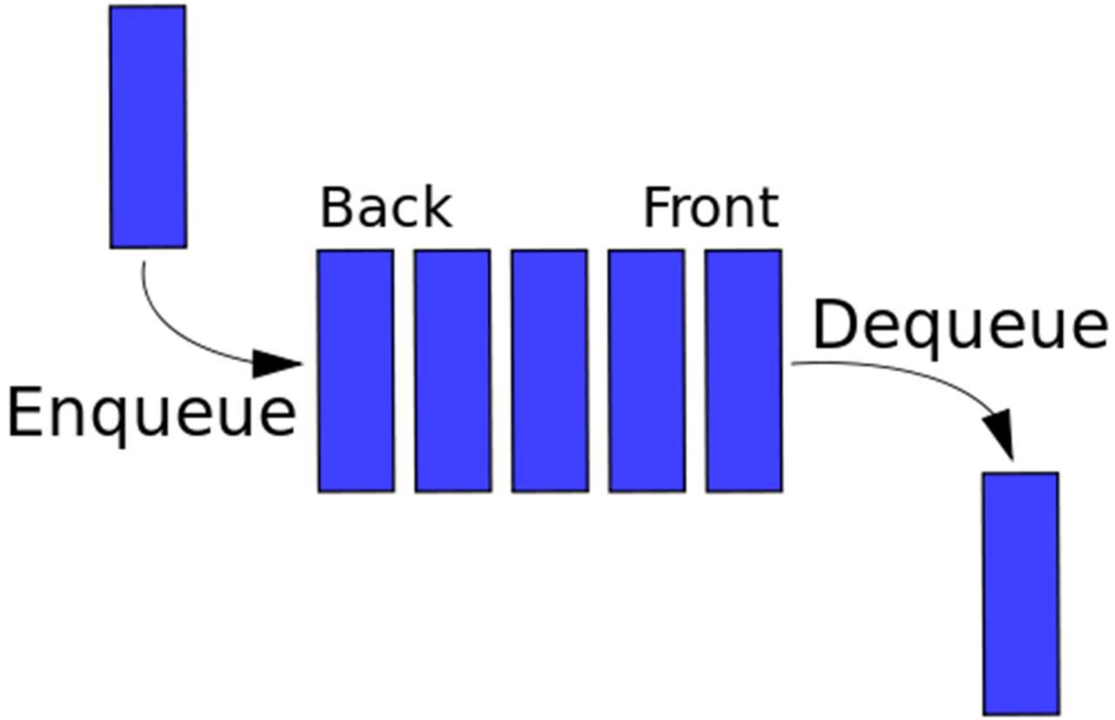
(06) Linear Data Structures Part 3: Queue and Priority Queue

Video (12 mins): https://youtu.be/xXFxcidR3tE

Queue

- Queue is a data structure with FIFO (First In, First Out) or FCFS (First-Come-First-Served) property
- Defined primarily by three main operations: `enqueue`, `dequeue`, `peek`

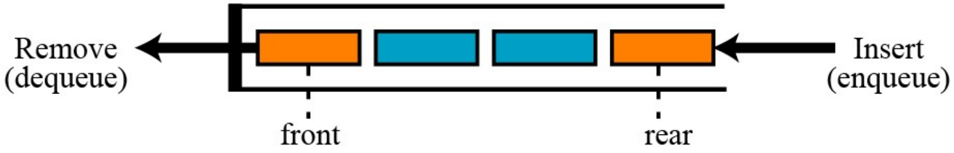


http://en.wikipedia.org/wiki/Queue_(data_structure)

Queue – Motivating Examples

- Many real-world situations
  - Example: A line to buy a movie ticket, waiting for the first available customer service representative, etc.
  - Scheduling based on first-come-first-served principle.
- Computer Applications
  - Printer spooling – jobs sent to the printer are queued until the printer finishes printing the previous job.
  - Asynchronous transmission of data – data sending at a different rate than receiving.

