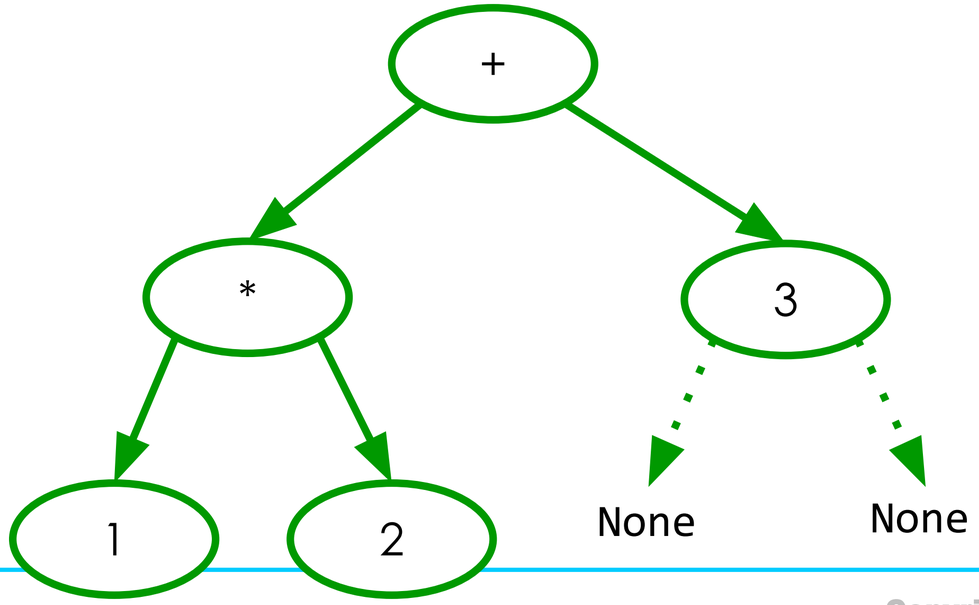
2:電卓を作ってみよう

Copyright © 2024 by INIAD

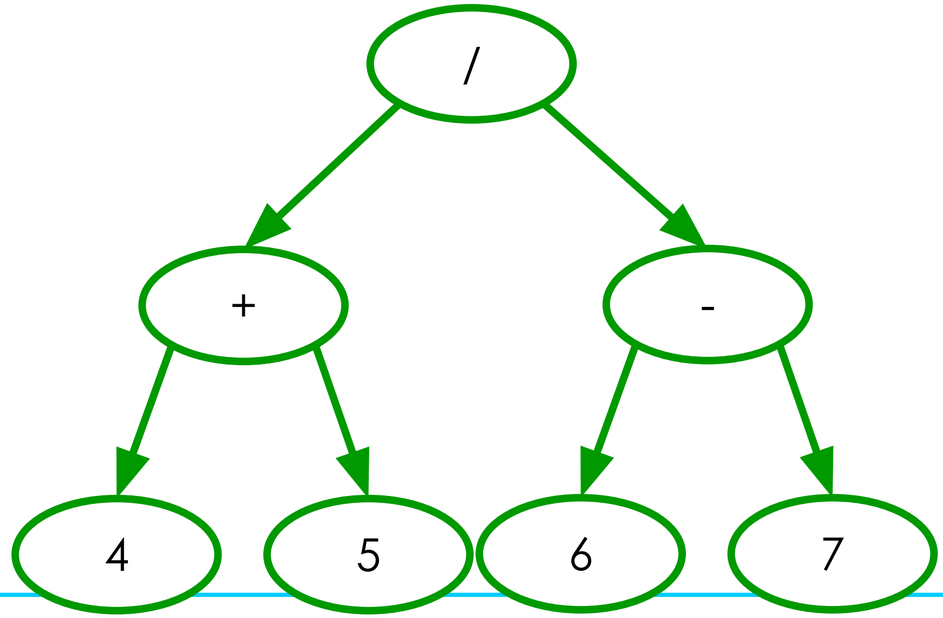
四則演算の表現方法

- 四則演算の計算は木構造を使って表現することができます
- 二分木を使い、ノードに数値もしくは演算子(+, -, *, /)を持たせると演算子の優先順位を含めて表現できるので便利です
  - 子ノードがない場合はNoneとします

