COL106: Data Structures and Algorithms

Ragesh Jaiswal, IIT Delhi

Data Structures: Heaps and Priority Queues

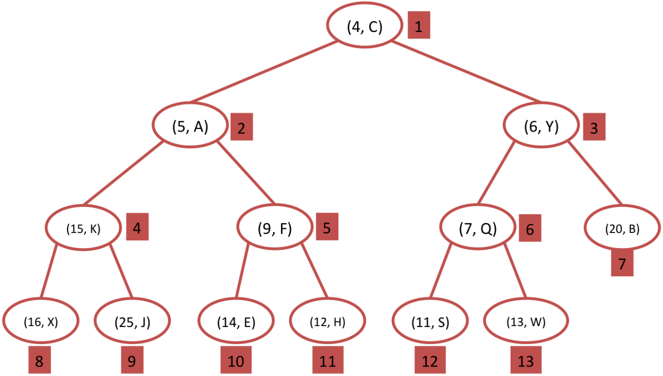
- What is the running time of each operation in the pointer based implementation:
  - `min()`: $O(1)$
  - `insert(k, v)`: $O(\log n)$
  - `removeMin()`: $O(\log n)$

- What is the running time of each operation in the pointer based implementation:
  - `min()`: $O(1)$
  - `insert(k, v)`: $O(\log n)$
  - `removeMin()`: $O(\log n)$
- Note that a Max-Heap can be defined in a similar manner as a Min-Heap.

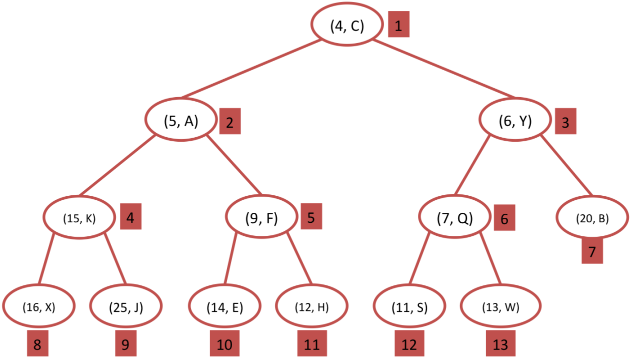
- What is the running time of each operation in the pointer based implementation:
    - `min()`: $O(1)$
    - `insert(k, v)`: $O(\log n)$
    - `removeMin()`: $O(\log n)$
- Pointer manipulations can sometimes be intricate (recall the `incrementLastNode` methods in the homework).
- Question: Can we implement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

- 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.

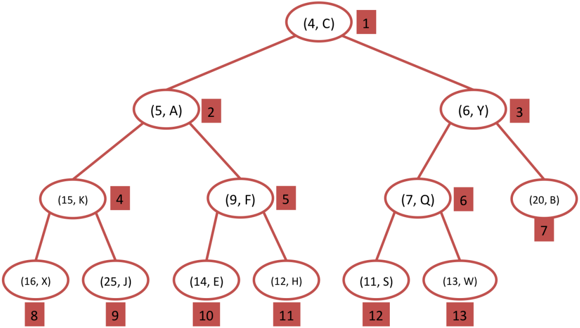
- Question: Can we implement Min-Heap using an Array?
- Consider the following example of a min-heap. Let us label the nodes of this complete binary tree level-wise as shown below.
- Do you observe some pattern in the labels?



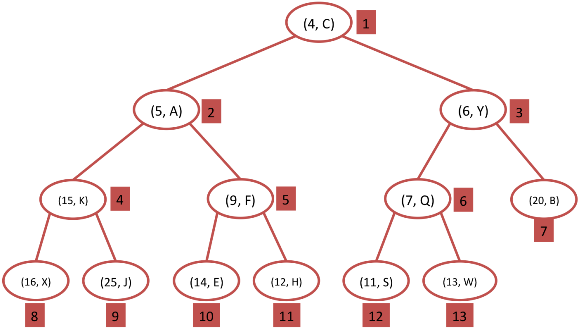
- Question: Can we implement Min-Heap using an Array?
- Consider the following example of a min-heap. Let us label the nodes of this complete binary tree level-wise as shown below.
- Do you observe some pattern in the labels?
  - Claim 1: Nodes of level i are labeled $2^i, ..., 2^{i+1} - 1$.



