Stacks and Queues

CMPT 145

Queue

Queue

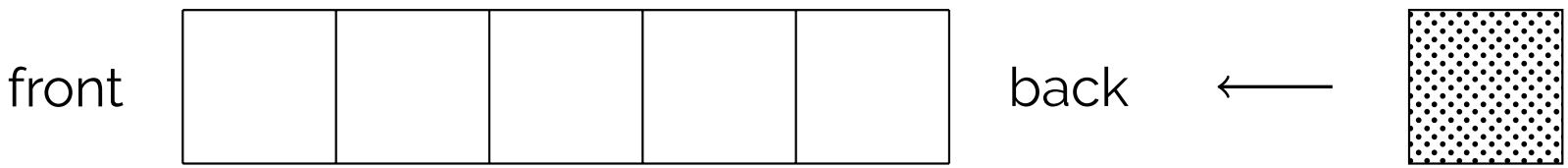
- Linear sequence of data
- Data values ordered via first-in first-out (FIFO) protocol

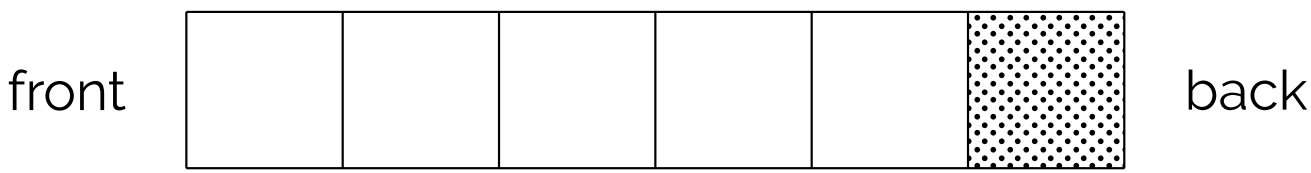
FIFO Protocol:

- Queues have a front and a back:
  - Data values are added **only** to the back
  - Data values are removed **only** from the front

FIFO Protocol: enqueue

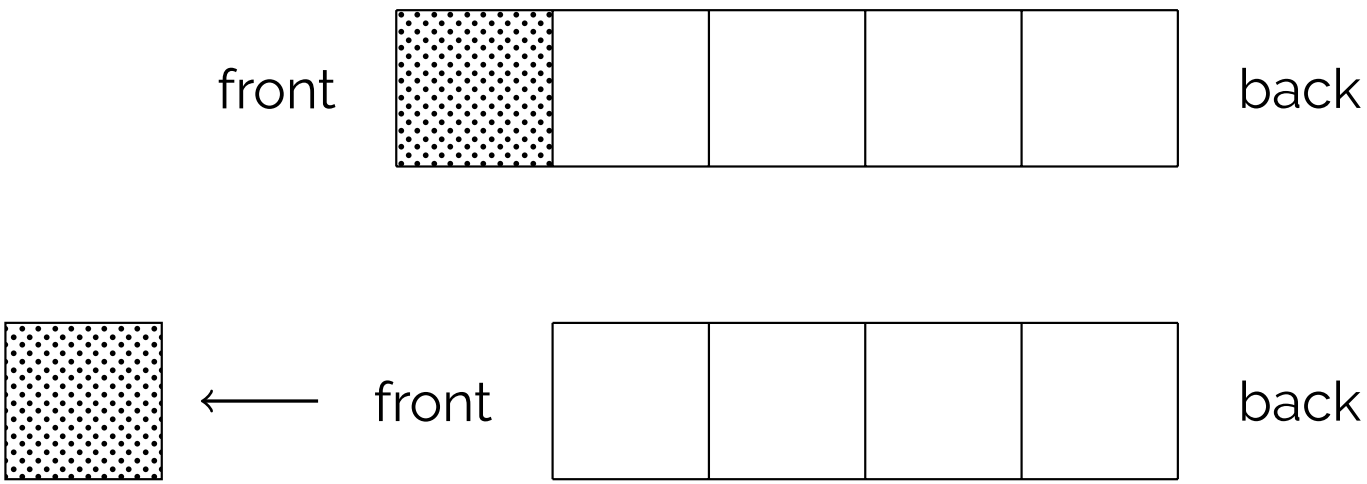
Enqueue adds a new data value to the back of a queue:





FIFO Protocol: dequeue

Dequeue removes the data value at the front of a queue:



Queue ADT

- Purpose:
  - Manage a FIFO-ordered sequence
- Implementation:
  - Data:
    - Sequence of values
  - Essential Operations:
    - Create empty queue
    - Query if queue is empty
    - Query size of queue
    - Enqueue value
    - Dequeue value
    - Peek at first value

Examples

- The FIFO protocol happens in everyday human interactions:
  - Customer line-ups anywhere
  - Walk-in patients at a medical clinic
  - Airport security screening
- The FIFO protocol ensures fairness:
  - The person at the front of the queue has been waiting the longest.