1 * 2 + 3



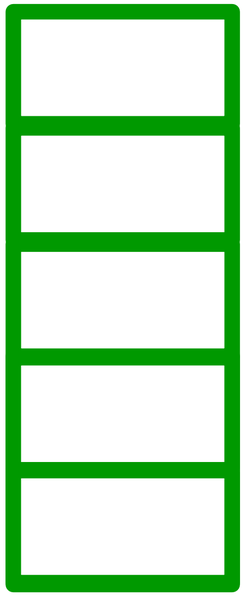
(4 + 5) / (6 - 7)

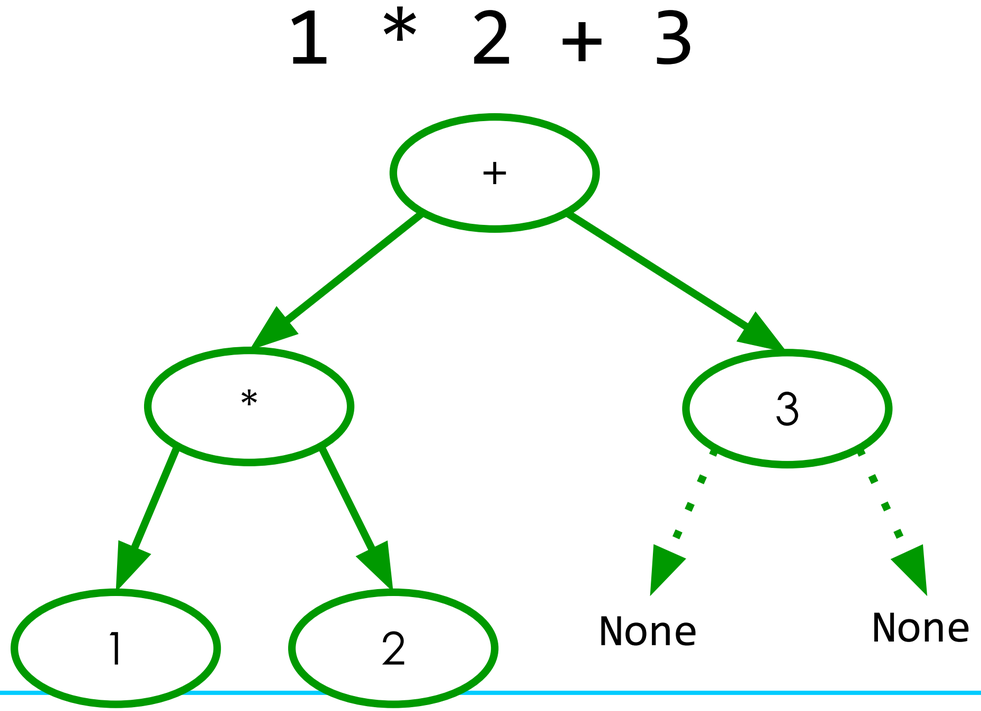


Copyright © 2024 by INIAD

逆ポーランド記法による四則演算の計算

- 木構造で表現された四則演算は以下の手順で計算することができる
  - 深さ優先探索（帰りがけ順の探索）で各ノードを探索する
  - ノードのvalueが値であればスタックにPush
  - 演算子であれば、引数分スタックからPopして適用し、戻り値をPushする

