



Dashboard



Compiler



Algorithms



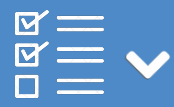
CC Assistant



Assignments



Exercises



Dashboard

Compiler

Algorithms

CC Assistant

Assignments

Exercises

LEXICAL ANALYSIS > INPUT REGULAR EXPRESSIONS

Lexeme

Regular expression



WHITESPACE

[\t\r\n\u00C] +

IntegerLiteral

(-|+)?[0-9] +

BooleanLiteral

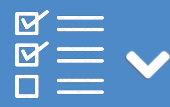
true|false



< PREVIOUS

UPLOAD TOKENS

NEXT >



Dashboard

Compiler

Algorithms

CC Assistant

Assignments

Exercises

LEXICAL ANALYSIS > INPUT REGULAR EXPRESSIONS > **REGULAR EXPRESSIONS TO NFA**



$(-|+)?[0-9]^+$

Regular expression parse tree



ADD NODE
CHECK INPUT

NFA

< PREVIOUS



NEXT >



Dashboard

Compiler

Algorithms

CC Assistant

Assignments

Exercises

LEXICAL ANALYSIS > INPUT REGULAR EXPRESSIONS > **REGULAR EXPRESSIONS TO NFA**

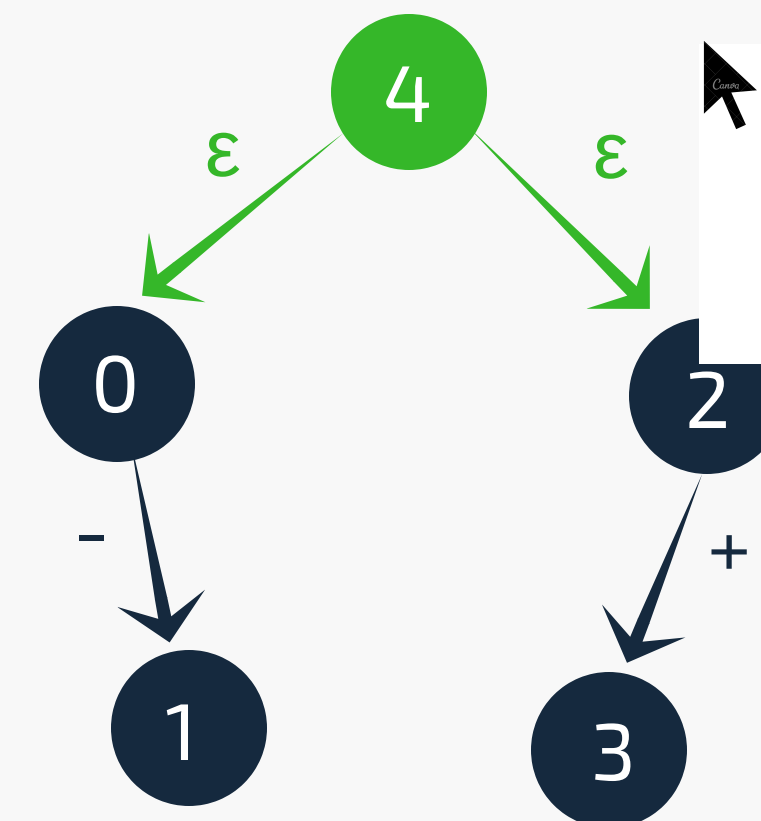


$(-|+)?[0-9]^+$

Regular expression parse tree



NFA



ADD STATE
CHECK INPUT

< PREVIOUS



NEXT >



Dashboard



Compiler



Full frontend

Lexical analysis

Syntax analysis

Semantic analysis

Garbage collection

CEK machine



Algorithms



CC Assistant



Assignments



Exercises

SYNTAX ANALYSIS > INPUT GRAMMAR



Grammar



```
program
  mainClass, { classDeclaration }, EOF

mainClass
  CLASS, Identifier, LBRACE, PUBLIC,
  STATIC,
  VOID, "main", LPAREN, RPAREN,
  LBRACE, statement, RBRACE, RBRACE
```

