cis112
Stack

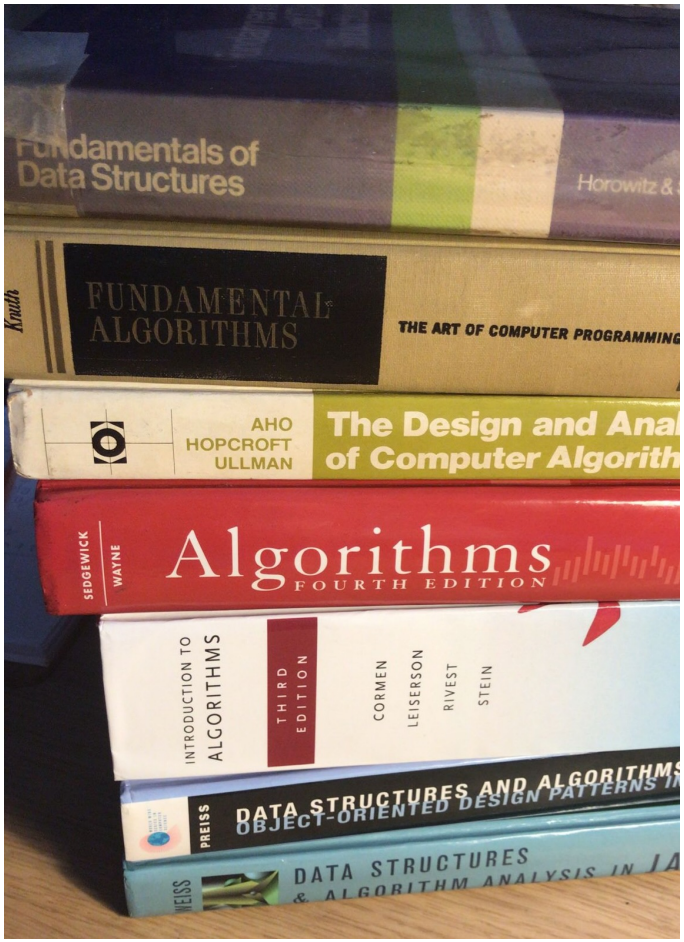
Haluk O. Bingol

Faculty of Computer and Information Sciences
Yeditepe University

v2024-11-04

Content

- Motivation
- Case: Browser visiting pages
- Implementation
- Abstraction
  - Container
  - Set
  - Stack
- Applications
  - Balanced parentheses
  - Postfix Evaluation
  - Infix to postfix conversion
  - Infix evaluation
  - Backtracking



Stack of books

Motivation

Motivation

How does

- browser keep track of previous visited pages?
- a text editor handle undo operations?
- compiler check if the parentheses balanced?
  - Balanced: (), (()), ()(), ((()(()))
  - Unbalanced:)(, (()), ())(, (((()(())))
- evaluate expression
  - 1 + 2 * 3

Case: Browser visiting pages

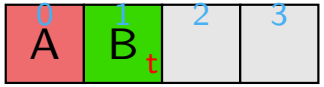
Case: Browser visiting pages

Visit page Page seen History

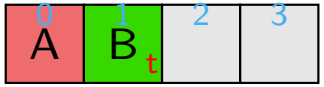
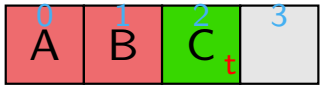
Case: Browser visiting pages

Visit page	Page seen	History
A	A	

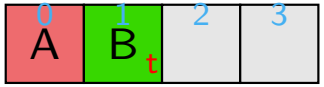
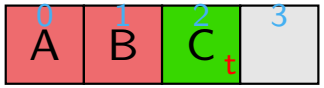
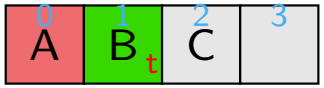
Case: Browser visiting pages