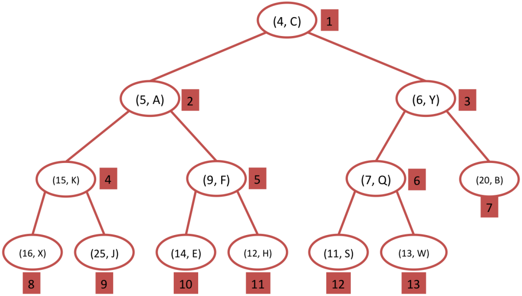
- Question: Can we implement Min-Heap using an Array?
- Consider the following example of a min-heap. Let us label the nodes of this complete binary tree level-wise as shown below.
- Do you observe some pattern in the labels?
  - Claim 1: Nodes of level i are labeled $2^i, ..., 2^{i+1} - 1$.
  - Claim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



- 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 $2^i, ..., 2^{i+1} - 1$.
  - Claim 2: For every node labeled i, the label of it children (if they exist) are $2i$ and $(2i + 1)$ and the label of its parent is $\lfloor i/2 \rfloor$.

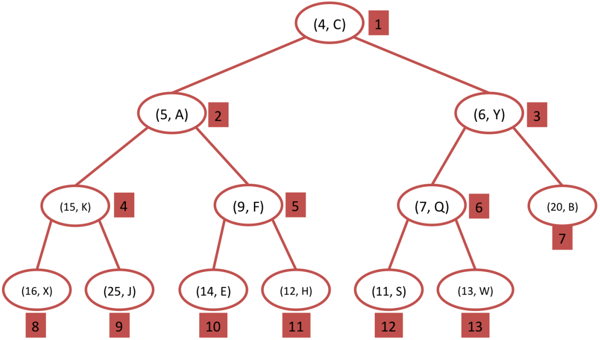


- Question: Can we implement Min-Heap using an Array?
- Consider the following example of a min-heap. Let us label the nodes of this complete binary tree level-wise as shown below.
- Do you observe some pattern in the labels?
  - Claim 1: Nodes of level i are labeled $2^i, ..., 2^{i+1} - 1$.
  - Claim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



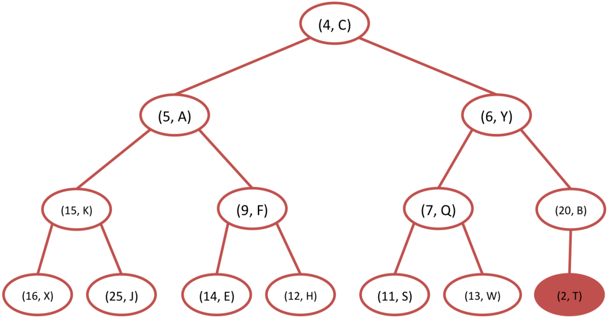
(4,C)	(5,A)	(6,Y)	(15,K)	(9,F)	(7,Q)	(20,B)	(16,X)	(25,J)	(14,E)	(12,H)	(11,S)	(13,W)

- 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?



