

ACOL106: Data Structures and Algorithms

Ragesh Jaiswal, IIT Delhi

Data Structures

Heaps and Priority Queues

- What is the running time of each operation in the pointer based implementation:
 - `min()`: $O(1)$
 - `insert(k, v)`: $O(\log n)$
 - `removeMin()`: $O(\log n)$
- Pointer manipulations can sometimes be intricate (recall the `incrementLastNode` methods in the homework).
- Question: Can we implement Min-Heap using an **Array**?

Data Structures

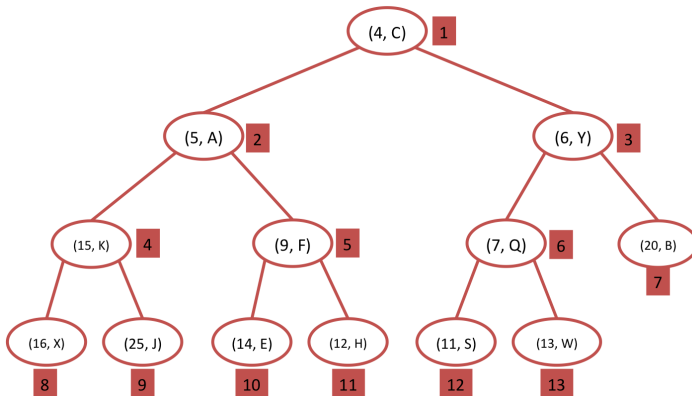
Heaps and Priority Queues

- What is the running time of each operation in the pointer based implementation:
 - `min()`: $O(1)$
 - `insert(k, v)`: $O(\log n)$
 - `removeMin()`: $O(\log n)$
- Pointer manipulations can sometimes be intricate (recall the `incrementLastNode` methods in the homework).
- Question: Can we implement Min-Heap using an **Array**?
 - Array elements are typically stored in contiguous locations in the memory and this has its advantages.

Data Structures

Heaps and Priority Queues

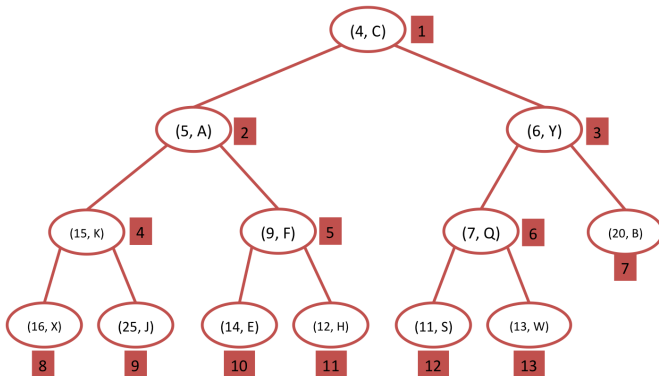
- Question: Can we implement Min-Heap using an **Array**?
- Consider the following example of a min-heap. Let us label the nodes of this **complete** binary tree level-wise as shown below.
- Do you observe some pattern in the labels?



Data Structures

Heaps and Priority Queues

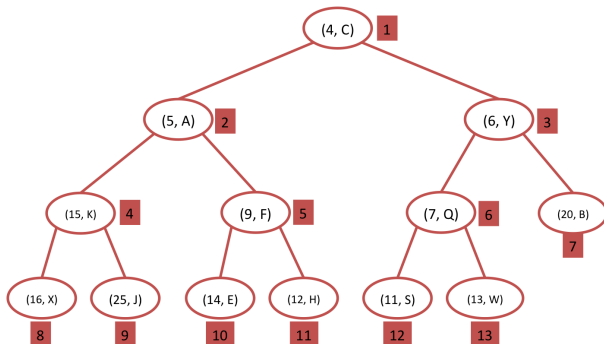
- Question: Can we implement Min-Heap using an **Array**?
- Consider the following example of a min-heap. Let us label the nodes of this **complete** binary tree level-wise as shown below.
- Do you observe some pattern in the labels?
 - Claim 1: Nodes of level i are labeled $2^i, \dots, 2^{i+1} - 1$.