Visit page	Page seen	History
A	A	
B	B	

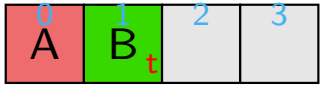
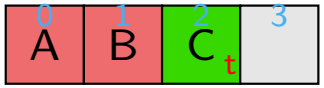
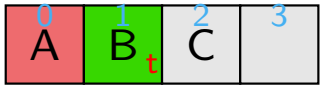
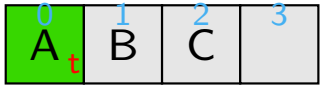
Case: Browser visiting pages

Visit page	Page seen	History
A	A	
B	B	
C	C	

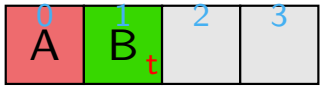
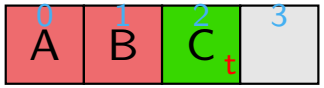
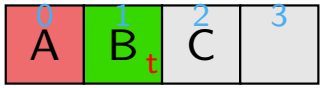
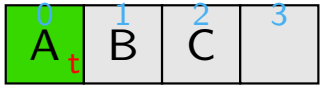
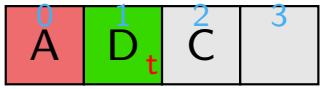
Case: Browser visiting pages

Visit page	Page seen	History
A	A	
B	B	
C	C	
Back	B	 C is still there!

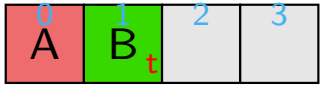
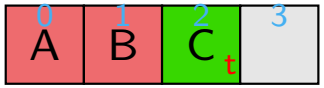
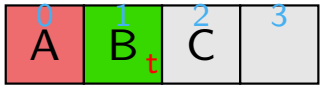
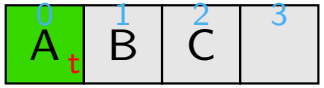
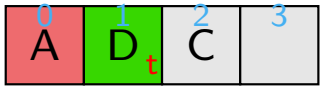
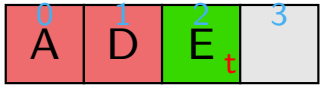
Case: Browser visiting pages

Visit page	Page seen	History
A	A	
B	B	
C	C	
Back	B	 C is still there!
Back	A	 what if we back once more?

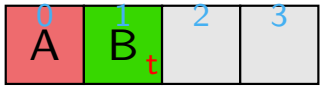
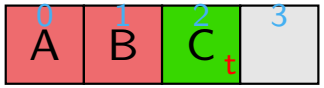
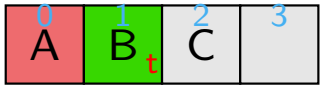
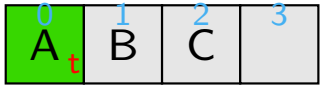
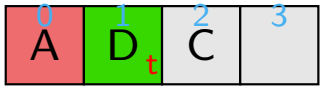
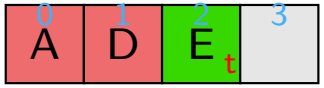
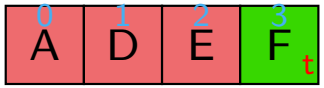
Case: Browser visiting pages

Visit page	Page seen	History
A	A	
B	B	
C	C	
Back	B	 C is still there!
Back	A	 what if we back once more?
D	D	

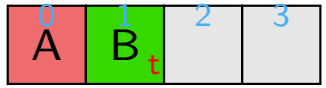
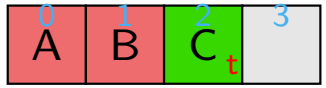
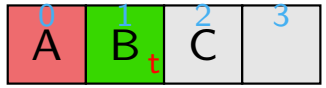
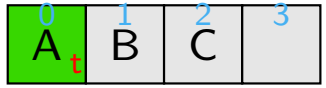
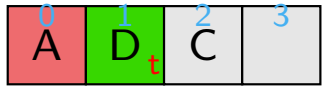
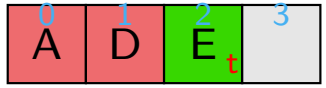
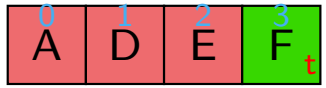
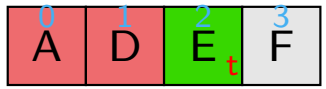
Case: Browser visiting pages

Visit page	Page seen	History
A	A	
B	B	
C	C	
Back	B	 C is still there!
Back	A	 what if we back once more?
D	D	
E	E	

Case: Browser visiting pages

Visit page	Page seen	History
A	A	
B	B	
C	C	
Back	B	 C is still there!
Back	A	 what if we back once more?
D	D	
E	E	
F	F	 what if we visit one more page?

