File Edit View Help

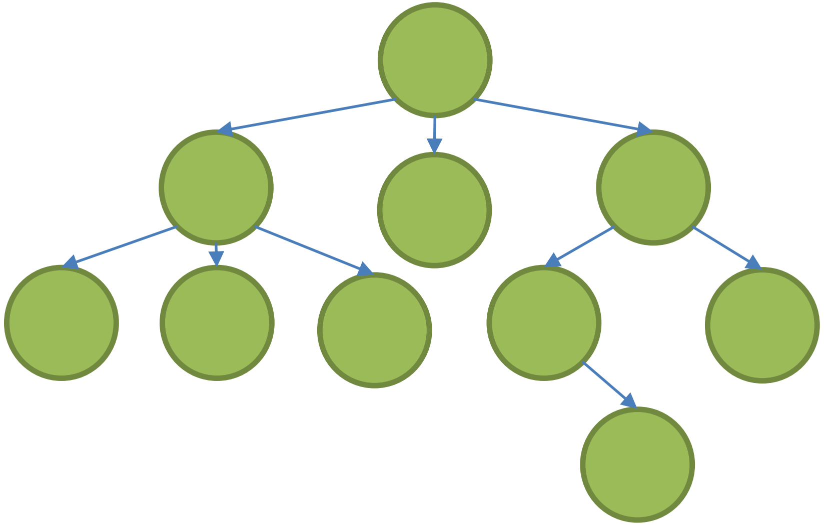
File Edit View Help

Heaps Part 02

Trees

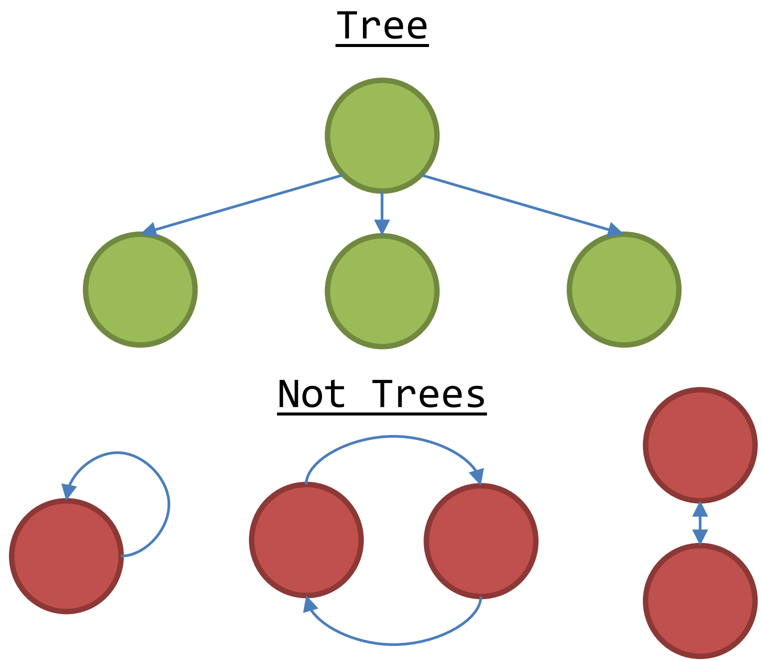
- Definition: A data structure that can be defined recursively as a collection of nodes, where each node is a data structure consisting of a value, together with a list of references (edges) to nodes, with the constraints that no reference is duplicated, and none points to the root.

Trees



Trees

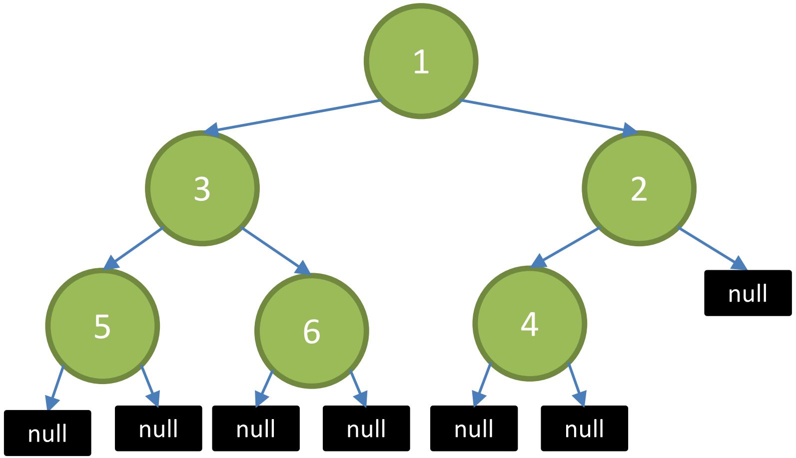
- Trees Have
  - Nodes
  - Edges
- Trees CANNOT
  - Contain Self-Referencing Edges
  - Have Cycles
  - Be Disjointed



Heaps

- Binary Tree Structure
- Node's data must be comparable
- Node's have at most two children
  - Left Child
  - Right Child
- Max Heap: Children must be less than or equal to the parent
- Min Heap: Children must be greater than or equal to the parent
- Assume Leaves are NULL references

Min Heap



File Edit View Help

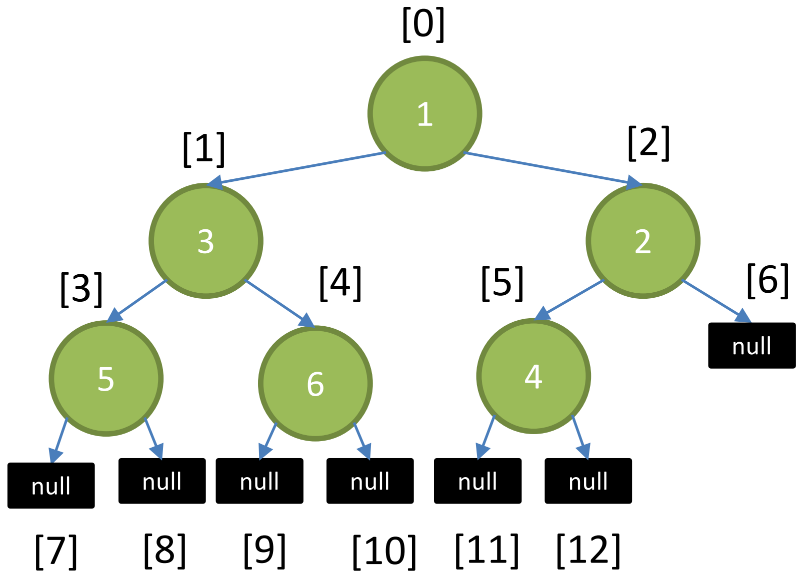
Heaps

- Array Heap
- Assume Root is at Index 0
- Left Child Index = Parent Index * 2 + 1
- Right Child Index = Parent Index * 2 + 2
- Parent Index = (Child Index-1)/2