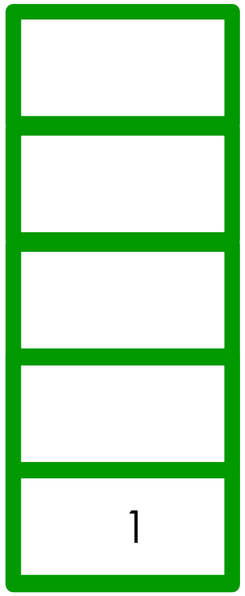


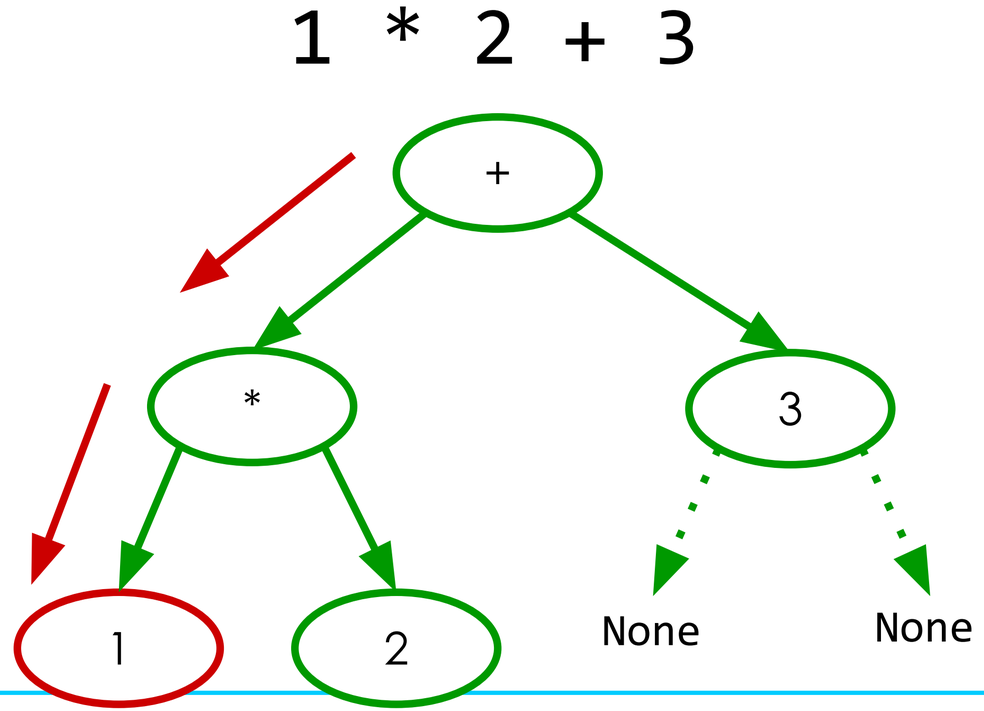


Copyright © 2024 by INIAD

逆ポーランド記法による四則演算の計算

- 木構造で表現された四則演算は以下の手順で計算することができる
  - 深さ優先探索（帰りがけ順の探索）で各ノードを探索する
  - ノードのvalueが値であればスタックにPush
  - 演算子であれば、引数分スタックからPopして適用し、戻り値をPushする