Case: Browser visiting pages

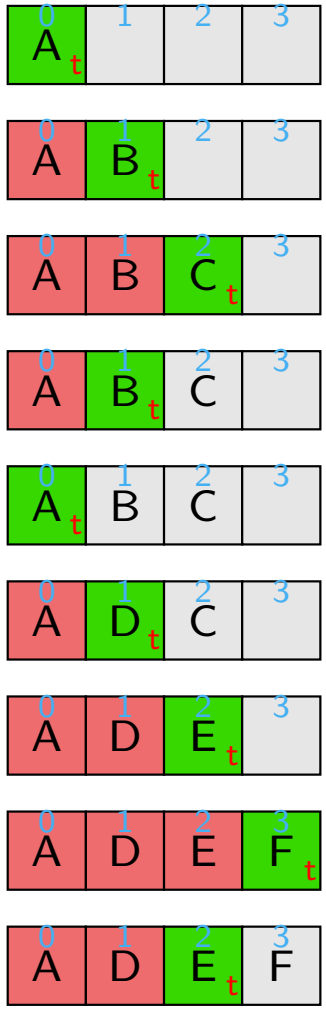
Visit page	Page seen	History
A	A	
B	B	
C	C	
Back	B	 C is still there!
Back	A	 what if we back once more?
D	D	
E	E	
F	F	 what if we visit one more page?
Back	E	

Implementation

Observations

- insert(item)
  - at lastUsedLocation + 1
- delete()
  - at lastUsedLocation
- Use array
  - keep track of lastUsedLocation

History



Implementation with array

- push(item) ~~insert(item)~~
  - at top + 1 ~~lastUsedLocation + 1~~
- pop() ~~delete(item)~~
  - at top ~~lastUsedLocation~~
- Use array
  - keep track of top ~~lastUsedLocation~~

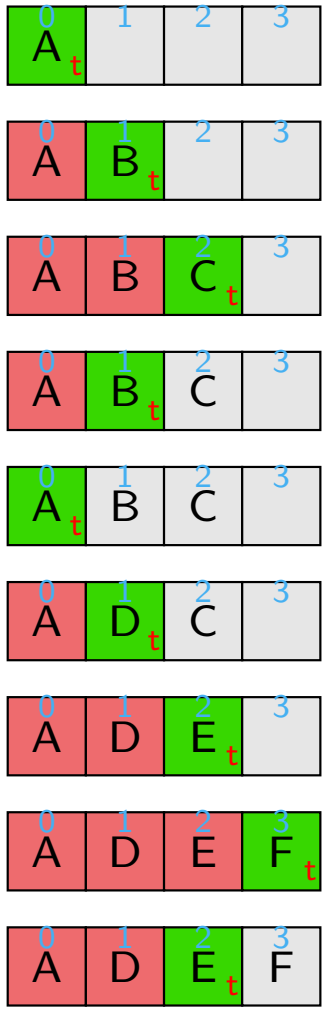
```
PUSH(S, x)
1  if S.top == S.size
2        error "overflow"
3  else S.top = S.top + 1
4       S[S.top] = x
```

```
POP(S)
1  if STACK-EMPTY(S)
2        error "underflow"
3  else S.top = S.top − 1
4       return S[S.top + 1]
```

```
STACK-EMPTY(S)
1  if S.top == 0
2        return TRUE
3  else return FALSE
```

[source: CLRS [1]]

Array



Abstraction

Abstract Data Types (ADT)

