Transform and Conquer

Heaps and Heapsort

- A *heap* is a binary tree with a key at each nodes such that it is essentially complete

- A *complete* tree has all its levels are full except possibly the last level, where only some rightmost keys may be missing

- The key at each node is $\geq$ keys at its children

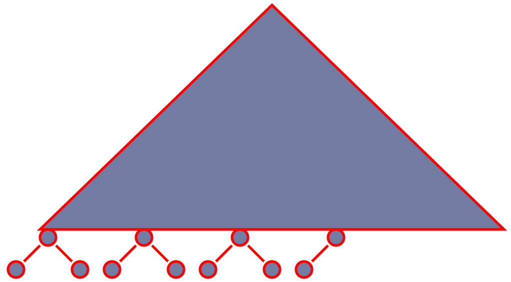


Illustration of the heap's definition

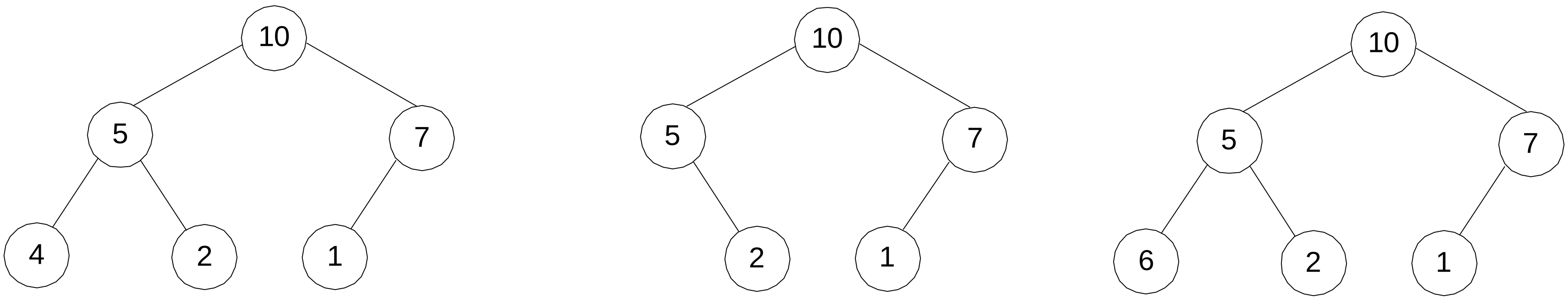
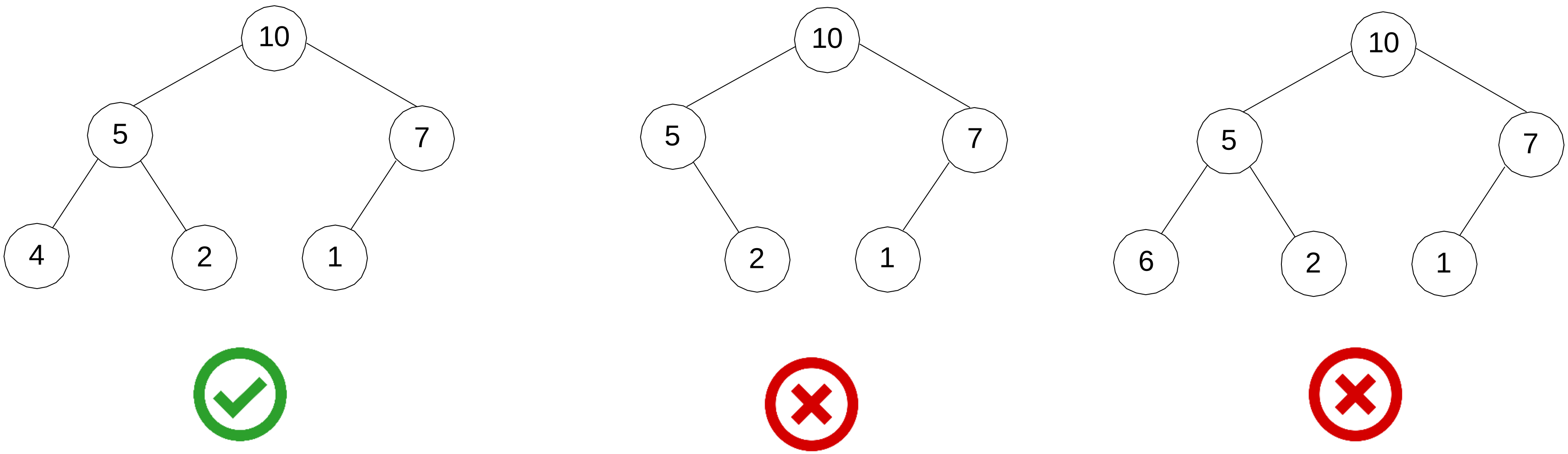


Illustration of the heap's definition



- Heap's elements are ordered top down (along any path down from its root)
- But they are not ordered left to right

Some Important Properties of a Heap

- Given n, there exists a unique binary tree with n nodes that is essentially complete, with $h = \lfloor \log_2 n \rfloor$

