Data Structures

Maarten Dhondt

Realdolmen

June 23, 2017

Who am I?

- Master of Engineering: Computer Science (KUL)
  - Computational informatics
- RealdolMen: acADDemICT in 09/2015
- Current project: Planning infrastructure @ Infrabel

Outline

Outline

What are Data Structures?

Data Structure[1]

A way in which data are stored for efficient search and retrieval. Different data structures are suited for different problems.

- Data type $\neq$ data structure
- `java.util.HashSet` vs. hash table
- `array` vs. array

[1] Encyclopædia Britannica

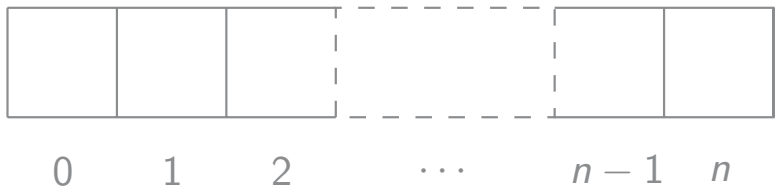
Outline

Array

Definition

- An indexed set of related elements.[2]
- An assemblage of items that are randomly accessible by integers, the index.[3]

- Example: linear array

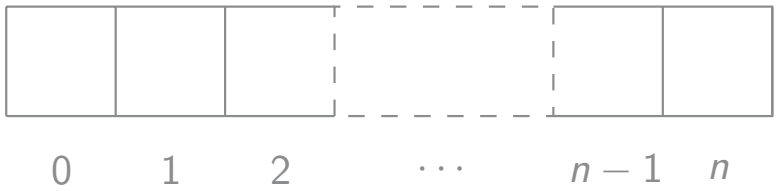


[2] Oxford Dictionary
[3] National Institute of Standards & Technology

Array

Operations

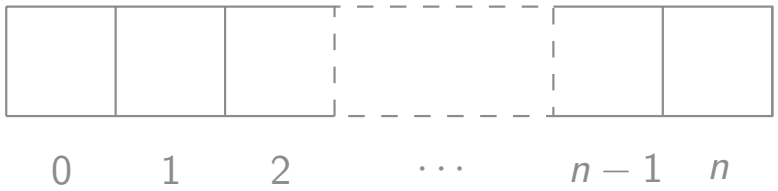
- `get`
- `set`
- `indexOf`



Array

Operations

- `get`
- `set`
- `indexOf`

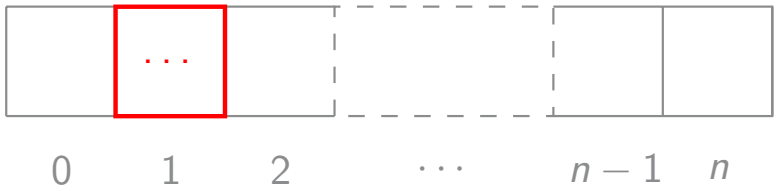


`get(1)`

Array

Operations

- `get`
- `set`
- `indexOf`

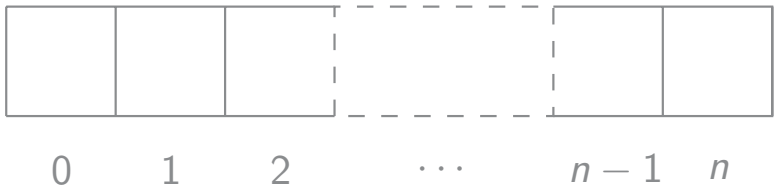


`get(1)`

Array

Operations

- `get` $O(1)$
- `set`
- `indexOf`

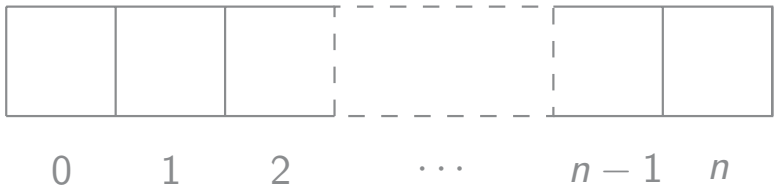


`get(1)`

Array

Operations

- `get` $O(1)$
- `set`
- `indexOf`



`set(2)`

Array

Operations

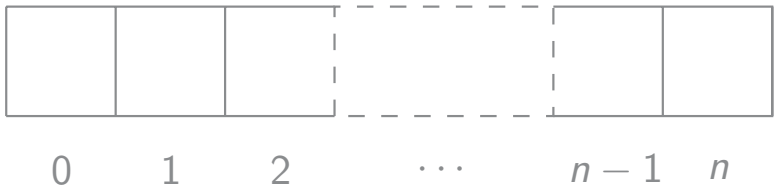
- `get` $O(1)$
- `set`
- `indexOf`

`set(2)`

Array

Operations

- `get` $O(1)$
- `set` $O(1)$
- `indexOf`

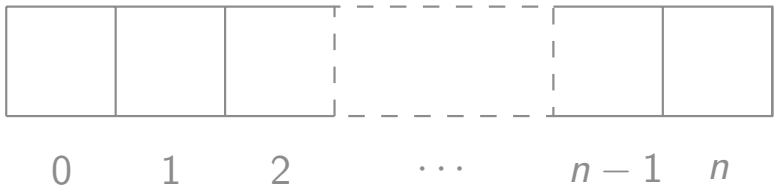


`set(2)`

Array

Operations

- `get` $O(1)$
- `set` $O(1)$
- `indexOf`

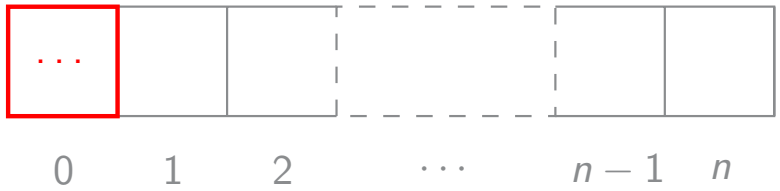


`indexOf(object)`

Array

Operations

- `get` $O(1)$
- `set` $O(1)$
- `indexOf`

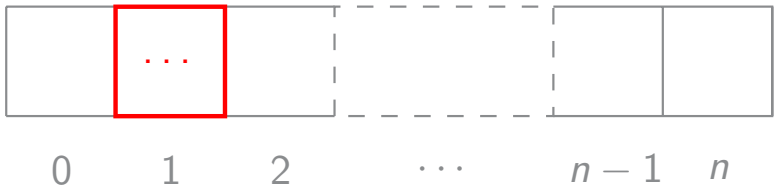


`indexOf(object)`

Array

Operations

- `get` $O(1)$
- `set` $O(1)$
- `indexOf`



`indexOf(object)`

Array

Operations

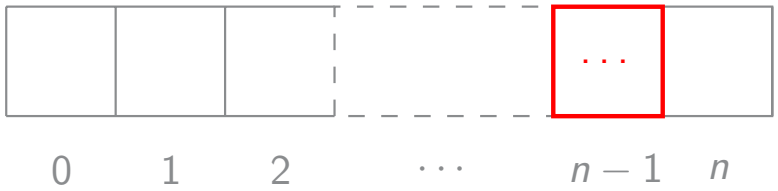
- `get` $O(1)$
- `set` $O(1)$
- `indexOf`

`indexOf(object)`

Array

Operations

- `get` $O(1)$
- `set` $O(1)$
- `indexOf`

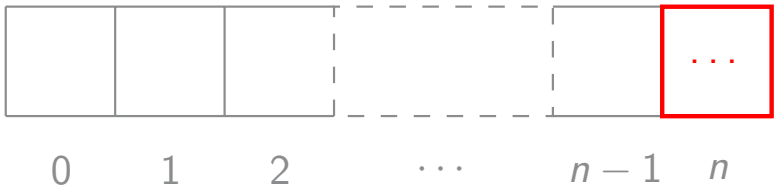


`indexOf(object)`

Array

Operations

- `get` $O(1)$
- `set` $O(1)$
- `indexOf`

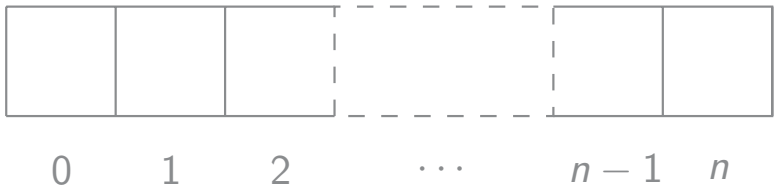


`indexOf(object)`

Array

Operations

- `get` $O(1)$
- `set` $O(1)$
- `indexOf` $O(n)$



Outline

Linked List

Definition

A linked list is a data structure in which the objects are arranged in a linear order. Unlike arrays in which the linear order is determined by indices, the order is determined by a pointer in each object.[4]

- Different types: singly, doubly, multiply, circular, ...
- Example: doubly linked list

[4] Introduction to Algorithms By Cormen, Leierson, Rivest & Stein

Linked List

Operations

- `add/remove first/last`
- `get/insertAt`
- `indexOf`

Linked List

Operations

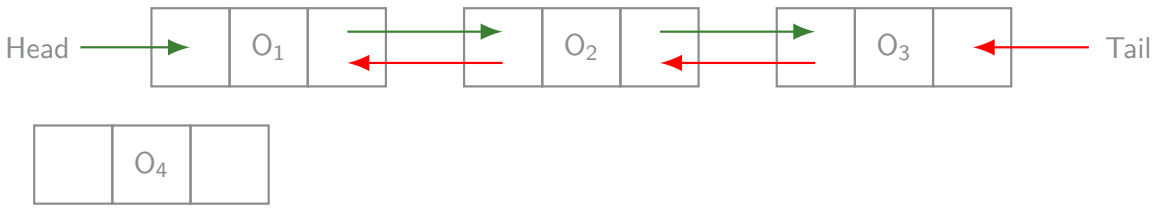
- `add/remove first/last`
- `get/insertAt`
- `indexOf`

