COSC 121: Computer Programming II

Dr. Bowen Hui

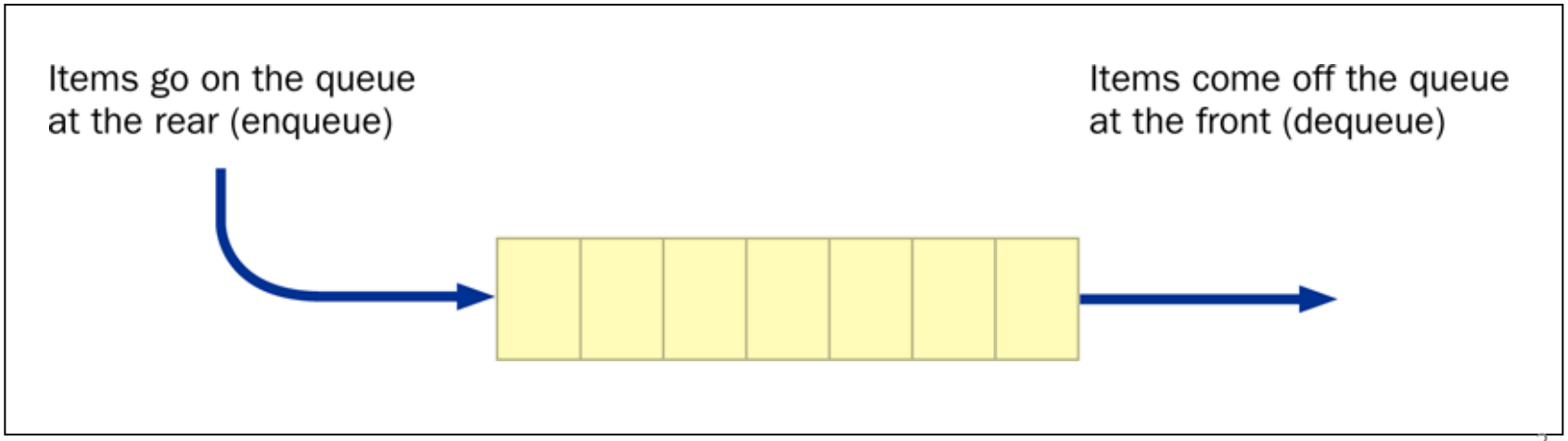
University of British Columbia Okanagan

Linear Data Structures

- Linear data structures are flat and traversal happens through elements one by one
- Examples:
  - Queue
  - Stack
- Non-linear data structures are not flat and traversal can skip over elements in an organized way
- Examples:
  - Tree
  - Graph

Queues

- A queue is a list that adds items only to the rear of the list and removes them only from the front
- It is a FIFO data structure: First-In, First-Out
- Analogy: a line of people at a bank teller's window



Queues

- Classic operations for a queue
  - enqueue - add an item to the rear of the queue
  - dequeue - remove an item from the front of the queue
  - empty - returns true if the queue is empty
- Queues often are helpful in simulations or any situation in which items get "backed up" while awaiting processing

Queues are an ADT

- Queues are a data type because:
  - Stores a set of information (e.g. people, cars, etc.)
- Queues are abstract because:
  - Can be implemented in several ways
  - Just need to know how to use the enqueue, dequeue, empty operations
- Queues are an ADT

Examples

- Real world applications of queues?