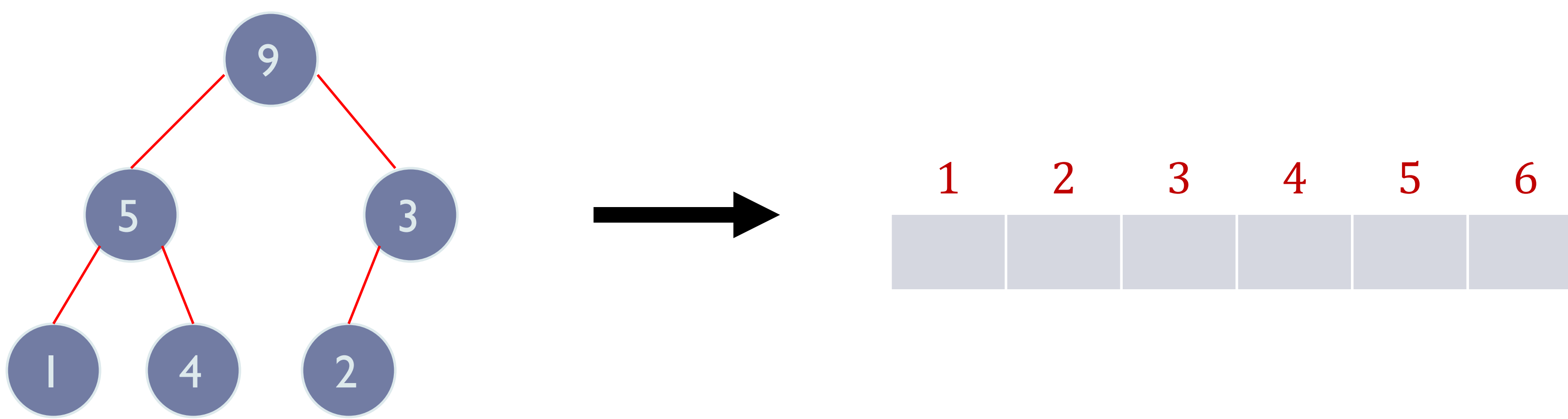
- The root contains the largest key

- The sub-tree rooted at any node of a heap is also a heap

- A heap can be represented as an array

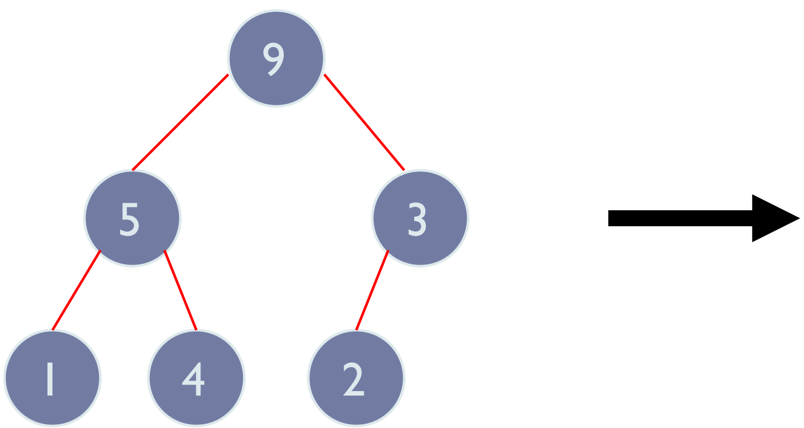
Heap's Array Representation

- Store heap's elements in an array
- Elements indexed, for convenience, 1 to n
- In top-down left-to-right order



Heap’s Array Representation

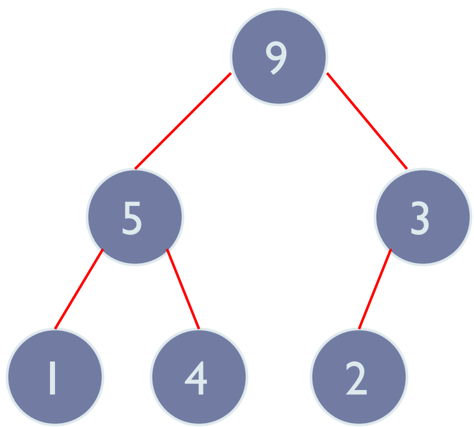
- Store heap’s elements in an array
- Elements indexed, for convenience, 1 to *n*
- In top-down left-to-right order



1	2	3	4	5	6
9	5	3	1	4	2

Heap's Array Representation

- Store heap's elements in an array
- Elements indexed, for convenience, 1 to n
- In top-down left-to-right order



1	2	3	4	5	6
9	5	3	1	4	2

- Left child of node j is at $2j$
- Right child of node j is at $2j + 1$
- Parent of node j is at $\lfloor \frac{j}{2} \rfloor$
- Parental nodes are represented in the first $\lfloor \frac{n}{2} \rfloor$ locations

Heap Construction (bottom-up)

Step 0: Initialize the structure with keys in the order given

Step 1: Starting with the last (rightmost) parental node, fix the heap rooted at it, if it doesn't satisfy the heap condition: keep exchanging it with its largest child until the heap condition holds

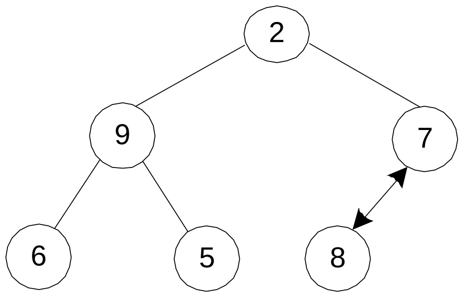
Step 2: Repeat Step 1 for the preceding parental node