`addFirst(`O_4`)`

Linked List

Operations

- `add/remove first/last`
- `get/insertAt`
- `indexOf`

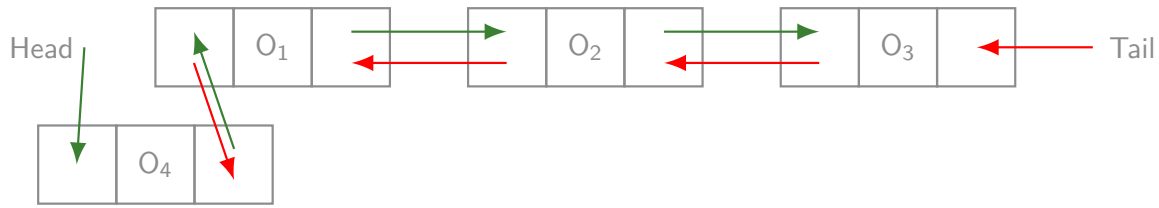


`addFirst(`O_4`)`

Linked List

Operations

- `add/remove first/last`
- `get/insertAt`
- `indexOf`

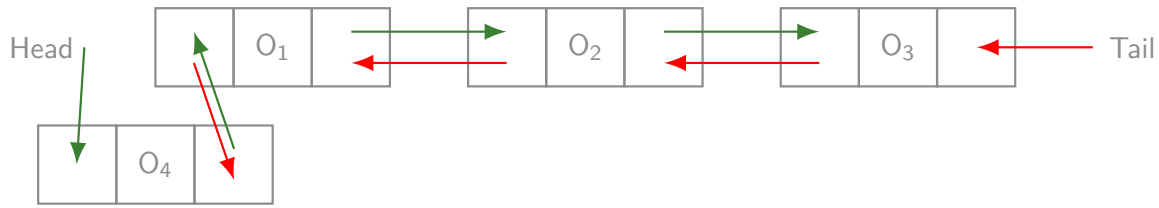


`addFirst(`O_4`)`

Linked List

Operations

- `add/remove first/last` $O(1)$
- `get/insertAt`
- `indexOf`



`addFirst(`O_4`)`

Linked List

Operations

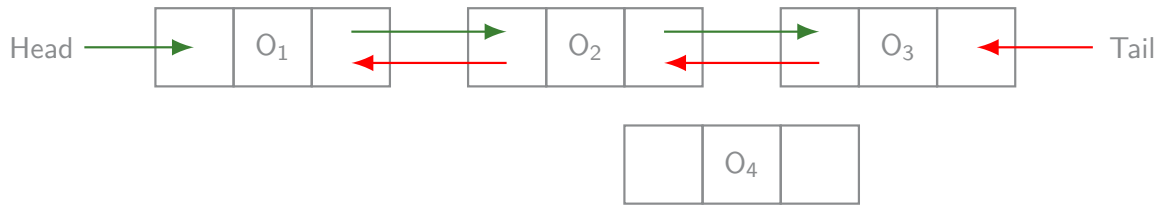
- `add/remove first/last` $O(1)$
- `get/insertAt`
- `indexOf`

`insertAt(2)`

Linked List

Operations

- `add/remove first/last` $O(1)$
- `get/insertAt`
- `indexOf`

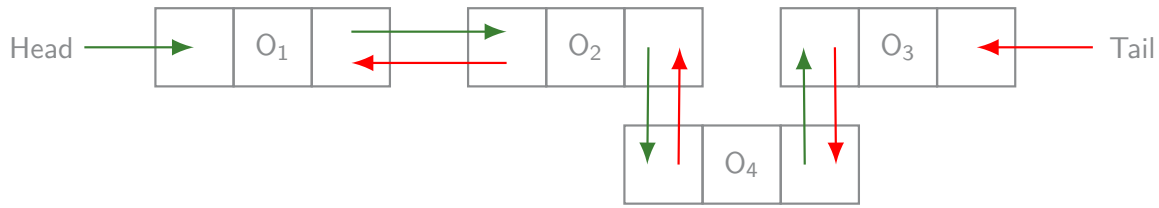


`insertAt(2)`

Linked List

Operations

- `add/remove first/last` $O(1)$
- `get/insertAt`
- `indexOf`

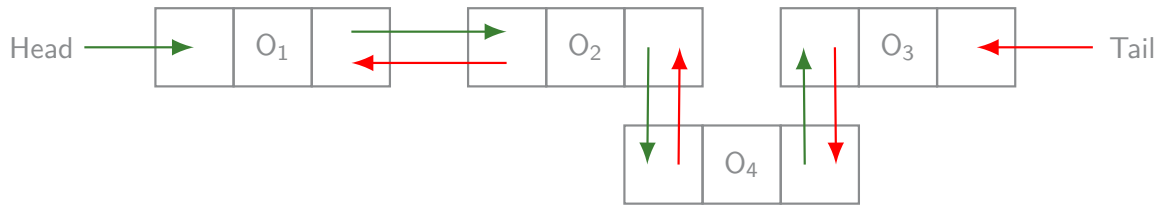


`insertAt(2)`

Linked List

Operations

- `add/remove first/last` $O(1)$
- `get/insertAt` $O(n)$
- `indexOf`



`insertAt(2)`

Linked List

Operations

- `add/remove first/last` $O(1)$
- `get/insertAt` $O(n)$
- `indexOf`

`indexOf(`O_2`)`

Linked List

Operations

- `add/remove first/last` $O(1)$
- `get/insertAt` $O(n)$
- `indexOf`

`indexOf(`O_2`)`

Linked List

Operations

- `add/remove first/last` $O(1)$
- `get/insertAt` $O(n)$
- `indexOf`

`indexOf(`O_2`)`

Linked List

Operations

- `add/remove first/last` $O(1)$
- `get/insertAt` $O(n)$
- `indexOf` $O(n)$

`indexOf`(O_2)

Linked List

Operations

- `add/remove first/last` $O(1)$
- `get/insertAt` $O(n)$
- `indexOf` $O(n)$

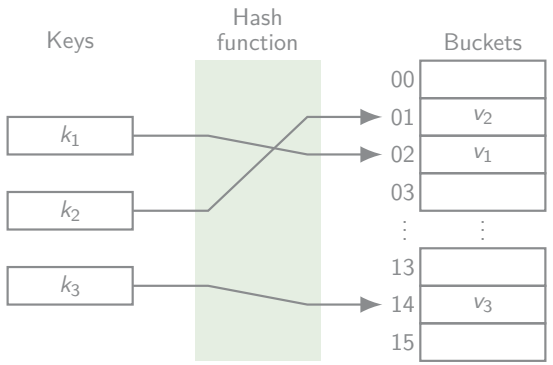
Outline

Hash Table

Definition

A dictionary in which keys are mapped to array positions by hash functions.[5]

- Hash functions: determinism, uniformity, defined range, data normalisation, non-invertible, perfect, ...
- Collisions resolution: chaining, open addressing, ...
- Example:

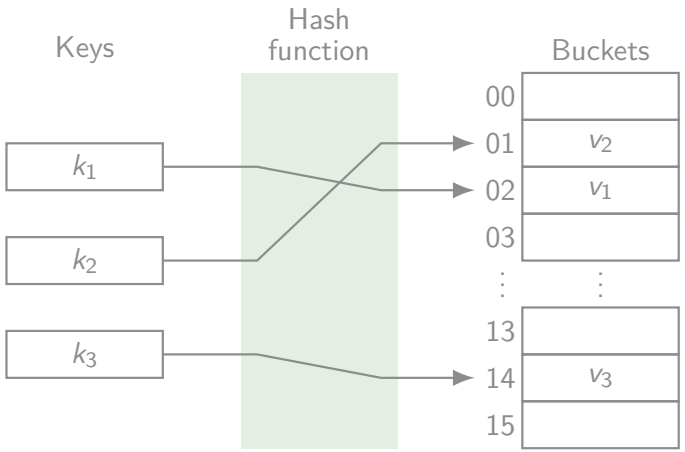


[5] National Institute of Standards & Technology

Hash Table

Operations

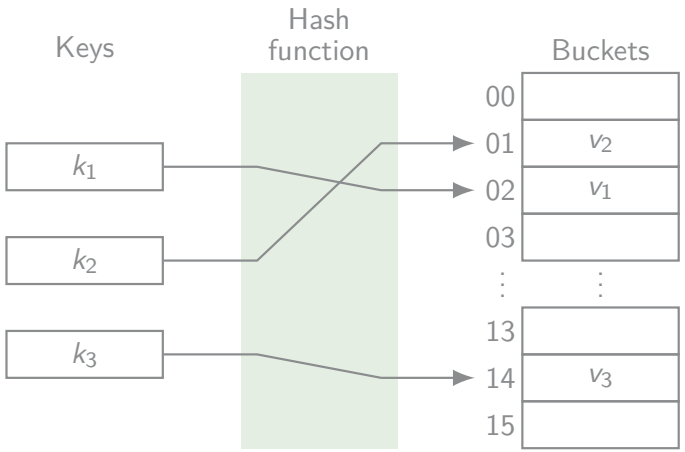
- `put`
- `remove`
- `get`



Hash Table

Operations

- `put`
- `remove`
- `get`

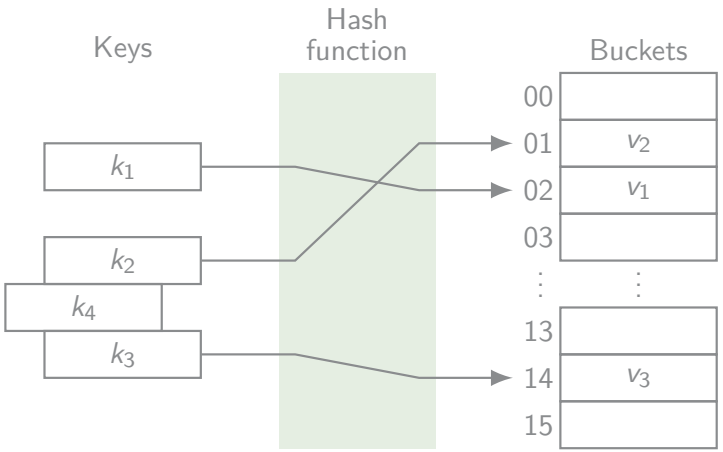


`put(`O_4`)`

Hash Table

Operations

- `put`
- `remove`
- `get`

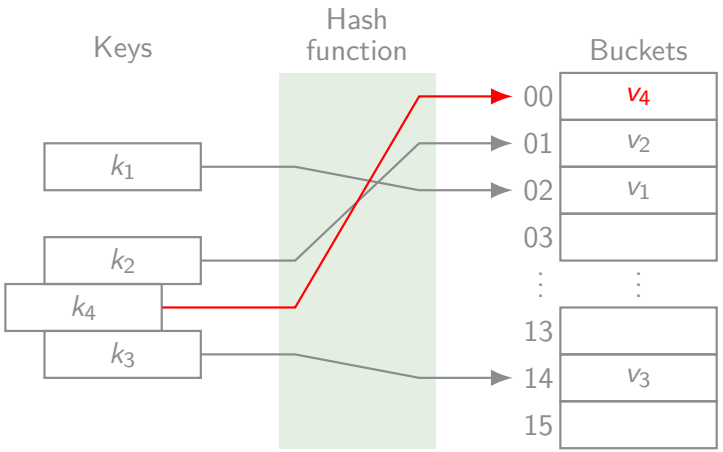


`put(O4)`

Hash Table

Operations

- `put`
- `remove`
- `get`

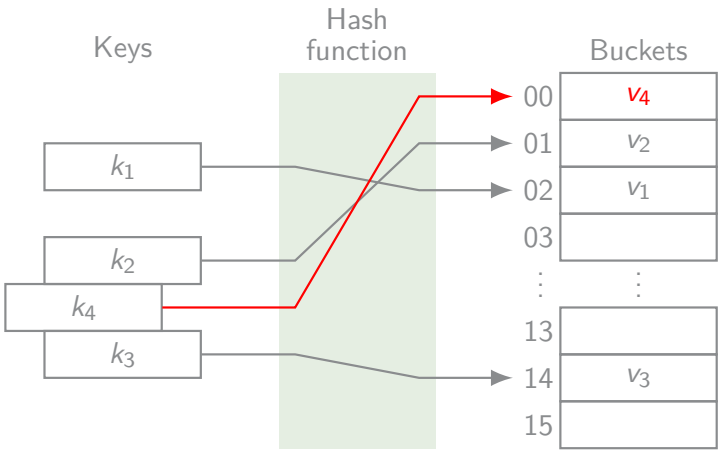


`put(`O_4`)`

Hash Table

Operations

- `put` $O(1)$ / $O(n)$
- `remove`
- `get`

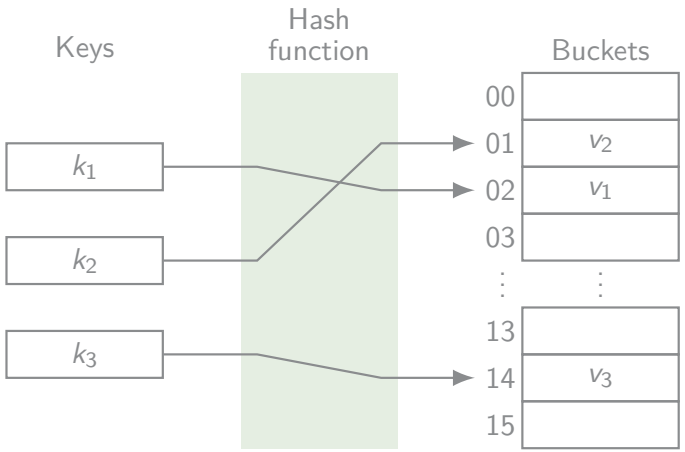


`put(O4)`

Hash Table

Operations

- `put` $O(1)$ / $O(n)$
- `remove`
- `get`

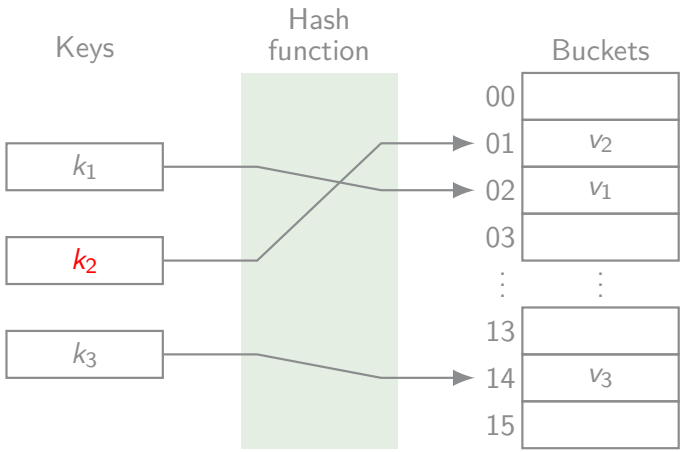


`remove(`0_2`)`

Hash Table

Operations

- `put` $O(1)$ / $O(n)$
- `remove`
- `get`

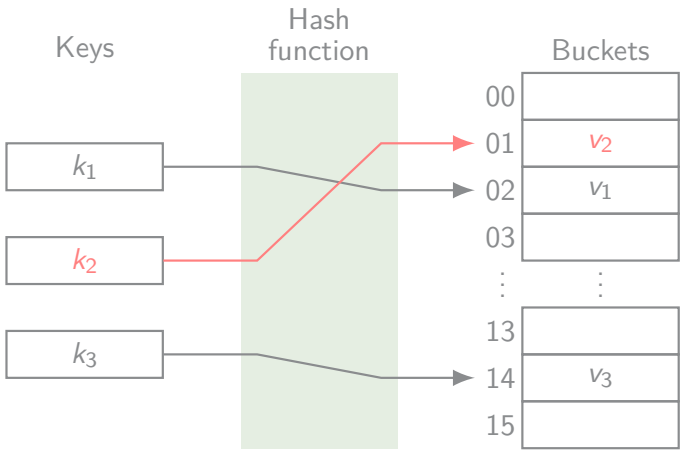


`remove(`0_2`)`

Hash Table

Operations

- `put` $O(1)$ / $O(n)$
- `remove`
- `get`

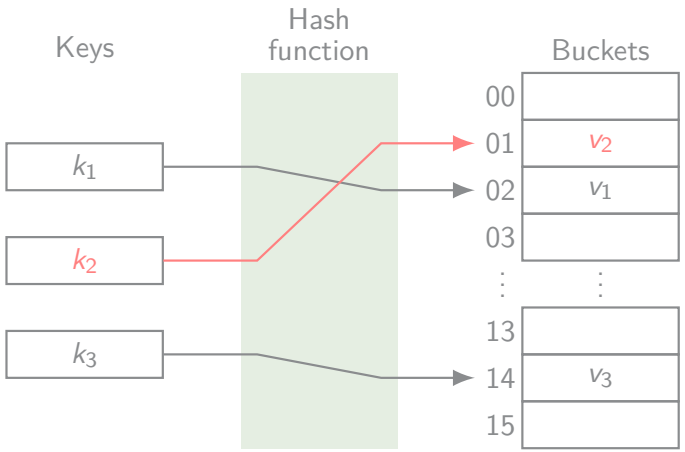


`remove(`0_2`)`

Hash Table

Operations

- `put` $O(1)$ / $O(n)$
- `remove` $O(1)$ / $O(n)$
- `get`

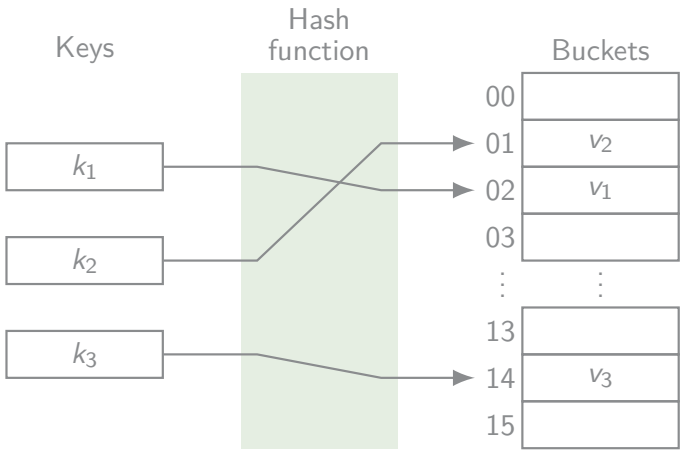


`remove(`0_2`)`

Hash Table

Operations

- `put` $O(1)$ / $O(n)$
- `remove` $O(1)$ / $O(n)$
- `get`

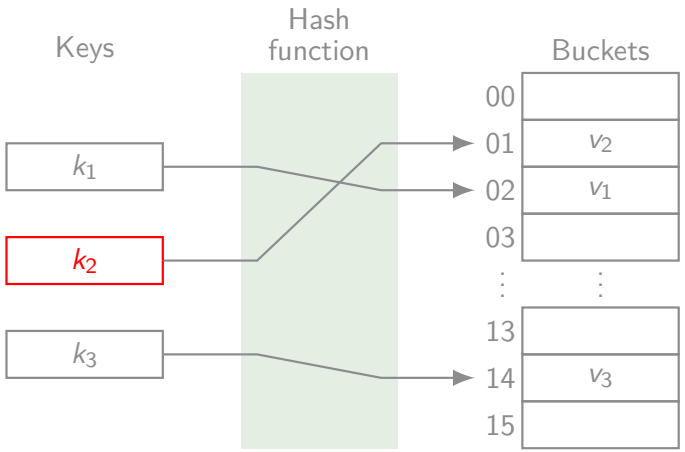


`get(`O_2`)`

Hash Table

Operations

- `put` $O(1)$ / $O(n)$
- `remove` $O(1)$ / $O(n)$
- `get`

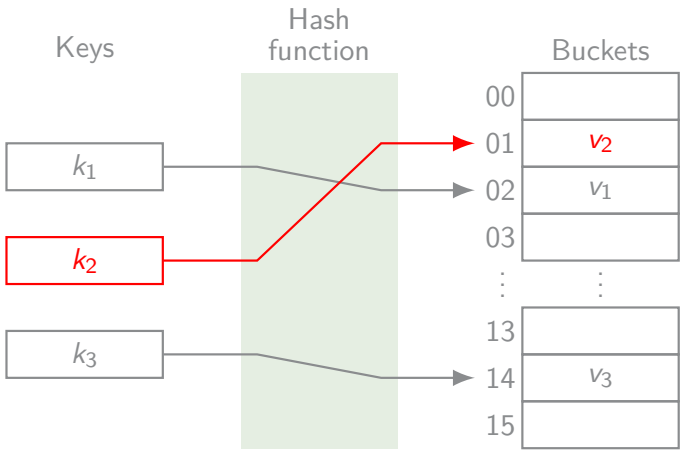


`get(`O_2`)`

Hash Table

Operations

- `put` $O(1)$ / $O(n)$
- `remove` $O(1)$ / $O(n)$
- `get`

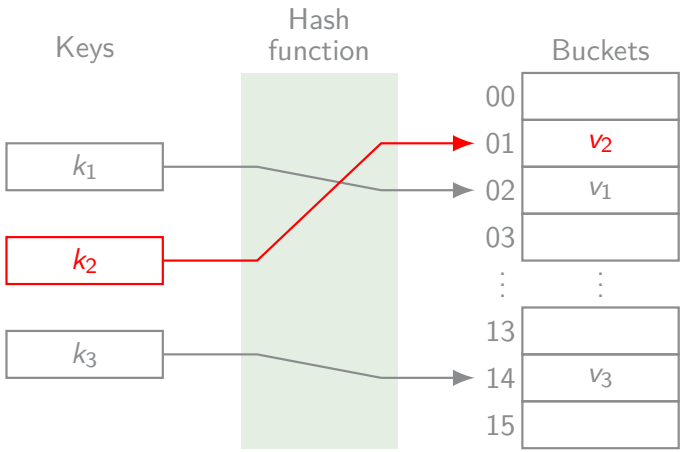


`get(`O_2`)`

Hash Table