Examples

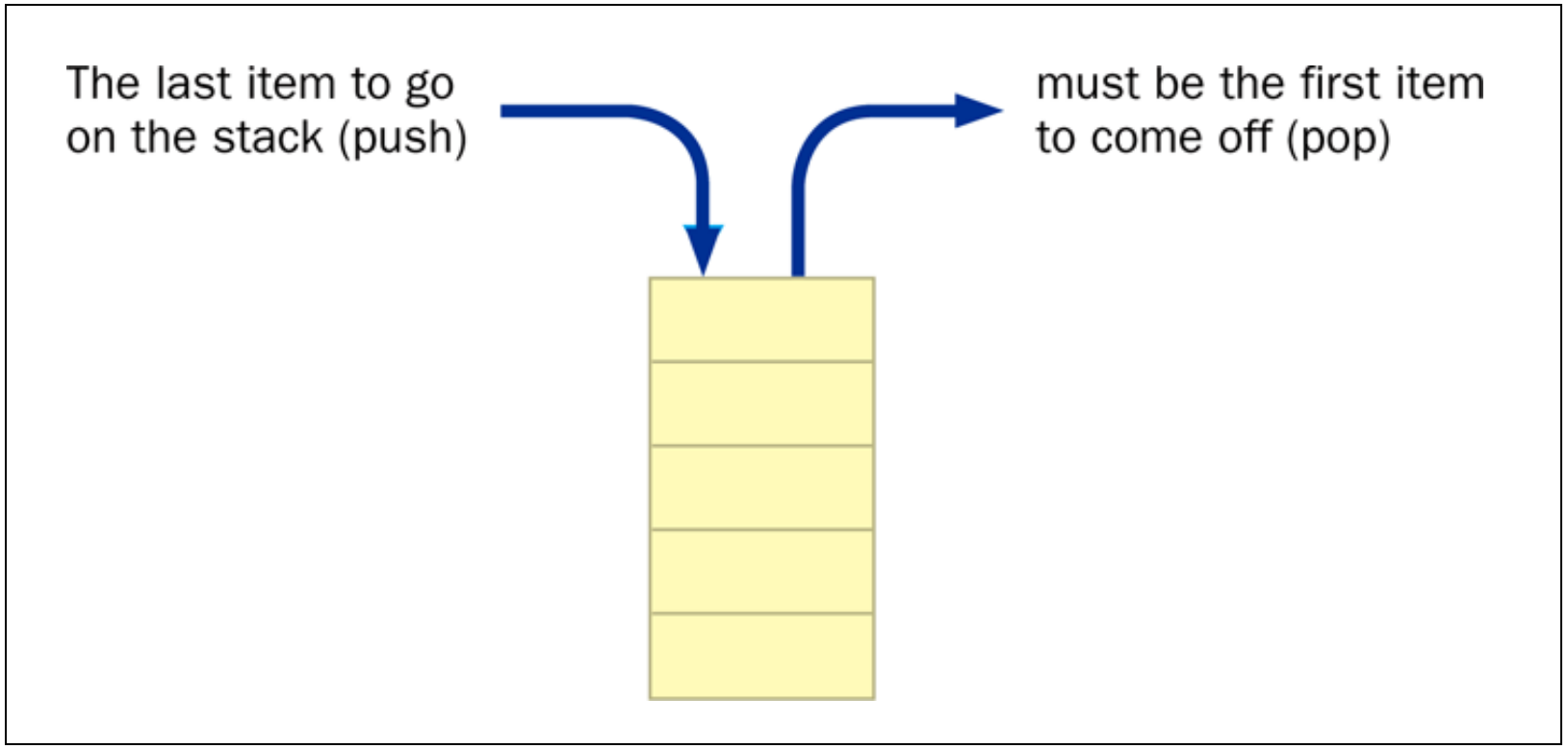
- Restaurant ordering system
  - List of orders
- Ticket purchasing system
  - Requests for specific seats
- Parks reservation system
  - Camp site registration
- Call centre phone routing system
  - Directing to the right support department
- Airline reservation system
  - Booking flights
- Text messaging system
  - Incoming messages

Stacks

- A stack is also linear, like a list or a queue
- Items are added and removed from only one end of a stack
- It is therefore LIFO: Last-In, First-Out
- Analogies: a stack of plates or a stack of books

Stacks

- Stacks often are drawn vertically:

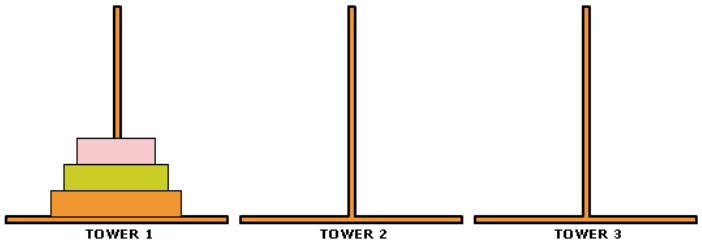


Stacks

- Clasic stack operations:
  - push - add an item to the top of the stack
  - pop - remove an item from the top of the stack
  - top - retrieves the top item without removing it
  - empty - returns true if the stack is empty
- Real examples of stacks?

