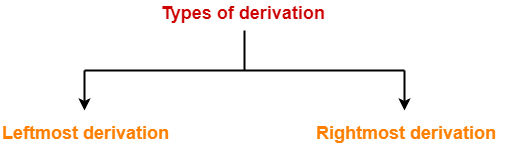
**Parse Tree-**

* The process of deriving a string is called as **derivation**.
* The geometrical representation of a derivation is called as a **parse tree** or **derivation tree.**



**1. Leftmost Derivation-**

* The process of deriving a string by expanding the leftmost non-terminal at each step is called as **leftmost derivation**.
* The geometrical representation of leftmost derivation is called as a **leftmost derivation tree**.

**Example-**

Consider the following grammar-

S → aB / bA

S → aS / bAA / a

B → bS / aBB / b

(**Unambiguous Grammar**)

Let us consider a string w = aaabbabbba

Now, let us derive the string w using leftmost derivation.

**Leftmost Derivation-**

S   → a**B**

→  aa**B**B                   (Using B → aBB)

→ aaa**B**BB                (Using B → aBB)

→ aaab**B**B                (Using B → b)

→ aaabb**B**                (Using B → b)

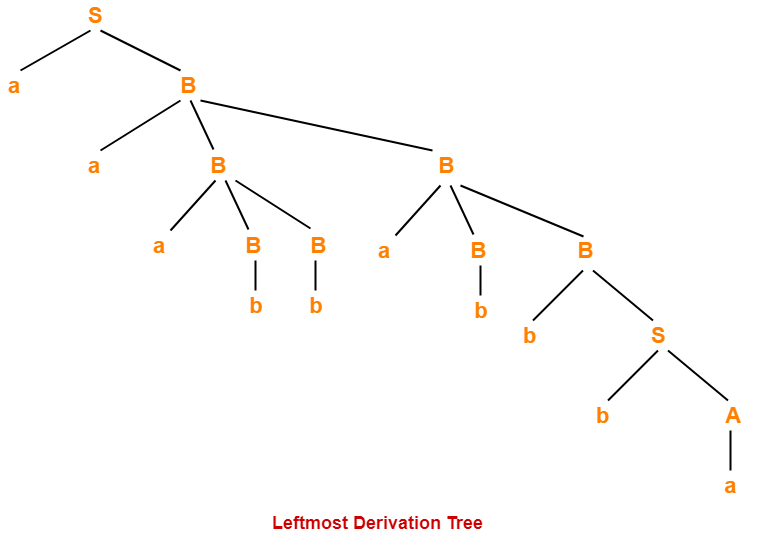
→ aaabba**B**B            (Using B → aBB)

→ aaabbab**B**            (Using B → b)

→ aaabbabb**S**          (Using B → bS)

→ aaabbabbb**A**        (Using S → bA)

→ aaabbabbba         (Using A → a)



**2. Rightmost Derivation-**

* The process of deriving a string by expanding the rightmost non-terminal at each step is called as **rightmost derivation**.
* The geometrical representation of rightmost derivation is called as a **rightmost derivation tree**.

**Example-**

Consider the following grammar-

S → aB / bA

S → aS / bAA / a

B → bS / aBB / b

(**Unambiguous Grammar**)

Let us consider a string w = aaabbabbba

Now, let us derive the string w using rightmost derivation.

**Rightmost Derivation-**

S   → a**B**

→  aaB**B**                    (Using B → aBB)

→ aaBaB**B**                 (Using B → aBB)

→ aaBaBb**S**               (Using B → bS)

→ aaBaBbb**A**             (Using S → bA)

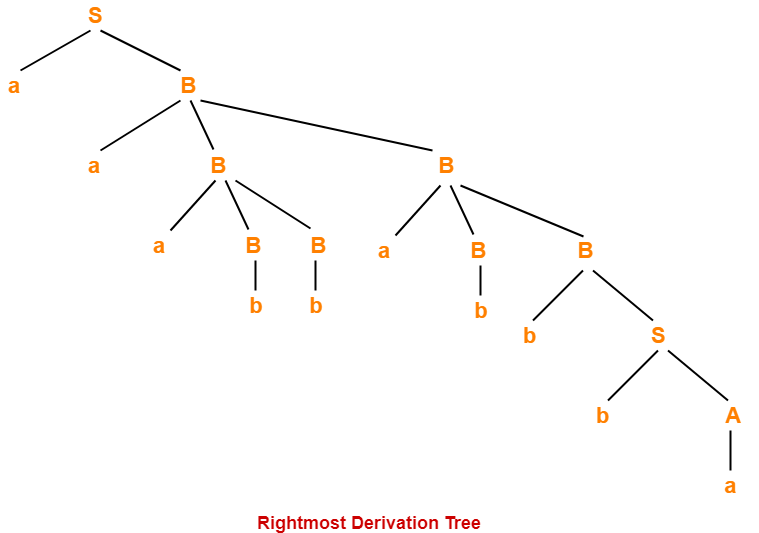
→ aaBa**B**bba              (Using A → a)

→ aa**B**abbba              (Using B → b)

→ aaaB**B**abbba          (Using B → aBB)