Abstraction-ADT

Container

Abstraction-ADT: Container

Container: contains items

- insert(item)
- delete(item)
- isIn(item)
- isEmpty()
- isFull()
- size()

- Insertion
  - Order of is not important
  - Duplication is allowed
- Deletion
  - Any item can be deleted

Abstraction-ADT

Set

Abstraction-ADT: Set

Container: contains items

- insert(item)
- delete(item)
- isIn(item)
- isEmpty()
- isFull()
- size()

- Insertion
  - Order of is not important
  - Duplication is not allowed
- Deletion
  - Any item can be deleted

Abstraction-ADT

Stack

Abstraction-ADT: Stack

Container: contains items

- push(item) ~~insert(item)~~
- pop() ~~delete(item)~~
- ~~isIn(item)~~
- isEmpty()
- isFull()
- ~~size()~~

- Last-In-First-Out (LIFO)
- Insertion
  - Order of is important
  - Duplication is allowed
- Deletion
  - ~~Any item can be deleted~~
  - Only a specific item can be deleted

Applications

Applications

Balanced parentheses

Balanced Parentheses

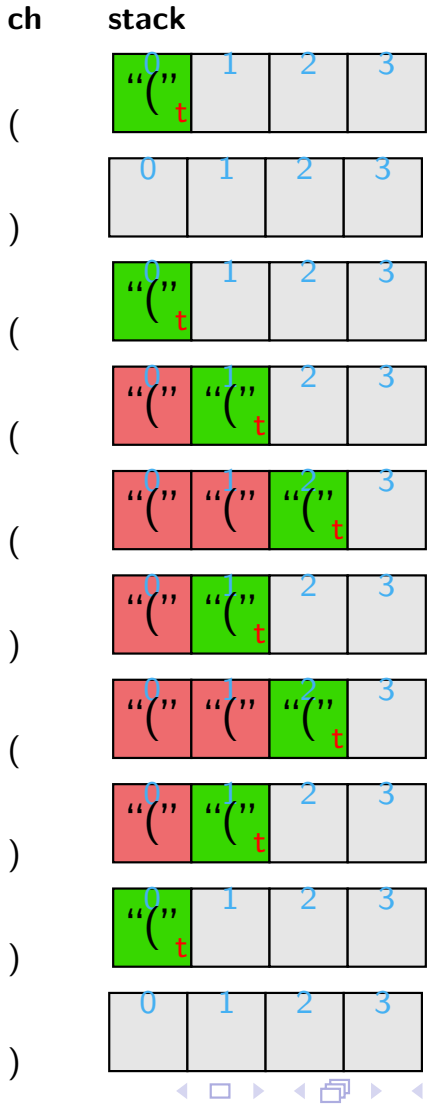
Algorithm 1: Balanced Parentheses

Data: algebraic expression
Result: true if parentheses are balanced

```
1  begin
2      foreach ch in expression from left to right do
3          if ch is an opening parenthesis then
4              push ch on the stack
5          else if ch is a closing parenthesis then
6              cs ← pop the stack
7              if the opening cs and closing ch
                 parentheses don't match then
8                  return FALSE
9      if at the end the stack is empty then
10         return TRUE
11     else
12         return FALSE
```

[Horstmann [4]]

expression: ( ) ((( ) ( )))



Applications

Postfix Evaluation

Infix, Prefix, and Postfix (RPN) Notations

infix notation	**prefix notation**	**postfix notation**
normal math expression	operators followed by operands	operands followed by operators
$a + b$	$+a\ b$	$a\ b+$
$a + b + c$	$+a + b\ c$	$a\ b + c+$
$a + b * c$	$+a * b\ c$	$a\ b\ c * +$
$(a + b) * c$	$* + a\ b\ c$	$a\ b + c*$
$a + b * c/(e - f)$	$+a\ / * b\ c - e\ f$	$a\ b\ c * e\ f - /+$

[HP-15 calculator online [5]]

Postfix notation is also called Reverse Polish Notation (RPN) [6]

[Infix, Prefix and Postfix Expressions [7]]

[Infix to Postfix/Prefix converter]

