Recursion

Algorithms - tutorial #9

Jakub Sýkora

Problem #1

Add + - symbols between number of list to get sum

- List [8, 4, 6, 2]
- Sum = 8
- Possible solutions
  - 8 + 4 - 6 + 2
  - 8 - 4 + 6 - 2

Problem #1

Add + - symbols between number of list to get sum

- List [8, 4, 6, 2]
- Idea - generating all possible {+,-} combinations
- Approach #1 - iteration
- Approach #1 - recursion - easier to implement

Problem #1

Inefficient recursion

- Pass previous +,- symbols in arguments as list
- Problem
  - recursion only use one list (at time)
  - list of symbol is copied at each level

Problem #1

More efficient recursion

- Keep symbols in single list of numbers size
- Recursion arguments
  - Next symbol (number) index
  - Current sum (faster)
- Order (program flow) of recursion assures single list is erased correctly

Problem #2

Binary tree - evaluate expression

- Binary tree represents some arithmetic expression
- Each node is operator or number
- Evaluation in prefix notation

Problem #3

Evaluation of postfix notation

- 4 3 * = 12
- 4 5 + 3 * = 27
- Solution with stack
  - Place operand on stack
  - Pop two operands to evaluate operater, push result

Homework

Evaluation of prefix notation

- Use recursion to evaluate expression in prefix notation
  - * + 2 3 4 = 20
- Solutions using binary tree representation not accepted