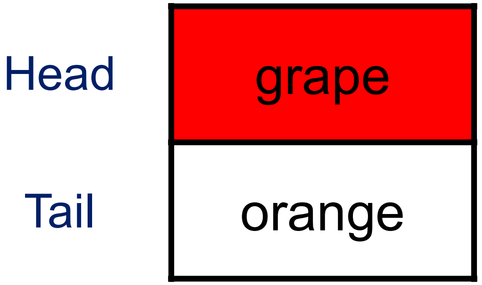
A queue of people



A computer queue

Queue Operation: `enqueue`

- Places a new data element to tail of the queue



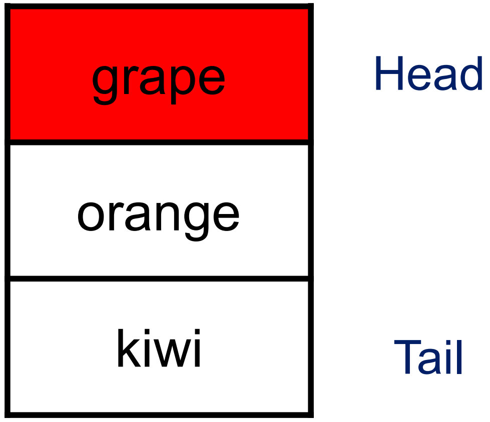
Before

Create queue (Before)

```
>>> q = Queue()
>>> q.enqueue("grape")
>>> q.enqueue("orange")
>>> q.display()
```

Place new data element to tail of queue (After)

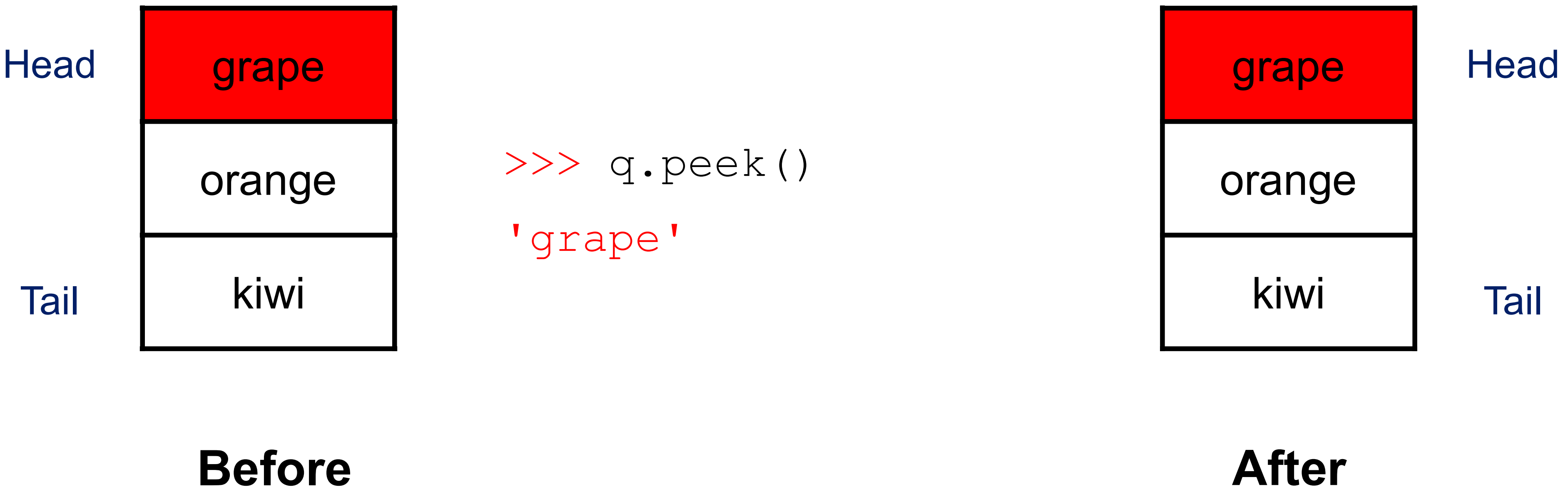
```
>>> q.enqueue("kiwi")
```



After

Queue Operation: `peek`

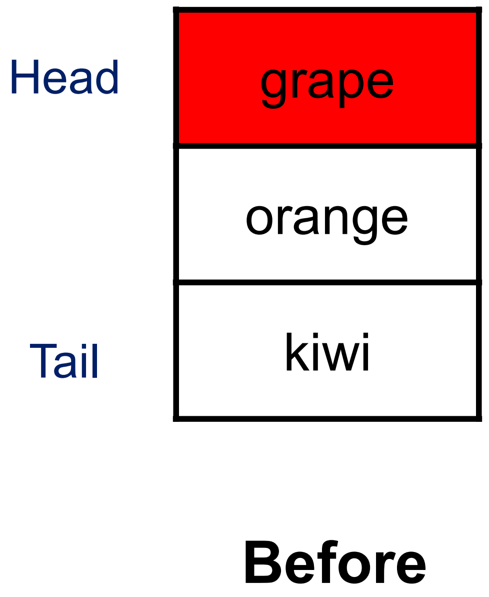
- Inspects the data element at the head of the queue without removing it



```
>>> q.peek()
'grape'
```