Operations

- `put` $O(1)$ / $O(n)$
- `remove` $O(1)$ / $O(n)$
- `get` $O(1)$ / $O(n)$

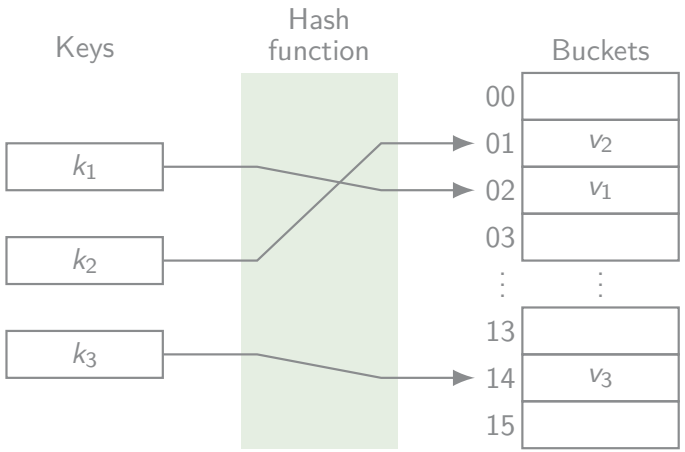


`get(`O_2`)`

Hash Table

Operations

- `put` $O(1)$ / $O(n)$
- `remove` $O(1)$ / $O(n)$
- `get` $O(1)$ / $O(n)$



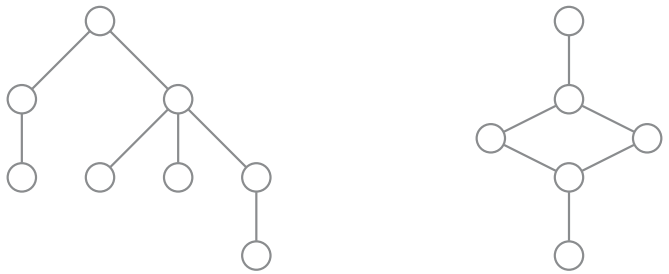
Outline

Tree

Definition

A data structure made up of nodes or vertices and edges without having any cycle. A tree that is not empty consists of a root node and potentially many levels of additional nodes that form a hierarchy.

- Depth, binary, (nearly) complete, ...
- Example:

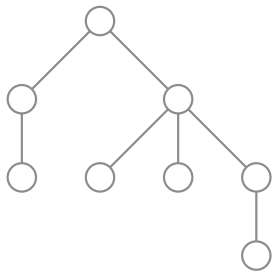


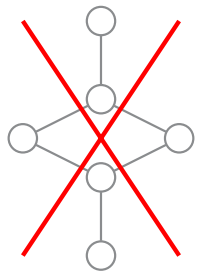
Tree

Definition

A data structure made up of nodes or vertices and edges without having any cycle. A tree that is not empty consists of a root node and potentially many levels of additional nodes that form a hierarchy.

- Depth, binary, (nearly) complete, ...
- Example:





Binary Heap

Definition (Heap)

A complete tree where every node has a key more extreme (greater or less) than or equal to the key of its parent.[6]

Definition (Binary Heap)

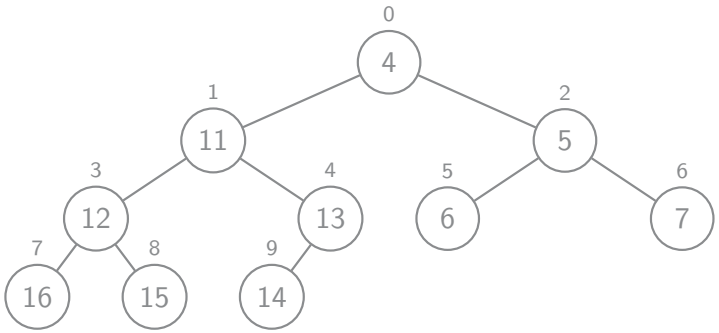
A binary heap data structure is an array object that we can view as a nearly complete binary tree that satisfies the min-heap or max-heap property.[7]

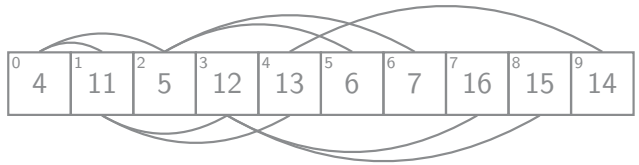
[6] National Institute of Standards & Technology

[7] Introduction to Algorithms By Cormen, Leierson, Rivest & Stein

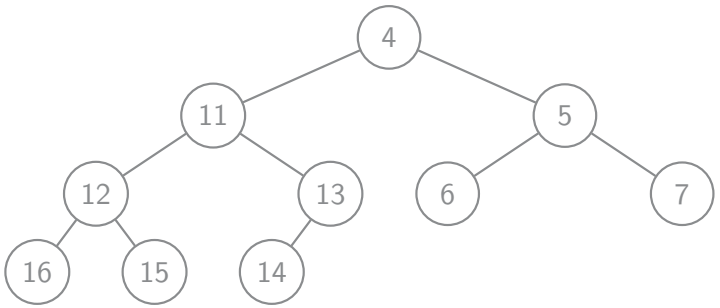
Binary Min-Heap

- `Parent(n)` $\lfloor \frac{n-1}{2} \rfloor$
- `Left(n)` $2n+1$
- `Right(n)` $2(n+1)$