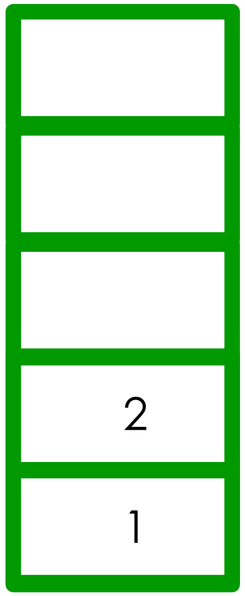


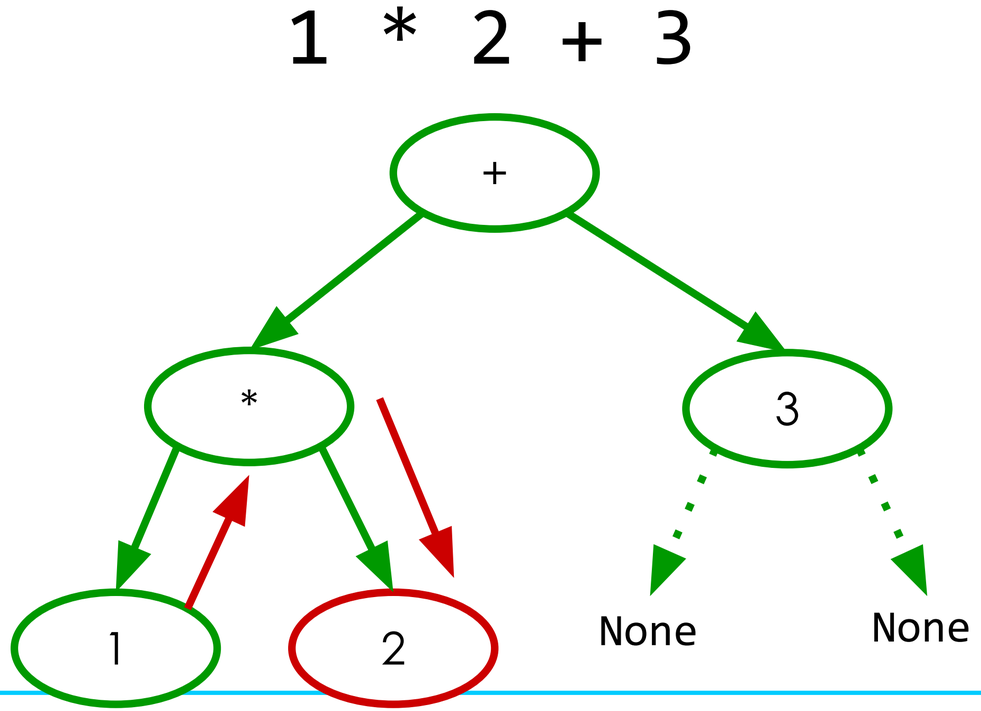


Copyright © 2024 by INIAD

逆ポーランド記法による四則演算の計算

- 木構造で表現された四則演算は以下の手順で計算することができる
  - 深さ優先探索（帰りがけ順の探索）で各ノードを探索する
  - ノードのvalueが値であればスタックにPush
  - 演算子であれば、引数分スタックからPopして適用し、戻り値をPushする