Example of Heap Construction

Construct a heap for the list 2, 9, 7, 6, 5, 8

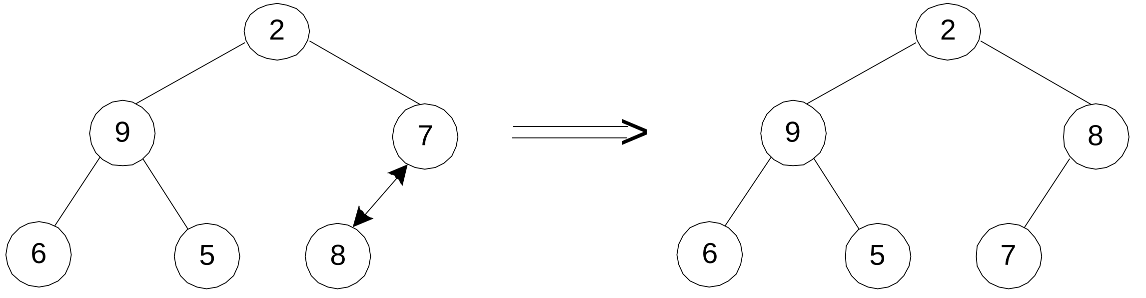
Example of Heap Construction

Construct a heap for the list 2, 9, 7, 6, 5, 8



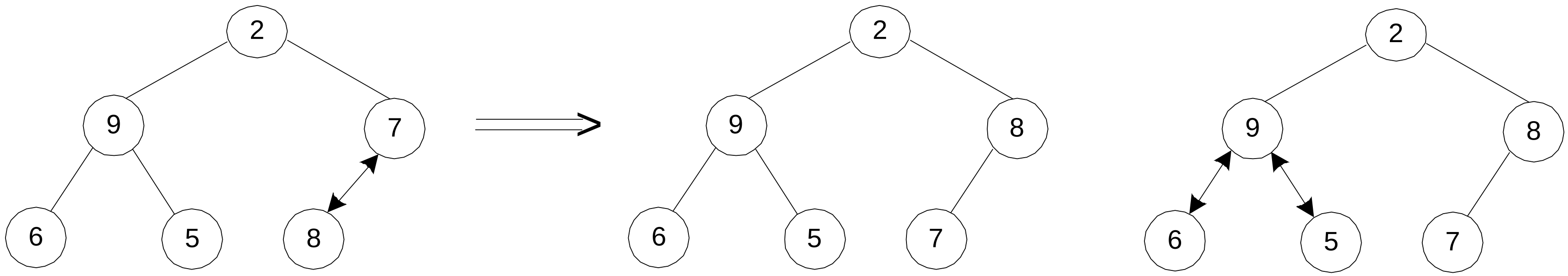
Example of Heap Construction

Construct a heap for the list 2, 9, 7, 6, 5, 8



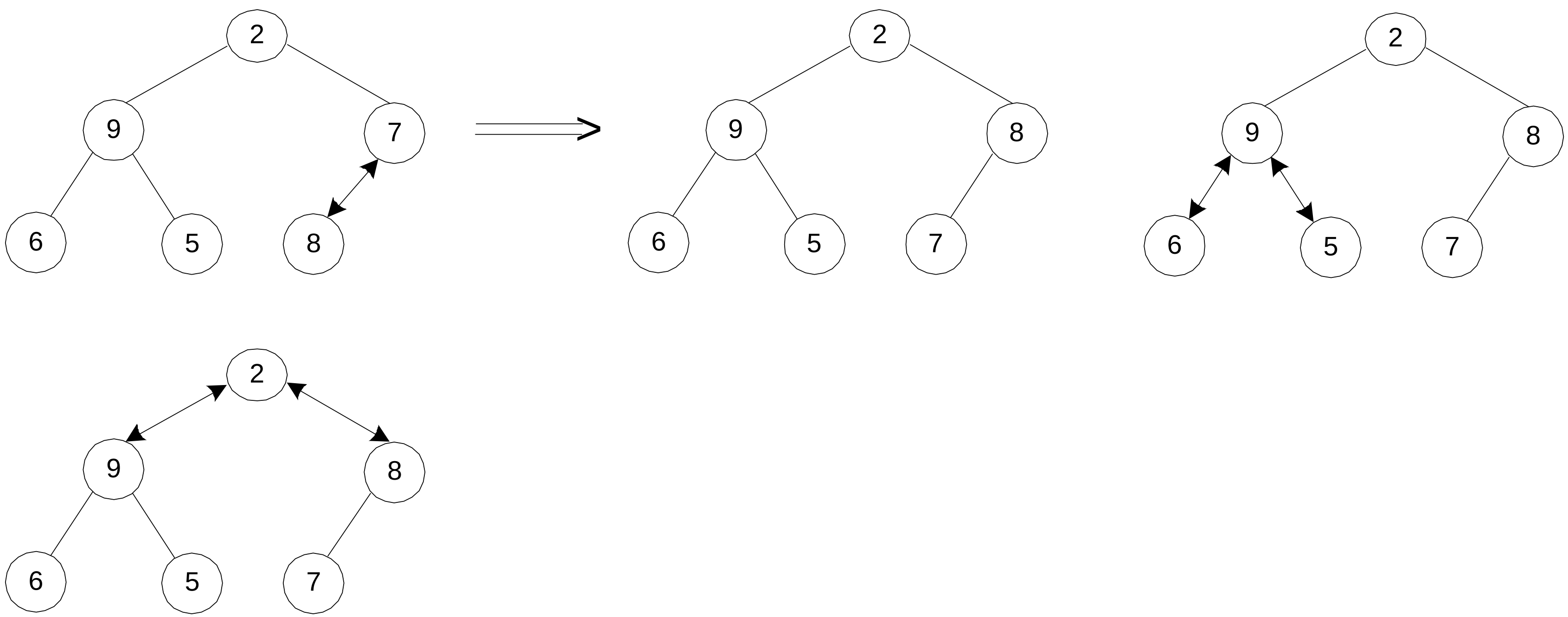
Example of Heap Construction

Construct a heap for the list 2, 9, 7, 6, 5, 8



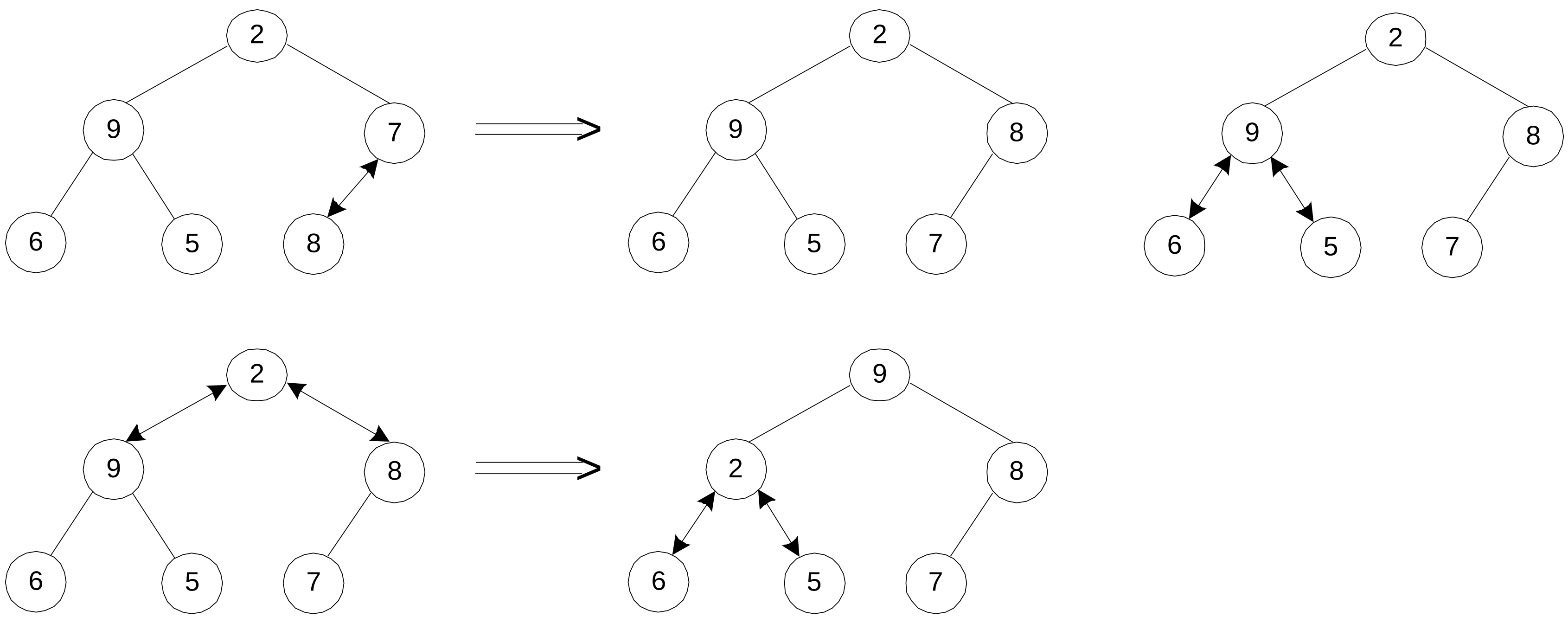
Example of Heap Construction

Construct a heap for the list 2, 9, 7, 6, 5, 8



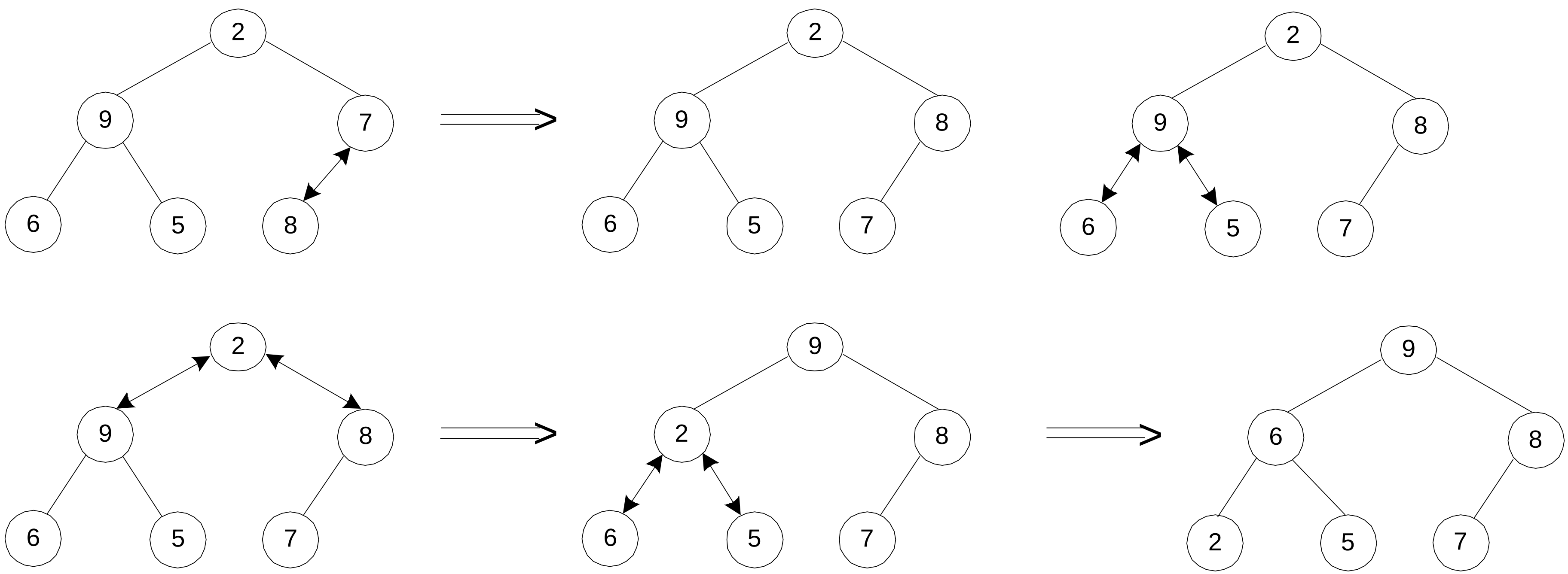
Example of Heap Construction

Construct a heap for the list 2, 9, 7, 6, 5, 8



Example of Heap Construction

Construct a heap for the list 2, 9, 7, 6, 5, 8



Bottom-up heap construction

Algorithm $HeapBottomUp(H[1..n])$
//Constructs a heap from the elements of a given array
// by the bottom-up algorithm
//Input: An array $H[1..n]$ of orderable items
//Output: A heap $H[1..n]$
for $i \leftarrow \lfloor n/2 \rfloor$ **downto** 1 **do**
    $k \leftarrow i$; $v \leftarrow H[k]$
    $heap \leftarrow$ **false**
    while not $heap$ **and** $2 * k \leq n$ **do**