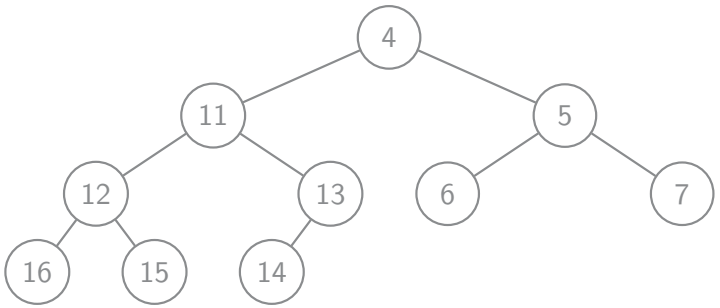




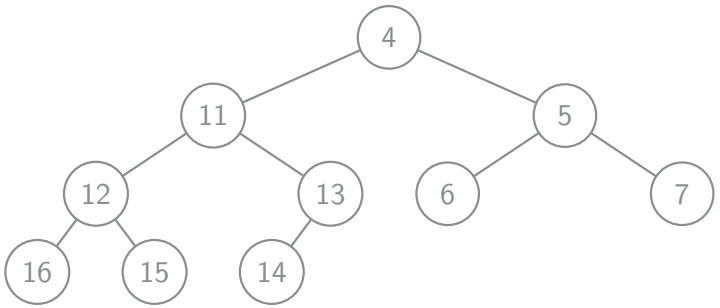
Binary Min-Heap

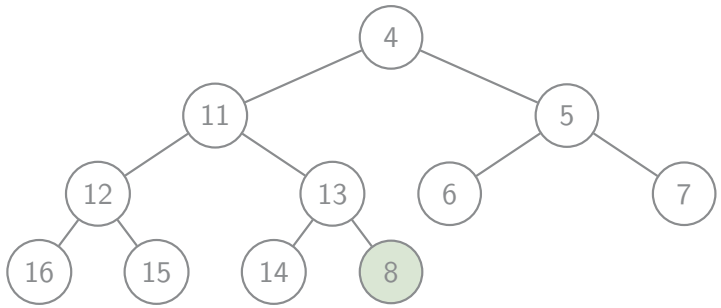


Binary Min-Heap

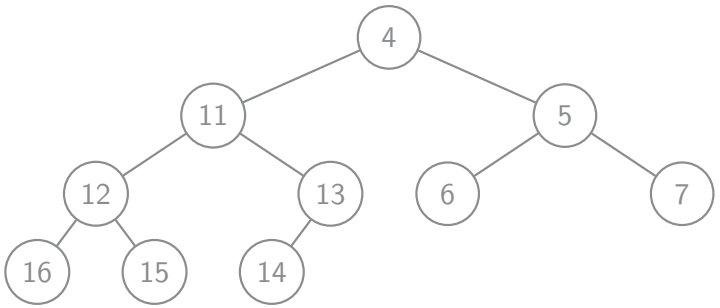


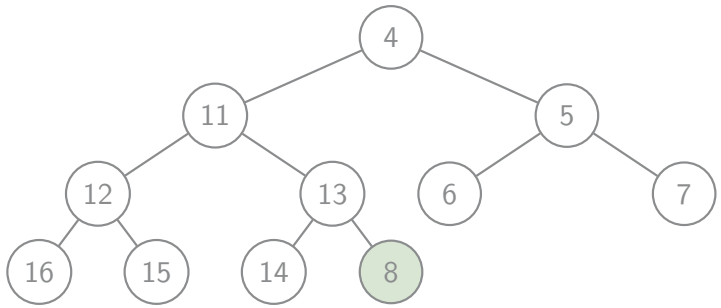
Binary Min-Heap

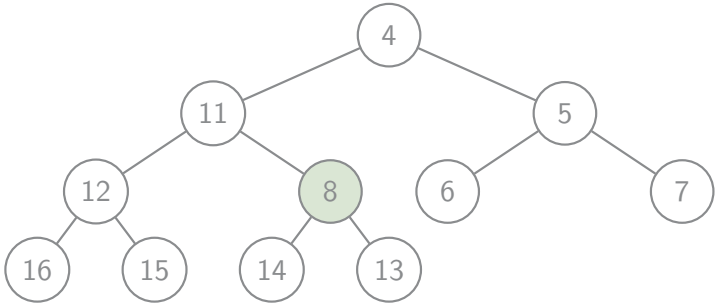




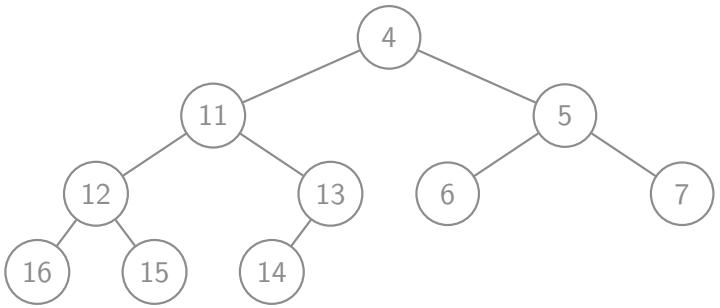
Binary Min-Heap

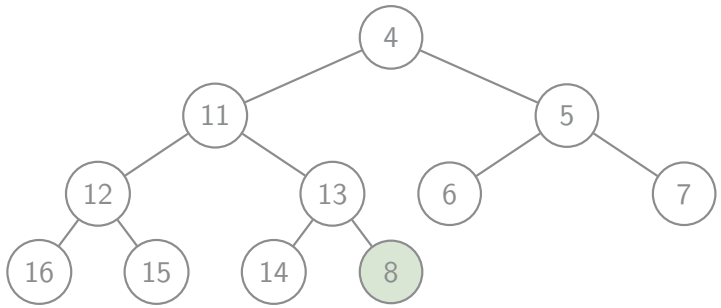


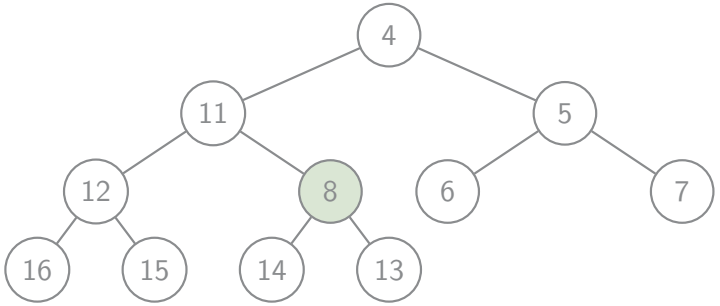


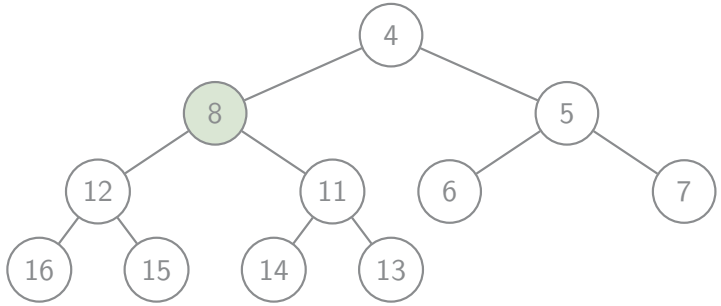


Binary Min-Heap

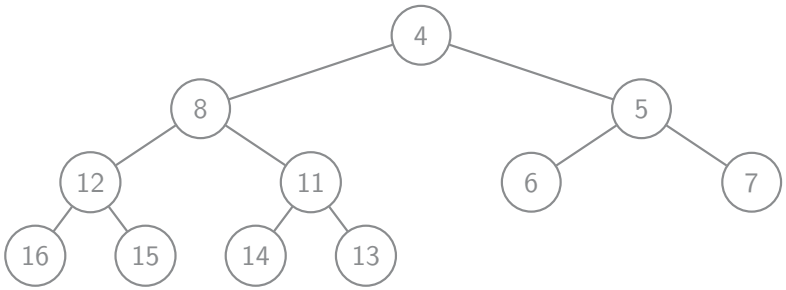




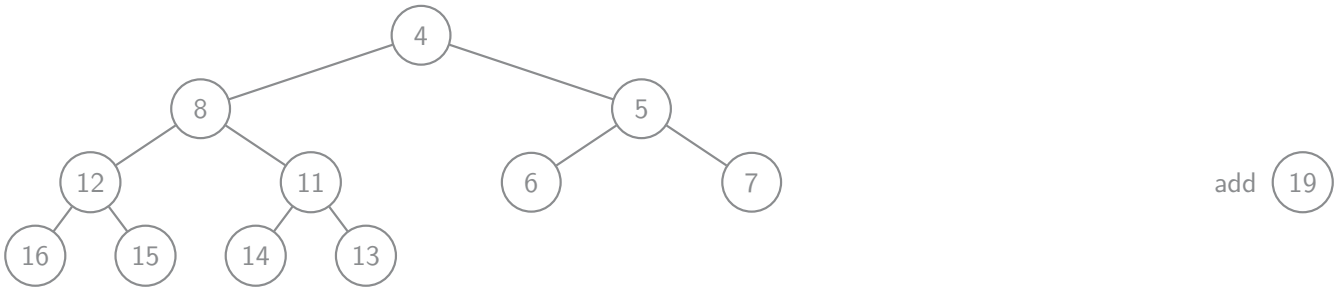




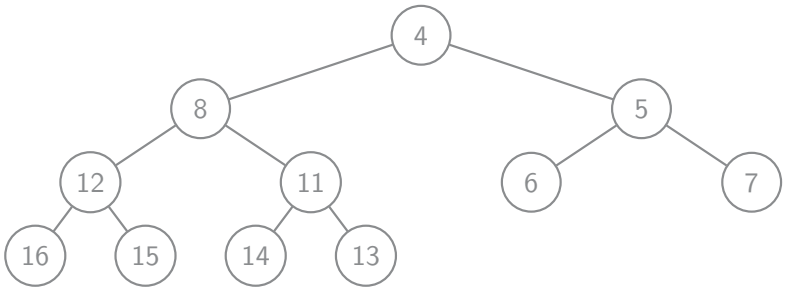
Binary Min-Heap