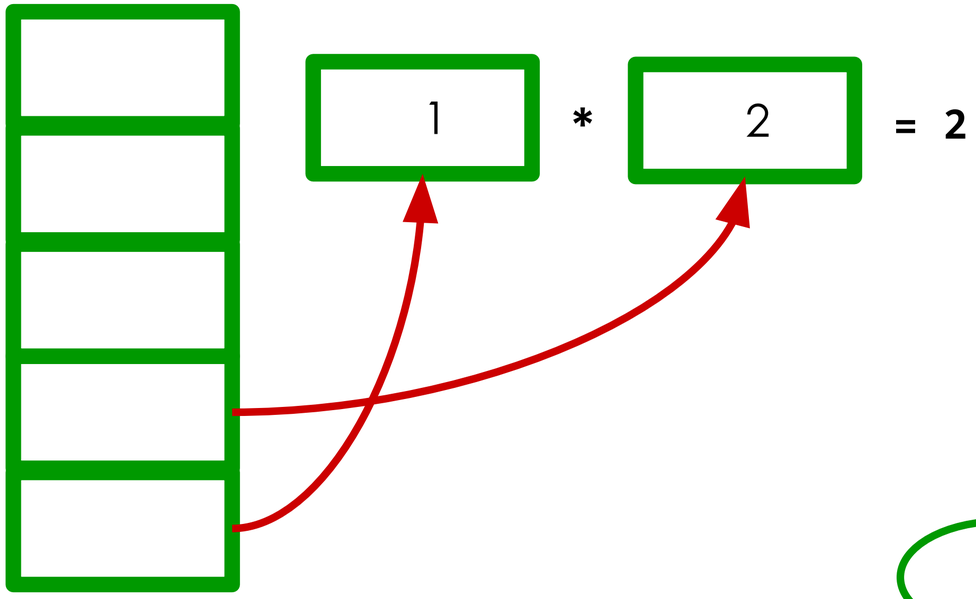


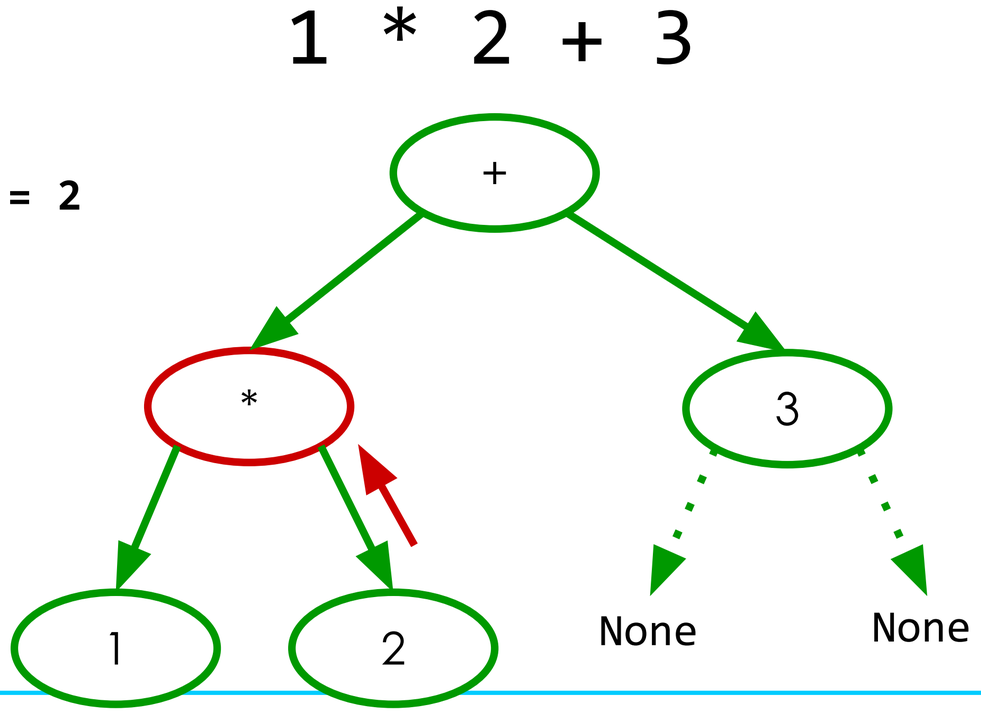


Copyright © 2024 by INIAD

逆ポーランド記法による四則演算の計算

- 木構造で表現された四則演算は以下の手順で計算することができる
  - 深さ優先探索（帰りがけ順の探索）で各ノードを探索する
  - ノードのvalueが値であればスタックにPush
  - 演算子であれば、引数分スタックからPopして適用し、戻り値をPushする