Data Structures

Heaps and Priority Queues

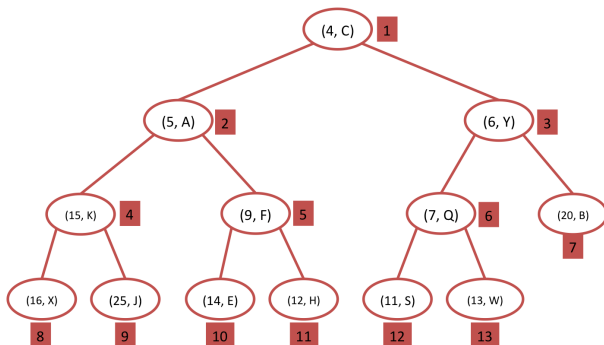
- Question: Can we implement Min-Heap using an **Array**?
- Consider the following example of a min-heap. Let us label the nodes of this **complete** binary tree level-wise as shown below.
- Do you observe some pattern in the labels?
 - Claim 1: Nodes of level i are labeled $2^i, \dots, 2^{i+1} - 1$.
 - Claim 2: For every node labeled i , the label of its children (if they exist) are _____ and _____ and the label of its parent is _____.



Data Structures

Heaps and Priority Queues

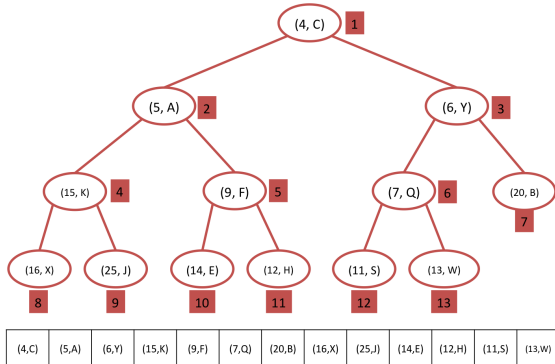
- Question: Can we implement Min-Heap using an **Array**?
- Consider the following example of a min-heap. Let us label the nodes of this **complete** binary tree level-wise as shown below.
- Do you observe some pattern in the labels?
 - Claim 1: Nodes of level i are labeled $2^i, \dots, 2^{i+1} - 1$.
 - Claim 2: For every node labeled i , the label of its children (if they exist) are $2i$ and $(2i + 1)$ and the label of its parent is $\lfloor i/2 \rfloor$.



Data Structures

Heaps and Priority Queues

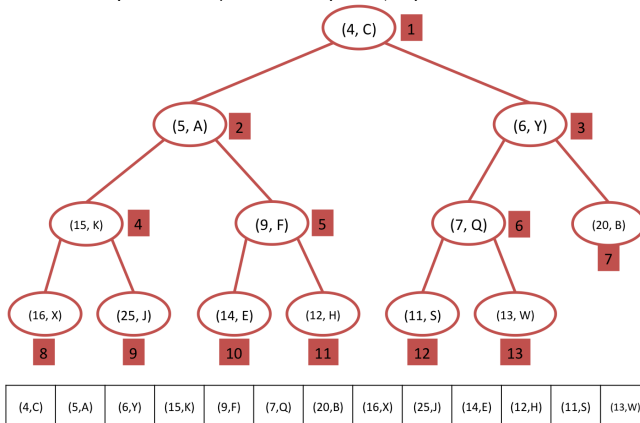
- Question: Can we implement Min-Heap using an **Array**?
- Consider the following example of a min-heap. Let us label the nodes of this **complete** binary tree level-wise as shown below.
- Do you observe some pattern in the labels?
 - Claim 1: Nodes of level i are labeled $2^i, \dots, 2^{i+1} - 1$.
 - Claim 2: For every node labeled i , the label of its children (if they exist) are $2i$ and $(2i + 1)$ and the label of its parent is $\lfloor i/2 \rfloor$.
- Main idea: Store node labeled i at index i of the array.



