Difference between LL and LR

LL Parser includes both the recursive descent parser and non-recursive descent parser. Its one type uses backtracking while another one uses parsing table. Theses are top down parser.

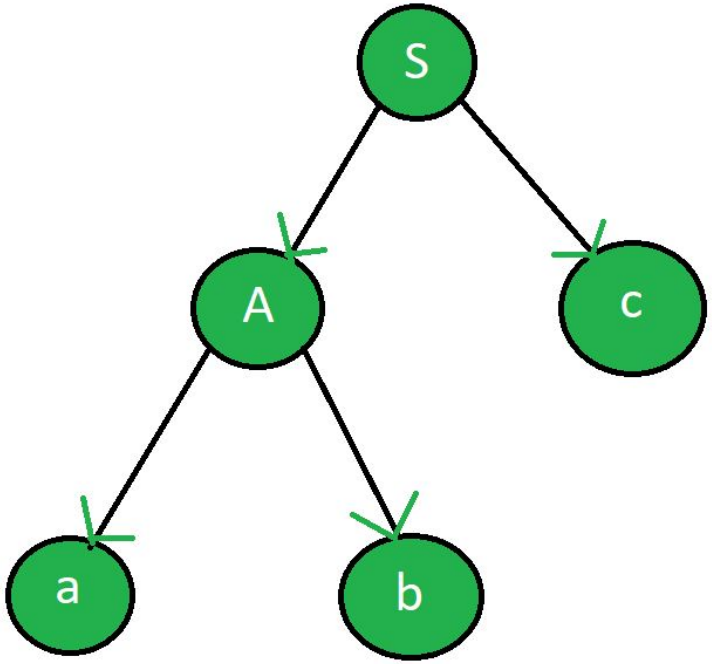
Example: Given grammar is

```
S -> Ac
A -> ab
```

where S is start symbol, A is non-terminal and a, b, c are terminals.

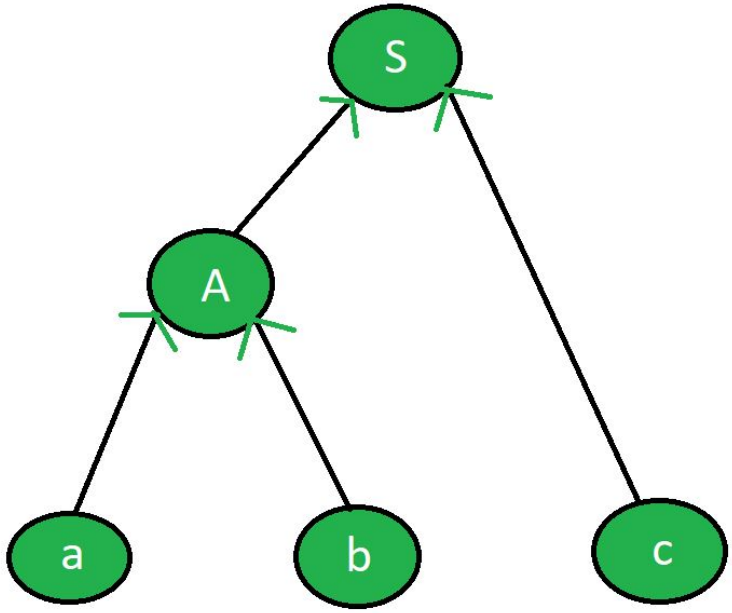
Input string: abc

Parse tree generated by LL parser:


S
A
c
a
b

LR Parser is one of the bottom up parser which uses parsing table (dynamic programming) to obtain the parse tree form given string using grammar productions.

Example: In the above example, parse tree generated by LR parser:



Difference between LL and LR parser:

LL PARSER	LR PARSER
First L of LL is for left to right and second L is for leftmost derivation.	L of LR is for left to right and R is for rightmost derivation.
It follows the left most derivation.	It follows reverse of right most derivation.

LL PARSER	LR PARSER
Using LL parser parser tree is constructed in top down manner.	Parser tree is constructed in bottom up manner.
In LL parser, non-terminals are expanded.	In LR parser, terminals are compressed.
Starts with the start symbol(S).	Ends with start symbol(S).
Ends when stack used becomes empty.	Starts with an empty stack.
Pre-order traversal of the parse tree.	Post-order traversal of the parser tree.
Terminal is read after popping out of stack.	Terminal is read before pushing into the stack.
It may use backtracking or dynamic programming.	It uses dynamic programming.
LL is easier to write.	LR is difficult to write.
Example: LL(0), LL(1)	**Example:** LR(0), SLR(1), LALR(1), CLR(1)

Recommended Posts:

Operator grammar and precedence parser in TOC

Shift Reduce Parser in Compiler

Recursive Descent Parser

Difference between Top down parsing and Bottom up parsing

Difference between High Level and Low level languages

Difference between Compiler and Assembler

Parsing ambiguos grammars using LR parser

Difference between Compile Time and Load Time address Binding

Difference between Load Time and Execution Time address binding

Difference between Compile Time and Execution Time address binding

Difference between Static allocation and Stack allocation

Difference between Static Allocation and Heap Allocation

Types of Three-address codes

Run-time Storage Organization

pp_pankaj
Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags : Compiler Design

Be the First to upvote.

0

No votes yet.

To-do Done

Feedback/ Suggest Improvement | Add Notes | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks
A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY
About Us
Careers
Privacy Policy
Contact Us

LEARN
Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE
Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE
Write an Article
Write Interview Experience
Internships
Videos