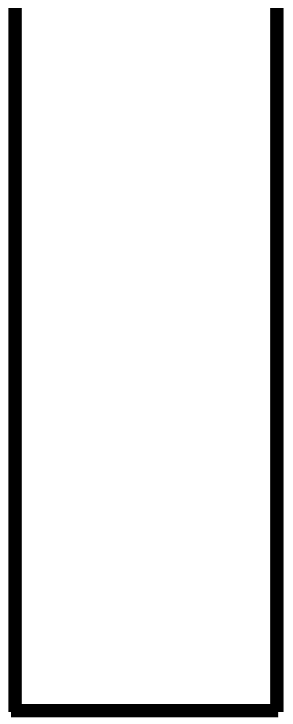
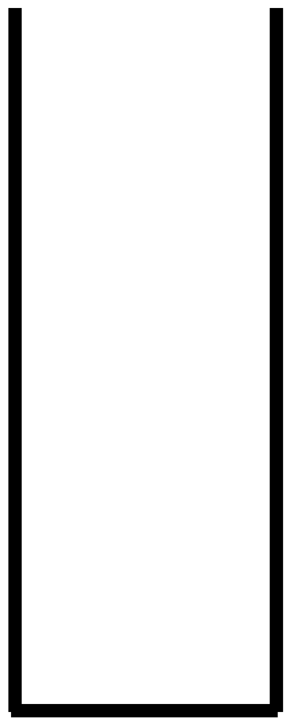
Building an Expression Tree