Data Structures

Heaps and Priority Queues

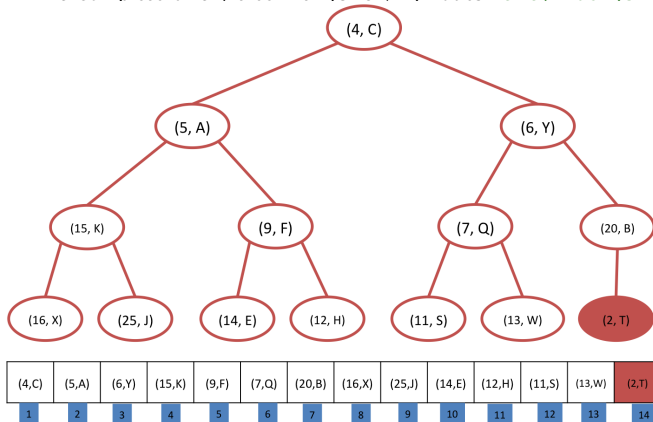
- Question: Can we implement Min-Heap using an **Array**?
- Let $size = 13$ denote the current size of the heap. Suppose we want to insert $(2, T)$.
 - Which array location should we put this entry in order to maintain a complete binary tree with $(size + 1)$ nodes?



Data Structures

Heaps and Priority Queues

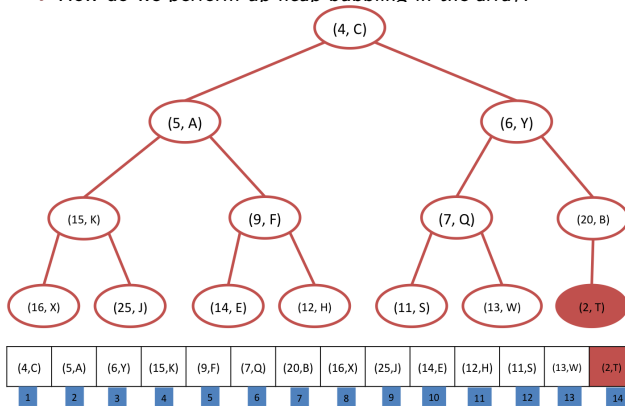
- Question: Can we implement Min-Heap using an **Array**?
- Let $size = 13$ denote the current size of the heap. Suppose we want to insert $(2, T)$.
 - Which array location should we put this entry in order to maintain a complete binary tree with $(size + 1)$ nodes? **array index $(size + 1)$**



Data Structures

Heaps and Priority Queues

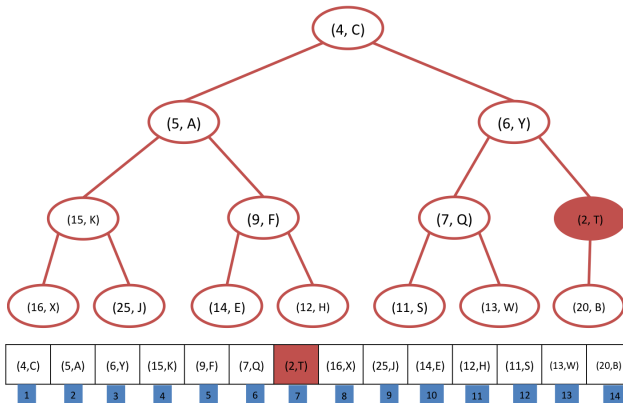
- Question: Can we implement Min-Heap using an **Array**?
- Let $size = 13$ denote the current size of the heap. Suppose we want to insert $(2, T)$.
 - Which array location should we put this entry in order to maintain a complete binary tree with $(size + 1)$ nodes? **array index $(size + 1)$**
 - How do we perform *up-heap bubbling* in the array?