Array Max Heap

0	1	2	3	4	5	6	7	8	9	10	11	12
1	3	2	5	6	4	-	-	-	-	-	-	-

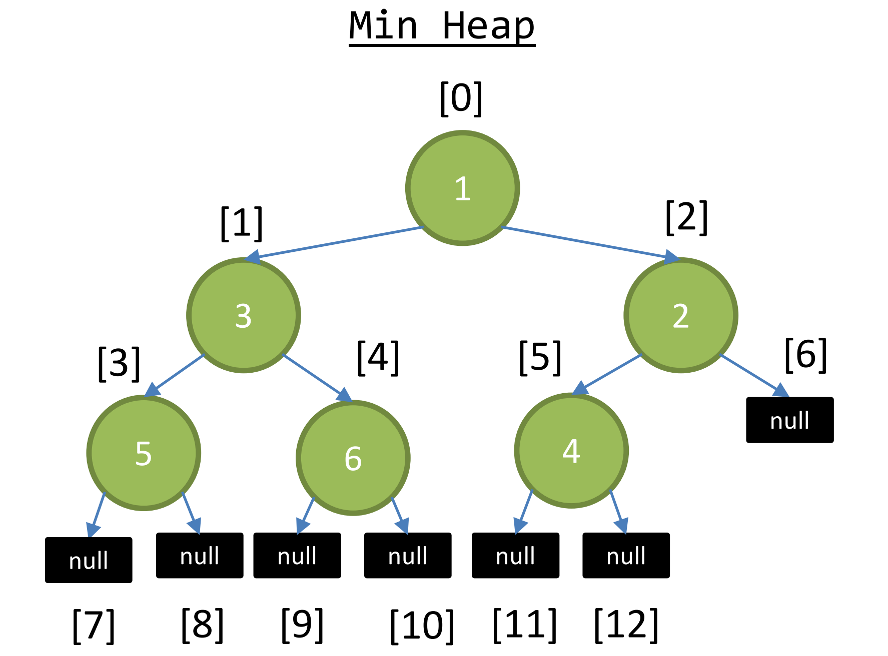
Min Heap



File Edit View Help

Heaps

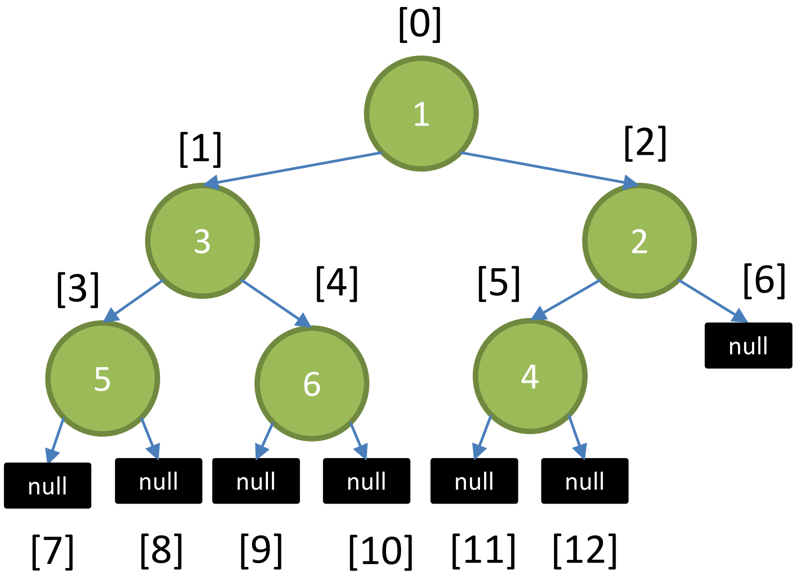
- Add
  - Replace the first leaf in breadth order with the new data
  - From that node "bubble up" the data if necessary
- Bubble Up
  - If the child's data is smaller than the parent then swap that information
  - Continue swapping child data with parent data until the parent is smaller than the child or we reach the root index



Heaps

- Add
  - Replace the first leaf in breadth order with the new data
  - From that node "bubble up" the data if necessary
- Bubble Up
  - If the child's data is smaller than the parent then swap that information
  - Continue swapping child data with parent data until the parent is smaller than the child or we reach the root index

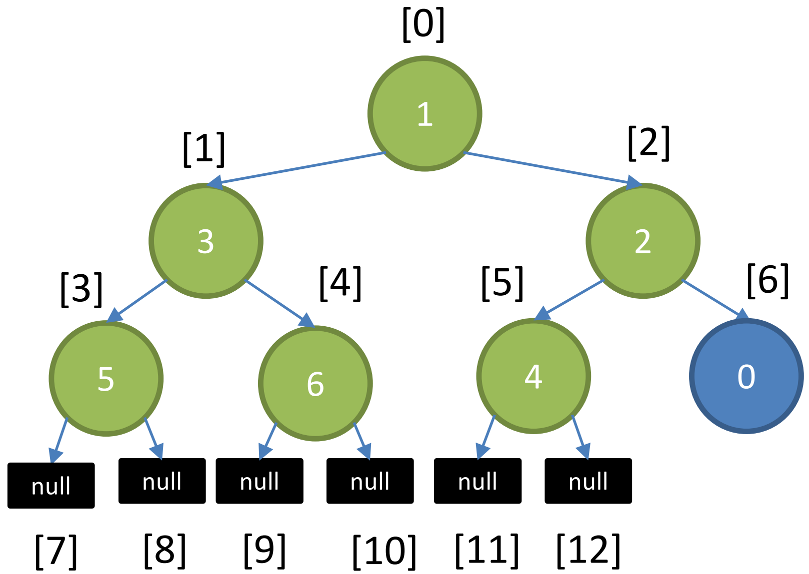
Example Adding "0"



Heaps

- Add
  - Replace the first leaf in breadth order with the new data
  - From that node "bubble up" the data if necessary
- Bubble Up
  - If the child's data is smaller than the parent then swap that information
  - Continue swapping child data with parent data until the parent is smaller than the child or we reach the root index

<u>Example Adding "0"</u>

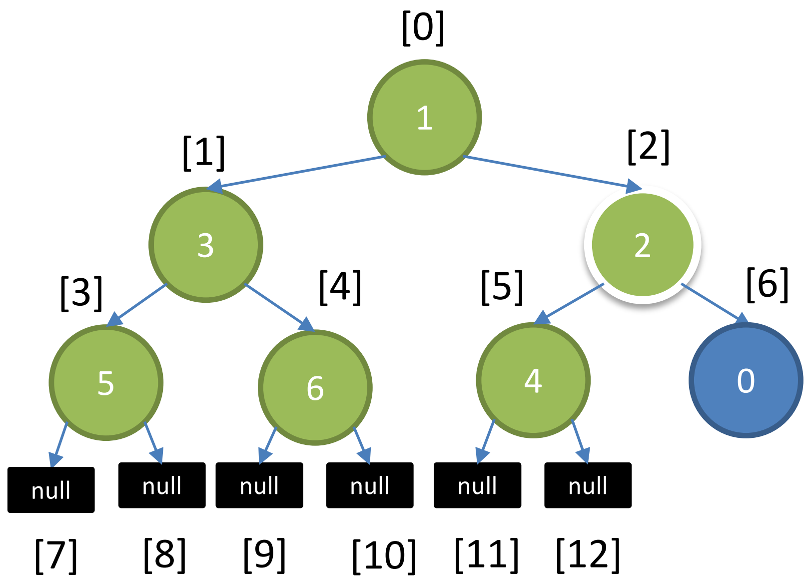


File Edit View Help

Heaps

- Add
  - Replace the first leaf in breadth order with the new data
  - From that node “bubble up” the data if necessary
- Bubble Up
  - If the child’s data is smaller than the parent then swap that information
  - Continue swapping child data with parent data until the parent is smaller than the child or we reach the root index

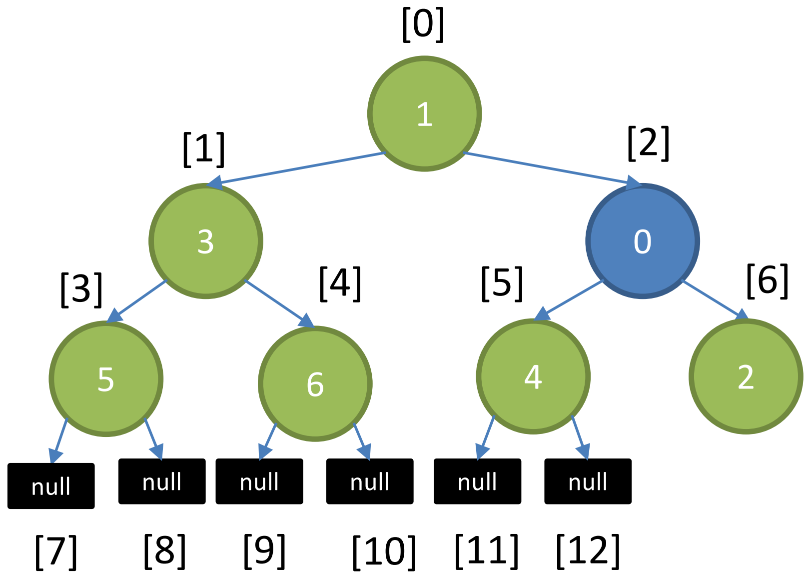
Example Adding “0”



Heaps

- Add
  - Replace the first leaf in breadth order with the new data
  - From that node “bubble up” the data if necessary
- Bubble Up
  - If the child’s data is smaller than the parent then swap that information
  - Continue swapping child data with parent data until the parent is smaller than the child or we reach the root index