$$3 \ 4 \ - \ 2 \ 5 \ + \ *$$

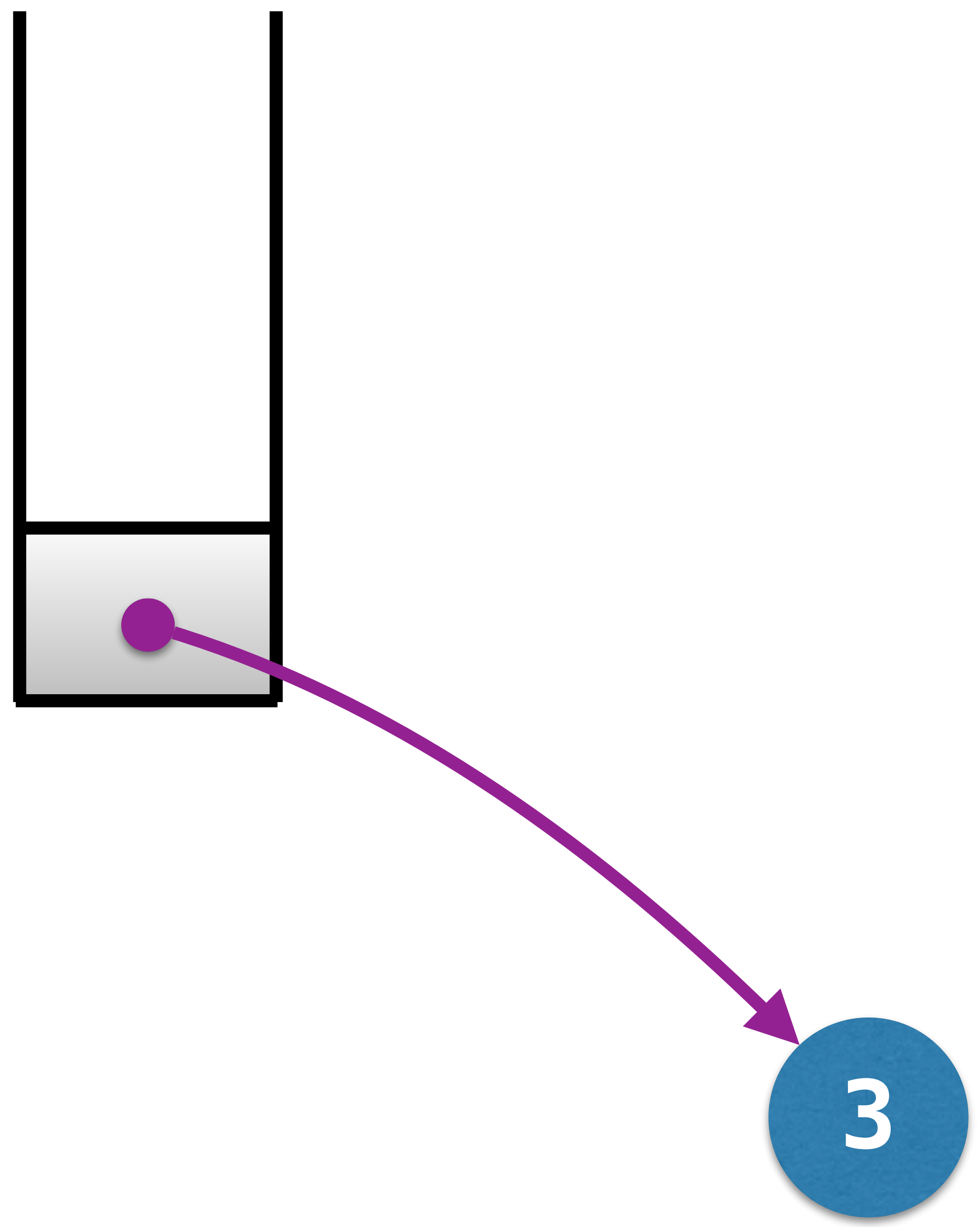
3 4 − 2 5 + *



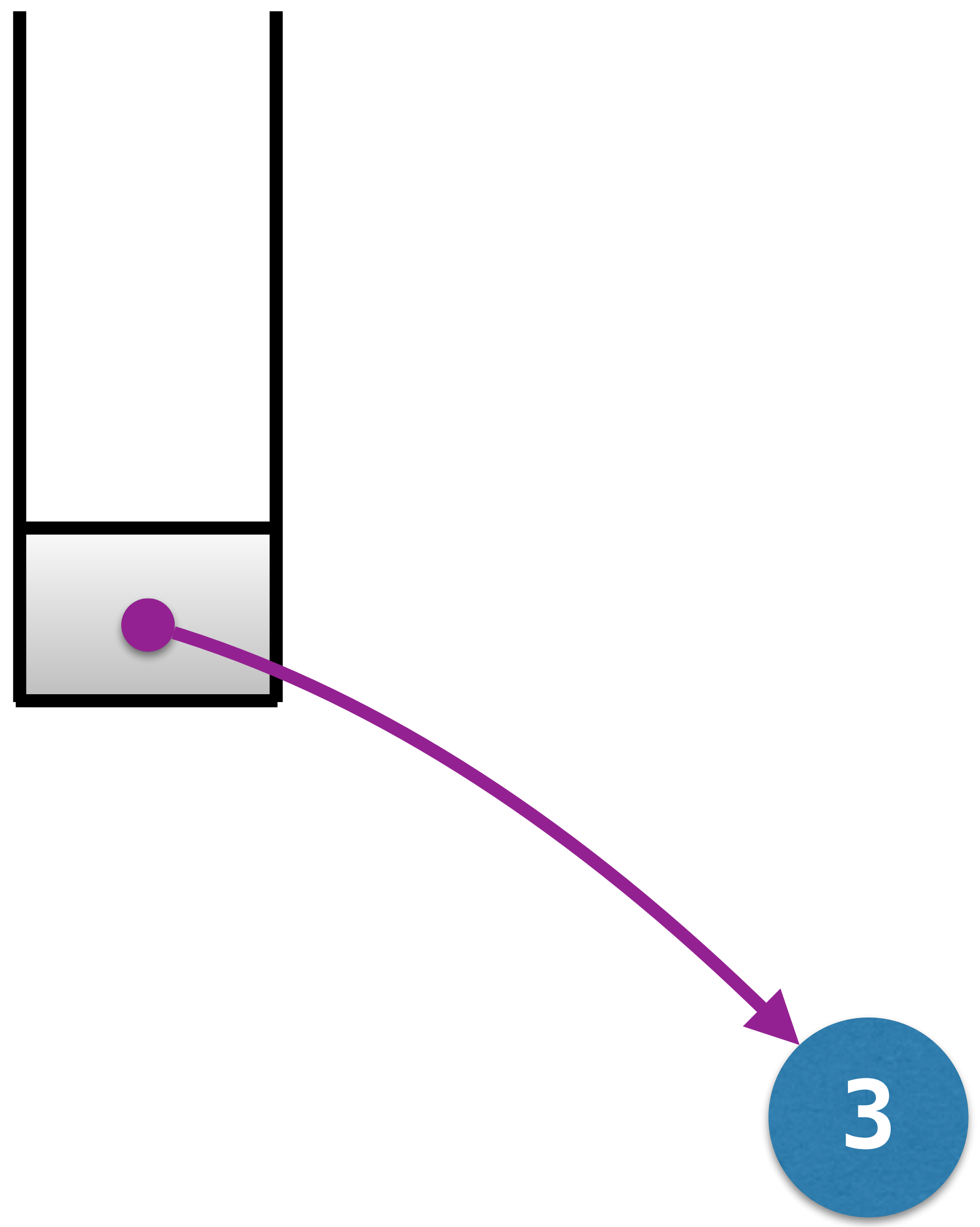
3 4 − 2 5 + *



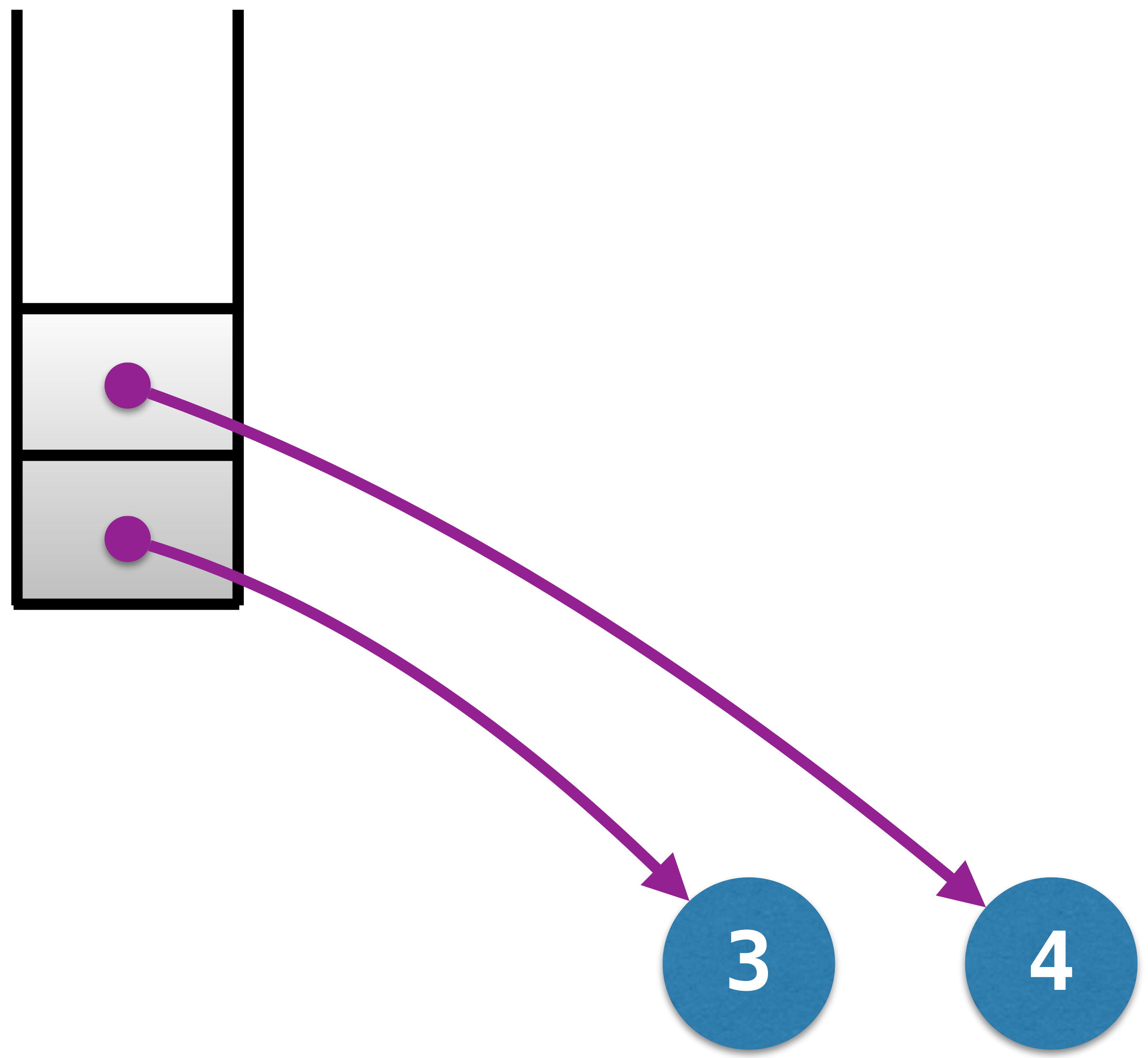
3 4 – 2 5 + *



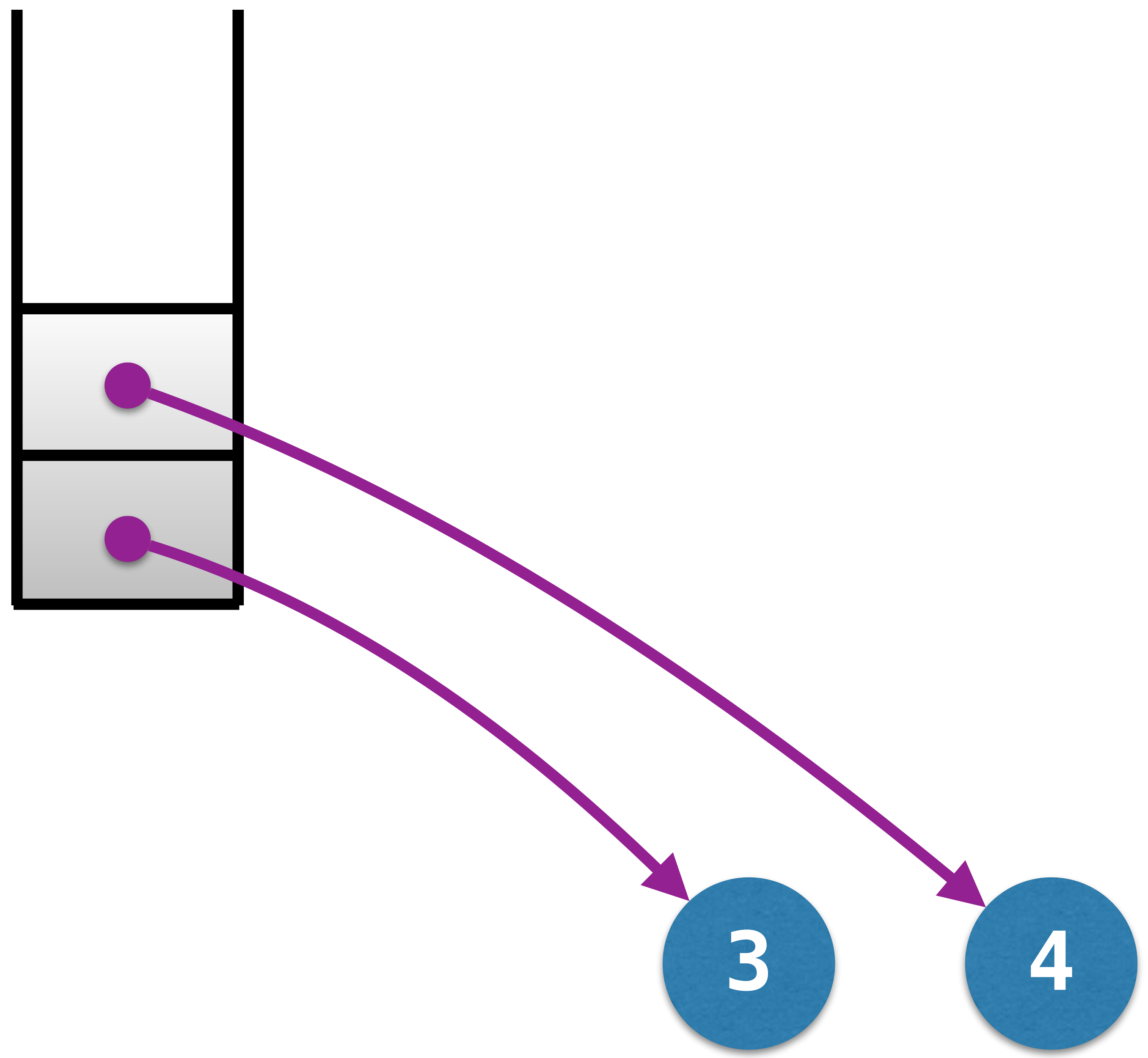
3 **4** − 2 5 + *



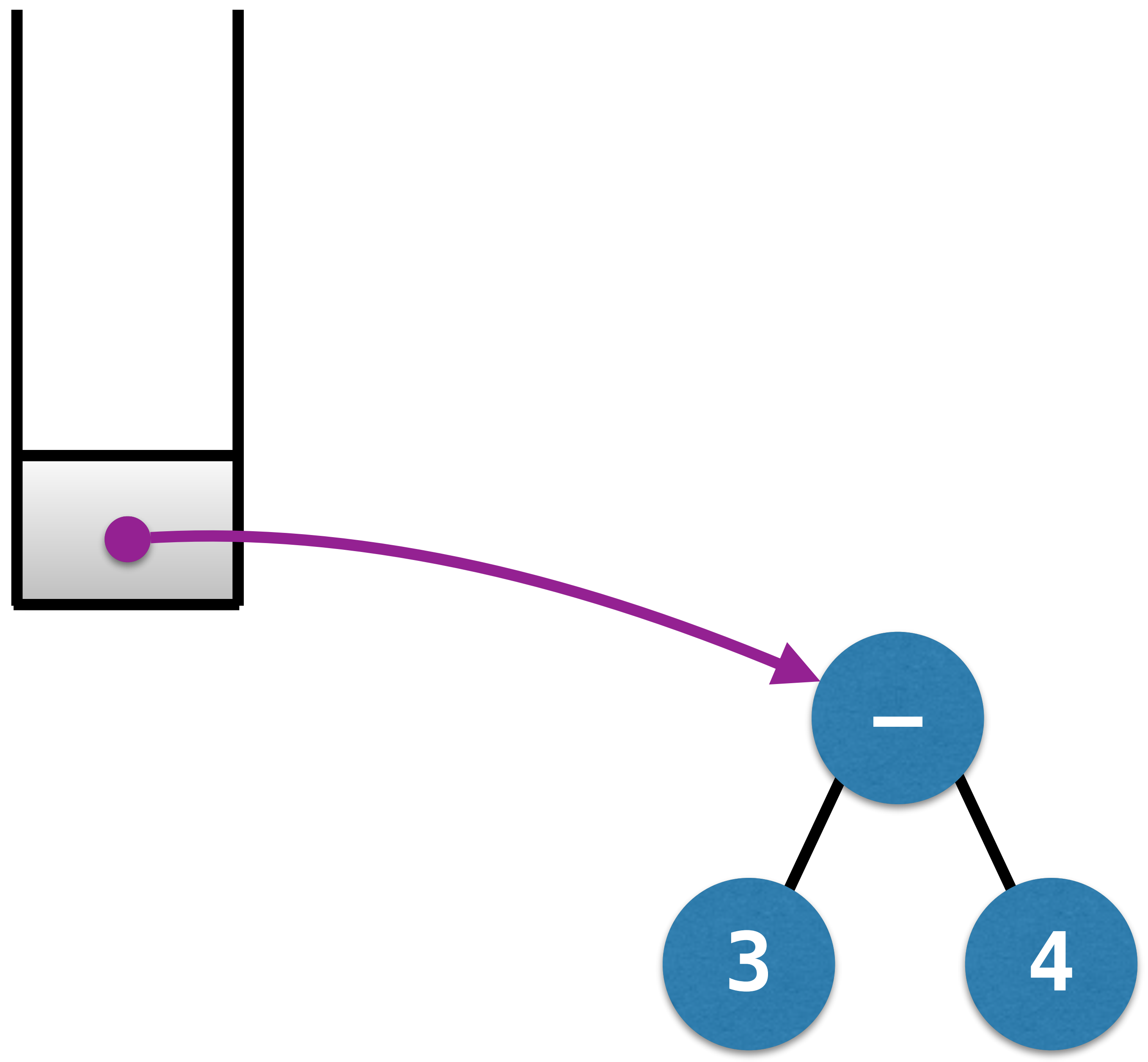
3 **4** − 2 5 + *