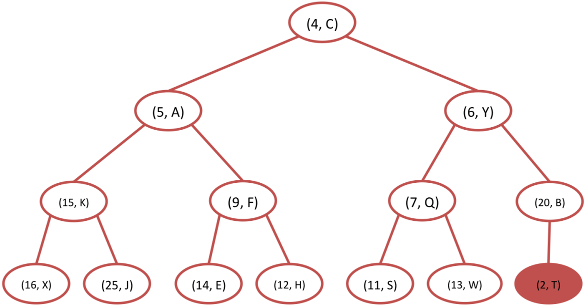
(4,C)	(5,A)	(6,Y)	(15,K)	(9,F)	(7,Q)	(20,B)	(16,X)	(25,J)	(14,E)	(12,H)	(11,S)	(13,W)

- 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 $(size + 1)$



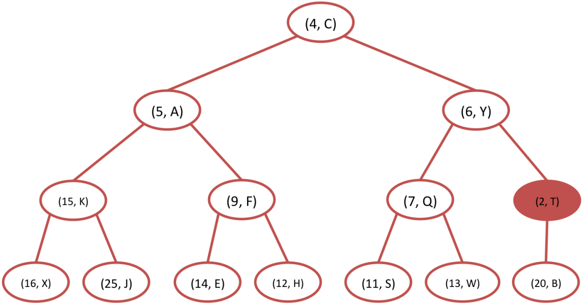
(4,C)	(5,A)	(6,Y)	(15,K)	(9,F)	(7,Q)	(20,B)	(16,X)	(25,J)	(14,E)	(12,H)	(11,S)	(13,W)	(2,T)
1	2	3	4	5	6	7	8	9	10	11	12	13	14

- 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?



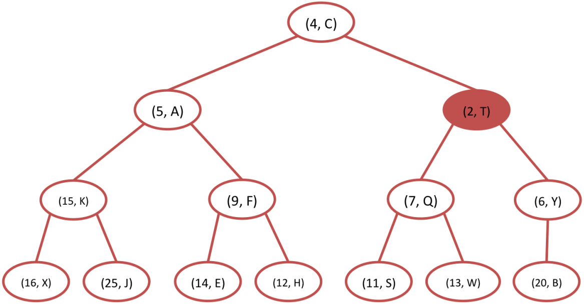
(4,C)	(5,A)	(6,Y)	(15,K)	(9,F)	(7,Q)	(20,B)	(16,X)	(25,J)	(14,E)	(12,H)	(11,S)	(13,W)	(2,T)
1	2	3	4	5	6	7	8	9	10	11	12	13	14

- 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?



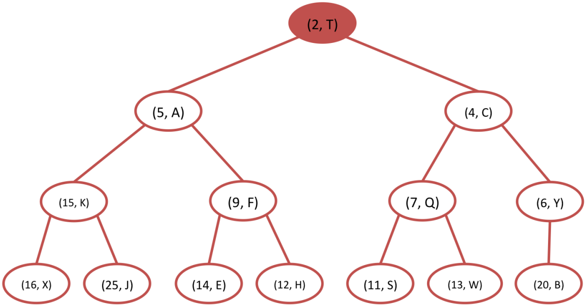
(4,C)	(5,A)	(6,Y)	(15,K)	(9,F)	(7,Q)	(2,T)	(16,X)	(25,J)	(14,E)	(12,H)	(11,S)	(13,W)	(20,B)
1	2	3	4	5	6	7	8	9	10	11	12	13	14

- 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?



(4,C)	(5,A)	(2,T)	(15,K)	(9,F)	(7,Q)	(6,Y)	(16,X)	(25,J)	(14,E)	(12,H)	(11,S)	(13,W)	(20,B)
1	2	3	4	5	6	7	8	9	10	11	12	13	14

- 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?



(2,T)	(5,A)	(4,C)	(15,K)	(9,F)	(7,Q)	(6,Y)	(16,X)	(25,J)	(14,E)	(12,H)	(11,S)	(13,W)	(20,B)
1	2	3	4	5	6	7	8	9	10	11	12	13	14

- Consider an array based implementation of min-heap.

Implementation

```
class Entry{
  public int key;
  public String value;
  public Entry(int k, String v){
     key = k;
     value = v;
  }
}
public class MinHeapArray{
  final int MAX_HEAP_SIZE = 1000;
  public Entry[] A;
  public int size;
  public MinHeapArray(){
     size = 0;
     A = new Entry[MAX_HEAP_SIZE];
  }
  public void upHeapBubble(int i){//To be written}
  public void downHeapBubble(int i){//To be written}
  public String min(){//To be written}
  public void insert(int k, String v){//To be written}
  public String removeMin(){//To be written}
}
```

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

- 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.