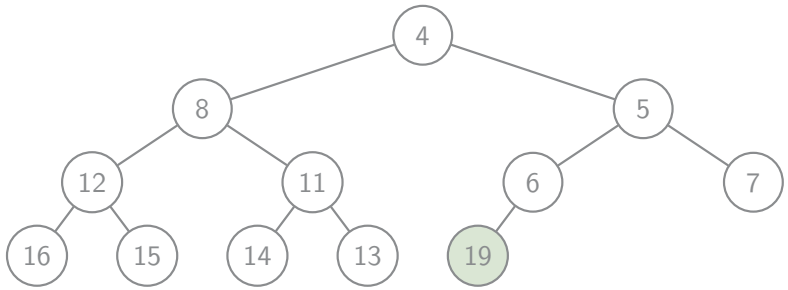


Binary Min-Heap

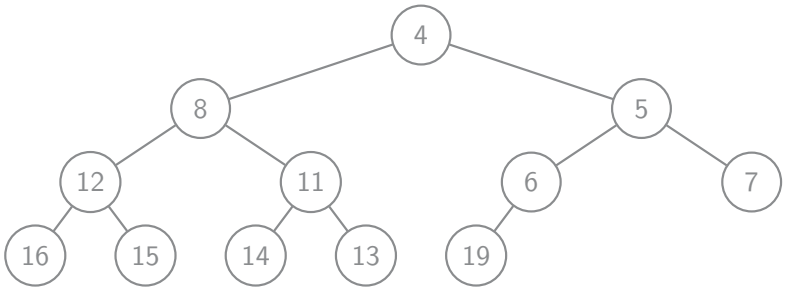


Binary Min-Heap

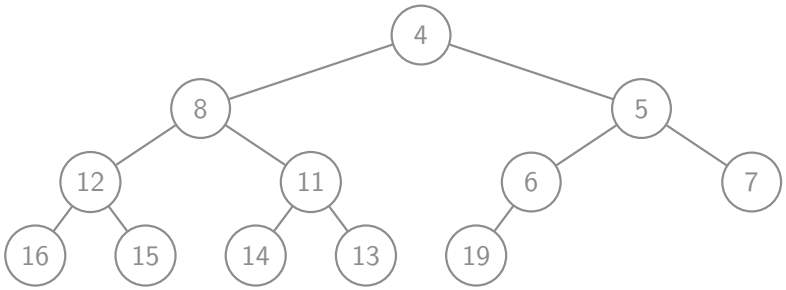




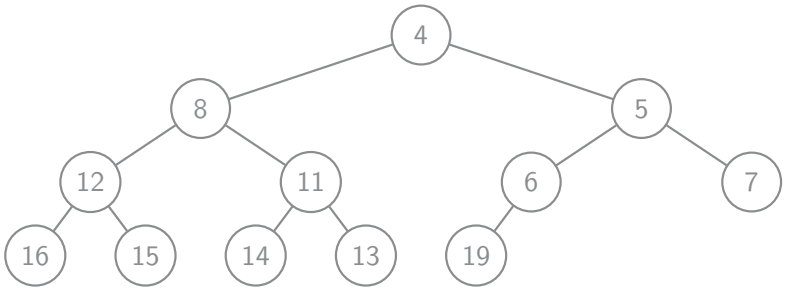
Binary Min-Heap

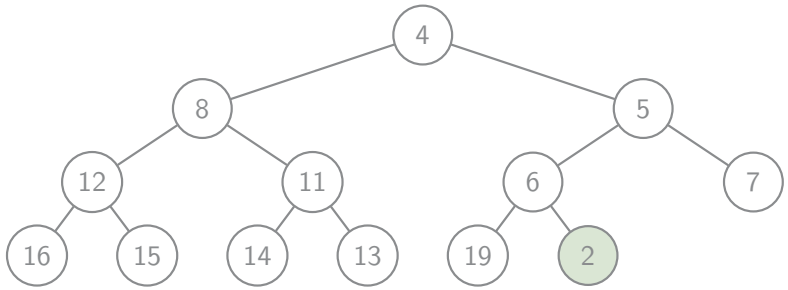


Binary Min-Heap

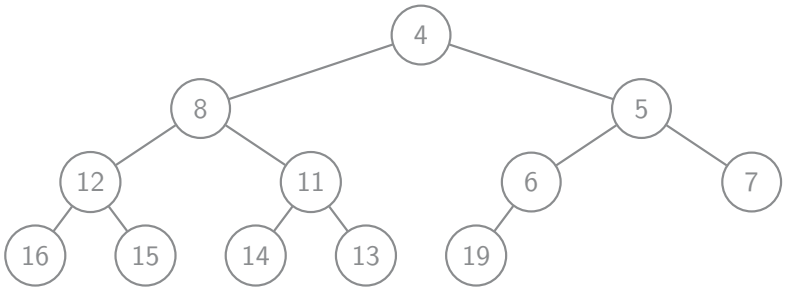


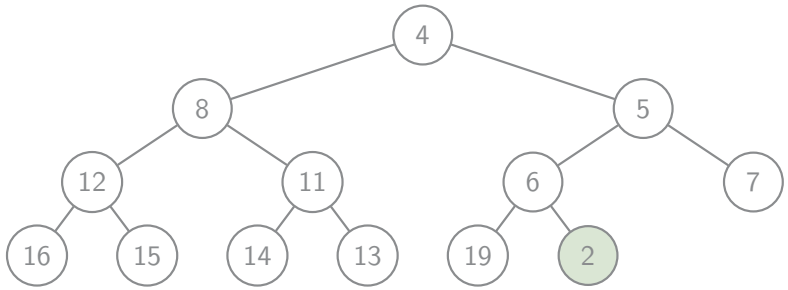
Binary Min-Heap

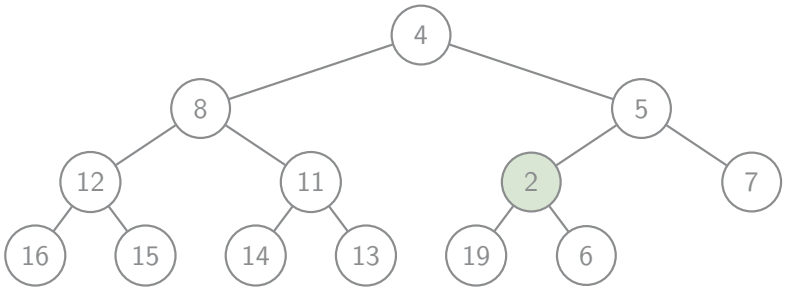




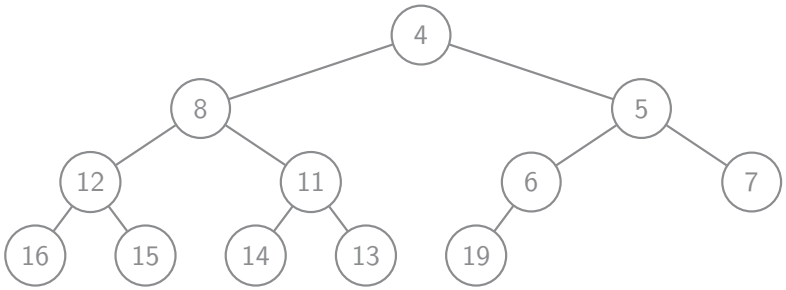
Binary Min-Heap

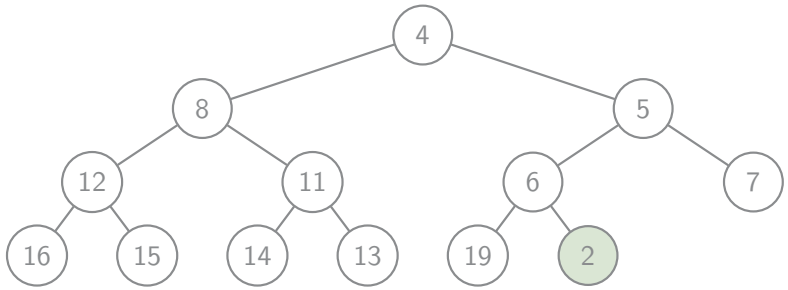


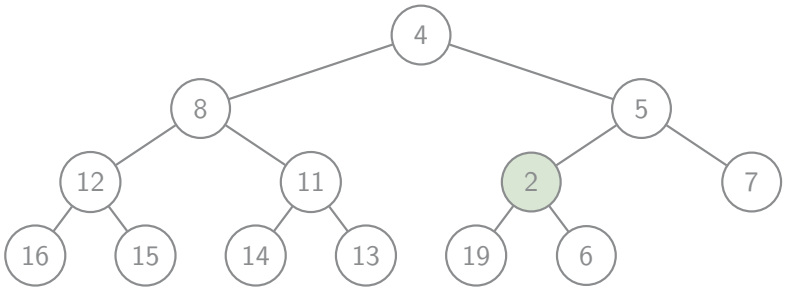


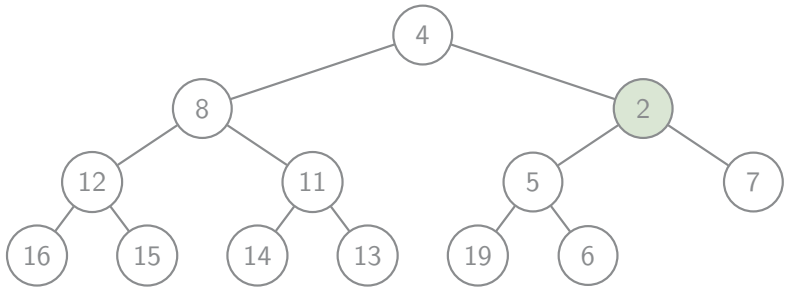


Binary Min-Heap

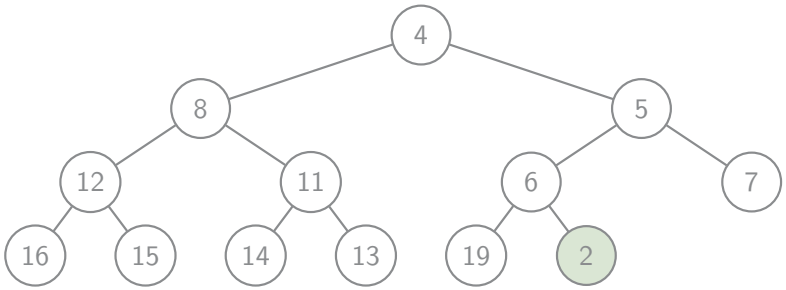