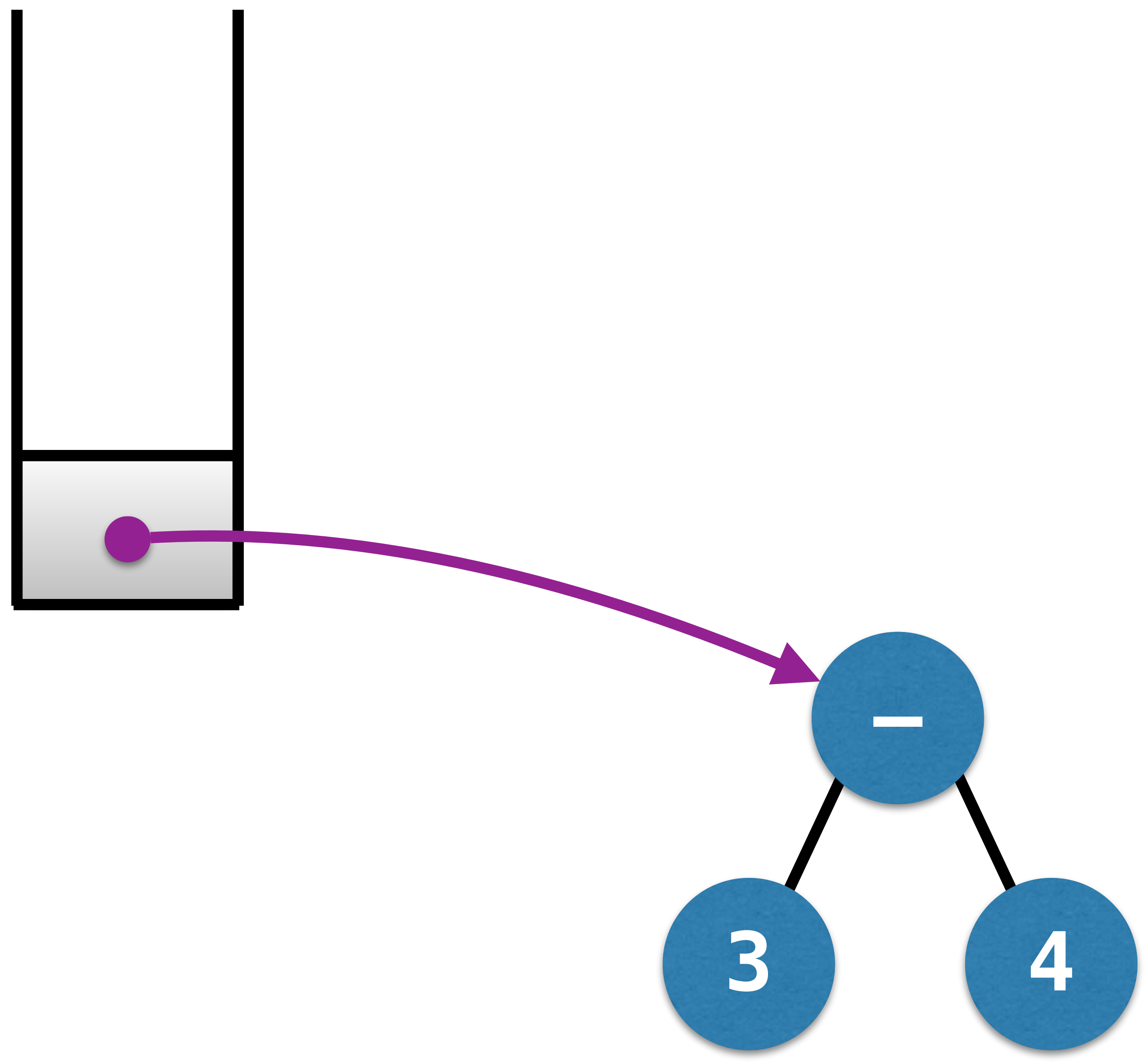
3 4 − 2 5 + *



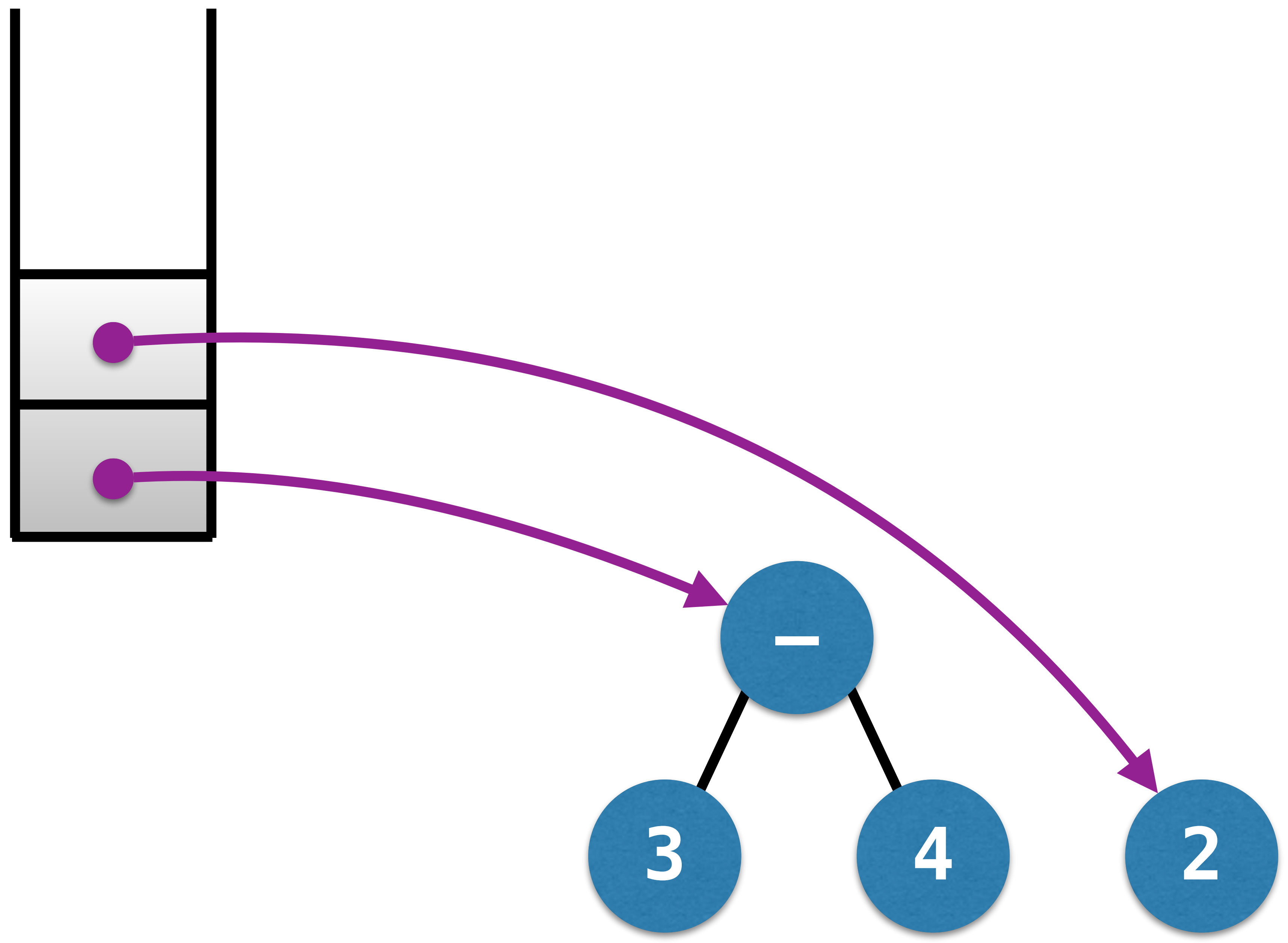
3 4 – 2 5 + *



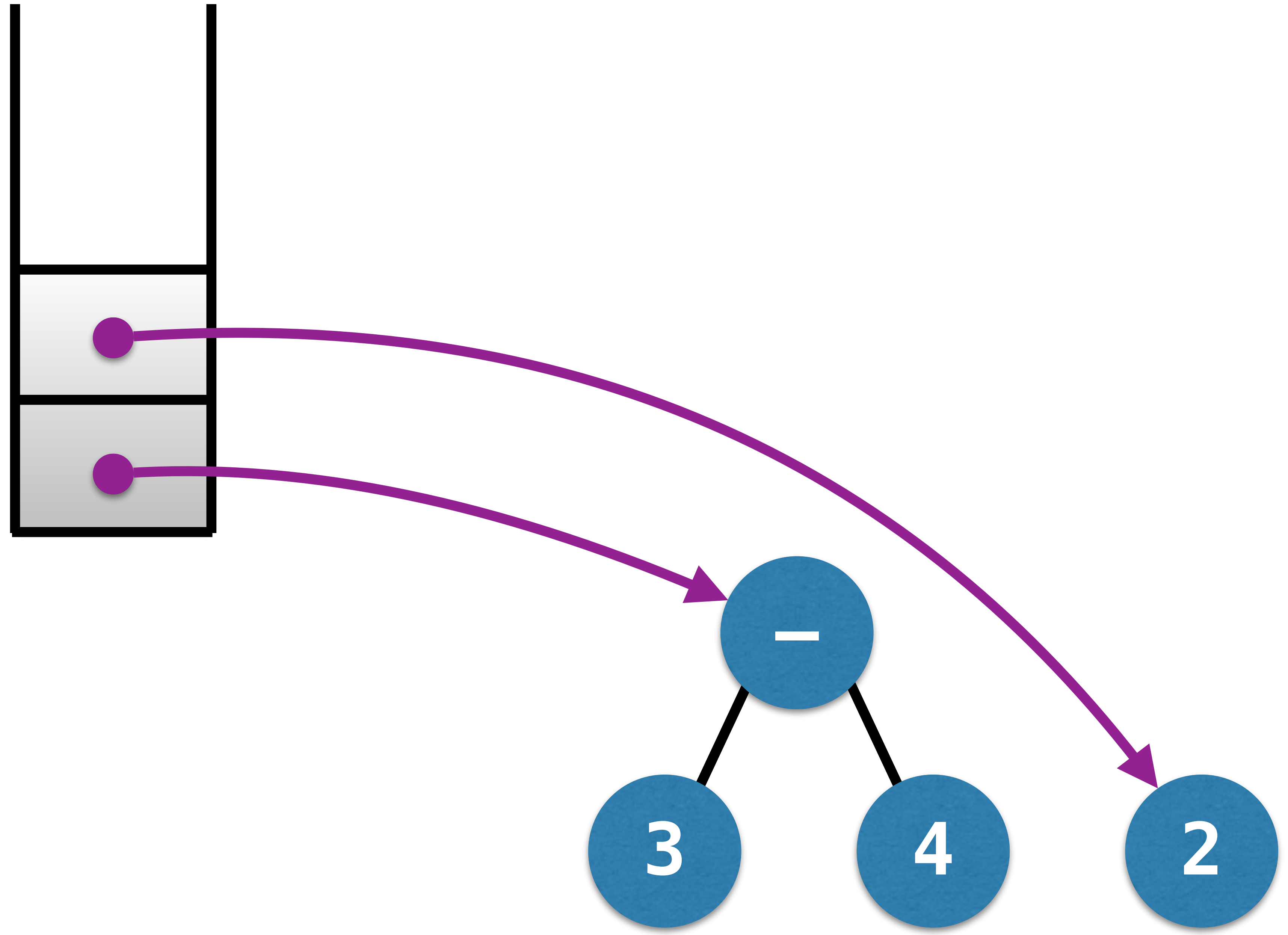
3 4 – 2 5 + *



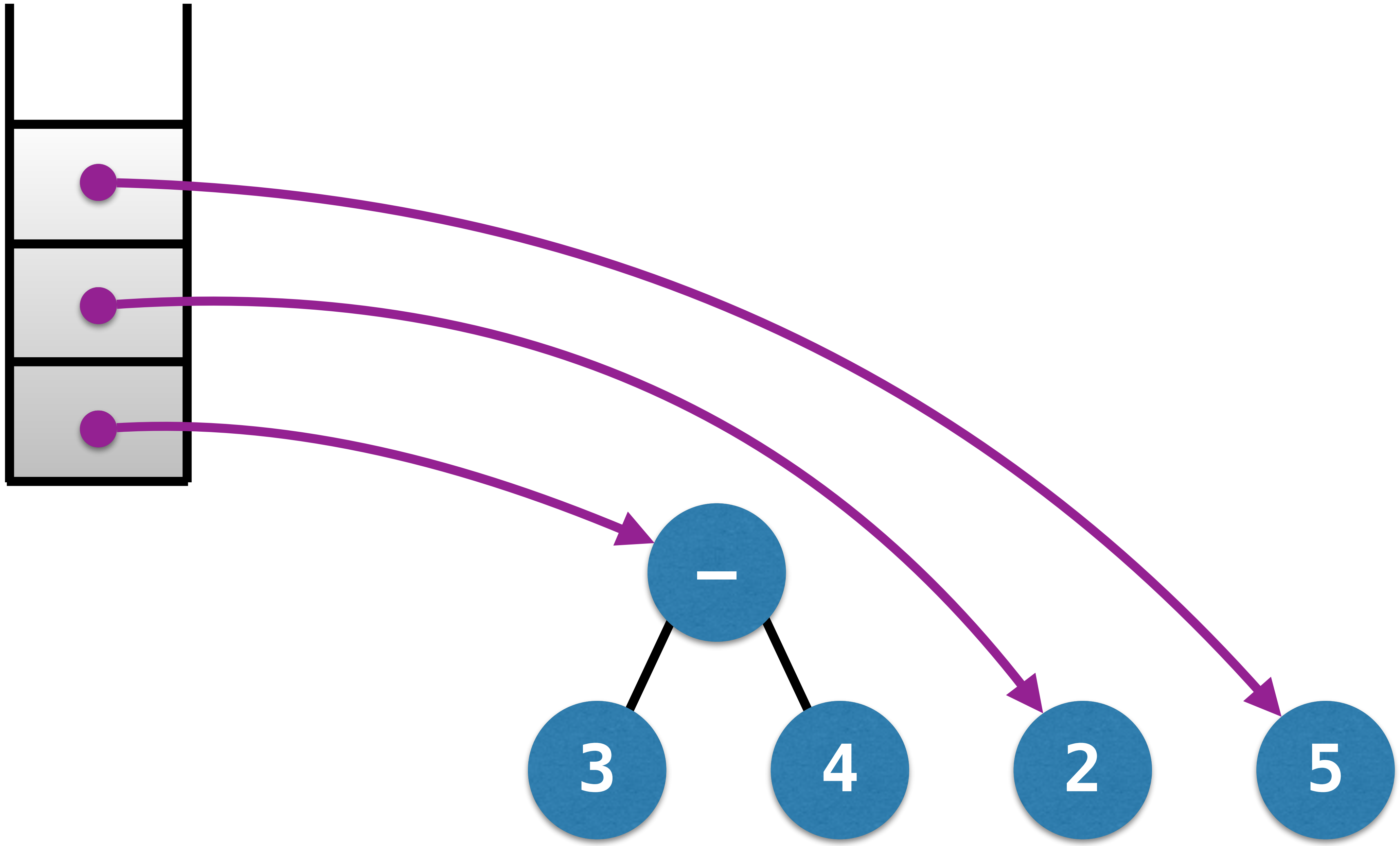
3 4 – 2 5 + *



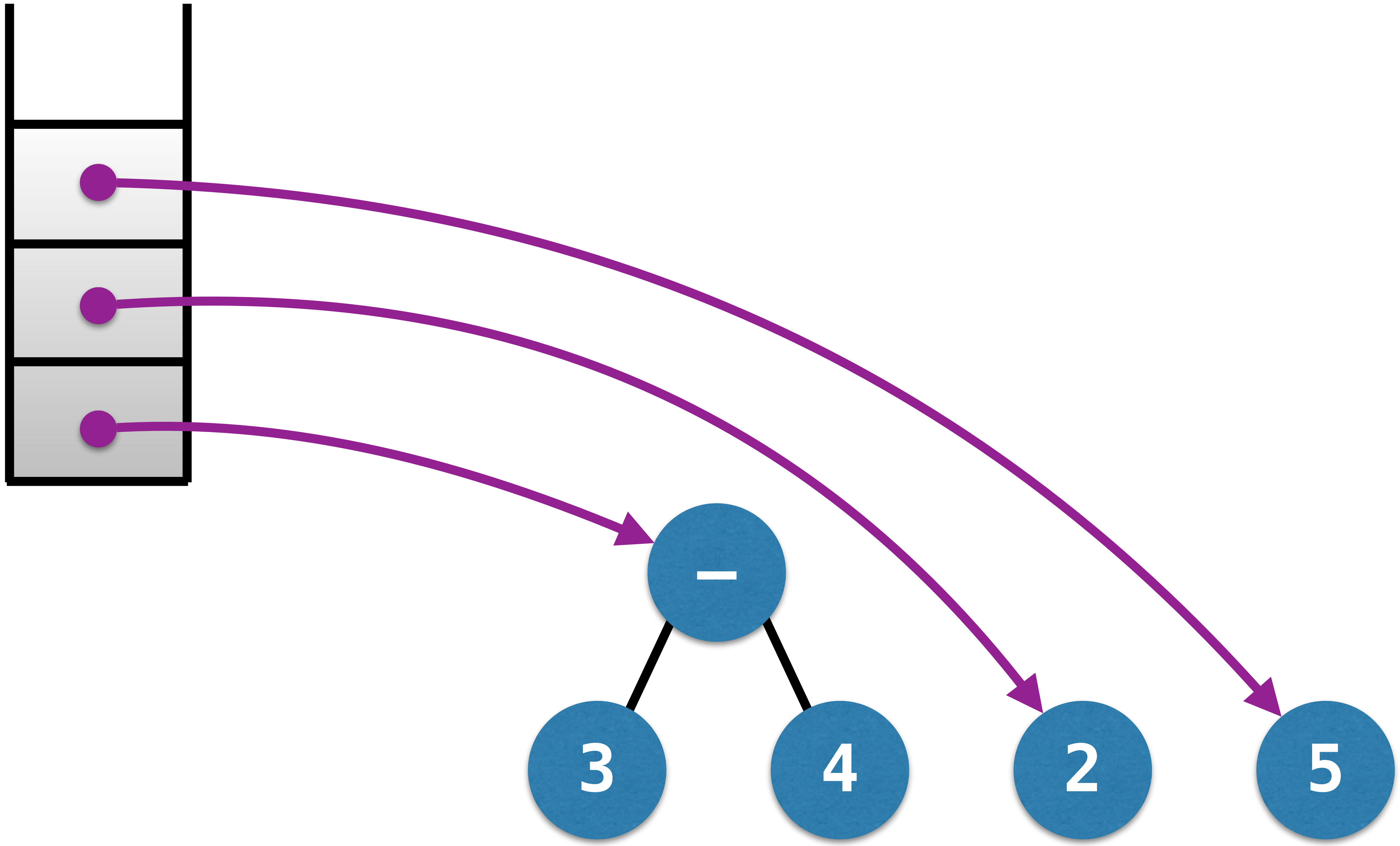
3 4 – 2 **5** + *