Example Adding “0”

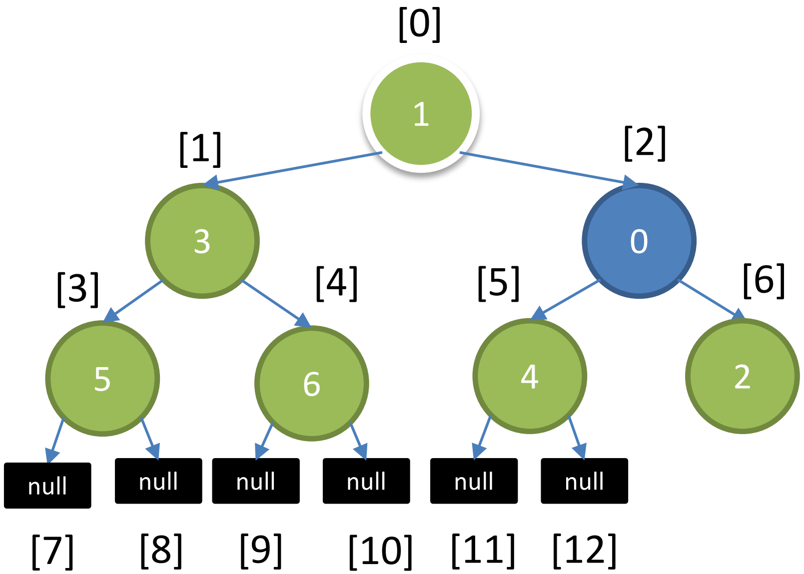


File Edit View Help

Heaps

- Add
  - Replace the first leaf in breadth order with the new data
  - From that node "bubble up" the data if necessary
- Bubble Up
  - If the child's data is smaller than the parent then swap that information
  - Continue swapping child data with parent data until the parent is smaller than the child or we reach the root index

Example Adding "0"

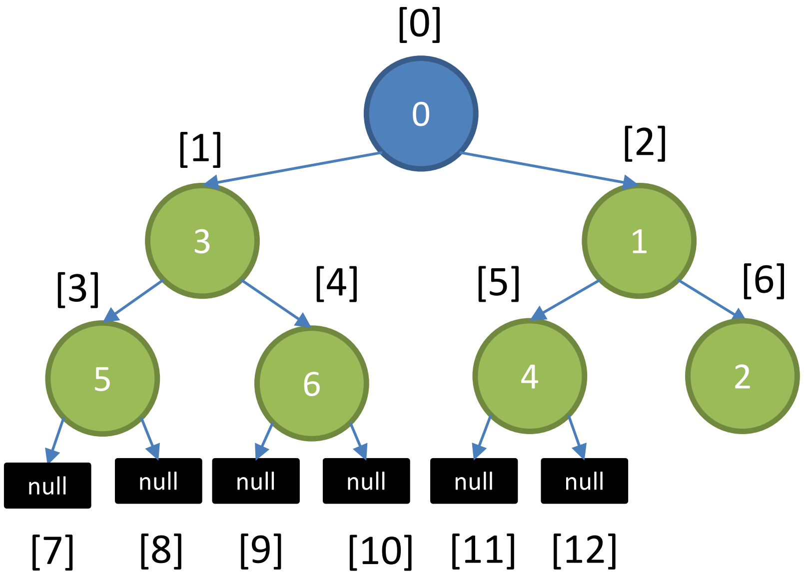


File Edit View Help

Heaps

- Add
  - Replace the first leaf in breadth order with the new data
  - From that node “bubble up” the data if necessary
- Bubble Up
  - If the child’s data is smaller than the parent then swap that information
  - Continue swapping child data with parent data until the parent is smaller than the child or we reach the root index

Example Adding “0”

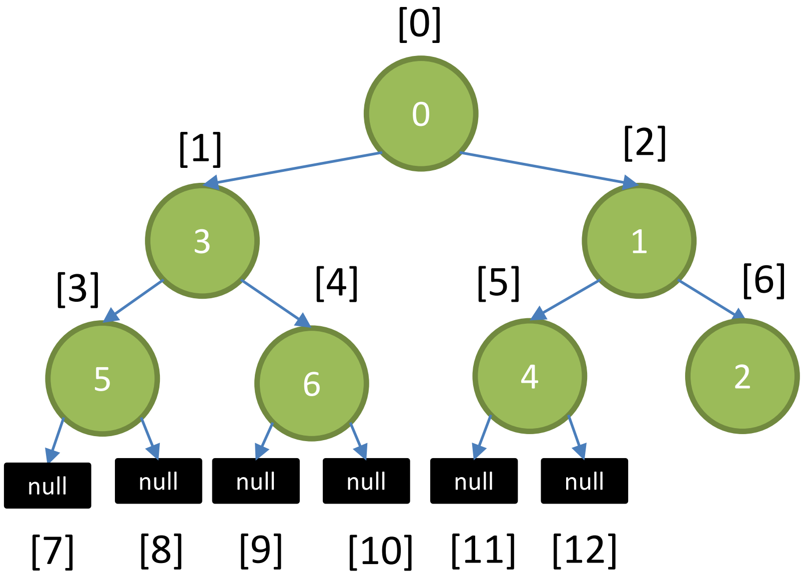


File Edit View Help

Heaps

- Add
  - Replace the first leaf in breadth order with the new data
  - From that node "bubble up" the data if necessary
- Bubble Up
  - If the child's data is smaller than the parent then swap that information
  - Continue swapping child data with parent data until the parent is smaller than the child or we reach the root index

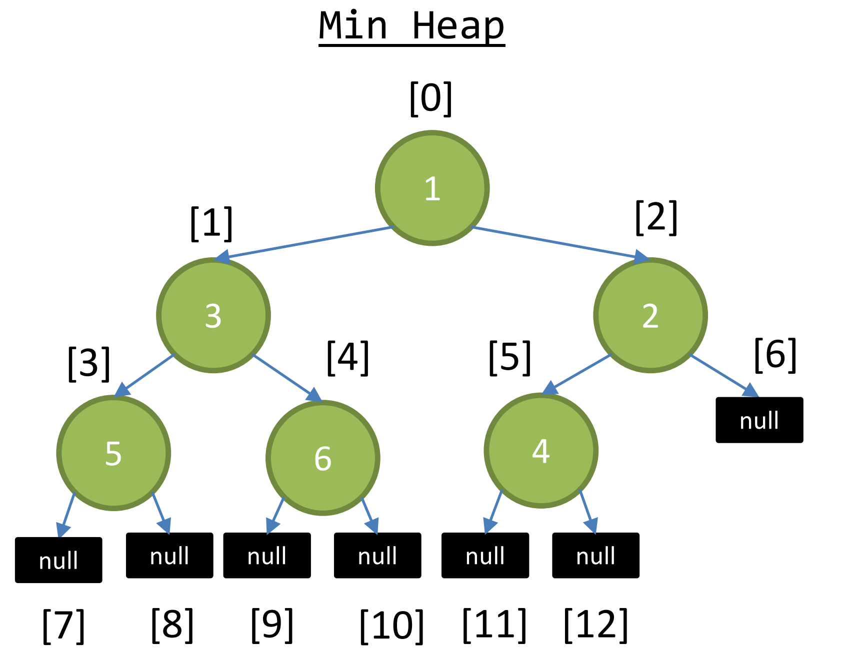
Example Adding "0"



