CH-231-A

Algorithms and Data Structures

ADS

Lecture 15

Dr. Kinga Lipskoch

Spring 2020

Heap as a Data Structure

- Heaps are a data structure that can be used for other purposes, as well.
- In particular, a max-heap is often used to build a max-priority queue.

Max-Priority Queues

Definition (priority queue):

- A priority queue is a data structure for maintaining a set S of elements, each with an associated value called a key.

Definition (max-priority queue):

- A max-priority queue is a priority queue that supports the following operations:
  - *Maximum*(S): return element from S with largest key.
  - *Extract-Max*(S): remove and return element from S with largest key.
  - *Increase-Key*(S, x, k): increase the value of the key of element x to k, where k is assumed to be larger or equal than the current key.
  - *Insert*(S, x): add element x to set S.

Maximum(S)

```
HEAP-MAXIMUM(A)
1  return A[1]
```

Time complexity: $O(1)$

Extract-Max(S)

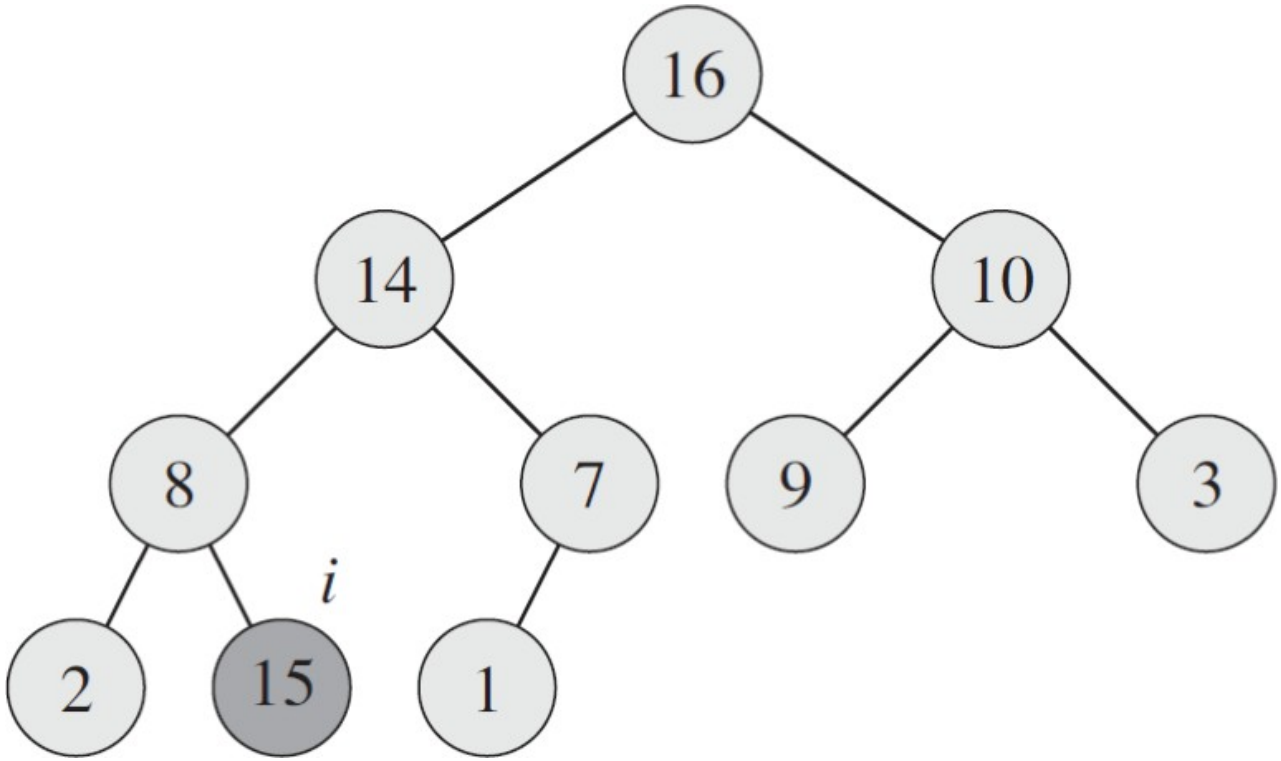
```
HEAP-EXTRACT-MAX(A)
1  if A.heap-size < 1
2        error "heap underflow"
3  max = A[1]
4  A[1] = A[A.heap-size]
5  A.heap-size = A.heap-size − 1
6  MAX-HEAPIFY(A, 1)
7  return max
```