Data Structures

Heaps and Priority Queues

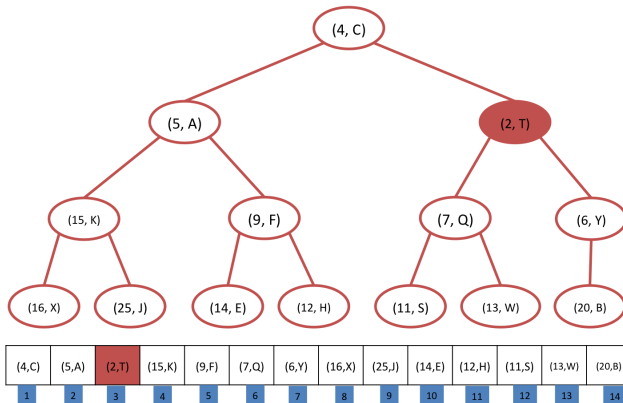
- Question: Can we implement Min-Heap using an **Array**?
- Let $size = 13$ denote the current size of the heap. Suppose we want to insert $(2, T)$.
 - Which array location should we put this entry in order to maintain a complete binary tree with $(size + 1)$ nodes? **array index $(size + 1)$**
 - How do we perform *up-heap bubbling* in the array?



Data Structures

Heaps and Priority Queues

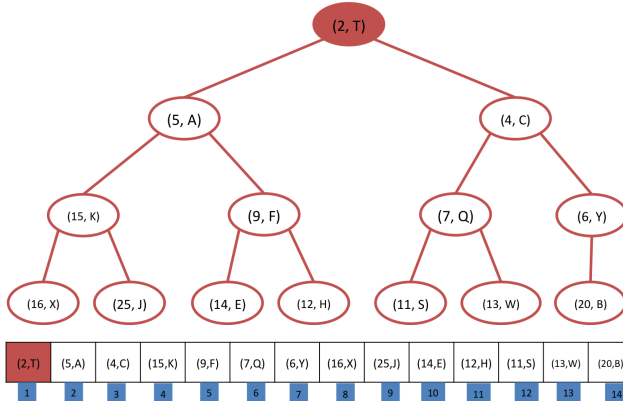
- Question: Can we implement Min-Heap using an **Array**?
- Let $size = 13$ denote the current size of the heap. Suppose we want to insert $(2, T)$.
 - Which array location should we put this entry in order to maintain a complete binary tree with $(size + 1)$ nodes? **array index $(size + 1)$**
 - How do we perform *up-heap bubbling* in the array?



Data Structures

Heaps and Priority Queues

- Question: Can we implement Min-Heap using an **Array**?
- Let $size = 13$ denote the current size of the heap. Suppose we want to insert $(2, T)$.
 - Which array location should we put this entry in order to maintain a complete binary tree with $(size + 1)$ nodes? **array index $(size + 1)$**
 - How do we perform *up-heap bubbling* in the array?



Data Structures

Heaps and Priority Queues

- Consider an array-based implementation of a min-heap.

Implementation

```
class minHeapList:
    """ The min heap using list class """
    def __init__(self):
        self._Heap = []
        self._size = 0

    def _upHeapBubble(self, index):

    def _downHeapBubble(self, index):

    def min(self):

    def insert(self, key, value):

    def deleteMin(self):
```

Data Structures

Heaps and Priority Queues

- What is the running time of each of these operations in the array-based implementation of Min-Heap?
 - `insert(k, v)`:
 - `min()`:
 - `removeMin()`:

Data Structures

Heaps and Priority Queues

- What is the running time of each of these operations in the array-based implementation of Min-Heap?
 - `insert(k, v)`: $O(\log n)$
 - `min()`: $O(1)$
 - `removeMin()`: $O(\log n)$

Problem

Given n entries, create a min-heap of these entries.

Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations.
 - What is the running time?

Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations.
 - What is the running time? $O(n \log n)$

Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**
 - Question: Suppose we have a min-heap H_1 and H_2 both containing $2^h - 1$ entries and an entry E . Can you construct a min-heap for all entries in H_1 , H_2 , and E combined? What is the running time for your combination algorithm?

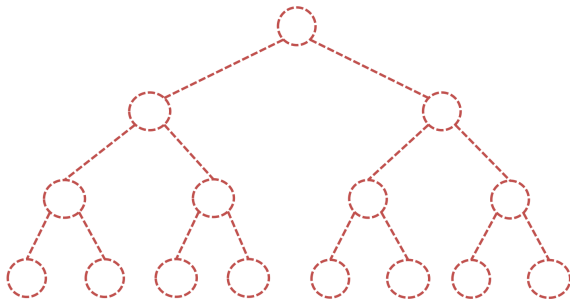
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



14	5	8	25	9	11	7	16	15	4	12	6	7	23	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

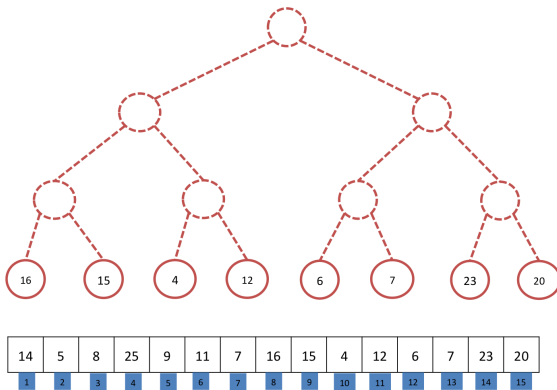
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



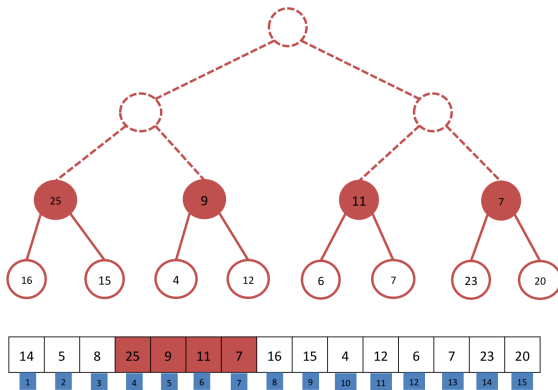
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



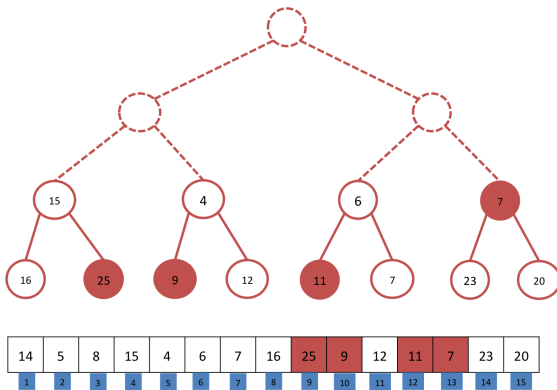
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



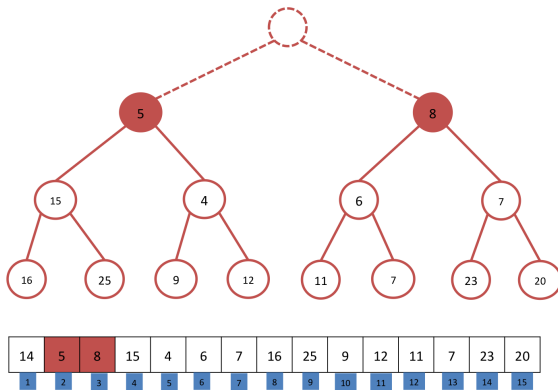
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



