        return (1)
    }
    else {
        return (3 * factorial (3 - 1))
    }
End factorial
```

```
Procedure factorial (2)
    if (2 == 0) {
        return (1)
    }
    else {
        return (2 * factorial (2 - 1))
    }
End factorial
```

```
Procedure factorial (1)
    if (1 == 0) {
        return (1)
    }
    else {
        return (1 * factorial (1 - 1))
    }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

```
Procedure factorial (5)
    if (5 == 0) {
        return (1)
    }
    else {
        return (5 * factorial (5 - 1))
    }
End factorial
```

```
Procedure factorial (4)
    if (4 == 0) {
        return (1)
    }
    else {
        return (4 * factorial (4 - 1))
    }
End factorial
```

```
Procedure factorial (3)
    if (3 == 0) {
        return (1)
    }
    else {
        return (3 * factorial (3 - 1))
    }
End factorial
```

```
Procedure factorial (2)
    if (2 == 0) {
        return (1)
    }
    else {
        return (2 * factorial (2 - 1))
    }
End factorial
```

```
Procedure factorial (1)
    if (1 == 0) {
        return (1)
    }
    else {
        return (1 * factorial (1 - 1))
    }
End factorial
```

```
Procedure factorial (0)
    if (0 == 0) {
        return (1)
    }
    else {
        return (0 * factorial (0 - 1))
    }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

```
Procedure factorial (5)
    if (5 == 0) {
        return (1)
    }
    else {
        return (5 * factorial (5 - 1))
    }
End factorial
```

```
Procedure factorial (4)
    if (4 == 0) {
        return (1)
    }
    else {
        return (4 * factorial (4 - 1))
    }
End factorial
```

```
Procedure factorial (3)
    if (3 == 0) {
        return (1)
    }
    else {
        return (3 * factorial (3 - 1))
    }
End factorial
```

```
Procedure factorial (2)
    if (2 == 0) {
        return (1)
    }
    else {
        return (2 * factorial (2 - 1))
    }
End factorial
```

```
Procedure factorial (1)
    if (1 == 0) {
        return (1)
    }
    else {
        return (1 * 1)
    }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

```
Procedure factorial (5)
    if (5 == 0) {
        return (1)
    }
    else {
        return (5 * factorial (5 - 1))
    }
End factorial
```

```
Procedure factorial (4)
    if (4 == 0) {
        return (1)
    }
    else {
        return (4 * factorial (4 - 1))
    }
End factorial
```

```
Procedure factorial (3)
    if (3 == 0) {
        return (1)
    }
    else {
        return (3 * factorial (3 - 1))
    }
End factorial
```

```
Procedure factorial (2)
    if (2 == 0) {
        return (1)
    }
    else {
        return (2 * 1)
    }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

```
Procedure factorial (5)
    if (5 == 0) {
        return (1)
    }
    else {
        return (5 * factorial (5 - 1))
    }
End factorial
```

```
Procedure factorial (4)
    if (4 == 0) {
        return (1)
    }
    else {
        return (4 * factorial (4 - 1))
    }
End factorial
```

```
Procedure factorial (3)
    if (3 == 0) {
        return (1)
    }
    else {
        return (3 * 2)
    }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

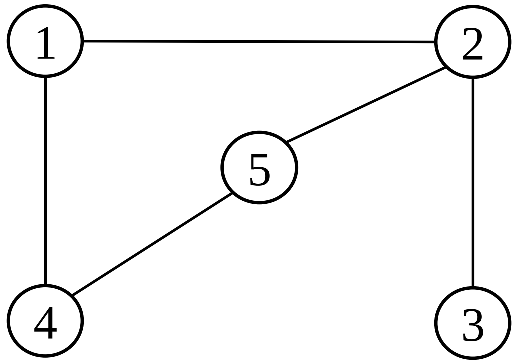
```
Procedure factorial (5)
    if (5 == 0) {
        return (1)
    }
    else {
        return (5 * factorial (5 - 1))
    }
End factorial
```

```
Procedure factorial (4)
    if (4 == 0) {
        return (1)
    }
    else {
        return (4 * 6))
    }
End factorial
```

Pseudocode to find the factorial using recursion (n = 5)

```
Procedure factorial (5)
        if (5 == 0) {
                return (1)
        }
        else {
                return (5 * 24)
        }
End factorial
```