Queue Operation: `dequeue`

- Removes the data element at the head of the queue



Head

grape

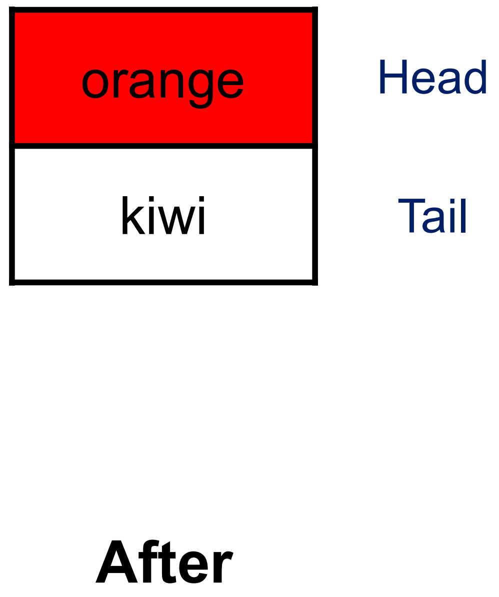
orange

Tail

kiwi

Before

```
>> q.dequeue()
'grape'
```



orange

Head

kiwi

Tail

After

Example: Recognizing Palindromes

- Palindromes are words that read the same from left and right
  - e.g., madam, refer, radar
- How do we detect palindromes using data structures?

Using a Stack and a Queue

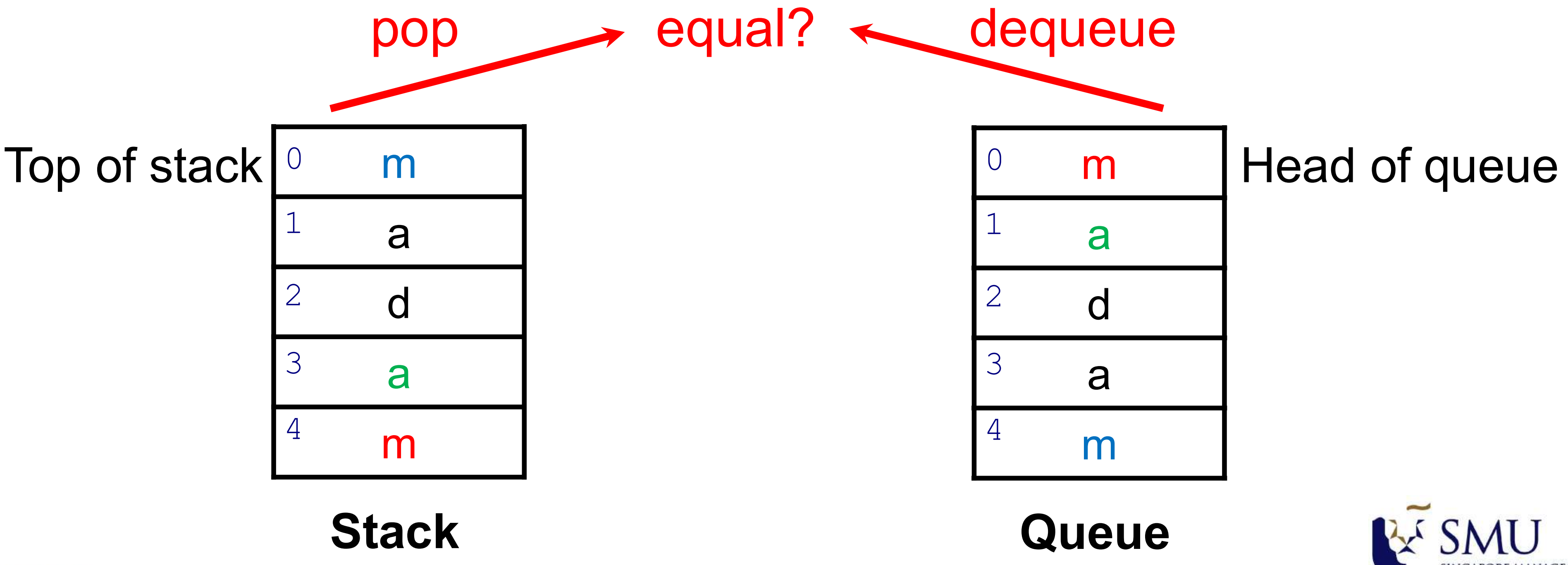
```
>>> word = "madam"

>>> s = Stack()
>>> q = Queue()

>>> for ch in word:
        s.push(ch)
        q.enqueue(ch)
```