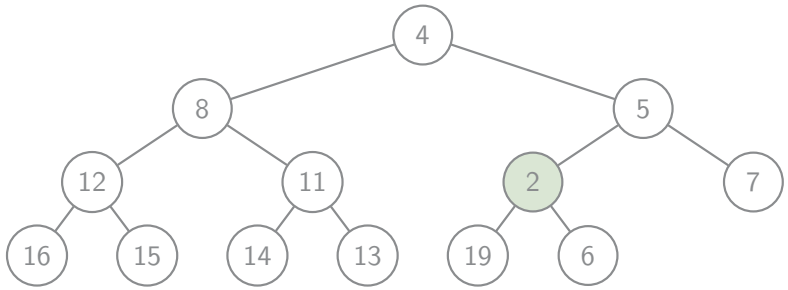


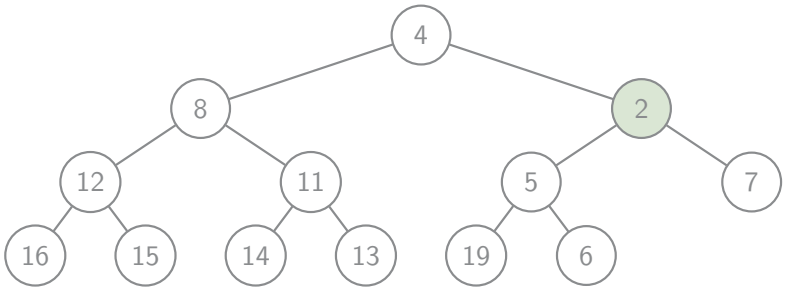




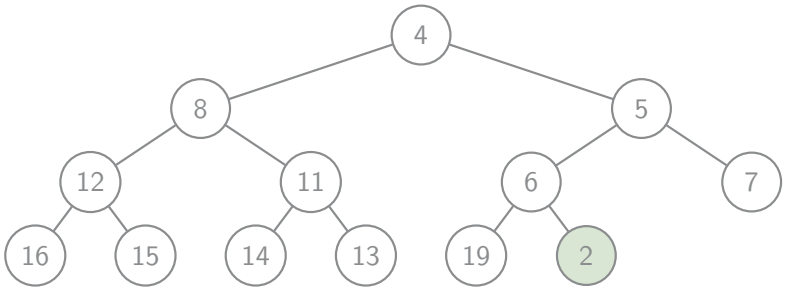
Binary Min-Heap

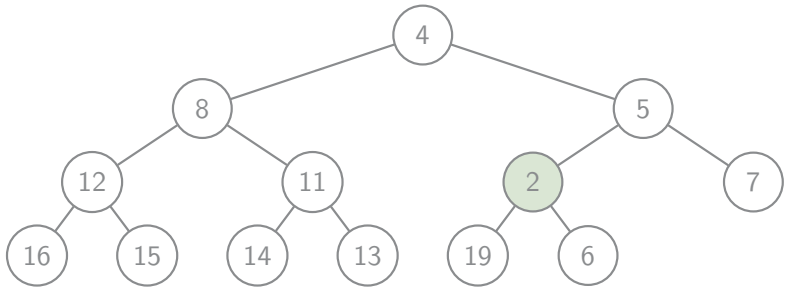


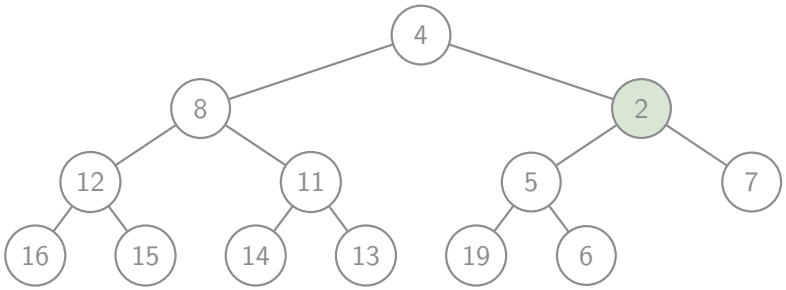


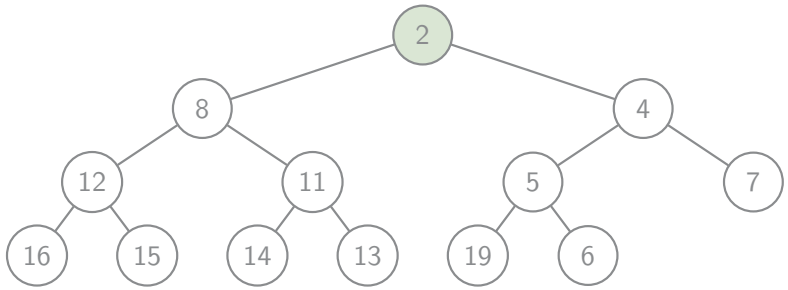


Binary Min-Heap

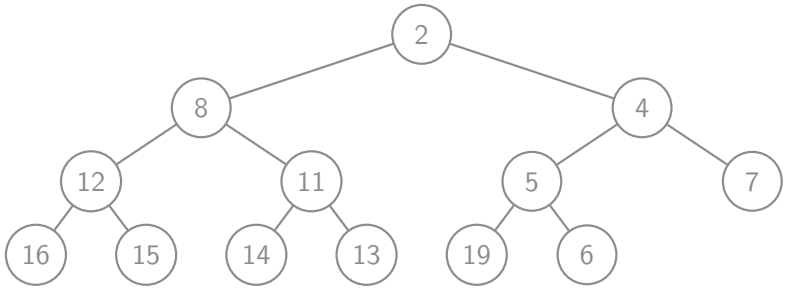




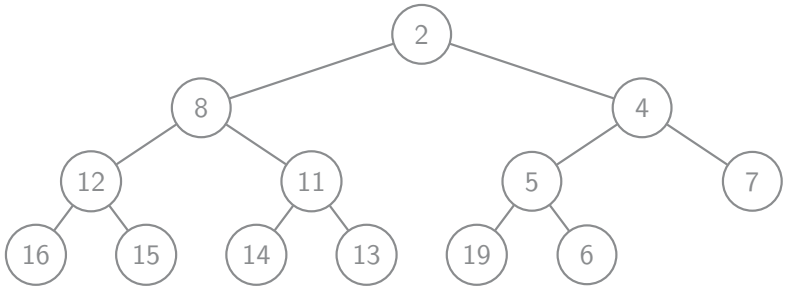




Binary Min-Heap

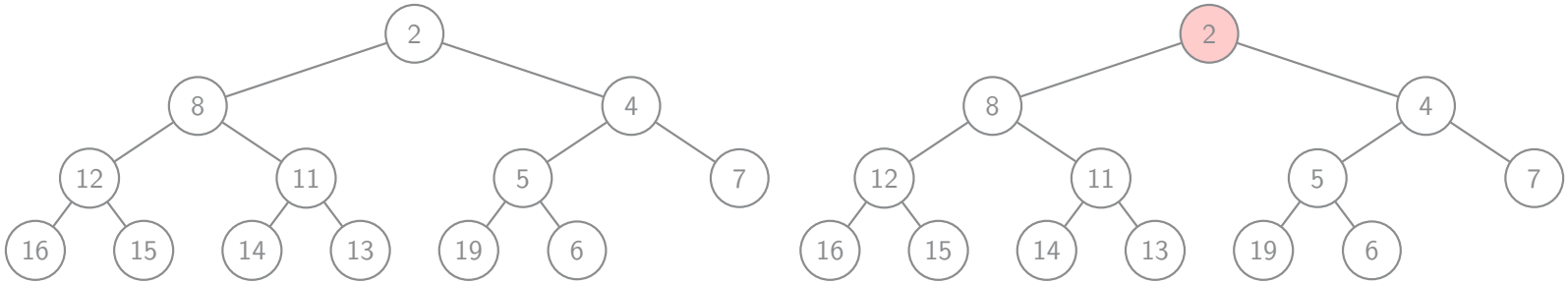


Binary Min-Heap

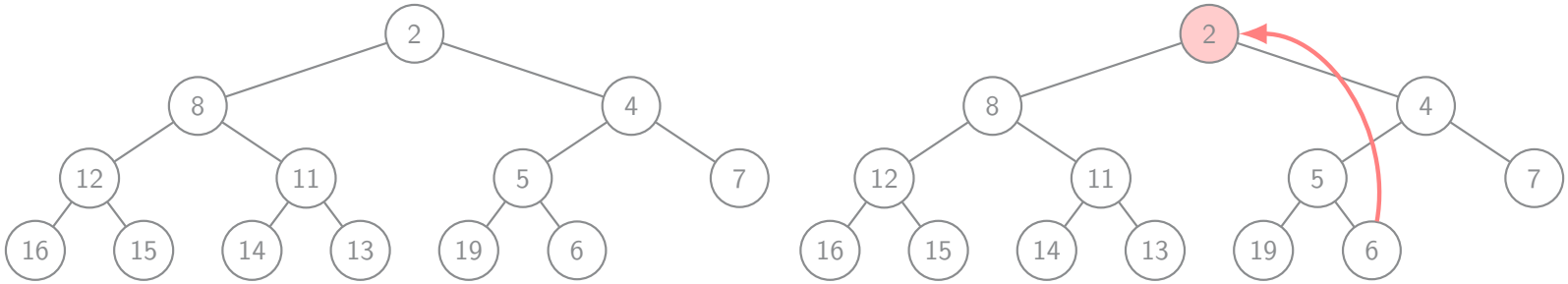


`poll`

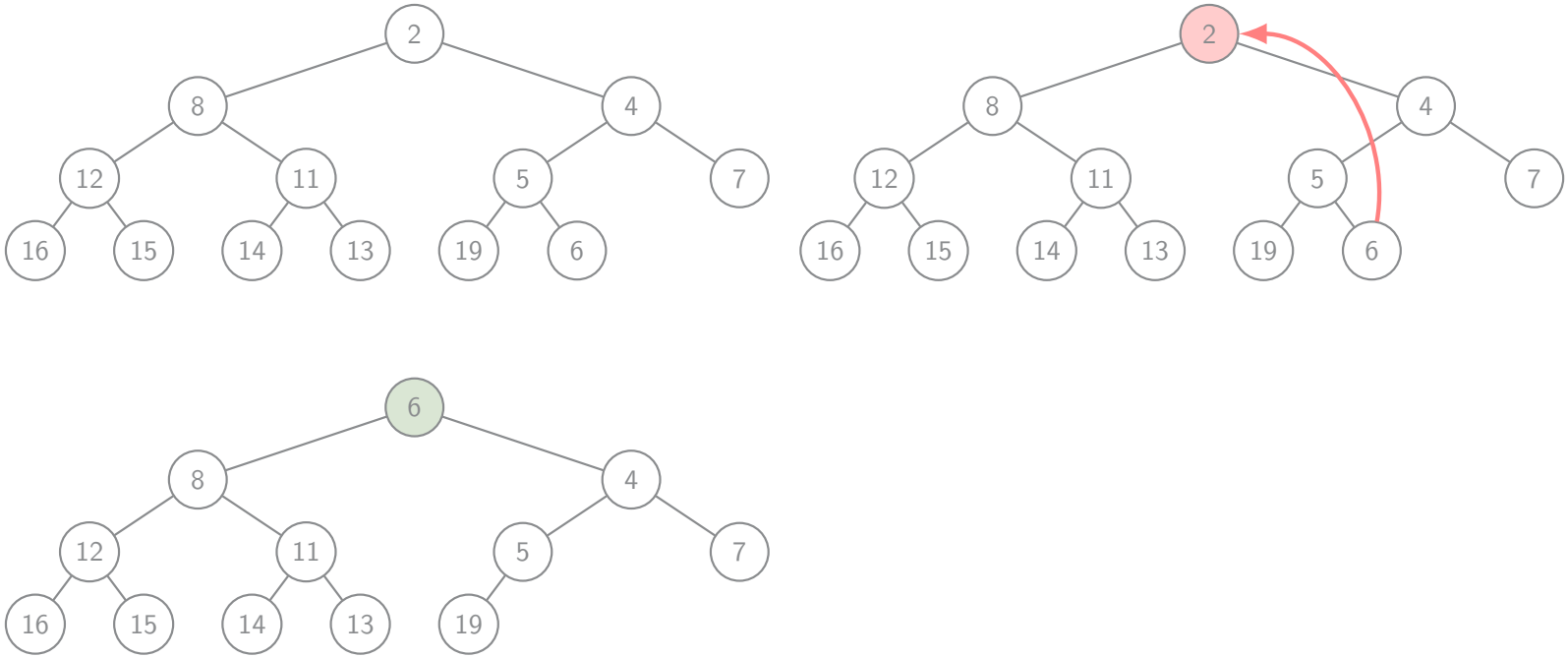
Binary Min-Heap



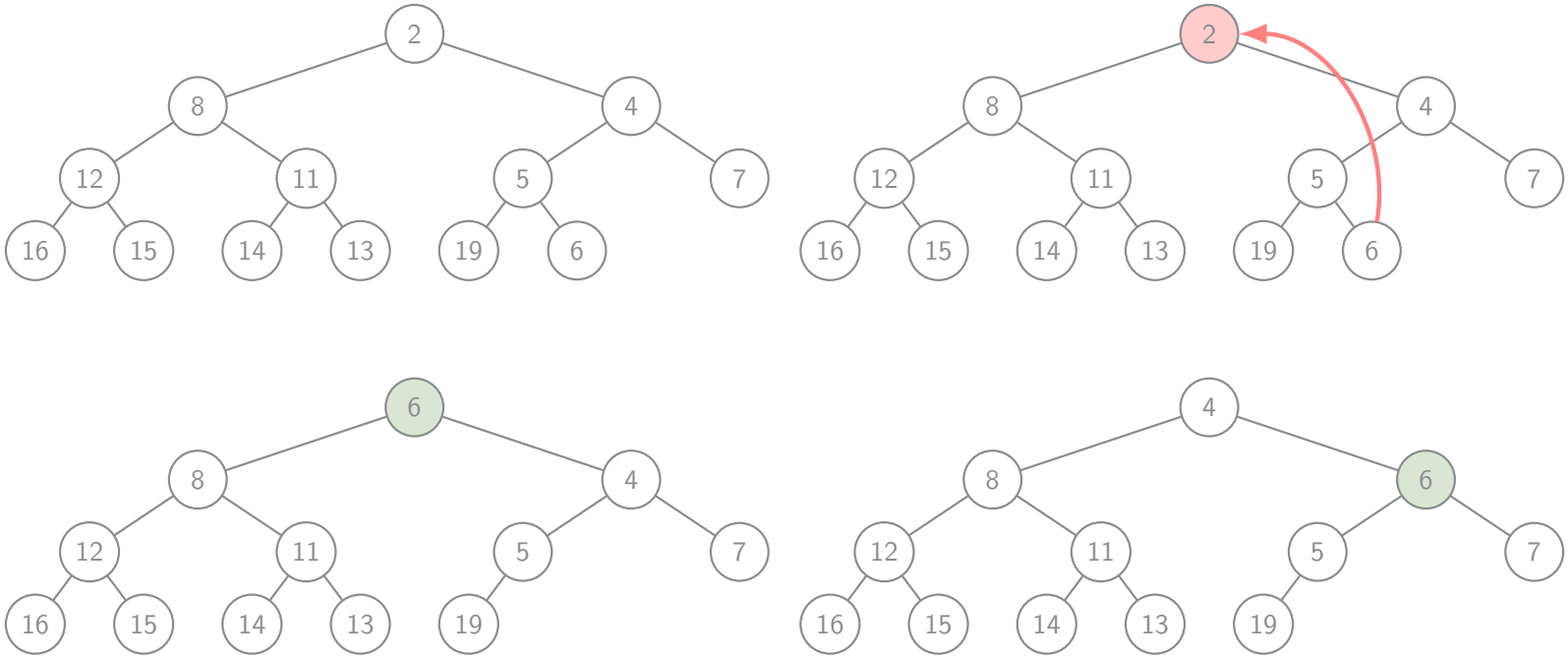
Binary Min-Heap



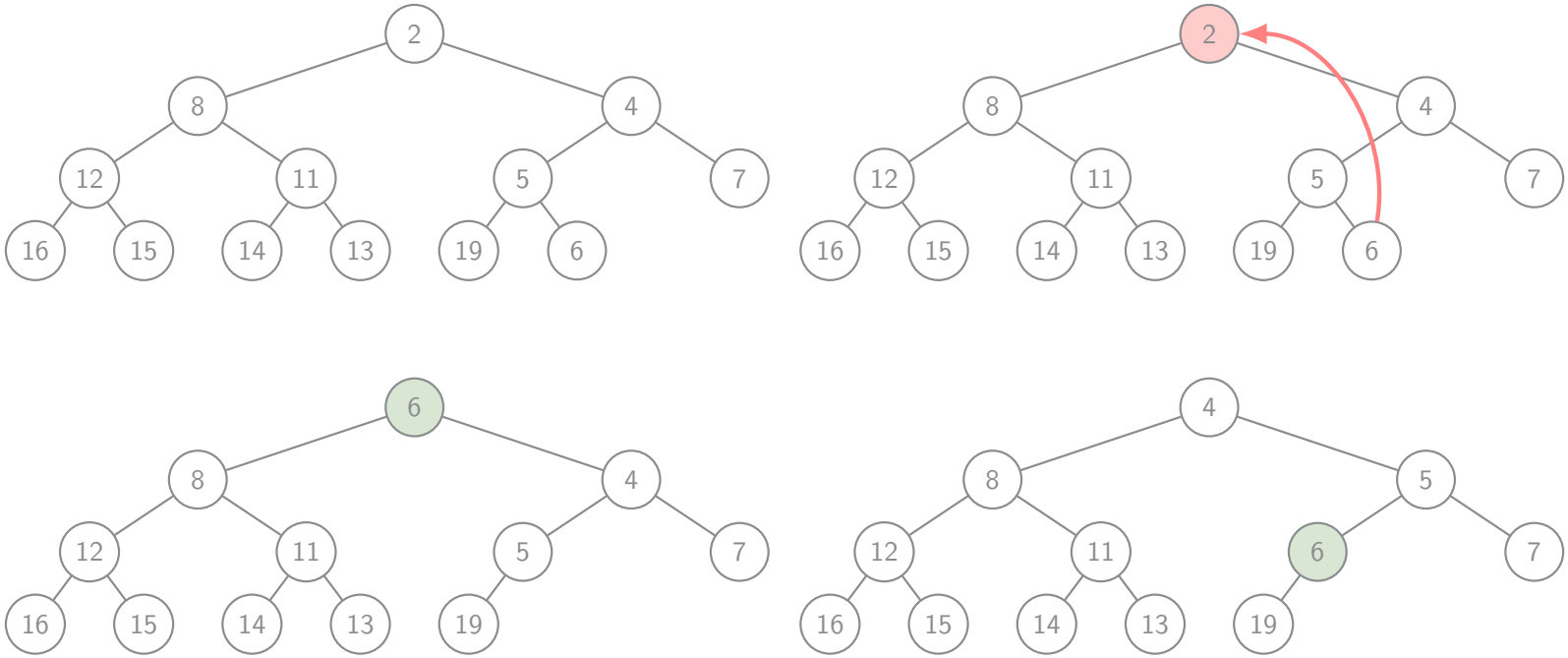