        $j \leftarrow 2 * k$
        if $j < n$ //there are two children
            if $H[j] < H[j+1]$ $j \leftarrow j + 1$
        if $v \geq H[j]$
            $heap \leftarrow$ **true**
        else $H[k] \leftarrow H[j]$; $k \leftarrow j$
    $H[k] \leftarrow v$

Heapsort

Stage 1: Construct a heap for a given list of n keys

Stage 2: Repeat operation of root removal n-1 times:

- Exchange keys in the root and in the last (rightmost) leaf
- Decrease heap size by 1
- If necessary, swap new root with larger child until the heap condition holds

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)

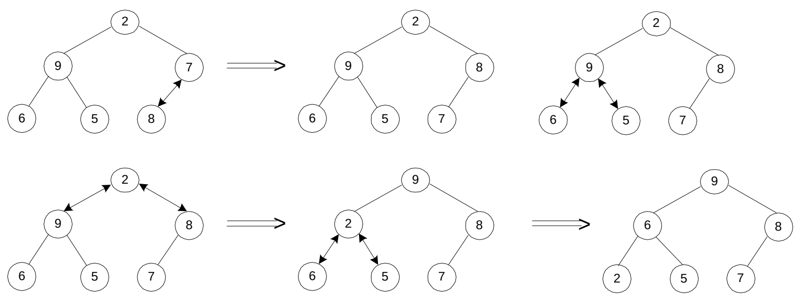
Stage 2 (root/max removal)

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)

Stage 2 (root/max removal)

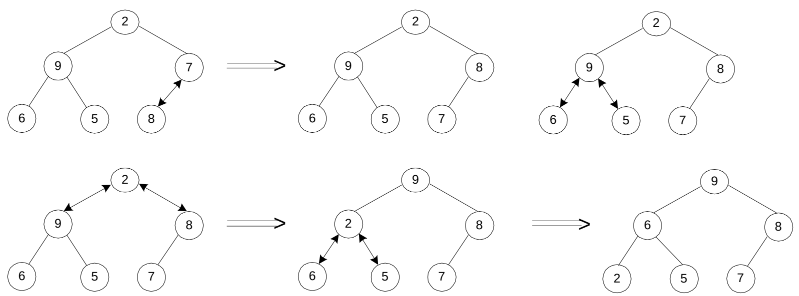


1	2	3	4	5	6
9	6	8	2	5	7

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

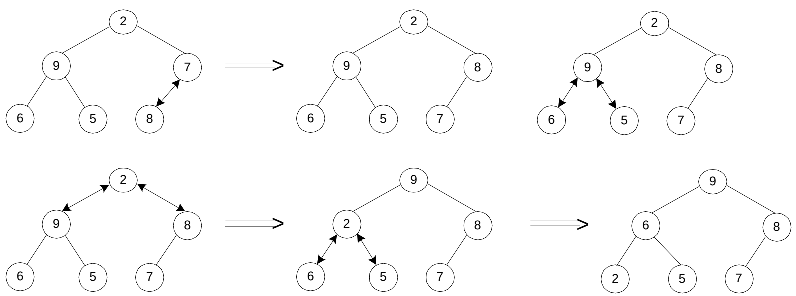
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

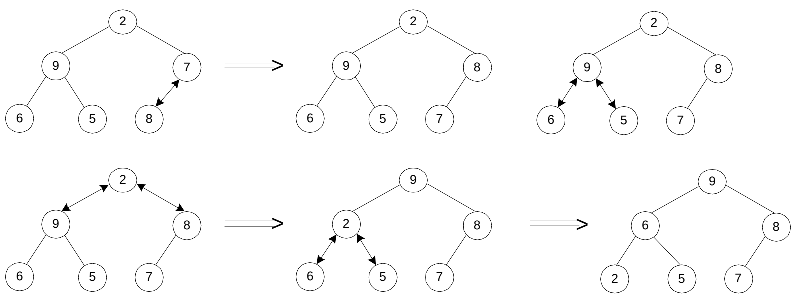
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
7	6	8	2	5	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