Time complexity: $O(\lg n)$

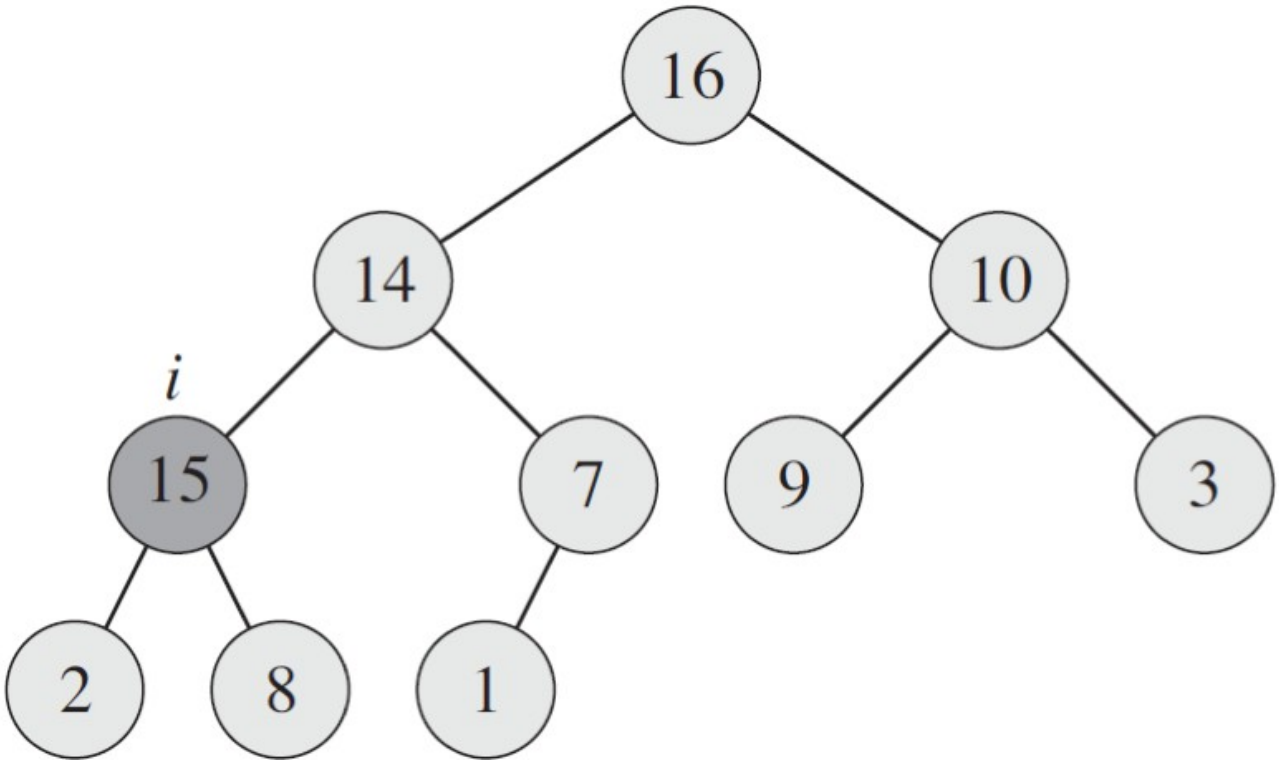
Increase-Key(S, x, k)

```
HEAP-INCREASE-KEY(A, i, key)
1  if key < A[i]
2      error "new key is smaller than current key"
3  A[i] = key
4  while i > 1 and A[PARENT(i)] < A[i]
5      exchange A[i] with A[PARENT(i)]
6      i = PARENT(i)
```