Problem

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

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

Problem

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

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

Problem

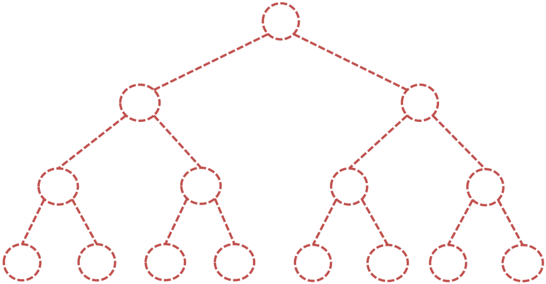
Given n entries create a min-heap of these entries.

- 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?

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

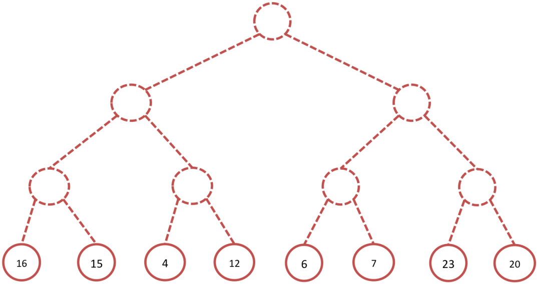


14	5	8	25	9	11	7	16	15	4	12	6	7	23	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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

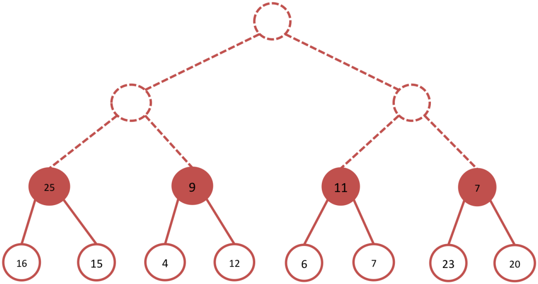


14	5	8	25	9	11	7	16	15	4	12	6	7	23	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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

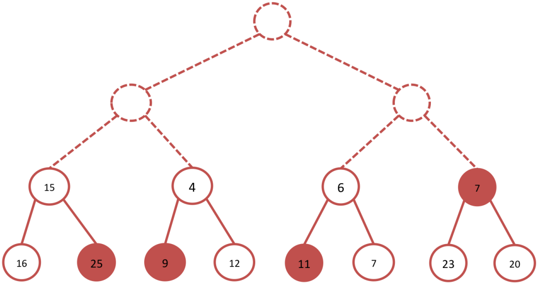


14	5	8	25	9	11	7	16	15	4	12	6	7	23	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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

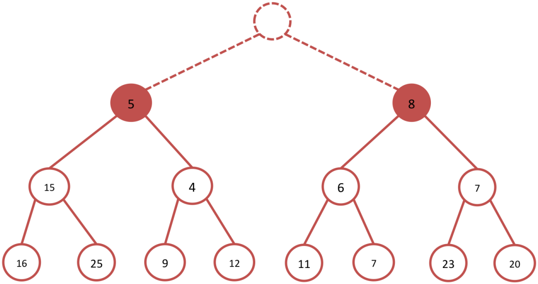


14	5	8	15	4	6	7	16	25	9	12	11	7	23	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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

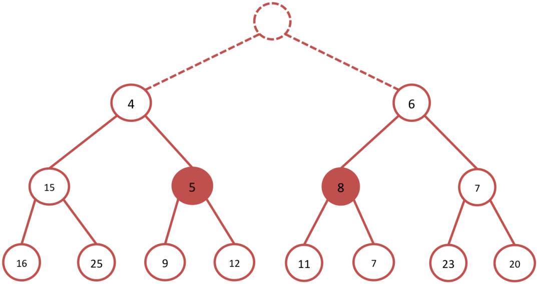


14	5	8	15	4	6	7	16	25	9	12	11	7	23	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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