File Edit View Help

Heaps

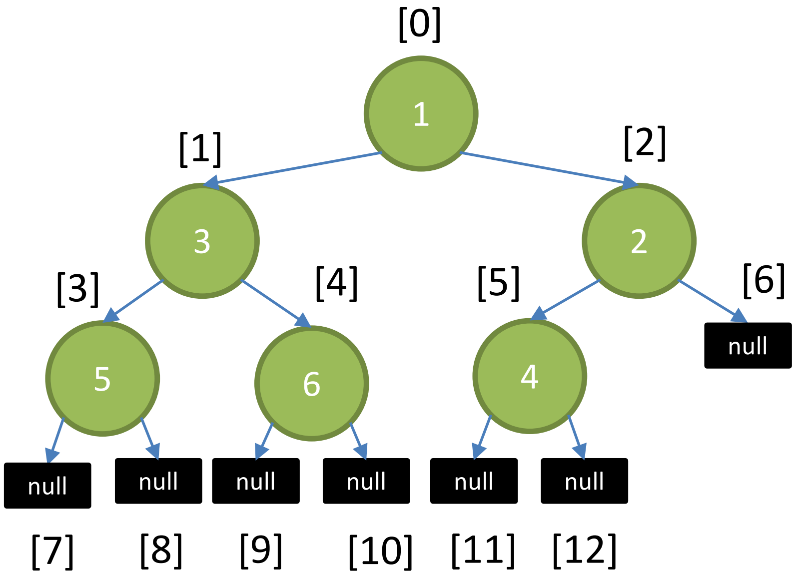
- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds



Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

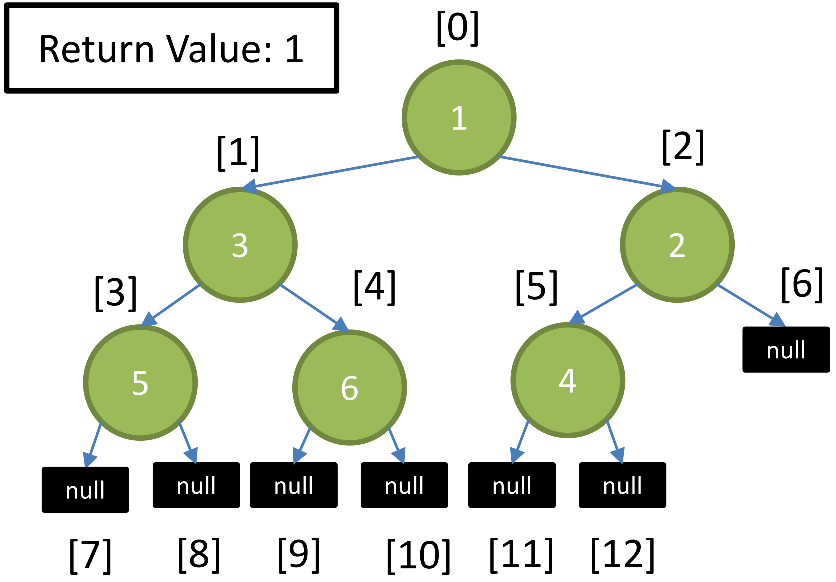
Example Remove



Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

Example Remove



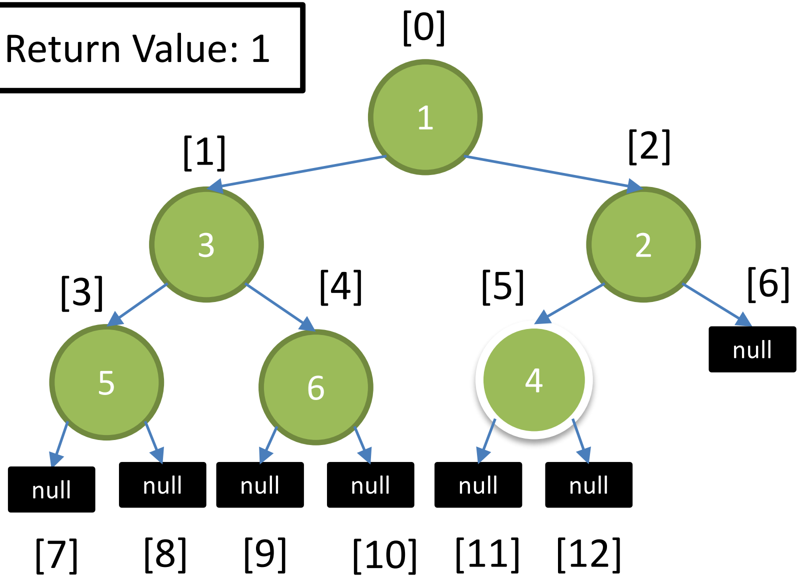
File Edit View Help

Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

Example Remove

Return Value: 1

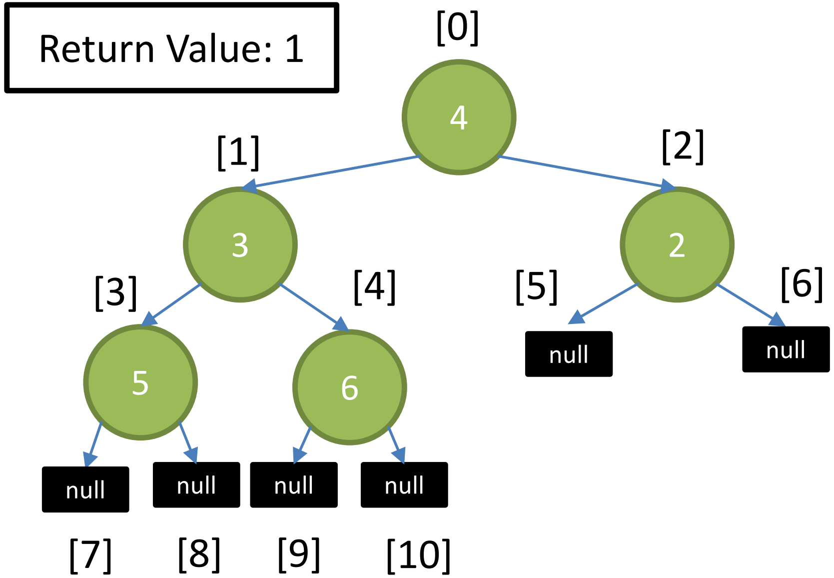


File Edit View Help

Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

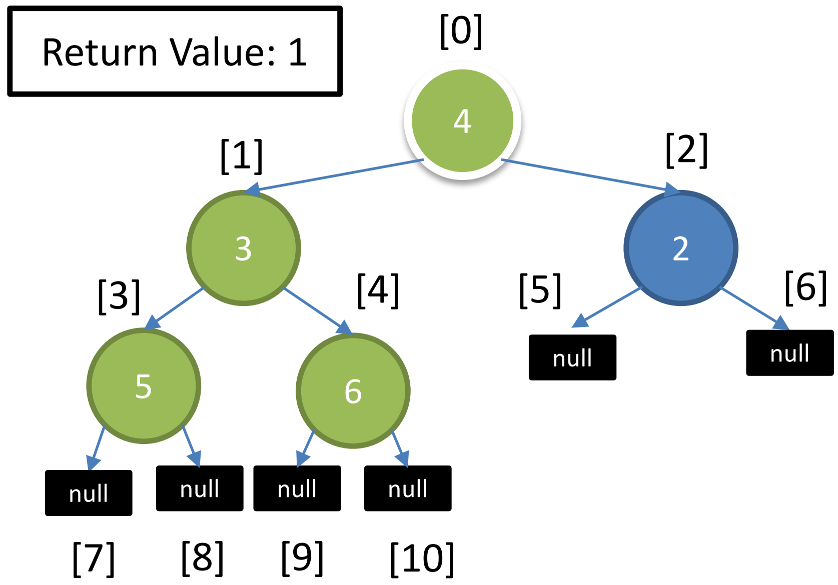
Example Remove



Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

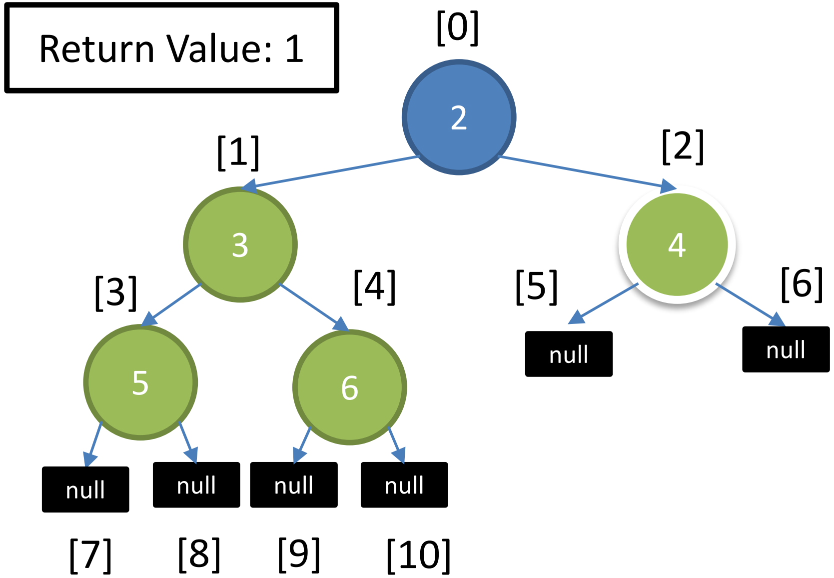
<u>Example Remove</u>



Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

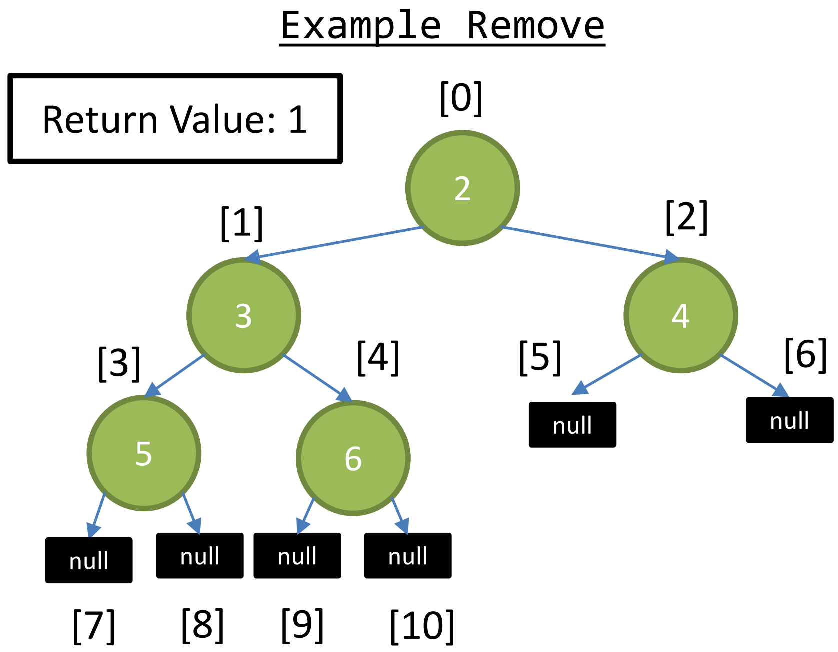
Example Remove



File Edit View Help

Heaps

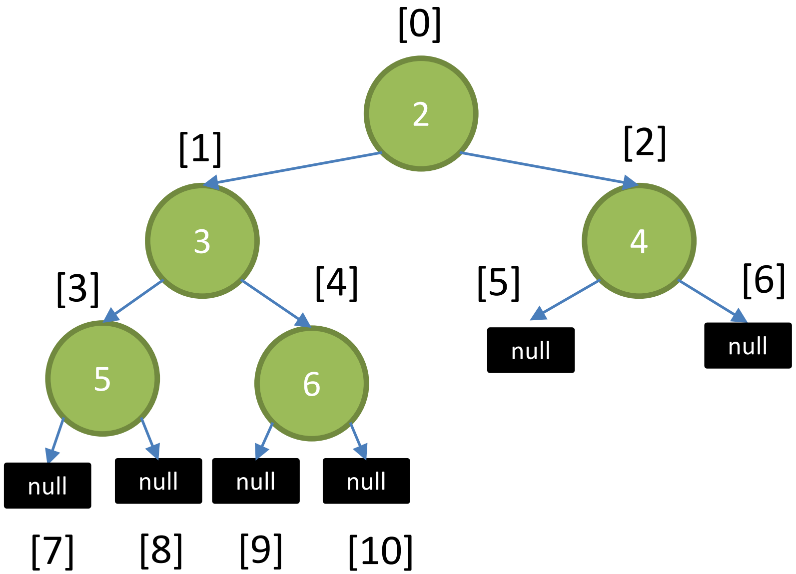
- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds



Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

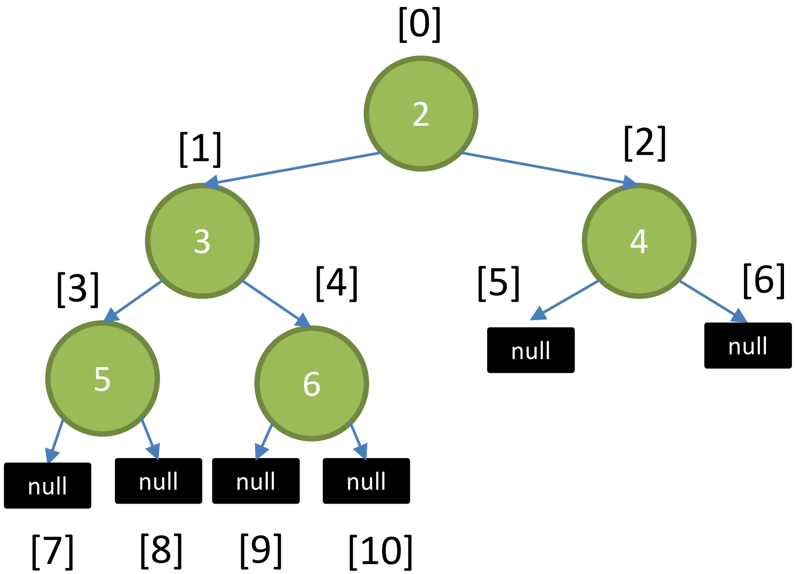
Example Remove



Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

Example Remove Again

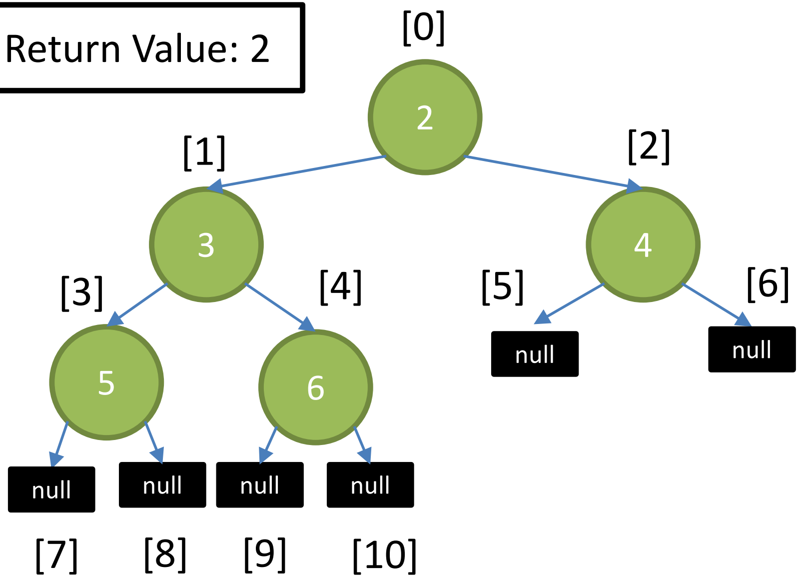


Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, "Bubble Down" that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent's value is smaller or we reach the tree's bounds

Example Remove Again

Return Value: 2

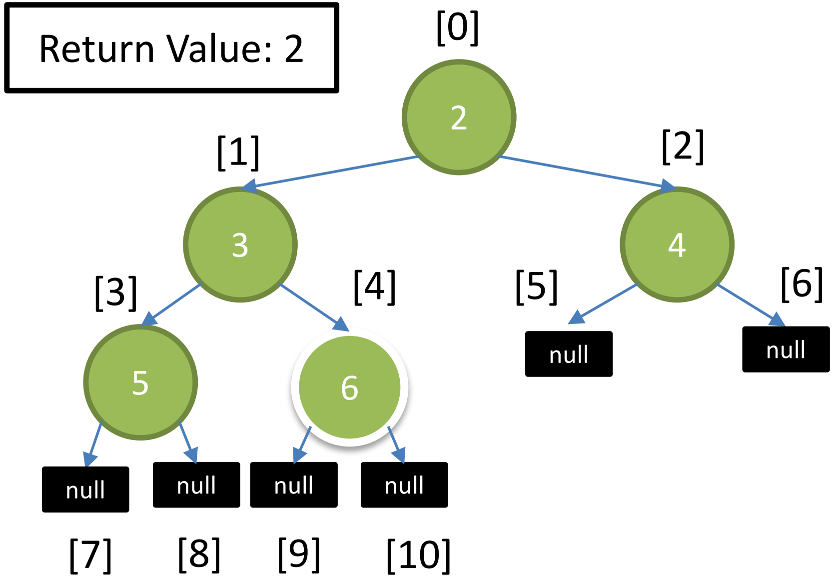


File Edit View Help

Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, "Bubble Down" that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent's value is smaller or we reach the tree's bounds