Video buffering

- YouTube sends video data ("producer"); video player displays data ("consumer")
- Variables that affect video experience:
  - Resolution of video (320, 720, 1080p, HD, etc)
  - Network lag, varies over time ("data delivery time")
  - Computer speed
- Video data is *buffered*: accumulated to allow smooth(er) viewing.

Producer-Consumer Model

- Program P (e.g., YouTube) produces some data elements.
- Program C (e.g., viewer) consumes those data elements.
- Data elements (e.g., video) are communicated in pieces.
- Potential problems:
  - If P produces faster than C consumes, P will have to wait for C.
  - If C consumes faster than P produces, C will have to wait for P.
  - Communication rate varies over time (sometimes faster/slower)

Buffers

- A buffer is temporary storage for data transmitted from producer to consumer.
  - All internet communication
  - All modern graphics “cards” or “chips.”
  - All secondary file storage (e.g., disk drives)
- Producer and consumer can work independently!
- Reduces the amount of time producer and consumer wait for the other.
- A buffer is a FIFO queue because data has to be consumed in the order it was produced.

Stack

Stack

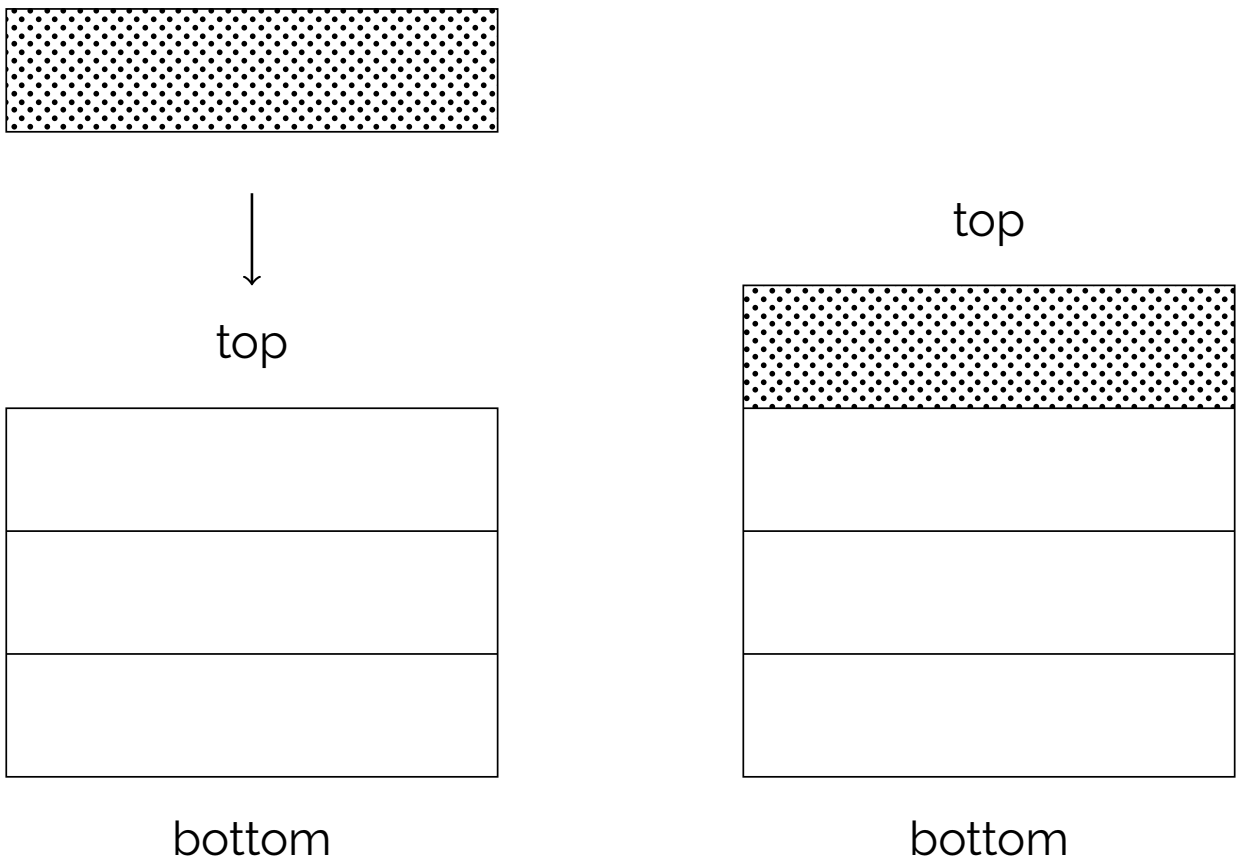
- Linear sequence of data
- Data values ordered via last-in first-out (LIFO) protocol