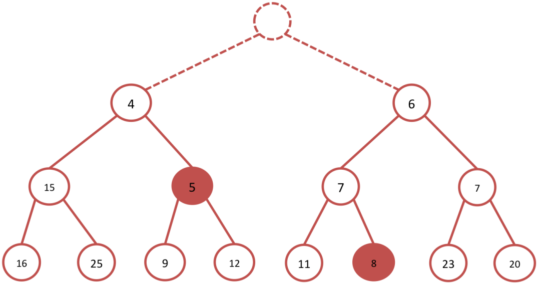


1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	4	6	15	5	8	7	16	25	9	12	11	7	23	20

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

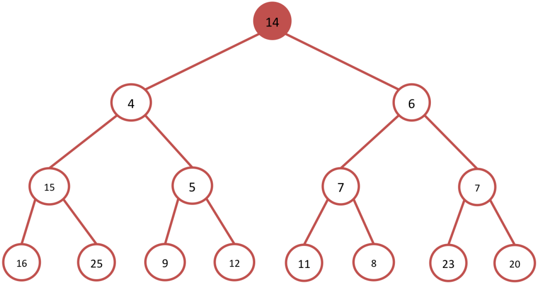


14	4	6	15	5	7	7	16	25	9	12	11	8	23	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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

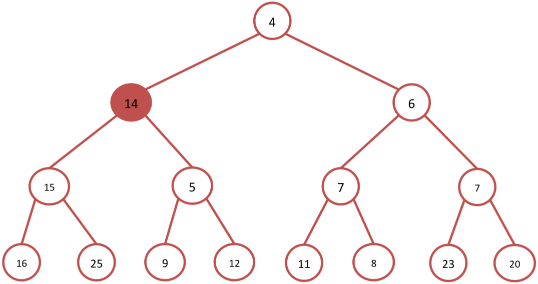


14	4	6	15	5	7	7	16	25	9	12	11	8	23	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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

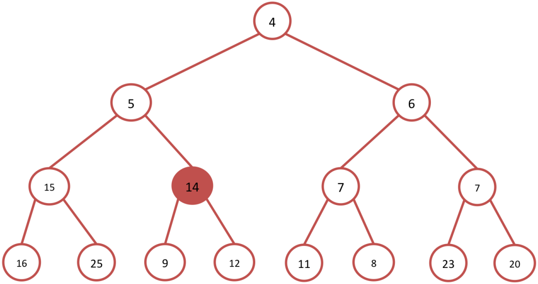


4	14	6	15	5	7	7	16	25	9	12	11	8	23	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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

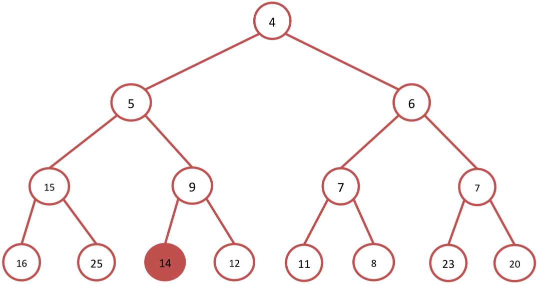


4	5	6	15	14	7	7	16	25	9	12	11	8	23	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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

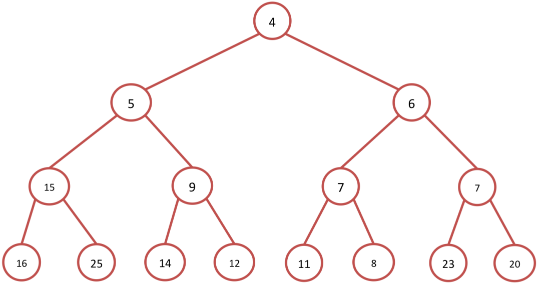


4	5	6	15	9	7	7	16	25	14	12	11	8	23	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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



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

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?

End