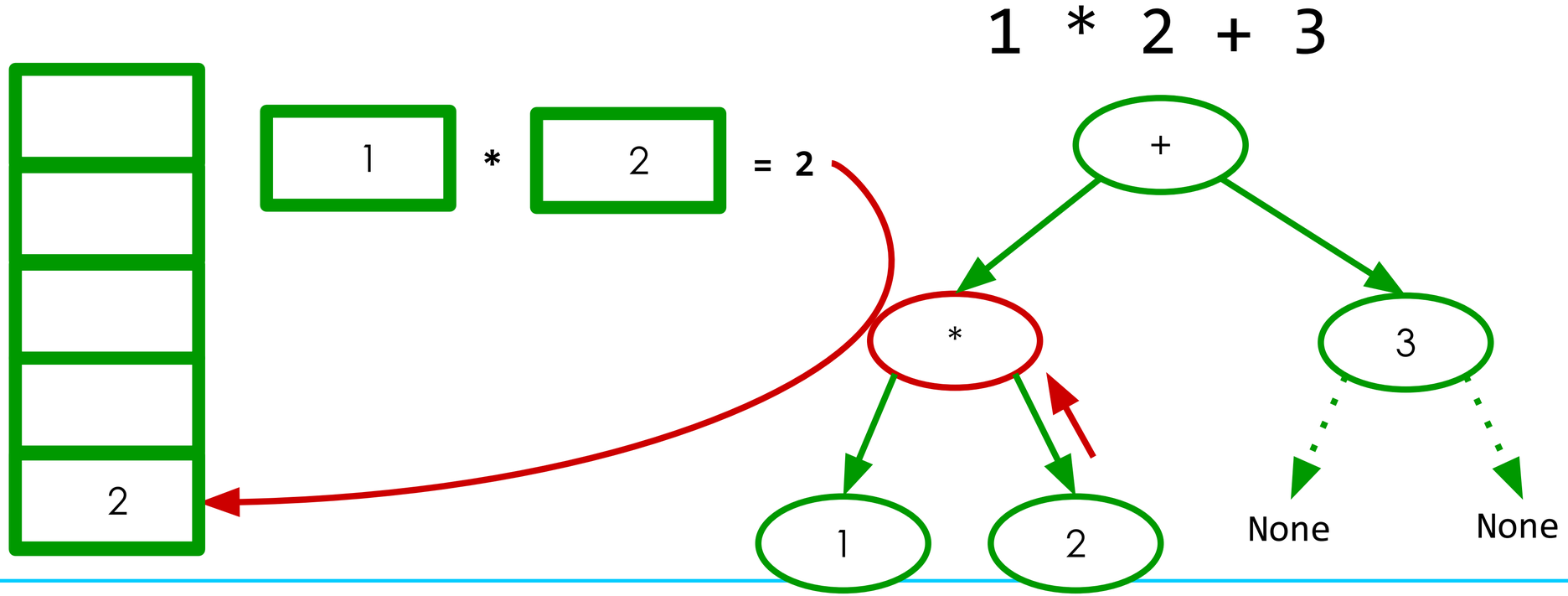




Copyright © 2024 by INIAD

逆ポーランド記法による四則演算の計算

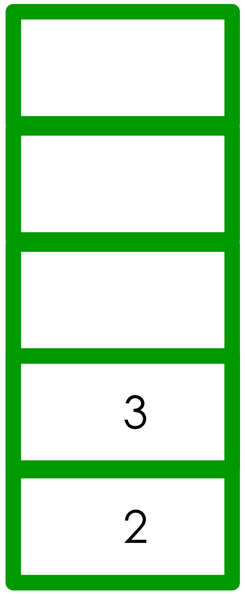
- 木構造で表現された四則演算は以下の手順で計算することができる
  - 深さ優先探索（帰りがけ順の探索）で各ノードを探索する
  - ノードのvalueが値であればスタックにPush
  - 演算子であれば、引数分スタックからPopして適用し、戻り値をPushする

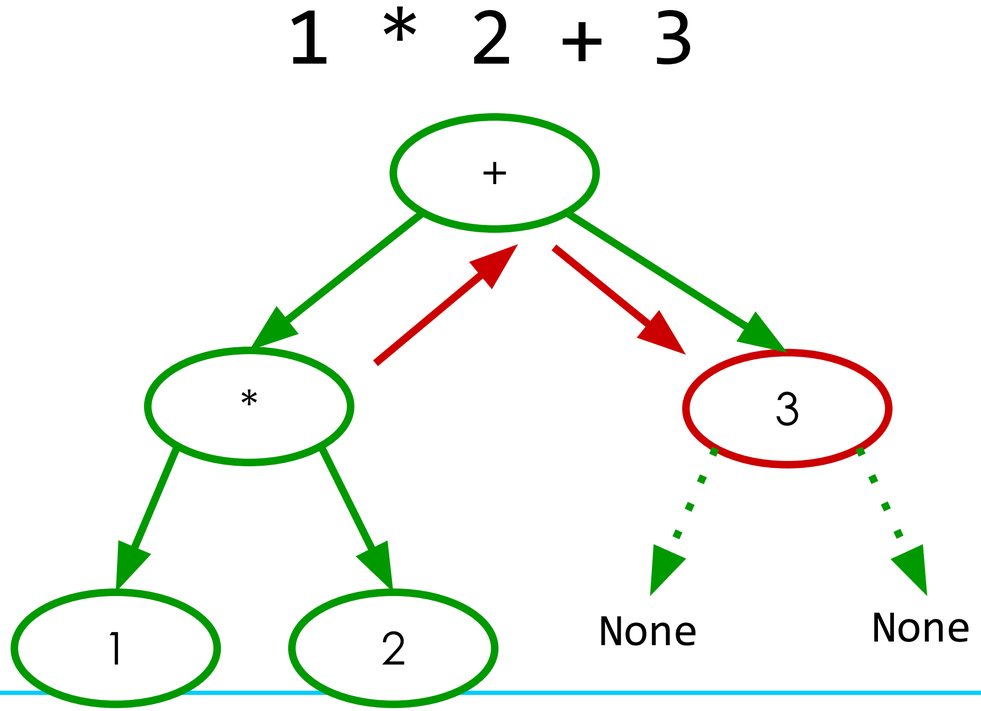


Copyright © 2024 by INIAD

逆ポーランド記法による四則演算の計算

- 木構造で表現された四則演算は以下の手順で計算することができる
  - 深さ優先探索（帰りがけ順の探索）で各ノードを探索する
  - ノードのvalueが値であればスタックにPush
  - 演算子であれば、引数分スタックからPopして適用し、戻り値をPushする