Example Remove Again

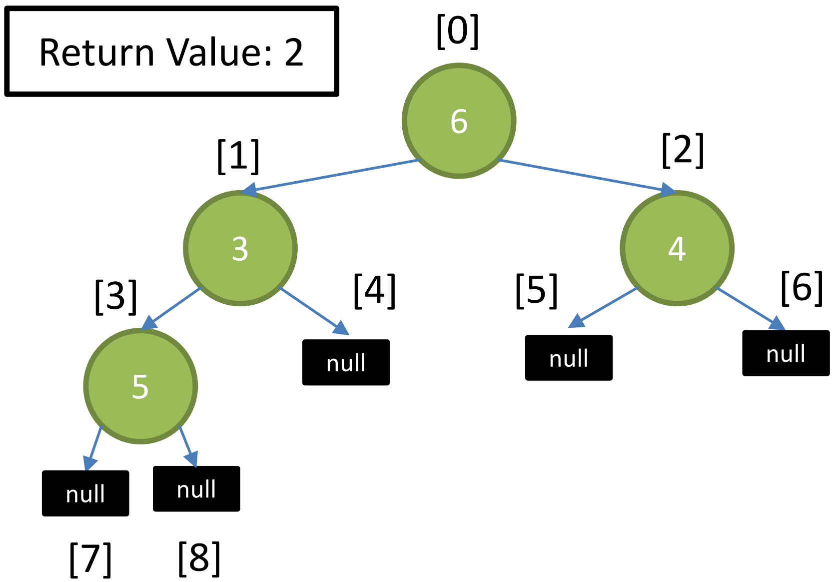


File Edit View Help

Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

Example Remove Again



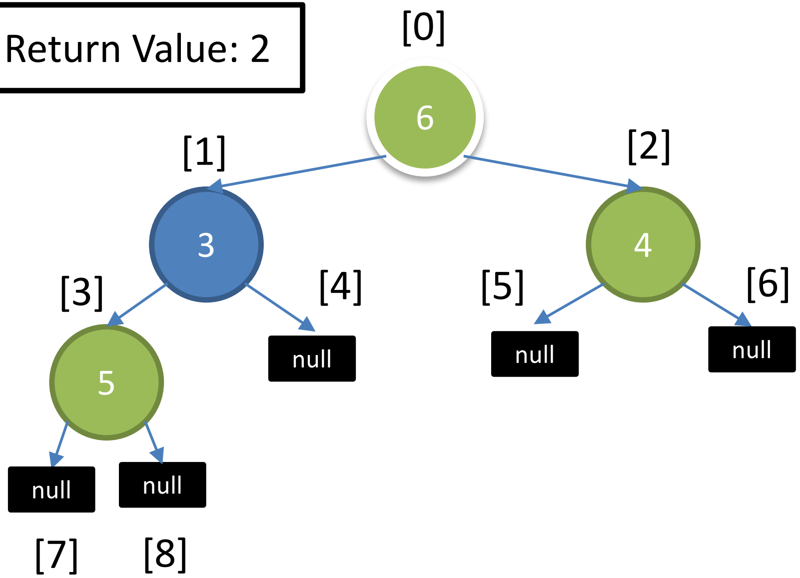
File Edit View Help

Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

Example Remove Again

Return Value: 2

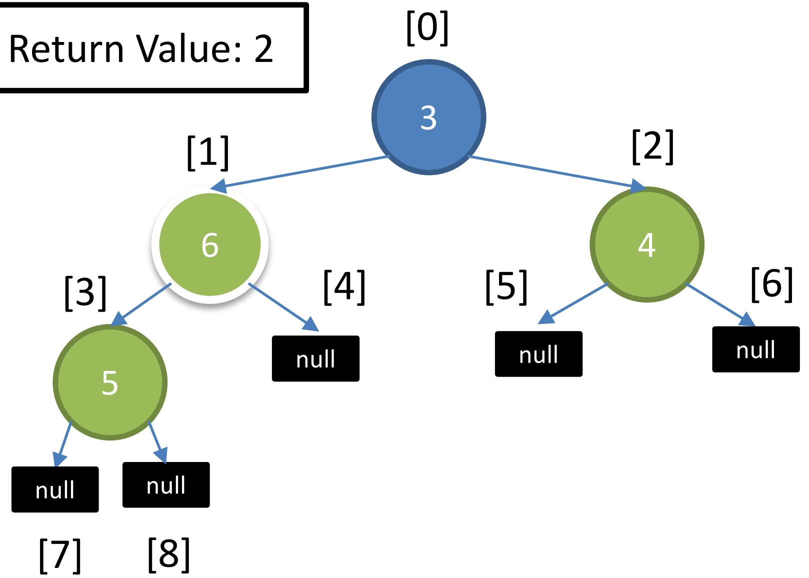


Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

Example Remove Again

Return Value: 2

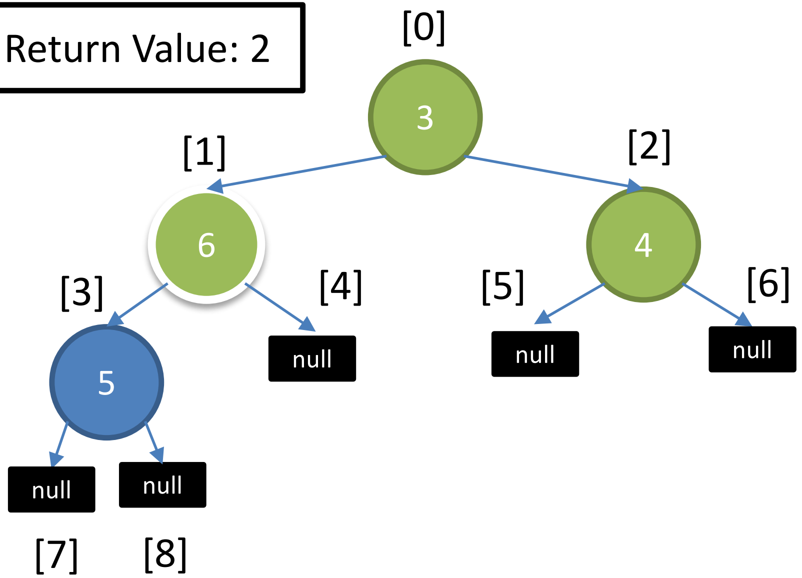


Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

Example Remove Again

Return Value: 2



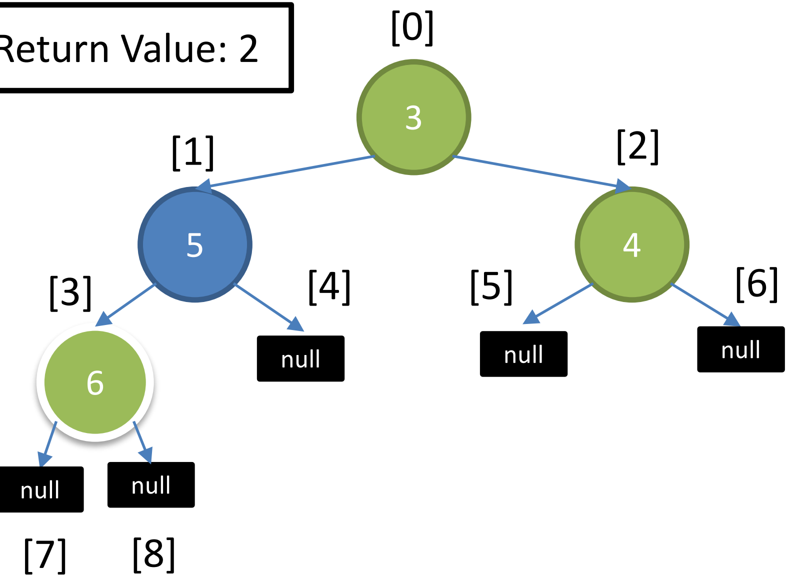
File Edit View Help

Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, "Bubble Down" that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent's value is smaller or we reach the tree's bounds