Text to parse



```
/**
 * Create an input stream from a socket.
 */
public In(java.net.Socket socket) {
  try {
    InputStream is = socket.getInputStream();
    scanner = new Scanner(new BufferedInputStream(is), CHARSET_NAME);
    scanner.useLocale(LOCALE);
  }
  catch (IOException ioe) {
    System.err.println("Could not open " + socket);
  }
}

/**
 * Create an input stream from a URL.
 */
public In(URL url) {
  try {
    URLConnection site = url.openConnection();
    InputStream is = site.getInputStream();
    scanner = new Scanner(new BufferedInputStream(is), CHARSET_NAME);
    scanner.useLocale(LOCALE);
  }
  catch (IOException ioe) {
    System.err.println("Could not open " + url);
  }
}
```

< PREVIOUS

LL(1)

NEXT >



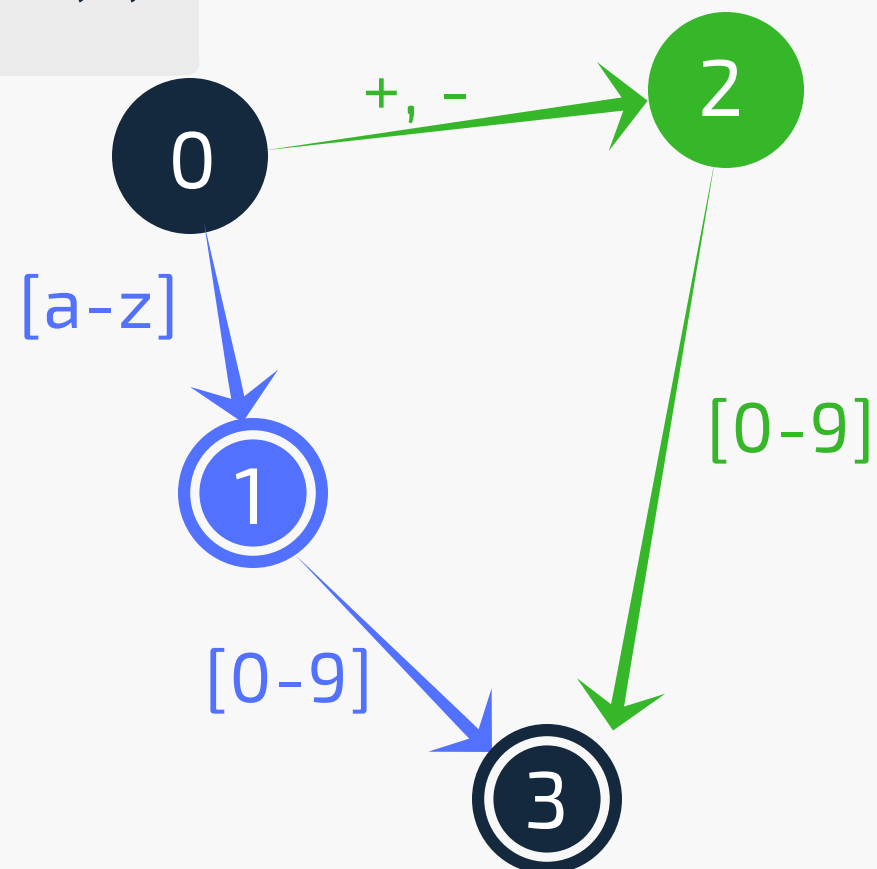
- Dashboard
- Compiler
- Full frontend
- Lexical analysis
- Syntax analysis
- Semantic analysis
- Garbage collection
- CEK machine
- Algorithms
- CC Assistant
- Assignments
- Exercises

SYNTAX ANALYSIS > INPUT GRAMMAR > LL(1) - TOKENIZATION

DFA

Initial state

NFA states: 0, 1, 2



Final state

NFA states: 1, 3, 4

Accept: [IntLiteral, Identifier]

Generated tokens



Lets the user replay as specific tokenization

Lets the user predict the next token

Change simulation settings. E.g. speed, whether to show the debugging page, etc...