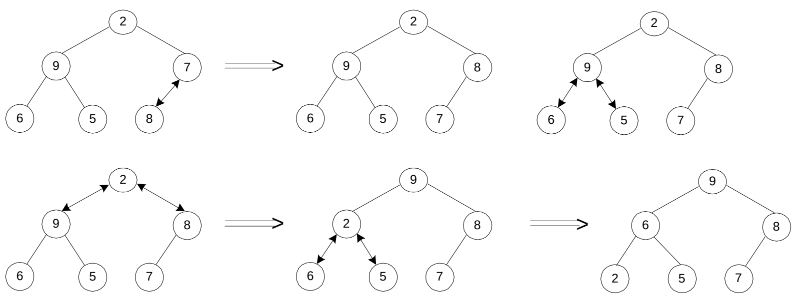
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
7	6	8	2	5	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

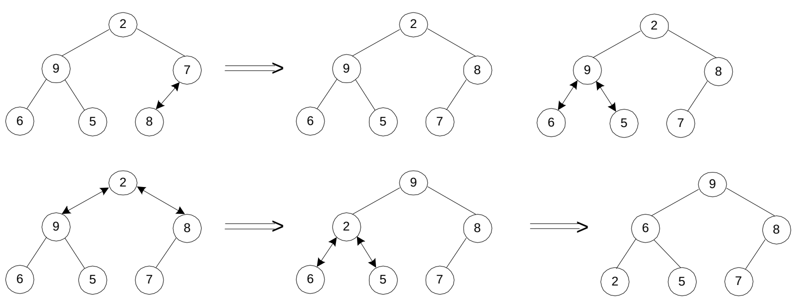
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
7	6	8	2	5	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

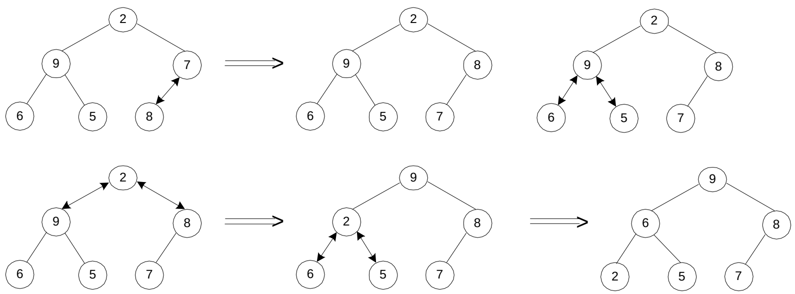
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
7	6	8	2	5	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

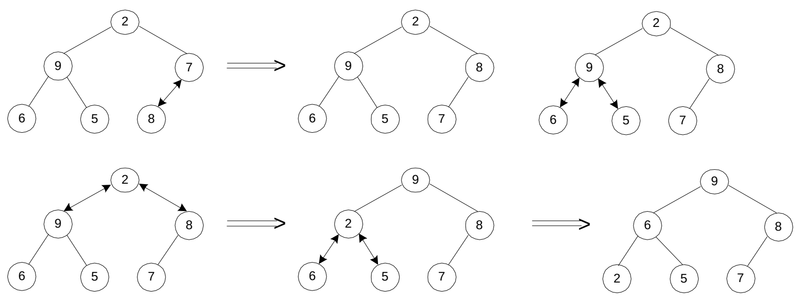
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
8	6	7	2	5	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

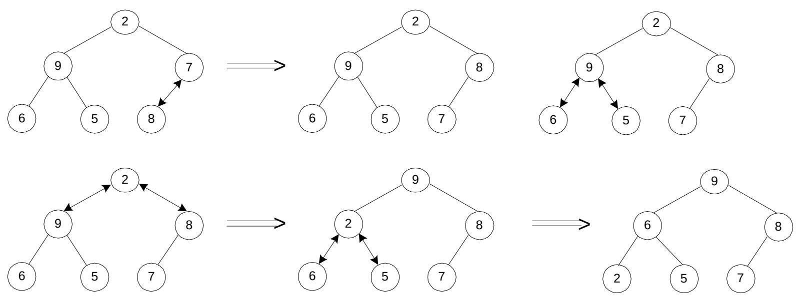
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
8	6	7	2	5	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

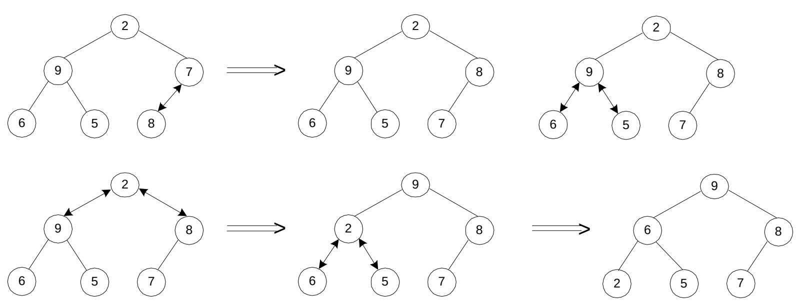
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
8	6	7	2	5	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

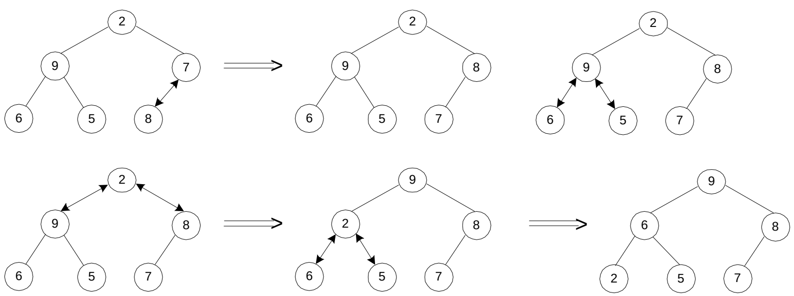
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
5	6	7	2	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

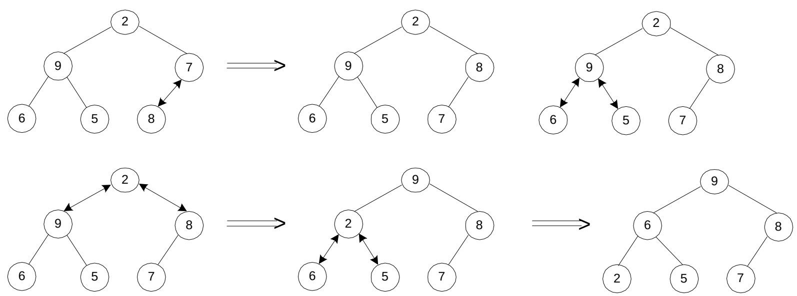
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
5	6	7	2	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