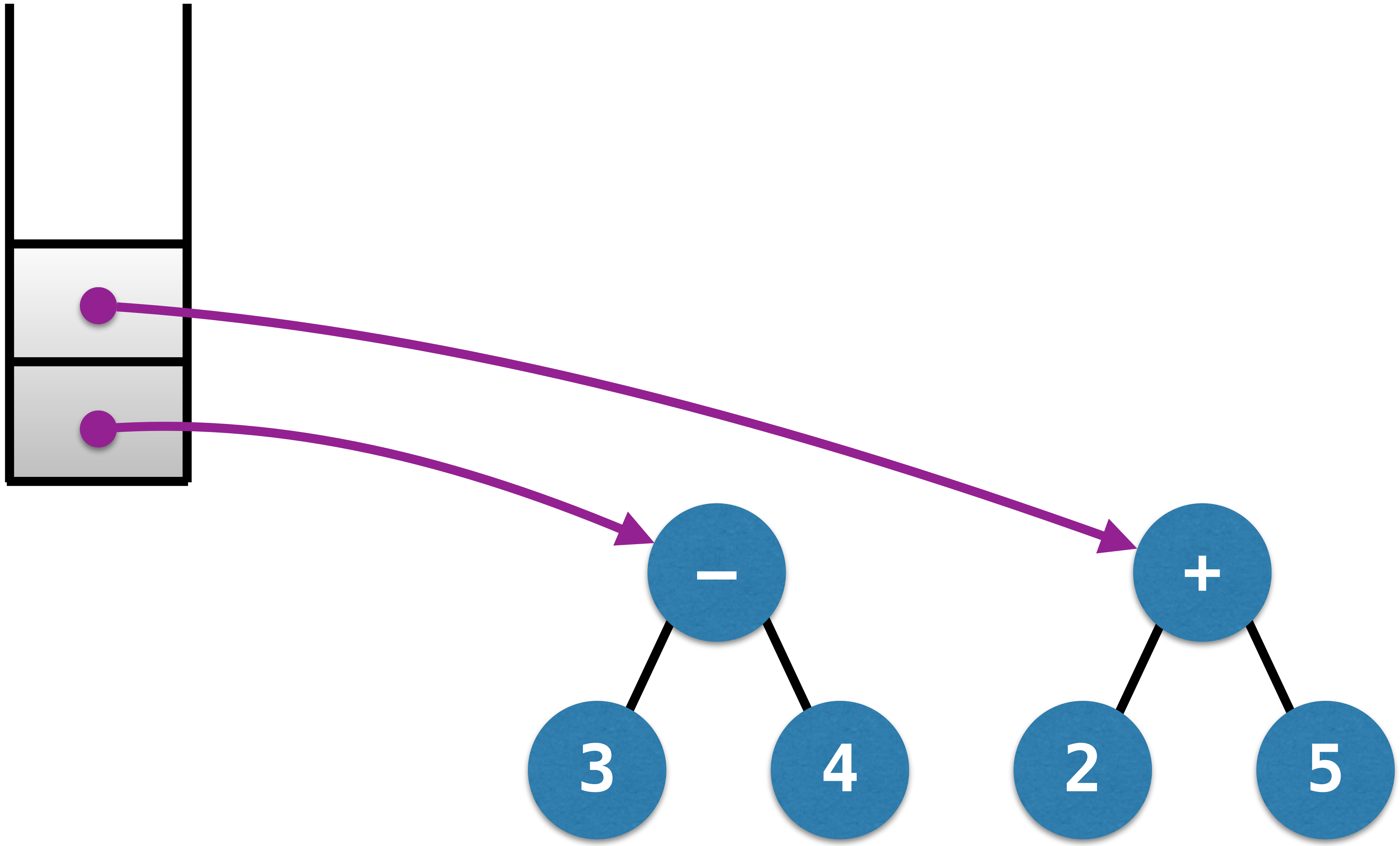
3 4 – 2 **5** + *



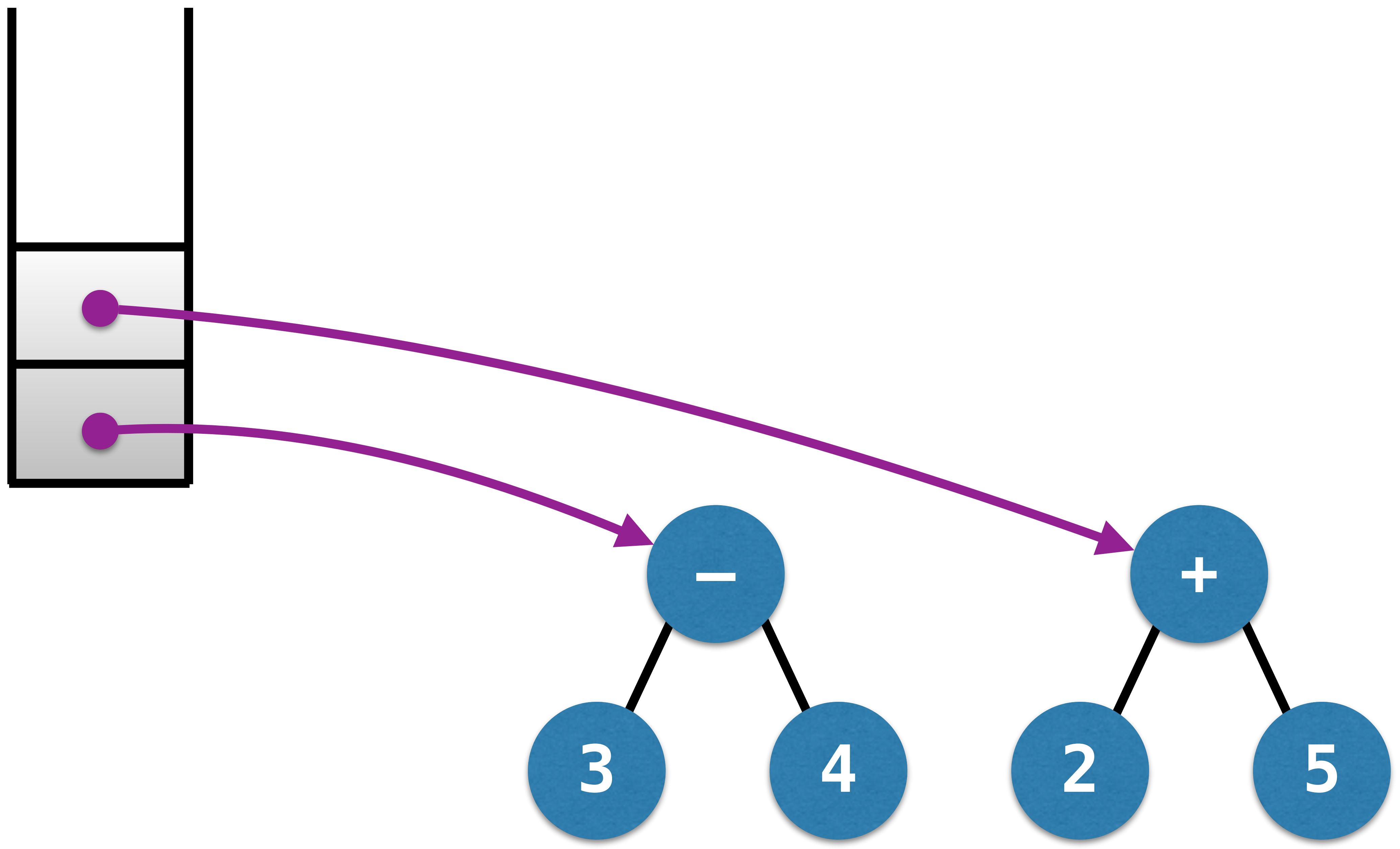
3 4 – 2 5 + *



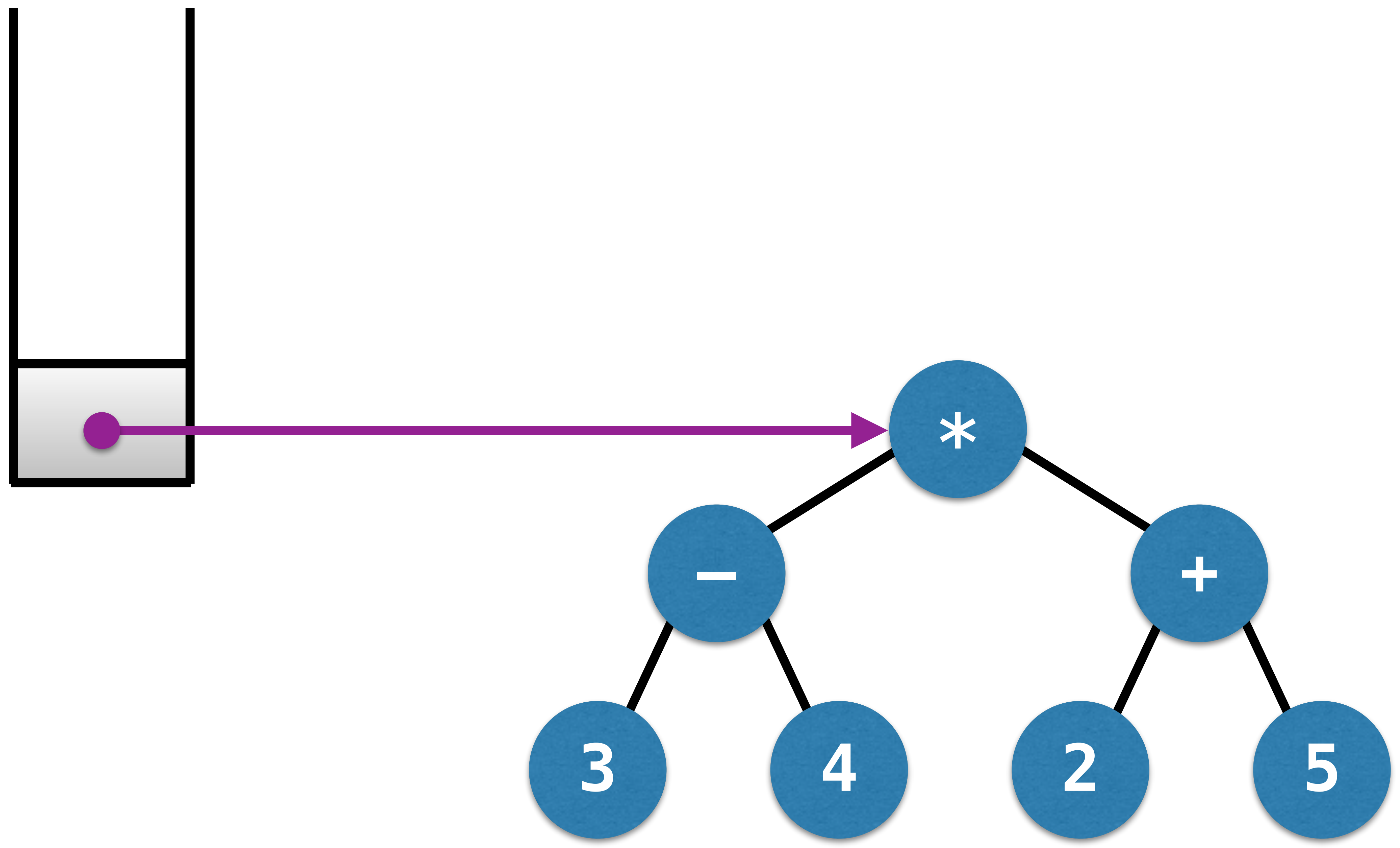
3 4 − 2 5 + *



3 4 − 2 5 + *



3 4 – 2 5 + *



Evaluating an expression tree

Evaluating an expression tree

- If the root is a leaf, it must be a number v.

Evaluating an expression tree

- If the root is a leaf, it must be a number v.
  - Return v.

Evaluating an expression tree

- If the root is a leaf, it must be a number v.
  - Return v.
- Otherwise, the root must be a binary operator op.

Evaluating an expression tree

- If the root is a leaf, it must be a number v.
  - Return v.
- Otherwise, the root must be a binary operator op.
  - Let x_L be the result of evaluating the left subtree.

Evaluating an expression tree

- If the root is a leaf, it must be a number v.
  - Return v.
- Otherwise, the root must be a binary operator op.
  - Let x_L be the result of evaluating the left subtree.
  - Let x_R be the result of evaluating the right subtree.

Evaluating an expression tree

- If the root is a leaf, it must be a number v.
  - Return v.
- Otherwise, the root must be a binary operator op.
  - Let x_L be the result of evaluating the left subtree.
  - Let x_R be the result of evaluating the right subtree.
  - Return x_L op x_R.

Evaluating an expression tree

- If the root is a leaf, it must be a number v.
  - Return v.
- Otherwise, the root must be a binary operator op.
  - Let x_L be the result of evaluating the left subtree.
  - Let x_R be the result of evaluating the right subtree.
  - Return x_L op x_R.

Post-order