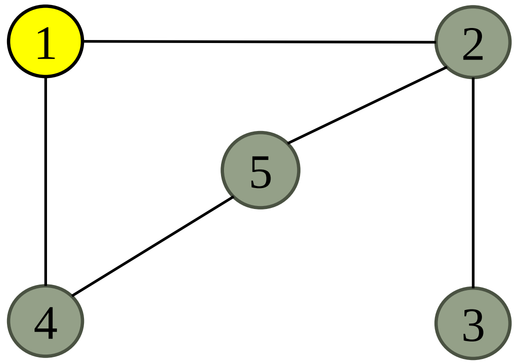
Depth First Search (DFS)



Input graph

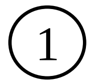
Starting vertex = 1

Depth First Search (DFS)

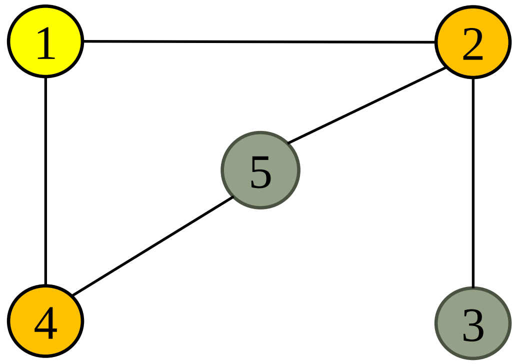


Input graph

Current vertex = 1



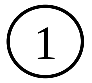
Depth First Search (DFS)



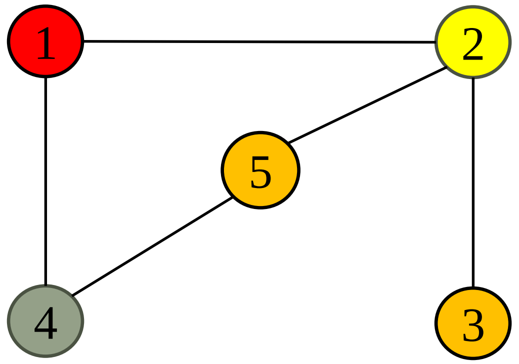
Input graph

Current vertex = 1

Neighbour = 2, 4



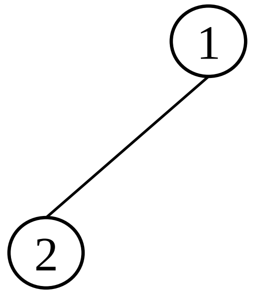
Depth First Search (DFS)



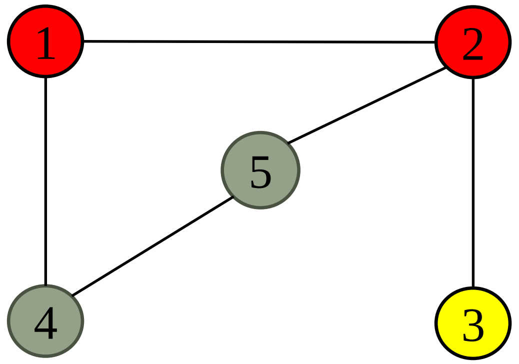
Input graph

Current vertex = 2

Neighbour = 3, 5



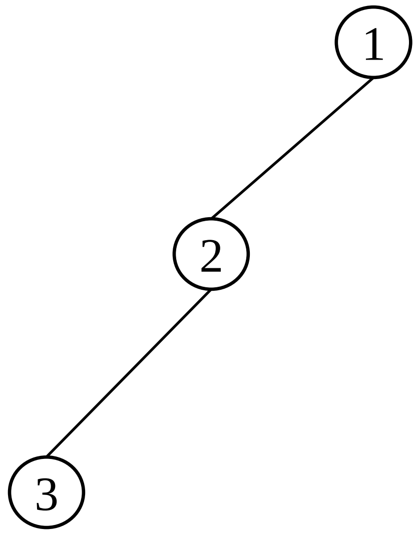
Depth First Search (DFS)