RPN Evaluation

Use one stack

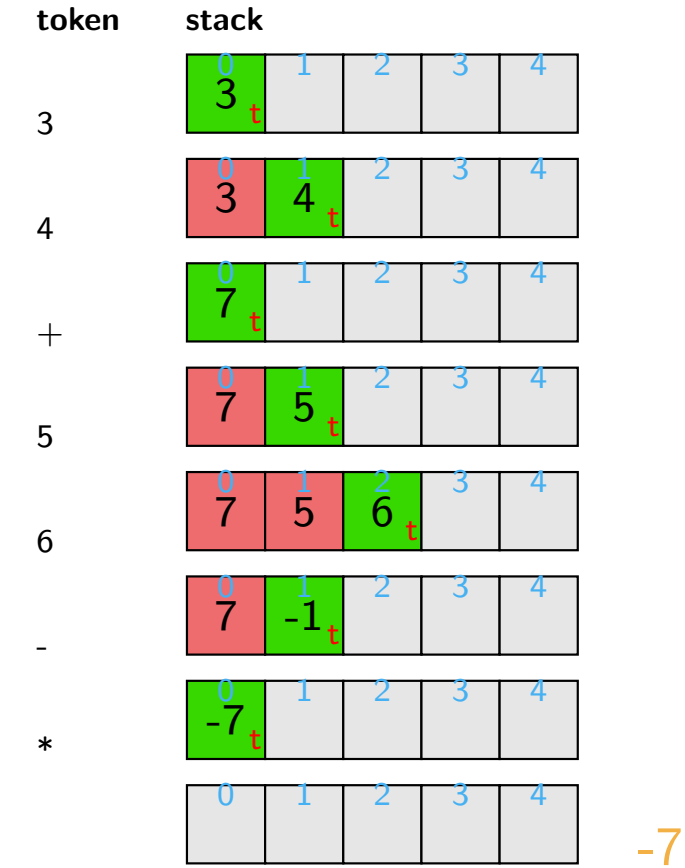
Algorithm 2: RPN Evaluation

Data: postfix expression
Result: value of the expression

```
1 begin
2     while there is a token do
3         if an operator is read then
4             Pop two values off the stack
5             Apply the operator to the two values
6             Push the result back onto the stack
7         else if a number is read then
8             Push it on the stack
9     Pop and display the result
```

[Horstmann [4]]

[HP-15 calculator online [5]]



HP-41CV,
RPN Calculator,
1979-1990

Applications

Infix to postfix conversion

Infix to postfix conversion

Use two stacks

- operator stack
- output stack

Input String	Stack Output	Stack Operator
(f-e)/c*b+a		(
(f-e)/c*b+a	f	(
(f-e)/c*b+a	f	(-
(f-e)/c*b+a	fe	(-
(f-e)/c*b+a	fe-	
(f-e)/c*b+a	fe-	/
(f-e)/c*b+a	fe-c	/
(f-e)/c*b+a	fe-c/	*
(f-e)/c*b+a	fe-c/b	*
(f-e)/c*b+a	fe-c/b*	+
(f-e)/c*b+a	fe-c/b*a	+
(f-e)/c*b+a	fe-c/b*a+	