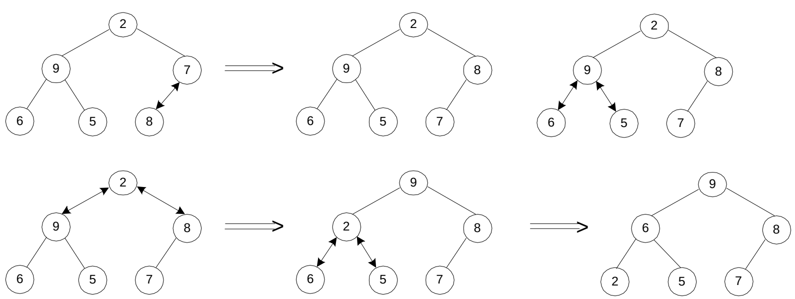
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
5	6	7	2	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

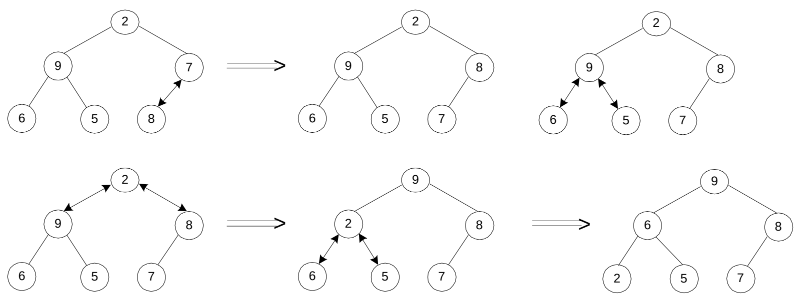
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
5	6	7	2	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

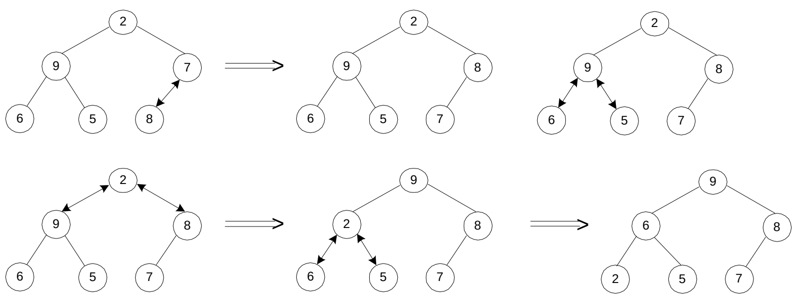
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
7	6	5	2	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

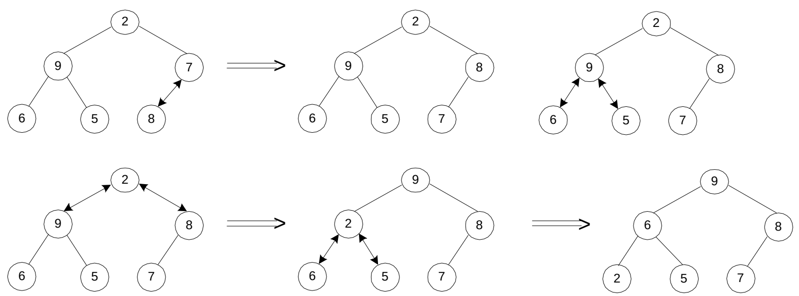
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
7	6	5	2	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

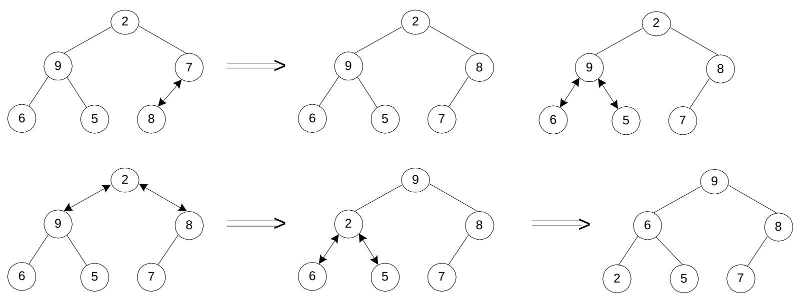
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
7	6	5	2	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

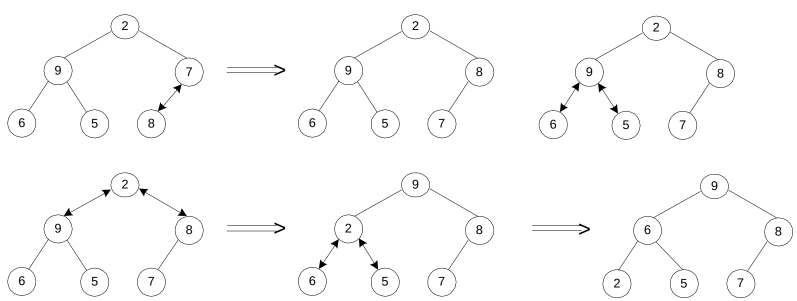
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
2	6	5	7	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

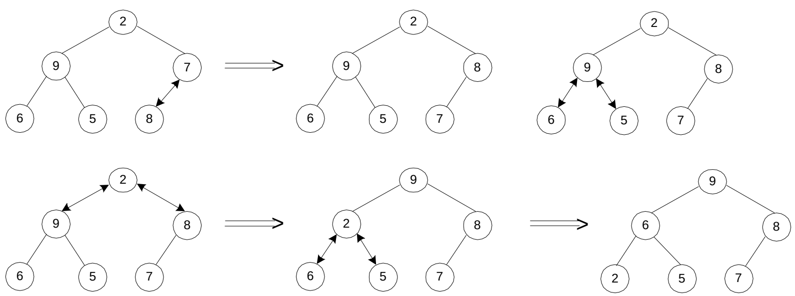
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
2	6	5	7	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

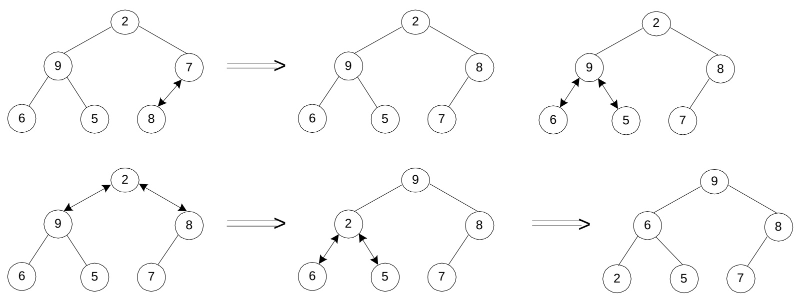
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
2	6	5	7	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

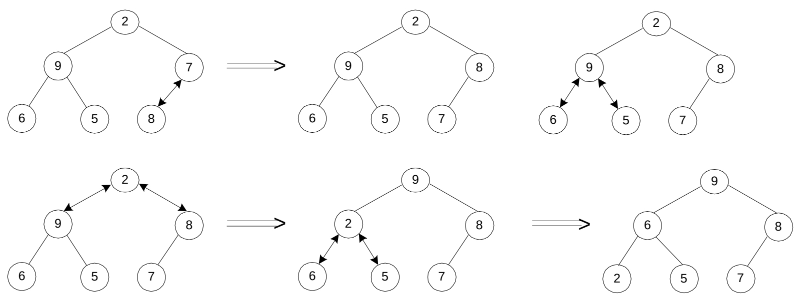
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
2	6	5	7	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