→ aaa**B**babbba          (Using B → b)

→ aaabbabbba           (Using B → b)



|  |
| --- |
| **NOTES**   * For unambiguous grammars, Leftmost derivation and Rightmost derivation represents the same parse tree. * For ambiguous grammars, Leftmost derivation and Rightmost derivation represents different parse trees. |

Here,

* The given grammar was unambiguous.
* That is why, leftmost derivation and rightmost derivation represents the same parse tree.

|  |
| --- |
| **Leftmost Derivation Tree = Rightmost Derivation Tree** |

**Also Read-** [**Ambiguous Grammar**](https://www.gatevidyalay.com/ambiguous-grammar-types-of-grammar/)

**Properties Of Parse Tree-**

* Root node of a parse tree is the start symbol of the grammar.
* Each leaf node of a parse tree represents a terminal symbol.
* Each interior node of a parse tree represents a non-terminal symbol.
* Parse tree is independent of the order in which the productions are used during derivations.

**Yield Of Parse Tree-**

* Concatenating the leaves of a parse tree from the left produces a string of terminals.
* This string of terminals is called as **yield of a parse tree**.

**PRACTICE PROBLEMS BASED ON DERIVATIONS AND PARSE TREE-**

**Problem-01:**

Consider the grammar-

S → bB / aA

A → b / bS / aAA

B → a / aS / bBB

For the string w = bbaababa, find-

1. Leftmost derivation
2. Rightmost derivation
3. Parse Tree

**Solution-**

**1. Leftmost Derivation-**

S   → b**B**

→ bb**B**B              (Using B → bBB)

→ bba**B**              (Using B → a)

→ bbaa**S**            (Using B → aS)

→ bbaab**B**          (Using S → bB)

→ bbaaba**S**        (Using B → aS)

→ bbaabab**B**      (Using S → bB)

→ bbaababa       (Using B → a)

**2. Rightmost Derivation-**

S   → b**B**

→ bbB**B**                (Using B → bBB)

→ bbBa**S**              (Using B → aS)

→ bbBab**B**            (Using S → bB)

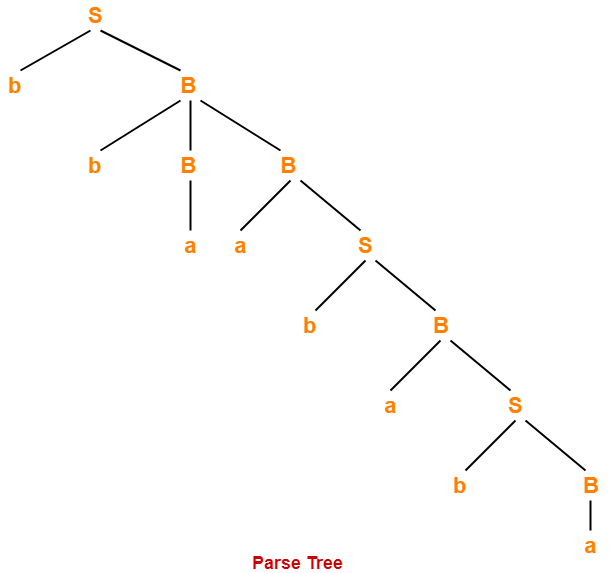
→ bbBaba**S**          (Using B → aS)

→ bbBabab**B**        (Using S → bB)

→ bb**B**ababa        (Using B → a)

→ bbaababa         (Using B → a)

**3. Parse Tree-**



* Whether we consider the leftmost derivation or rightmost derivation, we get the above parse tree.
* The reason is given grammar is unambiguous.

**Problem-02:**

Consider the grammar-

S → A1B

A → 0A / ∈

B → 0B / 1B / ∈

For the string w = 00101, find-

1. Leftmost derivation
2. Rightmost derivation
3. Parse Tree

**Solution-**

**1. Leftmost Derivation-**

S   → **A**1B

→ 0**A**1B              (Using A → 0A)

→ 00**A**1B            (Using A → 0A)

→ 001**B**              (Using A → ∈)

→ 0010**B**            (Using B → 0B)

→ 00101**B**          (Using B → 1B)

→ 00101             (Using B → ∈)

**2. Rightmost Derivation-**

S   → A1**B**

→ A10**B**                (Using B → 0B)

→ A101**B**              (Using B → 1B)

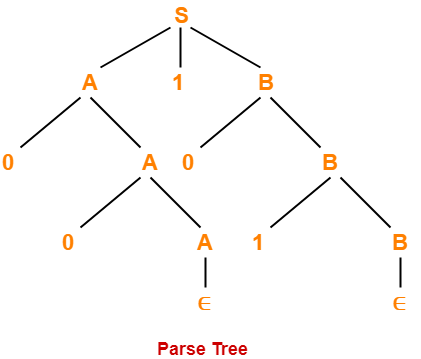
→ **A**101                (Using B → ∈)

→ 0**A**101              (Using A → 0A)

→ 00**A**101            (Using A → 0A)

→ 00101               (Using A → ∈)

**3. Parse Tree-**



* Whether we consider the leftmost derivation or rightmost derivation, we get the above parse tree.
* The reason is given grammar is unambiguous.