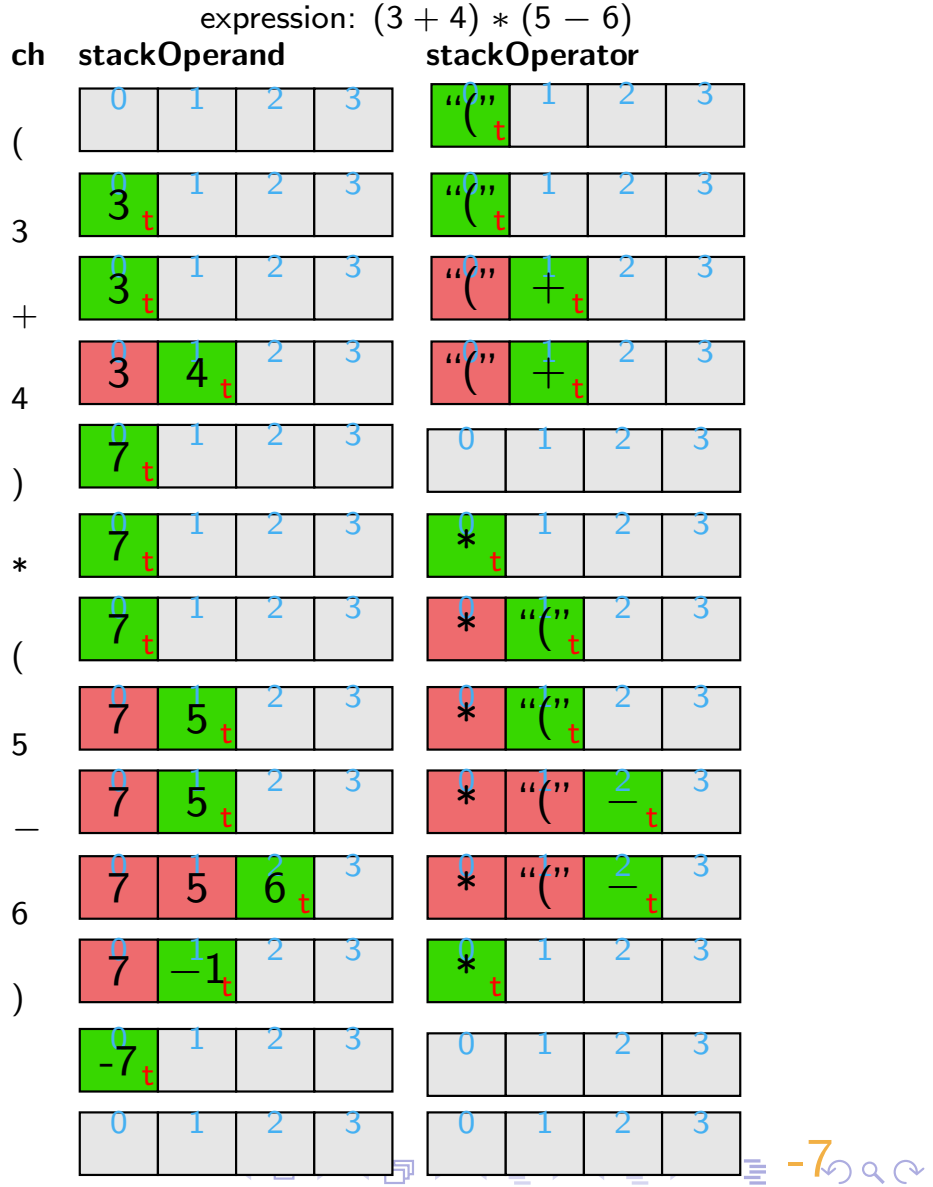
Applications

Infix evaluation

Infix Evaluation

Algorithm 3: Infix Evaluation

```
1  begin
2      if a number is read then
3          Push it on the stackOperand
4      else if a "(" is read then
5          Push it on the stackOperator
6      else if operator op is read then
7          while the top of stackOperator has a
             higher precedence than op do
8              Evaluate the top
9          Push op on stackOperator
10     else if a ")" is read then
11         while the top of stackOperator is not a
             "(" do
12             Evaluate the top.
13         Pop the "("
14     else if there is no more input then
15         while the stackOperator is not empty
             do
16             Evaluate the top
17     Pop and display the result
```