Example Remove Again

Return Value: 2



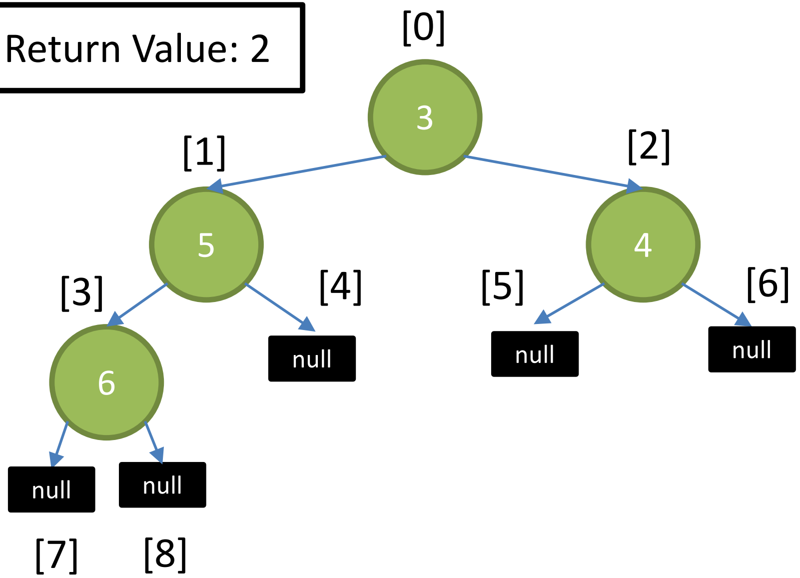
File Edit View Help

Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, "Bubble Down" that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent's value is smaller or we reach the tree's bounds

Example Remove Again

Return Value: 2

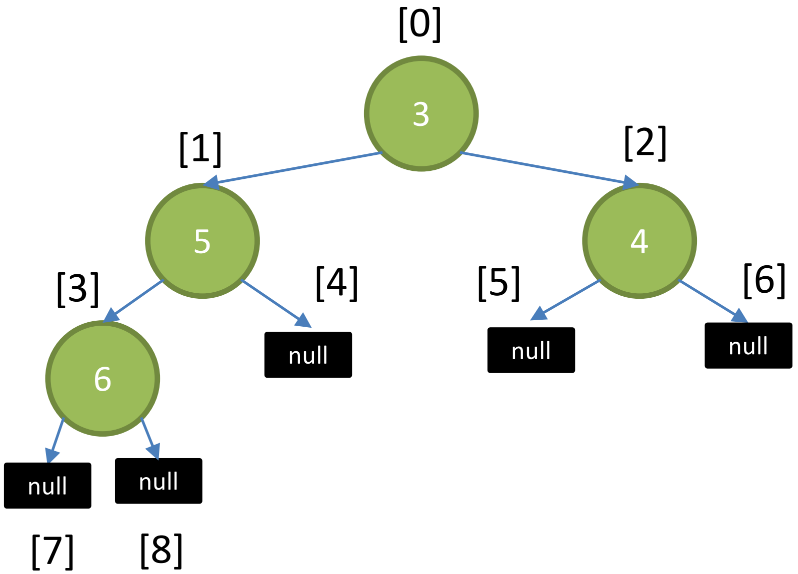


File Edit View Help

Heaps

- Remove
  - Store the data at the Root
  - Replace the Root data with the Data in the last node in Breadth Order
  - Starting from the root, “Bubble Down” that information
  - Return the stored value, previously at the root
- Bubble Down
  - Pick the smaller of the 2 children
  - If its value is smaller than the parent, then swap those values
  - Continue this until the parent’s value is smaller or we reach the tree’s bounds

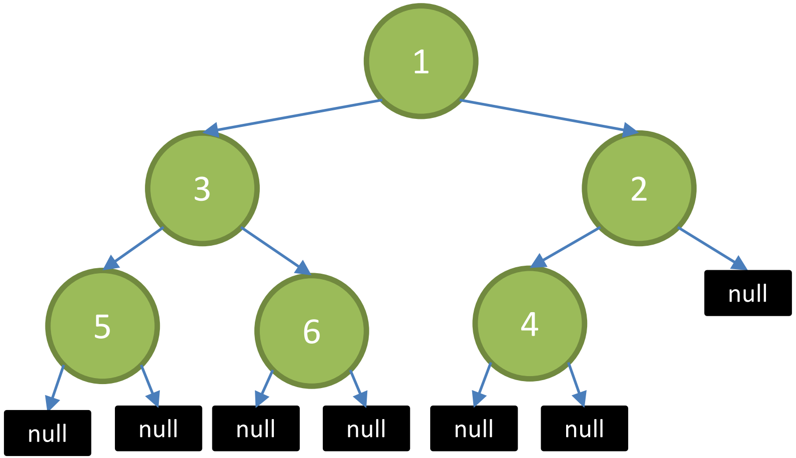
Example Remove Again



Heaps

- Heap Sort

Min Heap

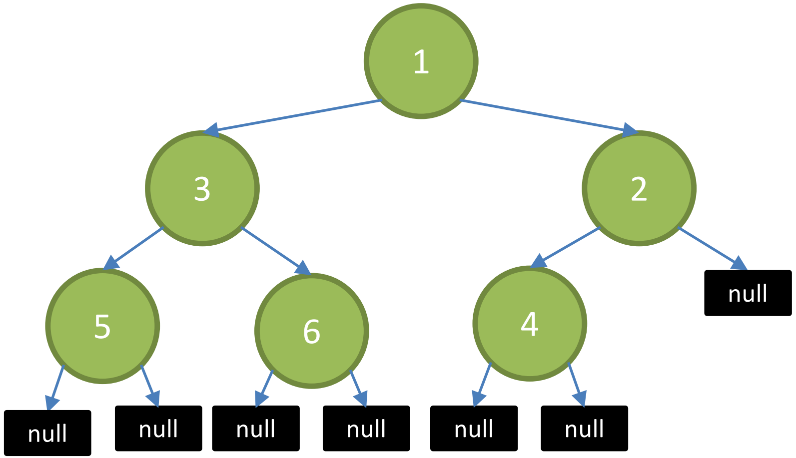


Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Min Heap



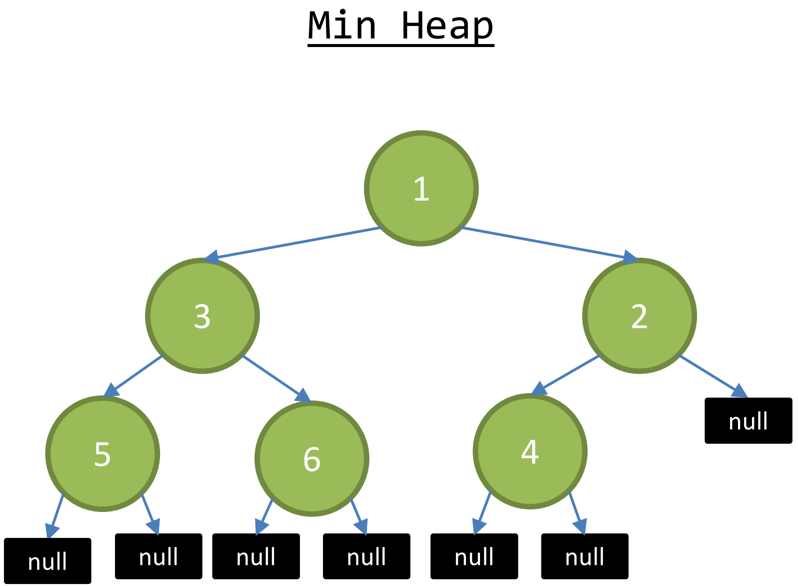
File Edit View Help

What?