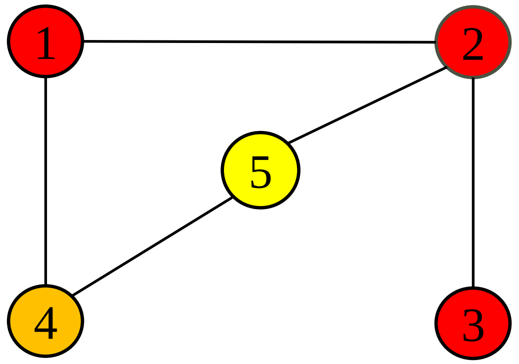
Input graph

Current vertex = 1

Neighbour = No



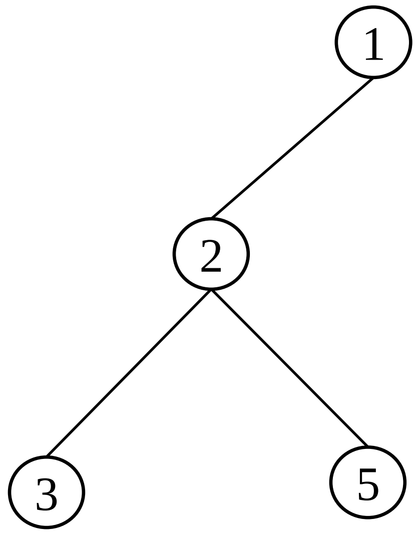
Depth First Search (DFS)



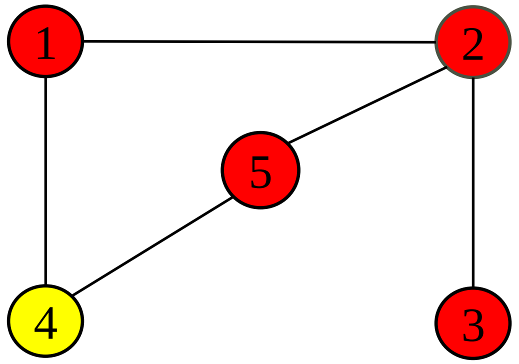
Input graph

Current vertex = 5

Neighbour = 4



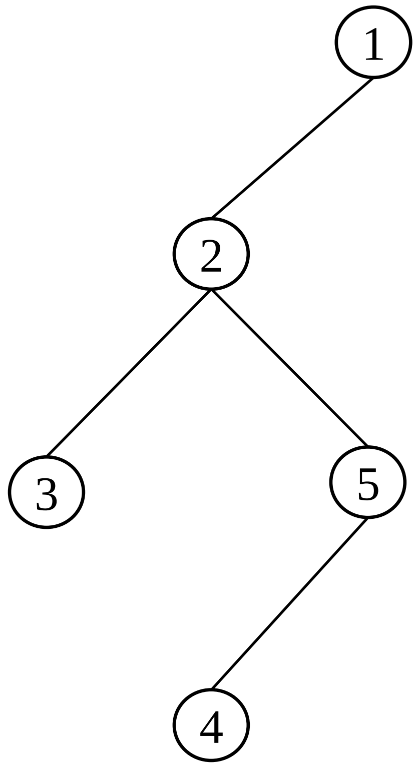
Depth First Search (DFS)



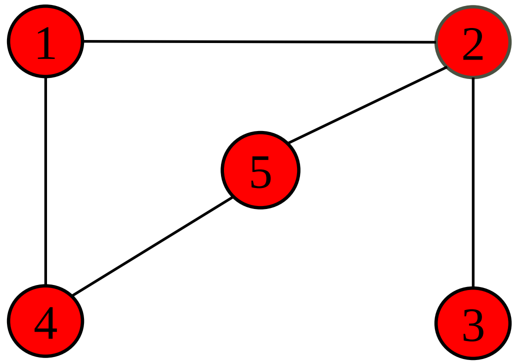
Input graph

Current vertex = 4

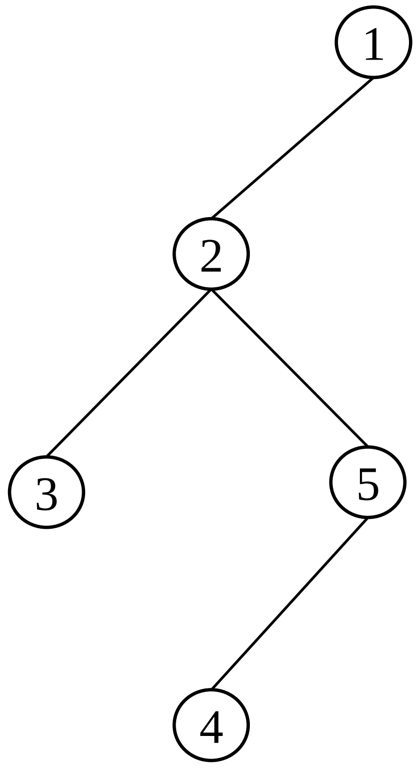
Neighbour = No



Depth First Search (DFS)



Input graph



Output DFS tree