Inserting from the top/head.

Inserting from the bottom/tail.



pop

equal?

dequeue

Top of stack

0	m
1	a
2	d
3	a
4	m

Stack

Head of queue

0	m
1	a
2	d
3	a
4	m

Queue

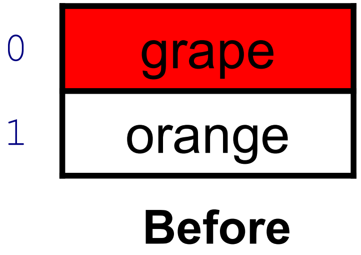
Recognizing Palindrome with Stack and Queue

```
def is_palindrome(word):
 s = Stack()
 q = Queue()
 for ch in word:
  s.push(ch)
  q.enqueue(ch)
 while s.count() > 0:
  left = s.pop()
  right = q.dequeue()
  if left != right
   return False
 return True
```

`List`-based Implementation of `enqueue`

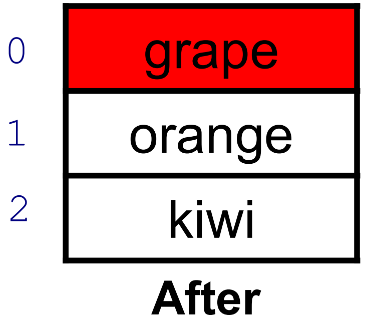
Use a list named `li` to contain data elements.
First element (index 0) is head of queue.

```
def enqueue(li, item):
    li.append(item)
```



Before

```
>>> li = ["grape","orange"]
>>> enqueue(li, "kiwi")
```



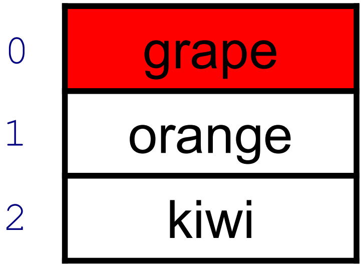
After

Complexity?

`List`-based Implementation of `peek`

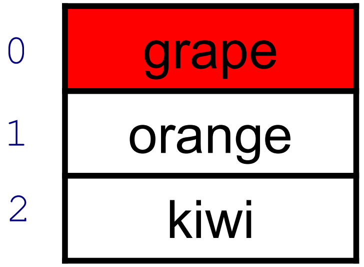
Use a list named `li` to contain data elements.
First element (index 0) is head of queue.

```
def peek(li):
    if len(li) > 0:
        return li[0]
```



Before

```
>>> q.peek()
'grape'
```



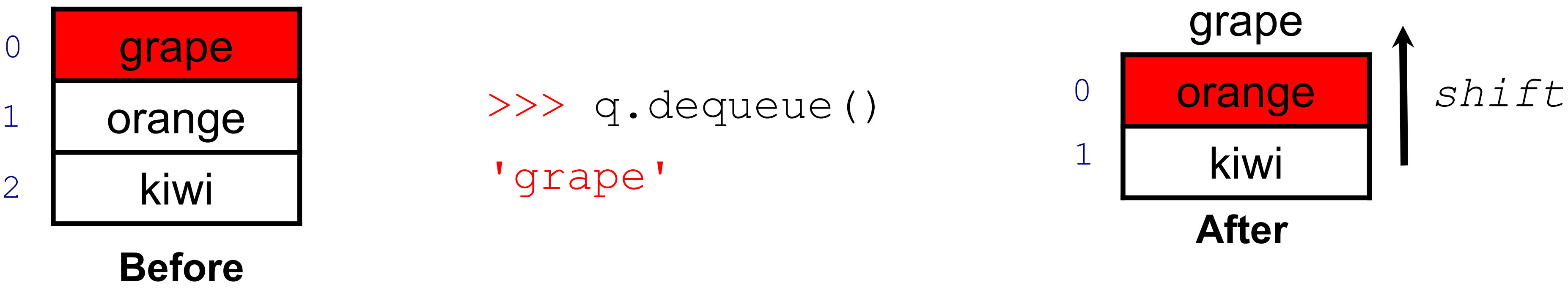
After

Complexity?