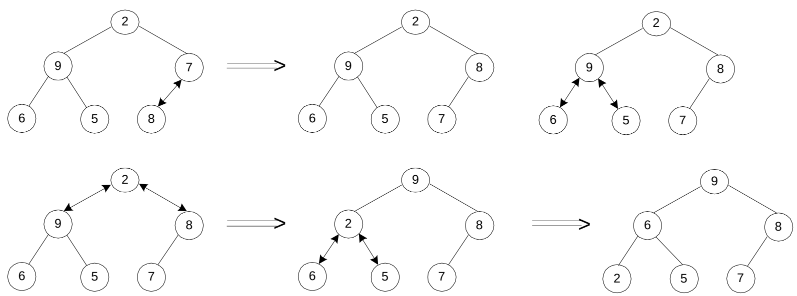
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
6	2	5	7	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

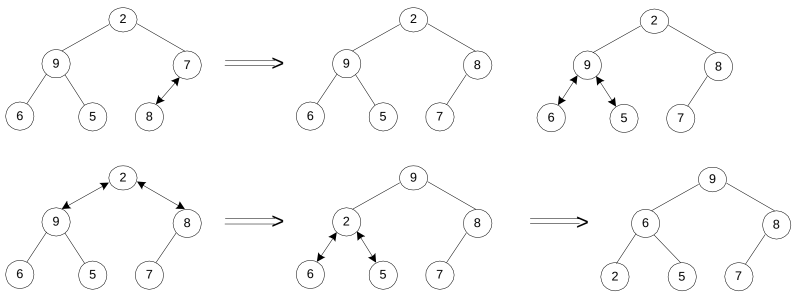
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
6	2	5	7	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

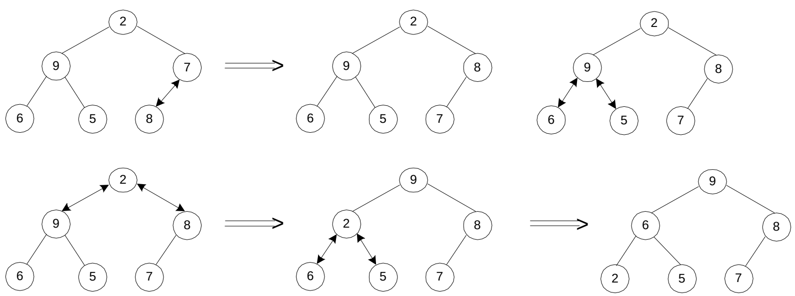
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
6	2	5	7	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

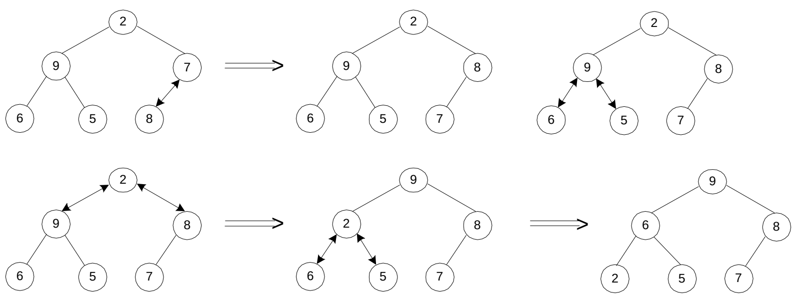
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
5	2	6	7	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

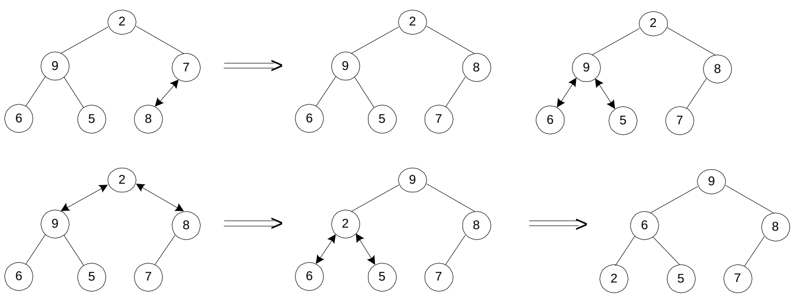
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
5	2	6	7	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

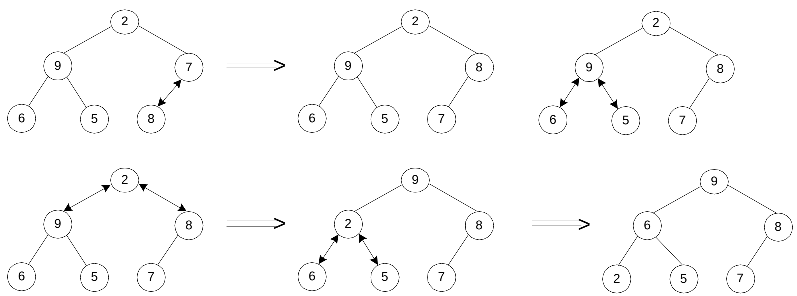
Stage 2 (root/max removal)

1	2	3	4	5	6
9	6	8	2	5	7
2	5	6	7	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)



1	2	3	4	5	6
9	6	8	2	5	7

Stage 2 (root/max removal)

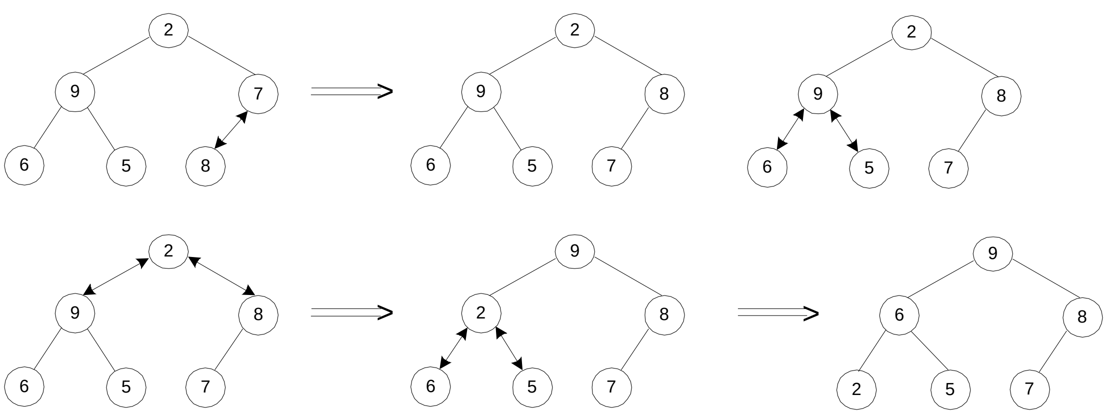
1	2	3	4	5	6
9	6	8	2	5	7
2	5	6	7	8	9

Example of Sorting by Heapsort

Sort the list 2, 9, 7, 6, 5, 8 by heapsort

Stage 1 (heap construction)