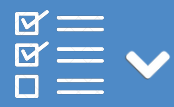
< BACK

< PREVIOUS



NEXT >

SKIP >



- Dashboard
- Compiler
- Full frontend
- Lexical analysis
- Syntax analysis
- Semantic analysis
- Garbage collection
- CEK machine
- Algorithms
- CC Assistant
- Assignments
- Exercises

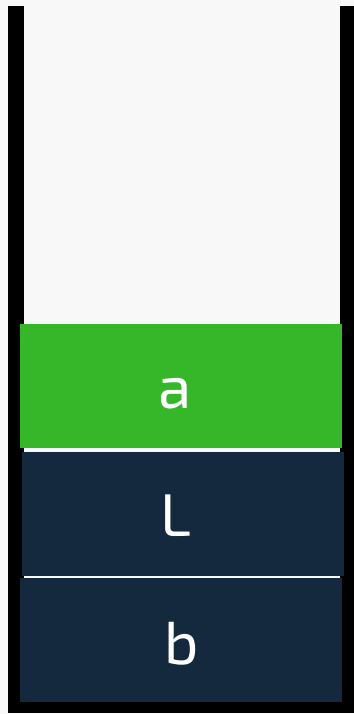
SYNTAX ANALYSIS > INPUT GRAMMAR > LL(1) - PARSING

CURRENT ACTION: **MATCH a**

Tokens



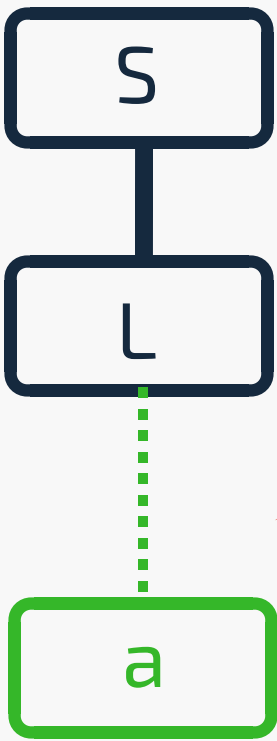
Stack



Grammar

$S \rightarrow Lb$
 $L \rightarrow aL$
 $L \rightarrow \epsilon$

Parse tree



This is the addition that will be visualized after doing a match (i.e. pressing next)

☒ Compact visualization

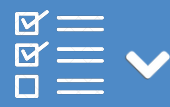
< BACK

< PREVIOUS



NEXT >

SKIP >



- Dashboard
- Compiler
- Full frontend
- Lexical analysis
- Syntax analysis
- Semantic analysis
- Garbage collection
- CEK machine
- Algorithms
- CC Assistant
- Assignments
- Exercises

SYNTAX ANALYSIS > INPUT GRAMMAR > LL(1) - PARSING

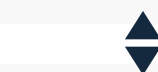
CURRENT ACTION: **PREDICT L -> aL**

Tokens

a a a b

Stack

L
b



Grammar

$S \rightarrow Lb$

$L \rightarrow aL$

$L \rightarrow \epsilon$

Parse tree

COMPUTING FIRST(aL)



After computing the FIRST set of 'aL', we check if the current character (a) is in this set and report to the user.

FIRST(aL) = {a}



☒ Compact visualization

< BACK

< PREVIOUS



NEXT >

SKIP >



- Dashboard
- Compiler
- Full frontend
- Lexical analysis
- Syntax analysis
- Semantic analysis
- Garbage collection
- CEK machine
- Algorithms
- CC Assistant
- Assignments
- Exercises