LIFO Protocol:

- Stacks have a top and a bottom:
  - Data values are added **only** to the top
  - Data values are removed **only** from the top

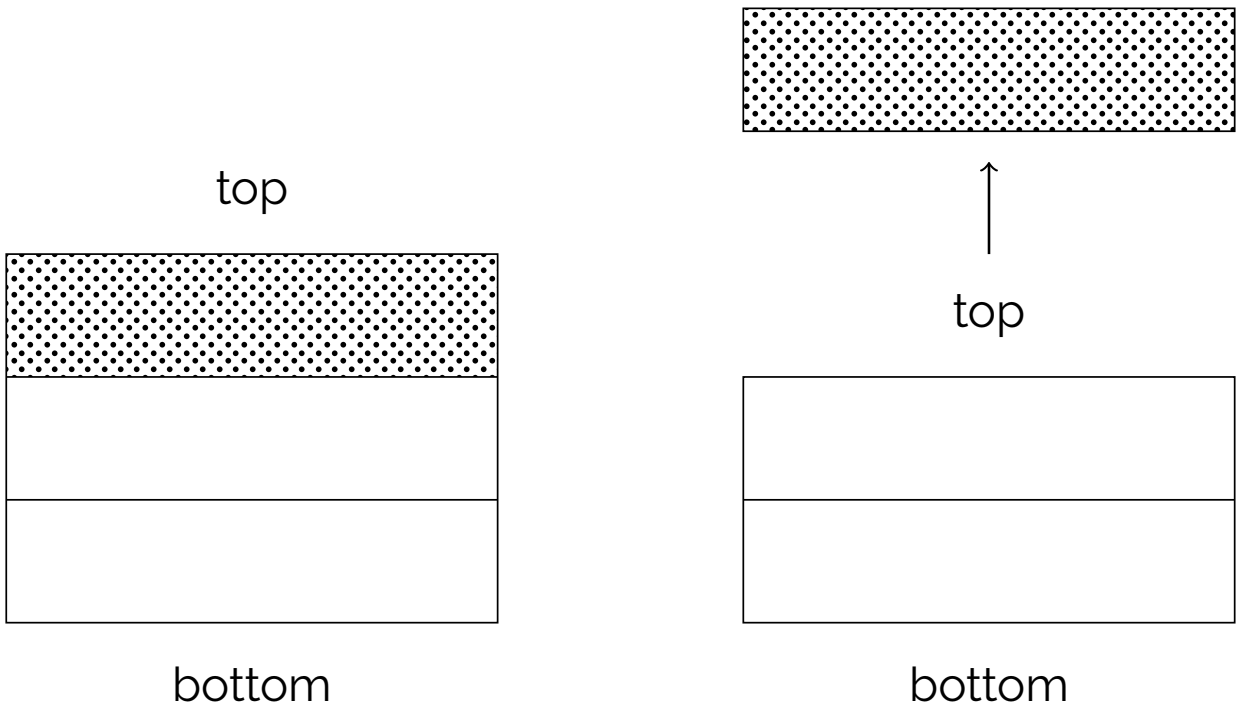
LIFO Protocol: push

Push adds a new data value to the top of a stack:



LIFO Protocol: pop

Pop removes the data value at the top of a stack:



Stack ADT

- Purpose:
  - Manage a LIFO-ordered sequence
- Implementations:
  - Data:
    - Sequence of values
  - Essential Operations:
    - Create empty stack
    - Query if stack is empty
    - Query size of stack
    - Push value onto stack
    - Pop value off of stack
    - Peek at top value

Examples

- The LIFO protocol happens in normal human experience:
  - Stacks of dishes, books, cards, etc.
  - Sedimentary rock formations
- Temporal and physical constraints allow easy access to the most recently added item.
  - The item at the bottom has been waiting in the stack the longest.

The ‘back’ button in your web-browser

- The browser keeps a stack of pages that you visit.
- The page currently displayed is the top of the stack.
- When we visit a new page, the URL gets pushed onto the stack.
- The ‘back’ button pops the top page, and we have one less page on the stack.

The ‘undo’ button in many applications

- The app keeps a stack of changes that you made.
- When when you make a change, information that can undo the change is pushed onto the stack.
- The undo button pops stack, and the changes can be reversed.

Exercise 1

- Every web browser also has a forward button.
  1. Can you describe how back and forward interact?
  2. What is going on when the user can't go back anymore?
- Most applications have a redo button.
  1. Can you describe how undo and redo interact?
  2. What is going on when the user can't redo anymore?

Exercise 2

Design a function to reverse a given string.

- String is immutable, so we have to build a new string from the old.
- Demonstrate mastery of stacks and queues by using one of each in your function.
- Use Python or pseudocode.

- Break the task into function interfaces only.
- No need to implement the functions at all.