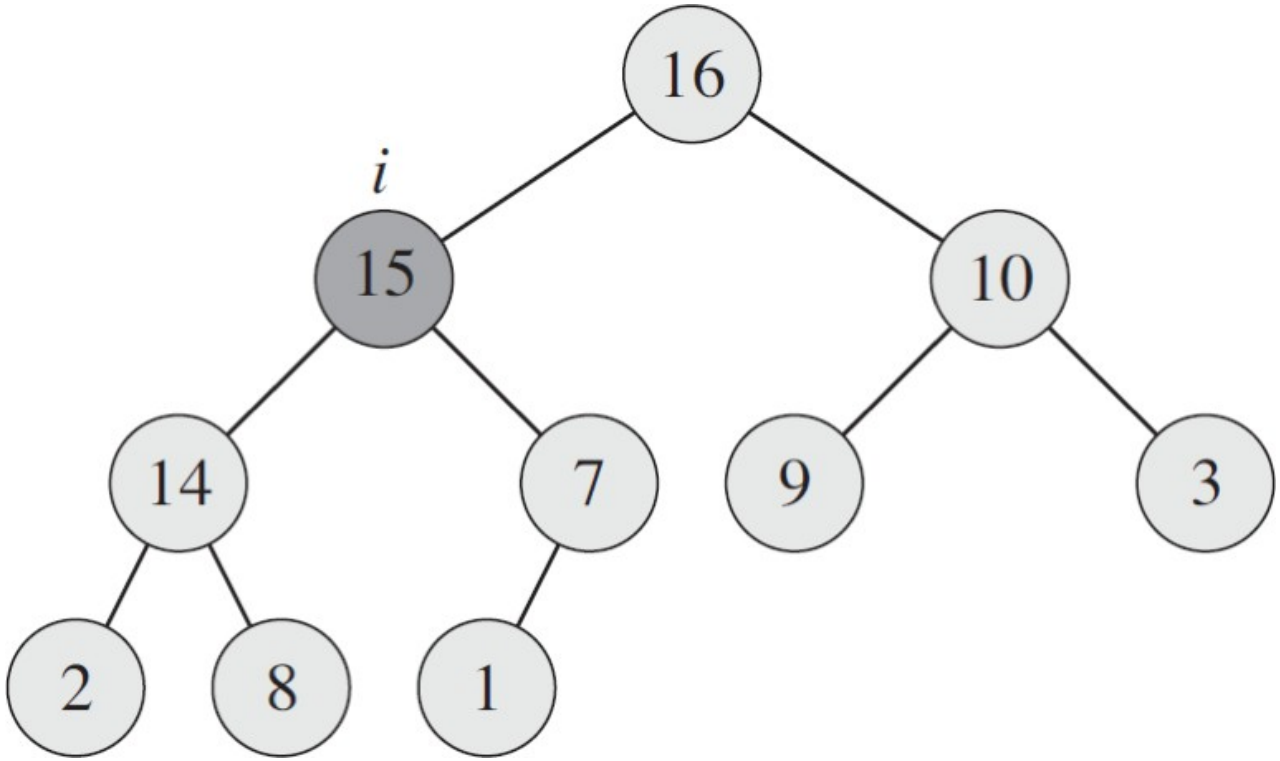
Increase-Key(S, x, k): Example (1)



Increase-Key(S, x, k): Example (2)



Increase-Key(S, x, k): Example (3)



Increase-Key(S, x, k)

```
HEAP-INCREASE-KEY(A, i, key)
1  if key < A[i]
2      error "new key is smaller than current key"
3  A[i] = key
4  while i > 1 and A[PARENT(i)] < A[i]
5      exchange A[i] with A[PARENT(i)]
6      i = PARENT(i)
```

Time complexity: $O(\lg n)$

Insert(S, x)

MAX-HEAP-INSERT(A, key)

1 $A.heap\text{-}size = A.heap\text{-}size + 1$
2 $A[A.heap\text{-}size] = -\infty$
3 HEAP-INCREASE-KEY$(A, A.heap\text{-}size, key)$

Time complexity: $O(\lg n)$