SYNTAX ANALYSIS > INPUT GRAMMAR > LL(1) - PARSING

CURRENT ACTION: **PREDICT L -> aL**



Tokens

a a a b

Stack

L
b

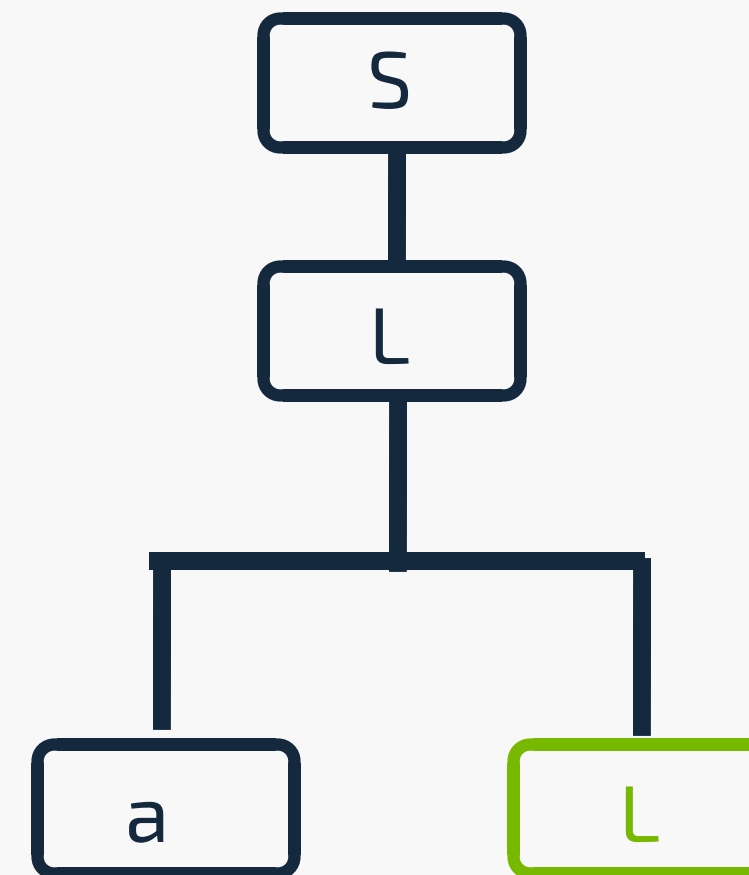


Grammar



$S \rightarrow Lb$
 $L \rightarrow aL$
 $L \rightarrow \epsilon$

Parse tree



Compact visualization

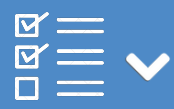
< BACK

< PREVIOUS



NEXT >

SKIP >



Dashboard

Compiler

Full frontend

Lexical analysis

Syntax analysis

Semantic analysis

Garbage collection

CEK machine

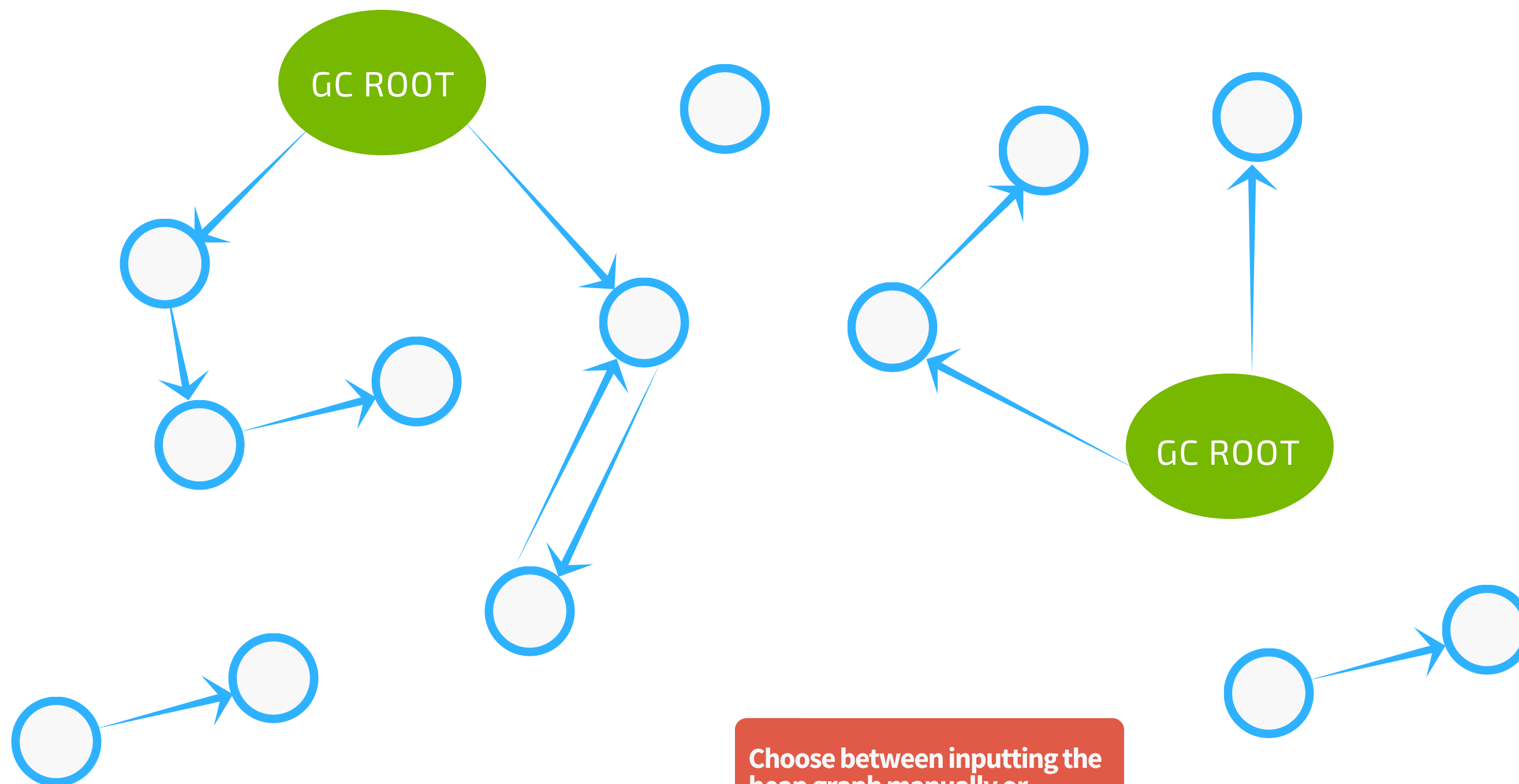
Algorithms

CC Assistant

Assignments

Exercises

GARBAGE COLLECTION > INPUT



Choose between inputting the heap graph manually or generating it from Java code

< PREVIOUS

HEAP GRAPH