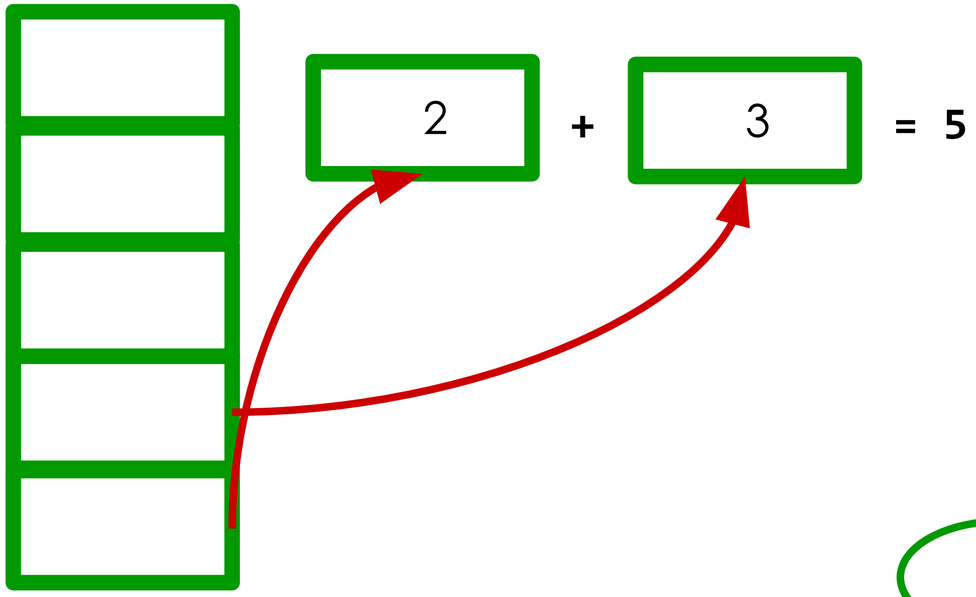


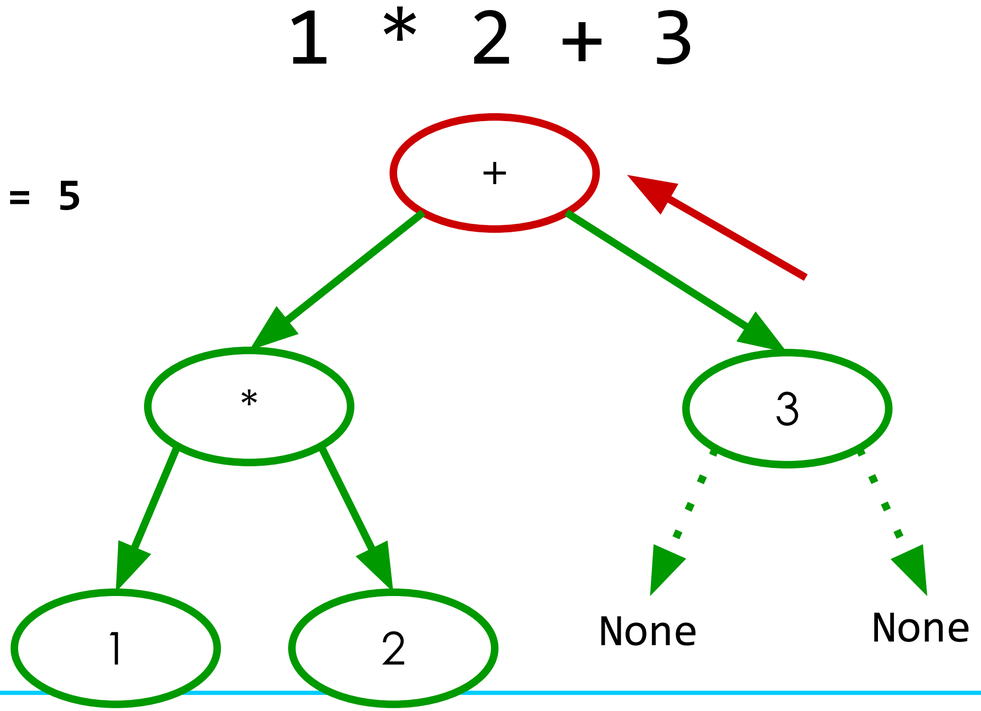


Copyright © 2024 by INIAD

逆ポーランド記法による四則演算の計算

- 木構造で表現された四則演算は以下の手順で計算することができる
  - 深さ優先探索（帰りがけ順の探索）で各ノードを探索する
  - ノードのvalueが値であればスタックにPush
  - 演算子であれば、引数分スタックからPopして適用し、戻り値をPushする