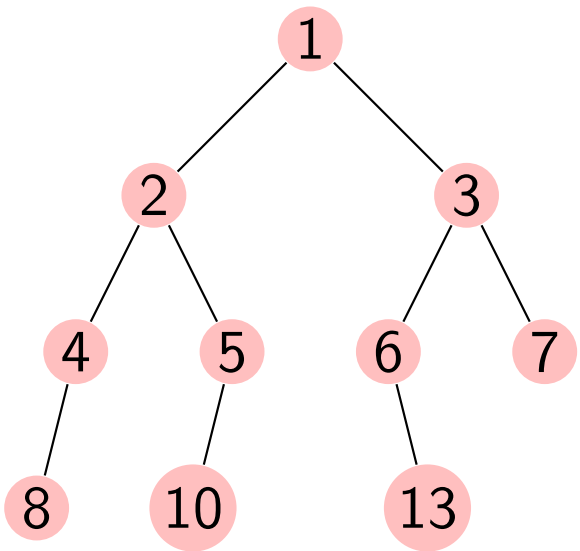


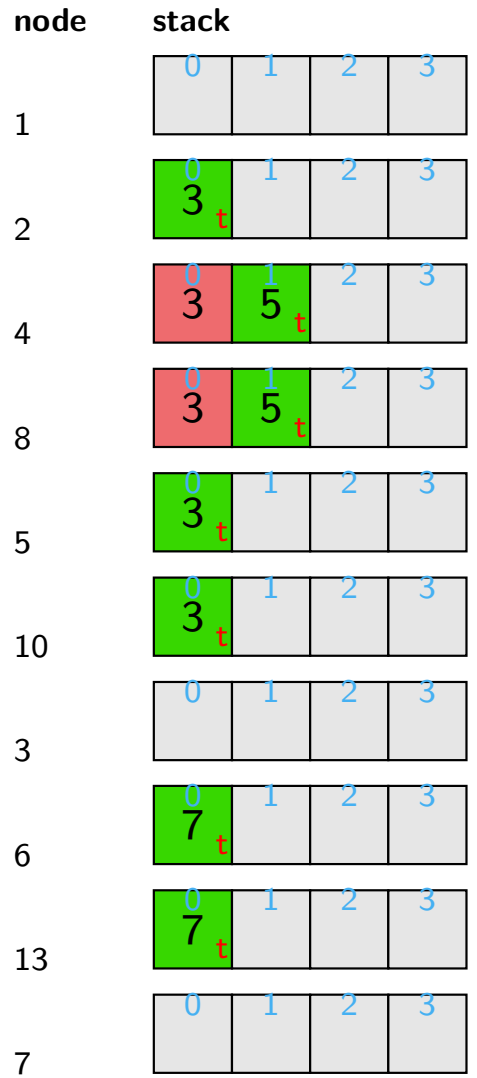
Applications

Backtracking

Decision Tree

- At a decision point, select one path, push the unselected paths to stack.
- If a path unsuccessfully ends, pop the stack for a new path.
- Keep doing as long as there is untested path in the stack.





Web resources

- Problem Solving with Algorithms and Data Structures using Python
  - Home [8]
  - Infix, Prefix and Postfix Expressions [7]
  - What is a Stack [9]
- The Java Tutorials by Oracle
  - Home [10]
  - Arrays [11]
  - Generics [12]
- HP Calculators (An RPN Calculator)
  - [HP-15 calculator online [5]]

References I

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (CLRS)*, 4th ed. MIT Press, 2022.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Addison Wesley, 1974.

[3] E. Horowitz and S. Sahni, *Fundamentals of Data Structures*. Pitman, 1982.

[4] C. S. Horstmann, *Big Java: Early Objects*, 7th ed. John Wiley & Sons, 2019.

[5] HP-15C calculator (online emulator). [Online]. Available: https://hp15c.com/web/hp15c.html

[6] Reverse polish notation. [Online]. Available: https://en.wikipedia.org/wiki/Reverse_Polish_notation

[7] Algorithms and data structures- infix, prefix and postfix expressions. [Online]. Available: https://www.openbookproject.net/books/pythonds/BasicDS/InfixPrefixandPostfixExpressions.html

[8] Algorithms and data structures. [Online]. Available: https://www.openbookproject.net/books/pythonds/index.html

[9] Algorithms and data structures- what is a stack. [Online]. Available: https://www.openbookproject.net/books/pythonds/BasicDS/WhatisaStack.html

[10] The java tutorials. [Online]. Available: https://docs.oracle.com/javase/tutorial/

[11] The java tutorials-arrays. [Online]. Available: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

References II

[12] The java tutorials-generics. [Online]. Available: https://docs.oracle.com/javase/tutorial/java/generics/index.html