NEXT >



Dashboard

Compiler

Full frontend

Lexical analysis

Syntax analysis

Semantic analysis

Garbage collection

CEK machine

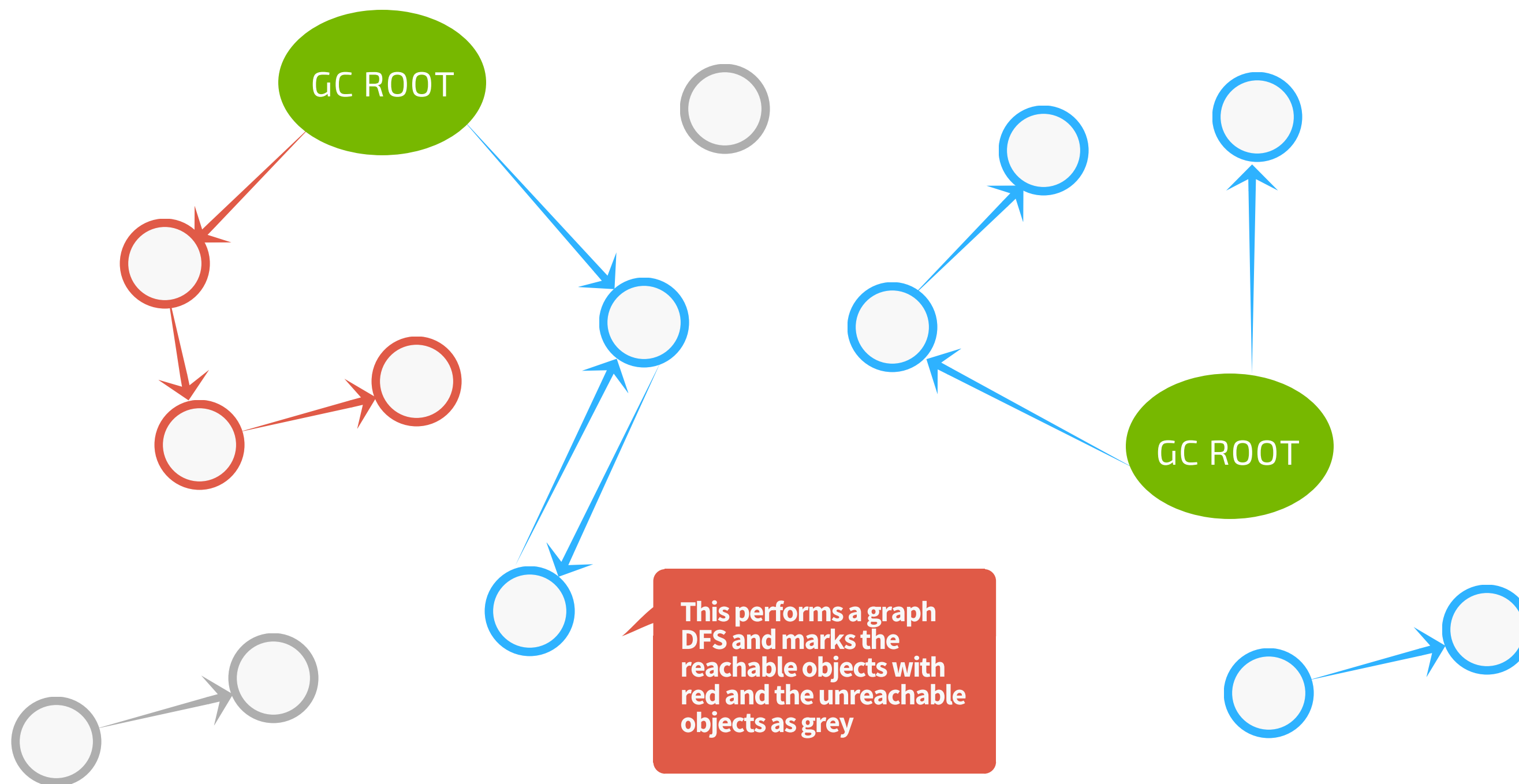
Algorithms

CC Assistant

Assignments

Exercises

GARBAGE COLLECTION > INPUT > MARK UNREACHABLE OBJECTS



< PREVIOUS

NEXT >



Dashboard

Compiler

Full frontend

Lexical analysis

Syntax analysis

Semantic analysis

Garbage collection

CEK machine

Algorithms

CC Assistant

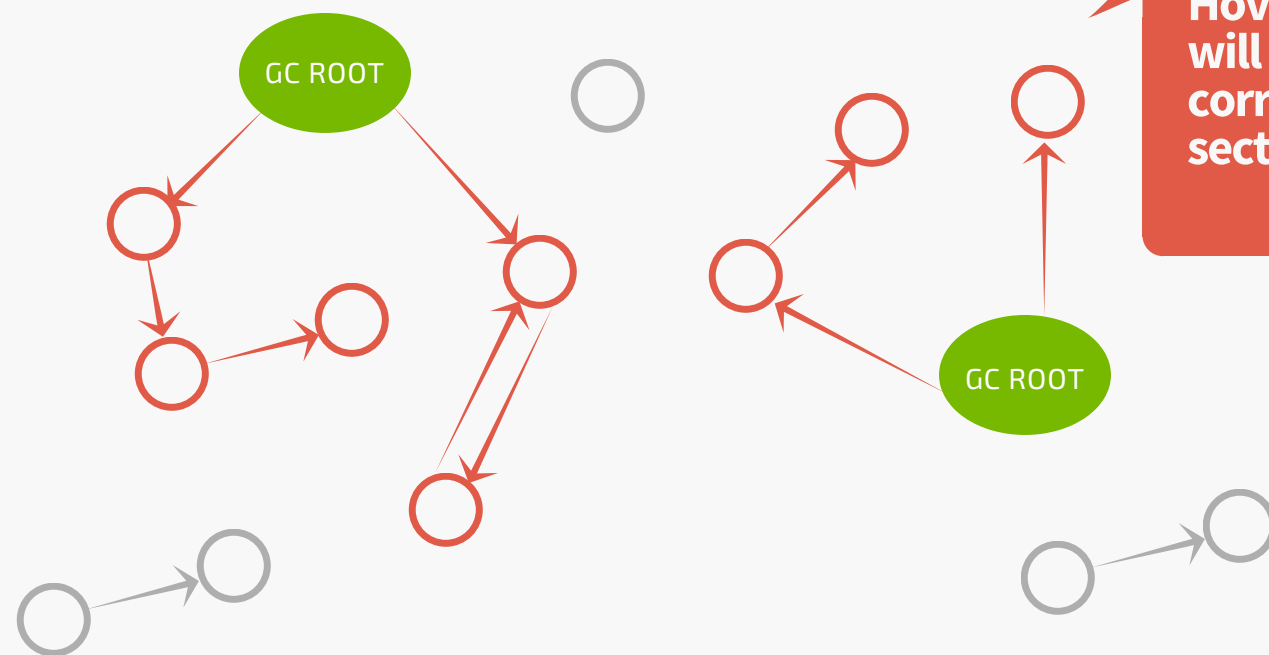
Assignments

Exercises

GARBAGE COLLECTION > INPUT > MARK UNREACHABLE OBJECTS > **SWEEP**

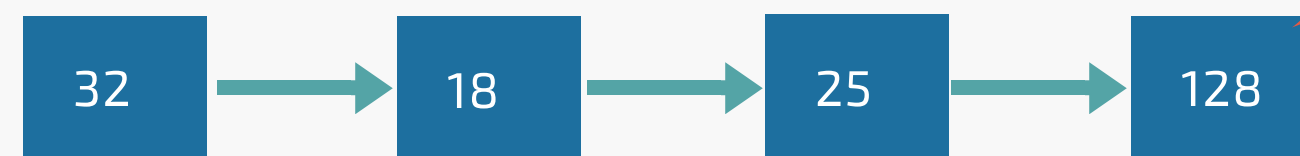


Heap graph

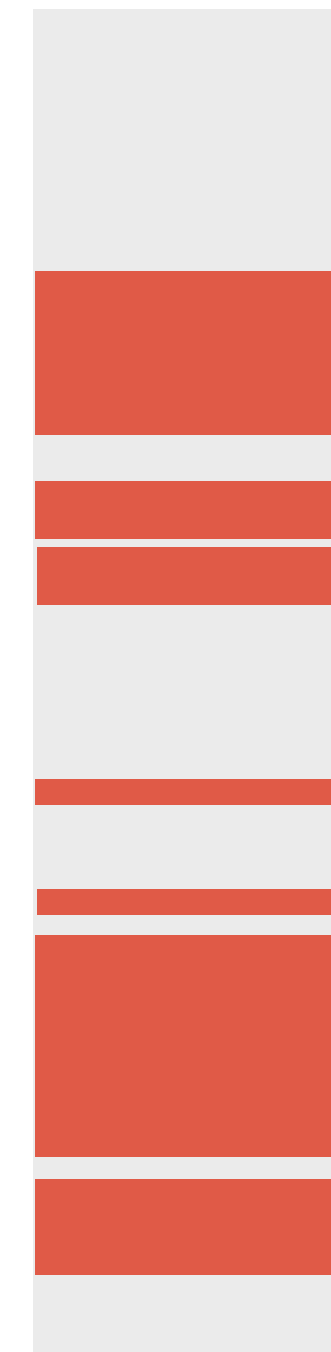