Binary Min-Heap



Binary Min-Heap



Binary Min-Heap



Binary Min-Heap

Operations

- `insert`
- `removeAt`
- `peek`
- `poll`

Binary Min-Heap

Operations

- `insert` $O(\log n)$
- `removeAt` $O(\log n)$
- `peek` $O(1)$
- `poll` $O(\log n)$

Binary Min-Heap

Operations

- `insert` $O(\log n)$
- `removeAt` $O(\log n)$
- `peek` $O(1)$
- `poll` $O(\log n)$

- Heapsort
- Frequently used in Priority Queues

Red-Black Tree

. . .

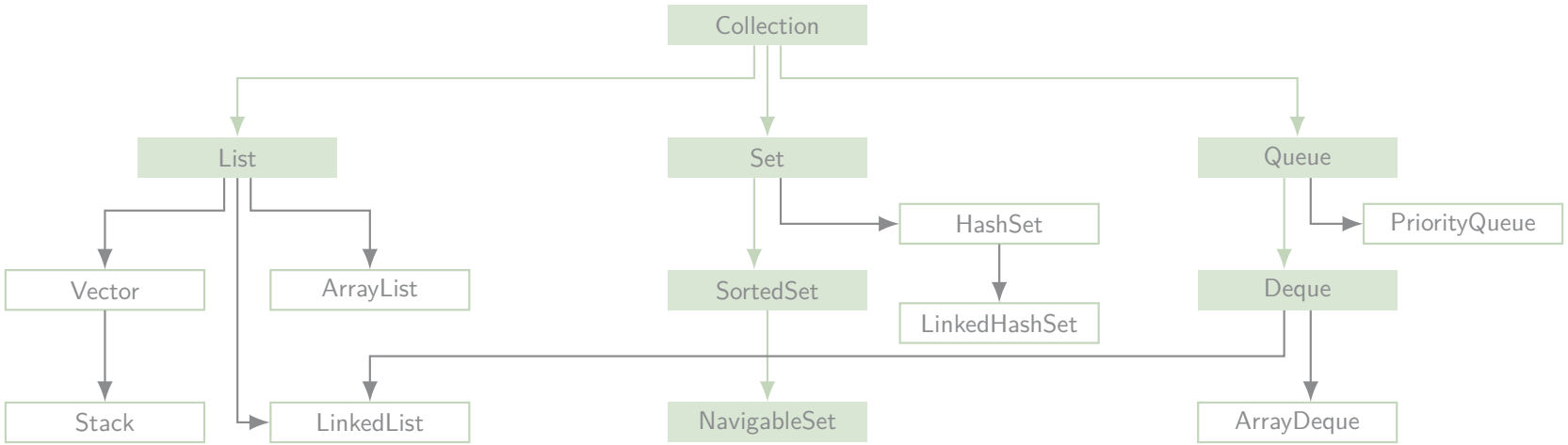
Red-Black Tree

. . .

Outline

Outline

Java Collection API



Java Collection API

...

Outline

Java Map API

. . .

Java Map API

. . .

Outline

Outline

Stuff. . .

. . .