Examples

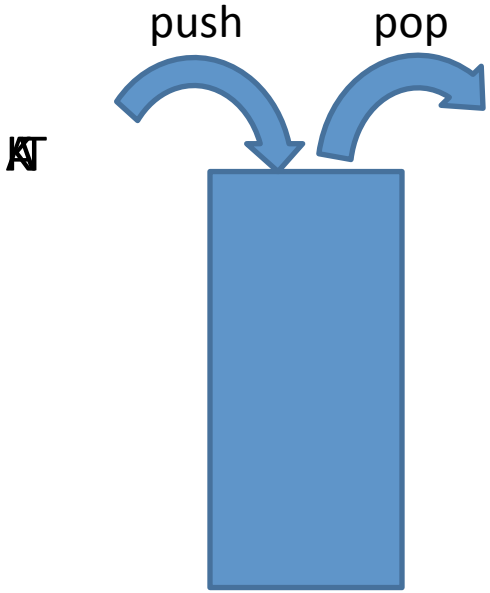
- Taking dishes and putting them away
  - Pile of dishes
- Tennis balls in a container
  - Top ball in/out
- Program stack trace
  - Method call stack in Java
- Tower of Hanoi puzzle solver
  - Each tower as a stack
- Variable scoping
  - Definitions of variables with the same name
- Things to “undo”
  - Undoing most recent items



Stacks are an ADT

- Stacks are a data type because:
  - Stores a set of information (e.g. people, cars, etc.)
- Stacks are abstract because:
  - Can be implemented in several ways
  - Just need to know how to use the push, pop, top, empty operations
- Stacks are an ADT

Queue vs Stack



How to Implement Queues and Stacks?

- As an array?

- As a singly linked list?

How to Implement Queues and Stacks?

- As an array?
  - Simpler
  - Difficult to maintain indices
- As a singly linked list?
  - Additional node class
  - Easier for queues, harder for stacks

How to Implement Queues and Stacks?

- As an array?
  - Simpler for stacks, harder for queues
  - Difficult to maintain indices
- As a singly linked list?
  - Additional node class
  - Easier for queues, harder for stacks
- As a doubly linked list?
  - Not yet needed
  - Assignment 3 – see word processor problem

Exercise: Implement Stack as Array

- Problem specification:
  - A stack to store a set of integers
  - Assume fixed stack size (given)
- Where to start in implementation?
  - Classes?
  - Methods?
  - What to keep track of?

Exercise: Implement Stack as Array

- Classes?
  - Test class with main()
  - Stack class with its operations
- Methods?
  - push(), pop(), top()
  - is_empty() (may also want is_full())
  - toString()
- What to keep track of?
  - Whether stack is empty
  - Index of array to push, pop, or top

```
~/Documents/121/code$ cat StackDemo.java
public class StackDemo
{
  public static void main( String[] args )
  {
    Stack mystack = new Stack( 5 );
    System.out.println( mystack.is_full() );
    mystack.push( 4 );
    mystack.push( 2 );
    mystack.push( 9 );
    System.out.println( mystack.is_full() );
    mystack.push( 3 );
    mystack.push( 7 );
    mystack.push( 6 );
    System.out.println( mystack.is_full() );
    System.out.println( mystack.toString() );
    System.out.println( mystack.top() );
    mystack.pop();
    mystack.push( 6 );
    System.out.println( mystack.toString() );
  }
}
```

Output?

Output

```
~/Documents/121/code$ javac Stack.java StackDemo.java
~/Documents/121/code$ java StackDemo
false
false
true
4 2 9 3 7
7
4 2 9 3 6
~/Documents/121/code$
```

How to visualize stack operations with array?

Stack Class Skeleton

```
public class Stack
{
   // attributes

   // methods
   public Stack( int sz ) { … }
   public boolean is_empty() { … }
   public boolean is_full() { … }
   public int top() { … }
   public void push( int item ) { … }
   public int pop() { … }
   public String toString() { … }
}
```

Summary of Linear Data Structures

- Queues and stacks
  - Know what they are conceptually
  - Know their basic operations
  - Understand implementation tradeoffs when:
    - Implemented as an array
    - Implemented as a linked list
- Next: continue implementation exercises