`List`-based Implementation of `dequeue`

Use a list named `li` to contain data elements.
First element (index 0) is head of queue.

```
def dequeue(li):
    if len(li) > 0:
        item = li[0]
        del li[0]
        return item
```



0 grape
1 orange
2 kiwi

Before

```
>>> q.dequeue()
'grape'
```

grape

0 orange
1 kiwi

After

shift

Complexity?

Priority Queue

- Priority queue has similar operations as regular queue:
  - `enqueue`, `peek`, `dequeue`
- Unlike regular queue, priority queue is not FIFO.
- Each data element in a priority queue has a “priority” value:
  - `dequeue` and `peek` will return the data element with the highest priority.
- What is priority?
  - Dependent on application scenarios
  - For our purpose, similar to sorted ordering:
    - Smaller numbers have higher priority over larger numbers
    - Earlier letters in the alphabet have higher priority over later letters

Examples of Priority Queue

- Personal to-do list.
- Patients' waiting list in a hospital emergency room.
- Special queues for express passes in a theme park.
- Scheduling computing jobs for your laptop's CPU.

Implementation using `List`

- The data elements are stored in an array object.
- `enqueue`:
  - Place the element in the last position in the array.
  - Complexity?
- `peek`:
  - Return the minimum data element in the array.
  - Complexity?
- `dequeue`:
  - Search for the minimum data element (highest priority) in the array, and delete it.
  - Complexity?