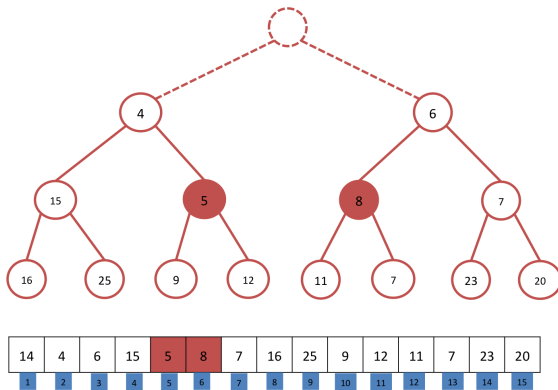
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



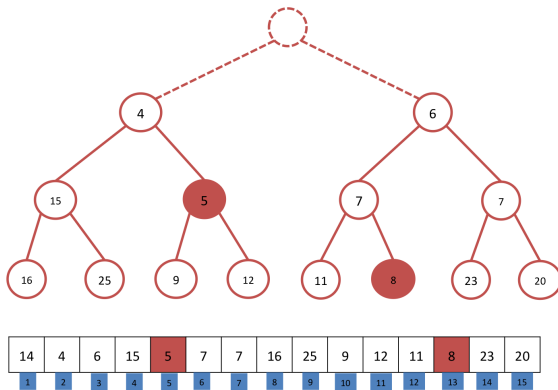
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



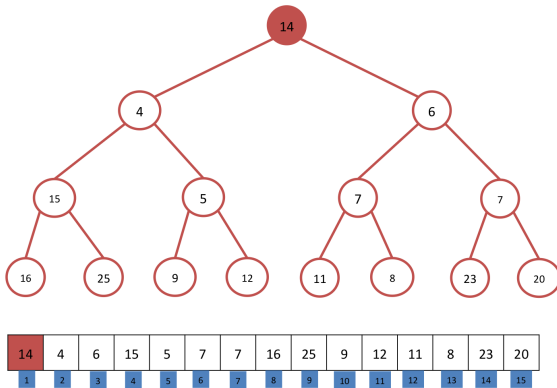
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



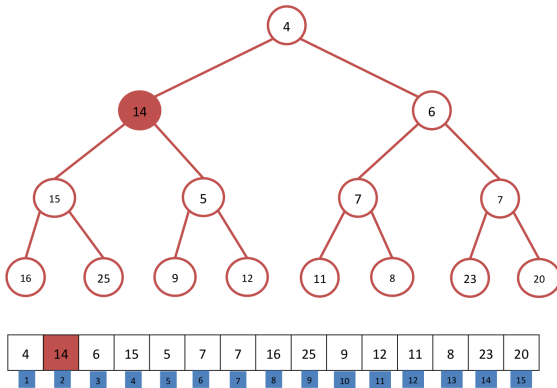
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



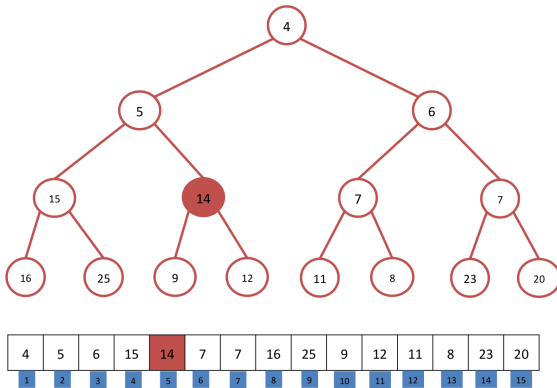
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



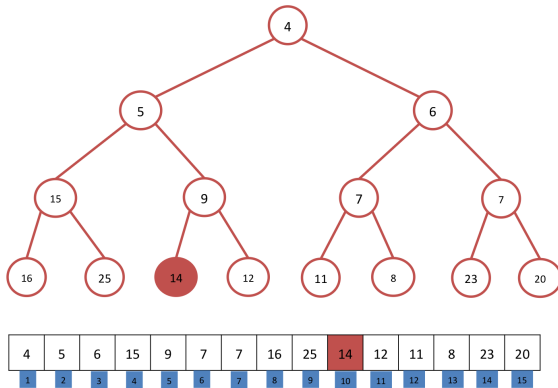
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