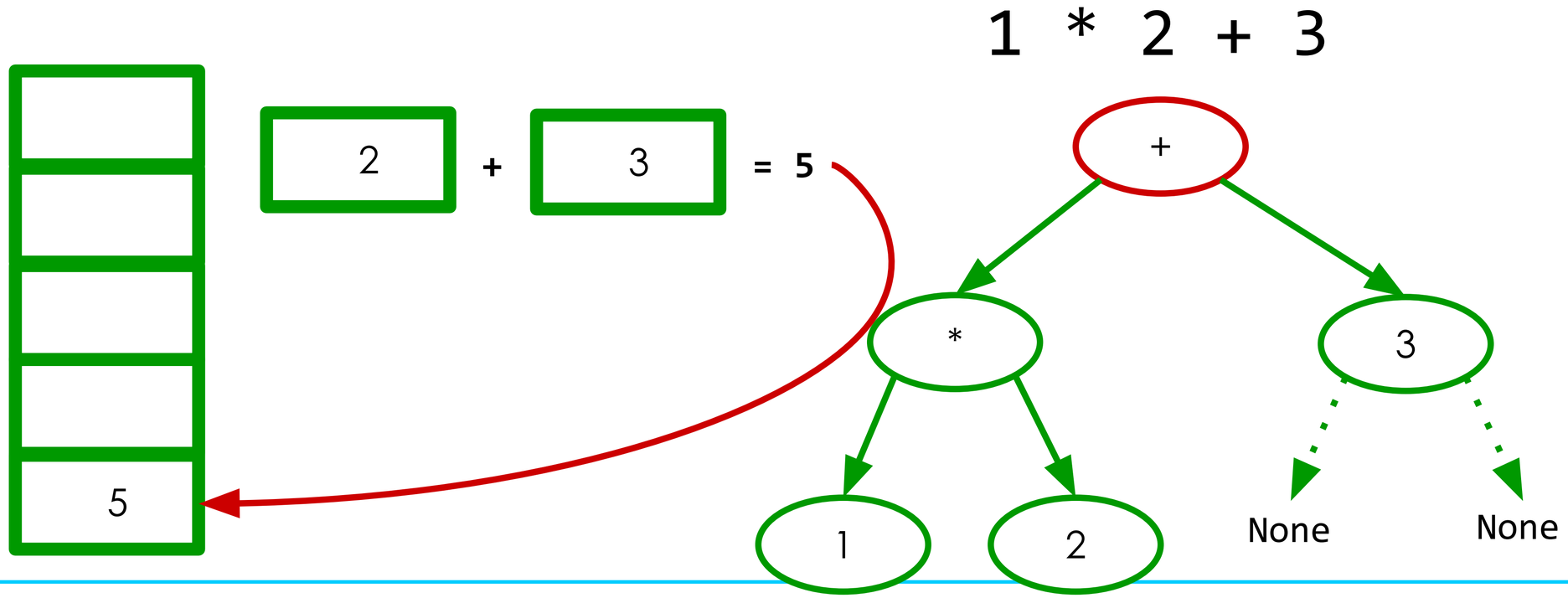




Copyright © 2024 by INIAD

逆ポーランド記法による四則演算の計算

- 木構造で表現された四則演算は以下の手順で計算することができる
  - 深さ優先探索（帰りがけ順の探索）で各ノードを探索する
  - ノードのvalueが値であればスタックにPush
  - 演算子であれば、引数分スタックからPopして適用し、戻り値をPushする



Copyright © 2024 by INIAD

課題2

- 以下の数式を木構造で表現し、スタックを用いて計算してください
- 計算に用いたプログラムを提出してください
  - ((9 – 3) + 1) * 6
  - 5 * 6 / (7 - 4)
  - 1 + 2 * 3 – 4 / 5 + (6 – 7) * 8

Copyright © 2024 by INIAD

課題2 ヒント

Code

```
def compute(node, stack):
    if node.left != None:     # 左のsubtreeにあるノードを計算する
        __(A)__
    if node.right != None:    # 右のsubtreeにあるノードを計算する
            __(B)__
     # stackを使って計算する
    if node.value == '+':
            r=__(C)__
            l=__(D)__
            stack.append(__(E)__)
    elif node.value == '-':
            r=__(C)__
            l=__(D)__
            stack.append(__(F)__)
    elif __(G)__ :      # 掛け算('*')の計算をする
            r=__(C)__
            l=__(D)__
            stack.append(__(H)__)
   __(I)__ :            # 割り算('/')の計算をする
            r=__(C)__
            l=__(D)__
            stack.append(__(J)__)
    else:
              __(K)__          # オペランドと仮定してスタックに積む
```

Code実行文

```
# (* (+ (- 9 3) 1) 6) == ((9 – 3) + 1) * 6 == 42
tree = Node('*', Node('+', Node('-', Node(9), Node(3)), Node(1)), Node(6))

from collections import deque
stack = deque()
compute(tree, stack)
print(stack[0])
```

実行結果

```
42
```

© 2024 by INIAD