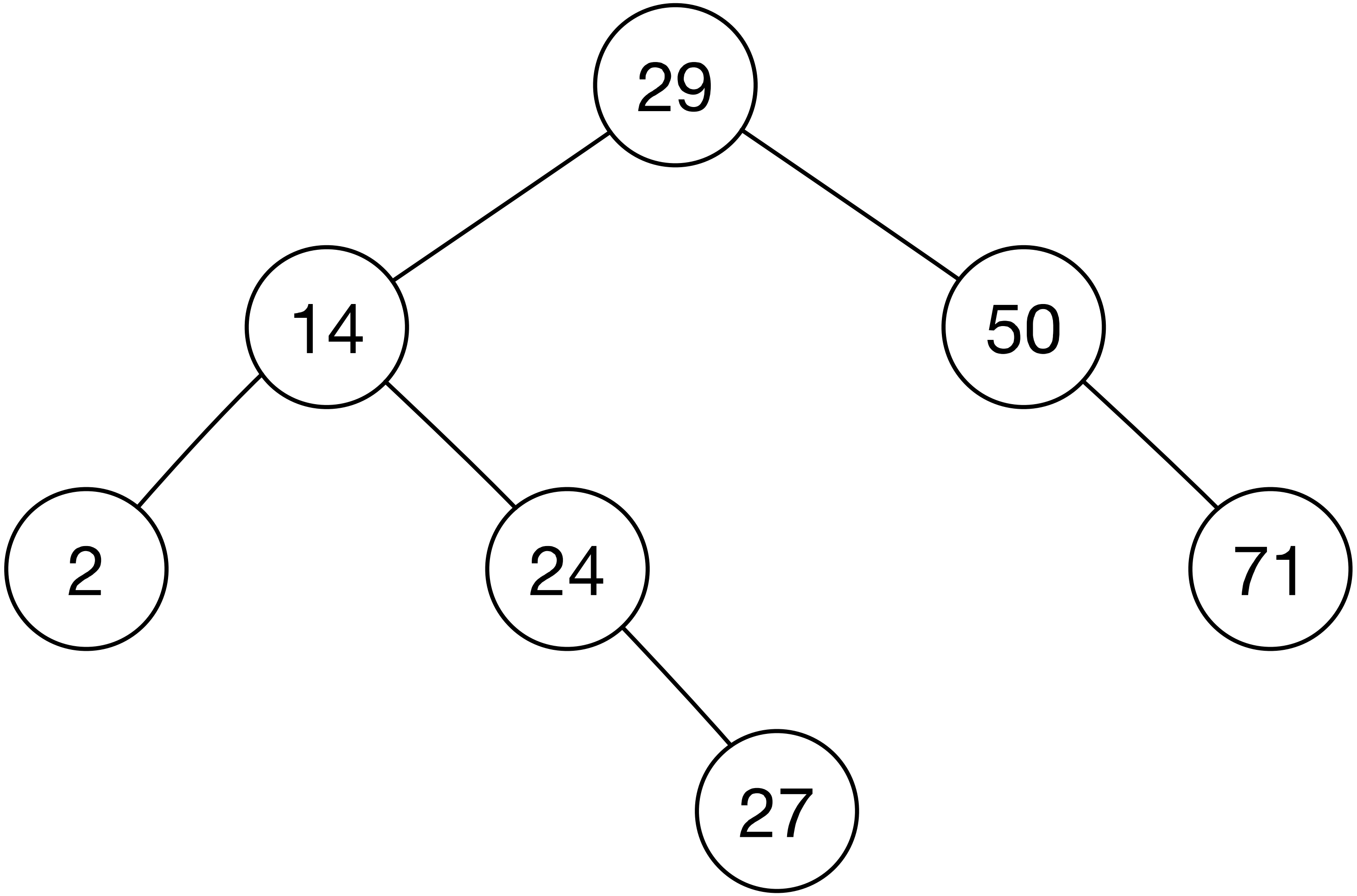
Implementation using a sorted `List`

- The data elements are stored in an array object, which is always in sorted order.
- `enqueue`:
  - Place the element in the "correct" position to keep the array sorted.
  - Complexity?
- `peek`:
  - Return the minimum data element in the array.
  - Complexity?
- `dequeue`:
  - Search for the minimum data element in the array, and delete it.
  - Complexity?

Implementation using a binary search tree (BST)

- The data elements are stored in a BST (a new data structure we will discuss next week), where the elements are always in order.
- `enqueue`:
  - Place the element in the “correct” position in the tree.
  - Complexity: O(`log n`) in average case, O(`n`) in worst case.
- `peek`:
  - Return the minimum data element in the BST.
  - Complexity? O(1)
- `dequeue`:
  - Search for the minimum data element in the BST, and delete it.
  - Complexity: O(`log n`) in average case, O(`n`) in worst case.

Preview: Binary Search Tree



Comparison of PQ Implementations

	Unsorted List	Sorted List	Binary Search Tree (worst case)	Binary Search Tree (average case)
enqueue	O(1)	O(n)	O(n)	O(log n)
peek	O(1)	O(1)	O(1)	O(1)
dequeue	O(n)	O(1)	O(n)	O(log n)

Which implementation is better?

Summary

- Linear data structures allow only one data element to be accessed at any point of time.
- Stack: *Last In, First Out (LIFO)*
  - Push places new item onto the top of the stack.
  - Pop removes the item currently at the top of the stack.
- Queue: *First In, First Out (FIFO)*
  - Enqueue places new item at the tail of the queue.
  - Dequeue removes the item currently at the head of the queue.
- Priority Queue: *Highest Priority, First Out*
  - Enqueue places new item in the queue.
  - Dequeue removes the item currently having the highest priority.

In-Class Exercises

- What is the output calling **q.dequeue()** after the following operations?

```
q = Queue()
q.enqueue(1)
q.enqueue(3)
q.dequeue()
q.enqueue(4)
q.dequeue()
q.enqueue(2)
```

In-Class Exercises

(a) Show how to “simulate” a queue using two stacks. The dequeue operation can be like this:

```
dequeue():
  if stack1 is empty:
    underflow error
  else:
    return stack1.pop()
```

Write the enqueue operation

(b) What is the complexity of each operation for your solution in (a)?

- ❖ Enqueue
- ❖ Dequeue

You may assume that each pop and push is O(1).

Road Map

Algorithm Design and Analysis

(Weeks 1 - 5)

Fundamental Data Structures

- Week 6: Linear data structures (stack, queue)
- Next week → Week 7: Hierarchical data structure (binary tree)
- Week 9: Networked data structure (graph)

Computational Intractability and Heuristic Reasoning

(Weeks 10 - 13)