2 9 7 6 5 8
2 9 8 6 5 7
2 9 8 6 5 7
9 2 8 6 5 7
9 6 8 2 5 7



Stage 2 (root/max removal)

9 6 8 2 5 7
7 6 8 2 5 | 9
8 6 7 2 5 | 9
5 6 7 2 | 8 9
7 6 5 2 | 8 9
2 6 5 | 7 8 9
6 2 5 | 7 8 9
5 2 | 6 7 8 9
5 2 | 6 7 8 9
2 | 5 6 7 8 9

Analysis of Heapsort

Stage 1: Build heap for a given list of n keys

worst-case

$$C(n) = \sum_{i=0}^{h-1} 2^i \times 2(h-i) = 2\big(n - log\,(n+1)\big) \in \Theta(\mathrm{n})$$

\# nodes at level *i*

Stage 2: Repeat operation of root removal n-1 times (fix heap)

worst-case

$$C(n) = \sum_{i=1}^{n-1} 2\log i \in \Theta(n \log n)$$

Both worst-case and average-case efficiency: $\theta(n\log n)$

In-place: yes

Priority Queue

- A *priority queue* is a set of elements with numerical priorities:
  - find element with highest priority
  - delete element with highest priority
  - insert element with assigned priority (see below)
- Heap is a very efficient way for implementing priority queues
- Two ways to handle priority queue in which

  highest priority = smallest number

  highest priority = largest number

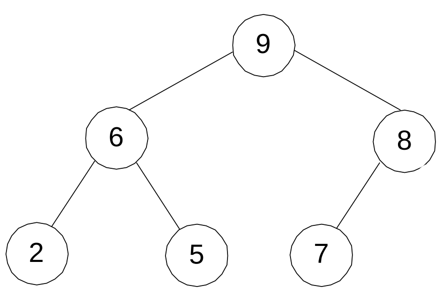
Insertion of a New Element into a Heap

- Insert the new element at last position in heap.
- Compare it with its parent and, if it violates heap condition, exchange them
- Continue comparing the new element with nodes up the tree until the heap condition is satisfied

Insertion of a New Element into a Heap

- Insert the new element at last position in heap.
- Compare it with its parent and, if it violates heap condition, exchange them
- Continue comparing the new element with nodes up the tree until the heap condition is satisfied

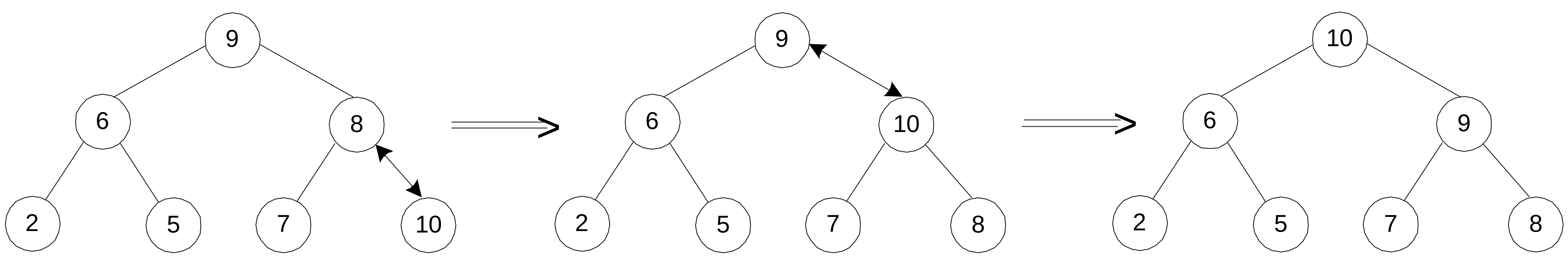
Example: Insert key 10



Insertion of a New Element into a Heap

- Insert the new element at last position in heap.
- Compare it with its parent and, if it violates heap condition, exchange them
- Continue comparing the new element with nodes up the tree until the heap condition is satisfied

Example: Insert key 10



Efficiency: O(log *n*)

Discussion

6. Sort the following lists by heapsort by using the array representation of heaps.

   a. 1, 2, 3, 4, 5 (in increasing order)

   b. 5, 4, 3, 2, 1 (in increasing order)

   c. S, O, R, T, I, N, G (in alphabetical order)

Discussion

6. a. Sort 1, 2, 3, 4, 5 by heapsort

Heap Construction

1	**2**	3	4	5
1	5	3	4	2
1	5	3	4	2
5	4	3	1	2

Maximum Deletions

5	4	3	1	2
2	4	3	1	$\vert$ **5**
4	2	3	1	
1	2	3	$\vert$ **4**	
3	2	1		
1	2	$\vert$ **3**		
2	1			
1	$\vert$ **2**			
1				

Discussion

b. Sort 5, 4, 3, 2, 1 (in increasing order) by heapsort

Heap Construction

5	**4**	3	2	1
5	4	3	2	1

Maximum Deletions

5	4	3	2		1
1	4	3	2	|	**5**
4	2	3	1		
1	2	3 |	**4**		
3	2	1			
1	2 |	**3**			
2	1				
1 |	**2**				
1					

Discussion

c. Sort S, O, R, T, I, N, G (in alphabetic order) by heapsort

Heap Construction

1	2	3	4	5	6	7
S	O	**R**	T	I	N	G
S	**O**	R	T	I	N	G
S	T	R	O	I	N	G
S	T	R	O	I	N	G
T	S	R	O	I	N	G

Maximum Deletions

1	2	3	4	5	6	7
T	S	R	O	I	N	G
G	S	R	O	I	N	**T**
S	O	R	G	I	N	
N	O	R	G	I	**S**	
R	O	N	G	I		
I	O	N	G	**R**		
O	I	N	G			
G	I	N	**O**			
N	I	G				
G	I	**N**				
I	G					
G	**I**					
G						

Discussion

Spaghetti sort Imagine a handful of uncooked spaghetti, individual rods whose lengths represent numbers that need to be sorted.

a. Outline a "spaghetti sort"—a sorting algorithm that takes advantage of this unorthodox representation.

b. What does this example of computer science folklore (see [Dew93]) have to do with the topic of this chapter in general and heapsort in particular?

Discussion

11. a. After the bunch of spaghetti rods is put in a vertical position on a tabletop, repeatedly take the tallest rod among the remaining ones out until no more rods are left. This will sort the rods in decreasing order of their lengths.

    b. The method shares with heapsort its principal idea: represent the items to be sorted in a way that makes finding and deleting the largest item a simple task. From a more general perspective, the spaghetti sort is an example, albeit a rather exotic one, of a representation-change algorithm.

Programming Exercise

- Implement the heapsort algorithm
- Develop test cases to test your code
- Compare Heapsort with other sorting algorithms