Stacks

Christophe Rhodes

Motivation

- last-in first-out collection
- useful for modelling (computational) state:
  - function calls / activation records
  - function-local temporary storage area

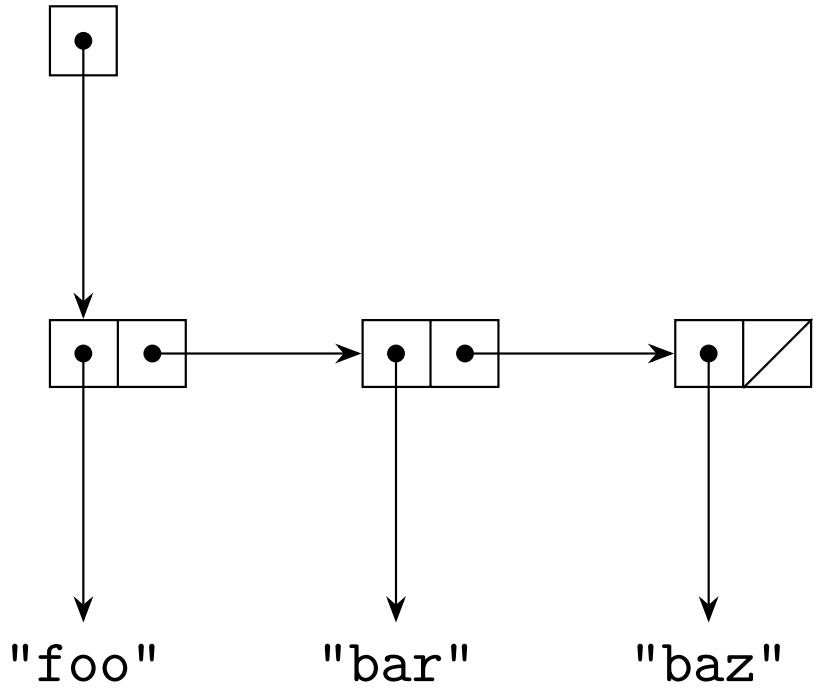
Definition

A stack is an extensible linear collection of data whose top element (only) is accessible

Operations

push![o]	add o to the top of the stack
top	return the top element of the stack
pop!	remove and return the top element of the stack
empty?	return *true* if the stack has no elements

Implementation



Complexity analysis

top

Two pointer reads $\Longrightarrow \Theta(1)$

pop!

Three pointer reads, one pointer write $\Longrightarrow \Theta(1)$

push!

One pair allocation, one pointer write $\Longrightarrow \Theta(1)$

empty?

One pointer read, one equality comparison $\Longrightarrow \Theta(1)$

Dynamic arrays, revisited

Can use a dynamic array directly as a stack

push[o]!, pop! directly supported

top select[length-1]

empty? length = 0

Compare with linked list implementation:

- Space complexity:

  linked list 2 words per item (+ 1 word overhead)

  dynamic array 1 word per item (+ 2+k words overhead)

  dynamic array up to twice as space efficient

- Time complexity:

  linked list operations $\Theta(1)$ in all cases

  dynamic array operations $\Theta(1)$ amortized