Hovering over the nodes will show the corresponding section in the heap

Free list



The numbers indicate the size of the free chunks



< PREVIOUS



NEXT >



Dashboard



Compiler



Algorithms



CC Assistant



Assignments



Exercises

CONTINUE DEVELOPMENT

Syntax Analysis

Currently on: Building AST.

Resume (88%)

SUGGESTED

Garbage Collection



Start

ASSIGNMENTS

[NFA to DFA \(Resume\)](#)

[Grammar Left Factoring \(Start\)](#)

[Regex to NFA \(Start\)](#)

CUSTOM TEST CASES

Regex to NFA

Edit

Test

Delete

Regex to NFA

Author: [John Smith](#)

Edit

Test

This is a test case submitted by the teacher



Learn

Implement the transition class for a finite automaton

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam luctus, lectus non tincidunt aliquet, nunc eros finibus nulla, vitae ultrices arcu urna sollicitudin risus.



Instructions



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam luctus, lectus non tincidunt aliquet, nunc eros finibus nulla, vitae ultrices arcu urna sollicitudin risus.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam luctus, lectus non tincidunt aliquet, nunc eros finibus nulla, vitae ultrices arcu urna sollicitudin risus.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam luctus, lectus non tincidunt aliquet, nunc eros finibus nulla, vitae ultrices arcu urna sollicitudin risus.

Transition.java

State.java

NFA.java



1

2

RUN

RESET