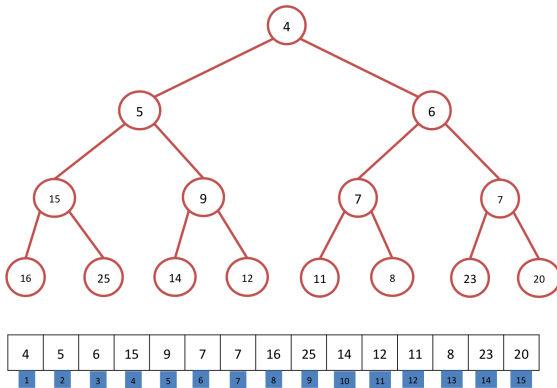
Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**



Data Structures

Heaps and Priority Queues

Problem

Given n entries, create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**
 - Suppose this construction is performed on an array with $n = 2^{h+1} - 1$ entries. What is the running time?

Data Structures

Heaps and Priority Queues

Problem

Given n entries create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**
 - Suppose this construction is performed on an array with $n = 2^{h+1} - 1$ entries. What is the running time?
 - Claim: The worst case running time is given by the expression:

$$F(h) = 2^{h-1} \cdot 1 + 2^{h-2} \cdot 2 + \dots + 2^{h-h} \cdot h$$

Data Structures

Heaps and Priority Queues

Problem

Given n entries create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**
 - Suppose this construction is performed on an array with $n = 2^{h+1} - 1$ entries. What is the running time?
 - Claim: The worst case running time is given by the expression:

$$F(h) = 2^{h-1} \cdot 1 + 2^{h-2} \cdot 2 + \dots + 2^{h-h} \cdot h$$

- How do we simplify the above expression?

Data Structures

Heaps and Priority Queues

Problem

Given n entries create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**
 - Suppose this construction is performed on an array with $n = 2^{h+1} - 1$ entries. What is the running time?
 - Claim: The worst case running time is given by the expression:

$$F(h) = 2^{h-1} \cdot 1 + 2^{h-2} \cdot 2 + \dots + 2^{h-h} \cdot h$$

- We can write:

$$\begin{aligned} F(h) &= \sum_{i=0}^{h-1} 2^i + \sum_{i=0}^{h-2} 2^i + \dots + \sum_{i=0}^0 2^i \\ &= (2^h - 1) + (2^{h-1} - 1) + \dots + (2^1 - 1) \\ &= \sum_{i=1}^h 2^i - h \\ &= 2^{h+1} - 2 - h \end{aligned}$$

Data Structures

Heaps and Priority Queues

Problem

Given n entries create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction**
 - Suppose this construction is performed on an array with $n = 2^{h+1} - 1$ entries. What is the running time?
 - Claim: The worst case running time is given by the expression:

$$F(h) = 2^{h-1} \cdot 1 + 2^{h-2} \cdot 2 + \dots + 2^{h-h} \cdot h$$

- We can write:

$$\begin{aligned} F(h) &= \sum_{i=0}^{h-1} 2^i + \sum_{i=0}^{h-2} 2^i + \dots + \sum_{i=0}^0 2^i \\ &= (2^h - 1) + (2^{h-1} - 1) + \dots + (2^1 - 1) \\ &= \sum_{i=1}^h 2^i - h \\ &= 2^{h+1} - 2 - h \leq 2^{h+1} - 1 = n \end{aligned}$$

- So, the running time of bottom-up heap construction is $O(n)$.

Data Structures

Heaps and Priority Queues

Problem

Given n entries create a min-heap of these entries.

- Method 1: Perform n insert operations in $O(n \log n)$ time.
- Method 2: **Bottom-up heap construction** in $O(n)$ time.

End