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!



File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

<u>Heap Sort Values {5,1,4,3,6,2}</u>

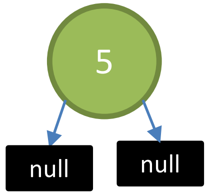
File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

<u>Heap Sort Values {5,1,4,3,6,2}</u>



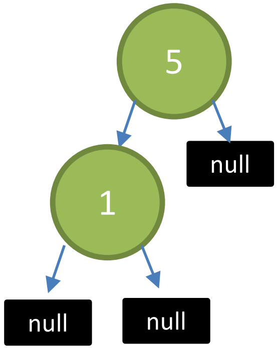
File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



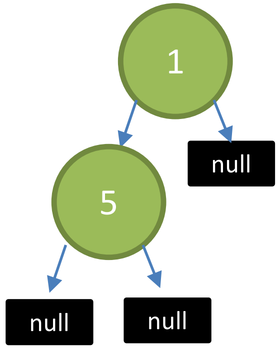
File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



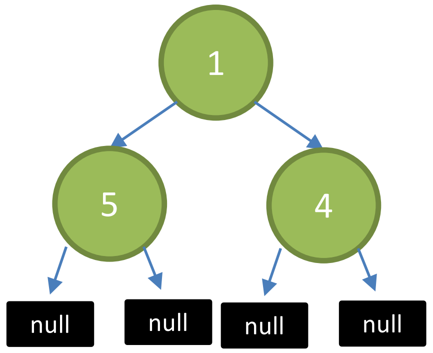
File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



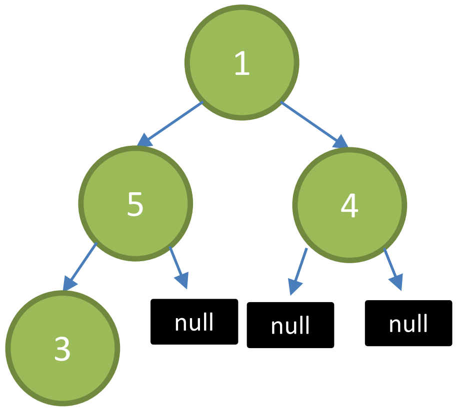
File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



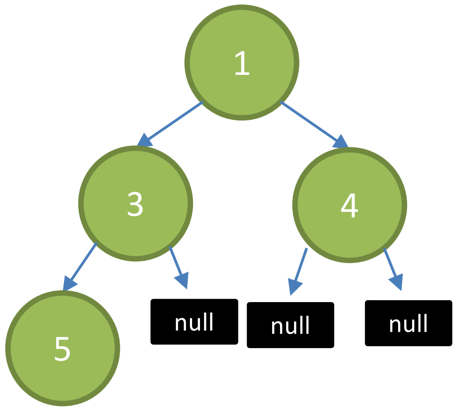
File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

<u>Heap Sort Values {5,1,4,3,6,2}</u>

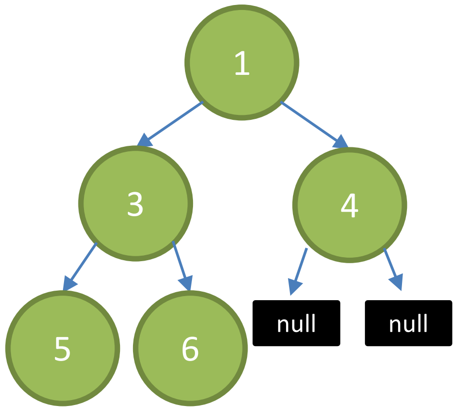


Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}

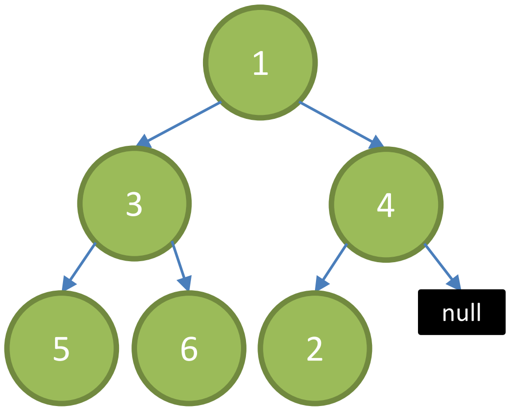


Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



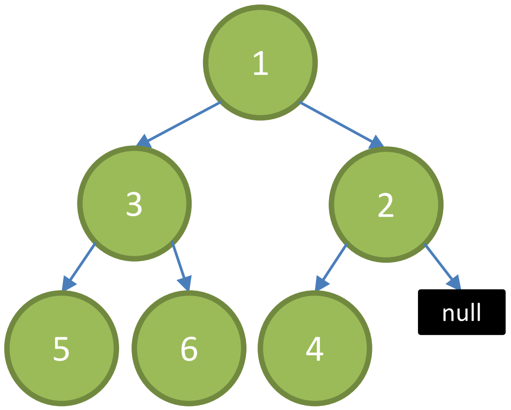
File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



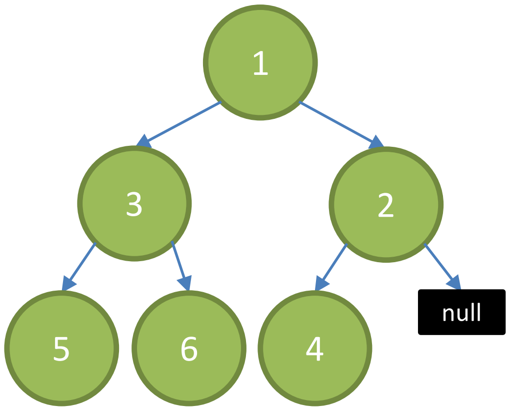
File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



Returned Values:

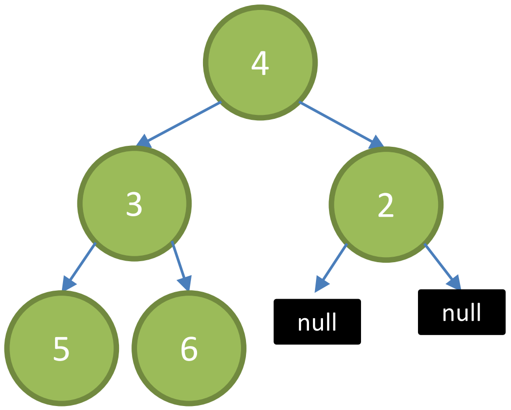
File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



Returned Values: 1,

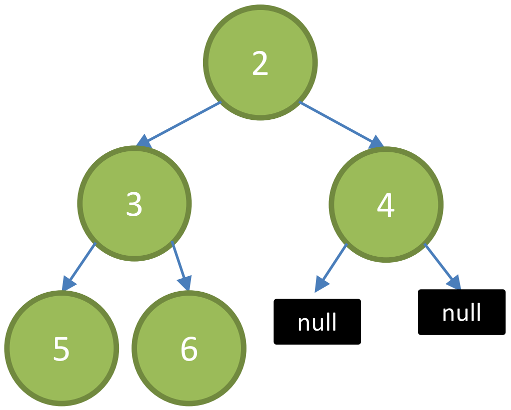
File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



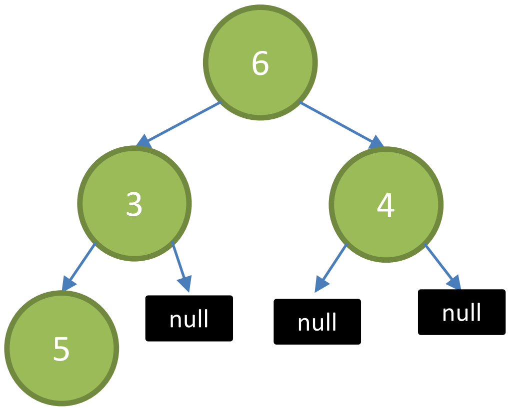
Returned Values: 1,

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

<u>Heap Sort Values {5,1,4,3,6,2}</u>



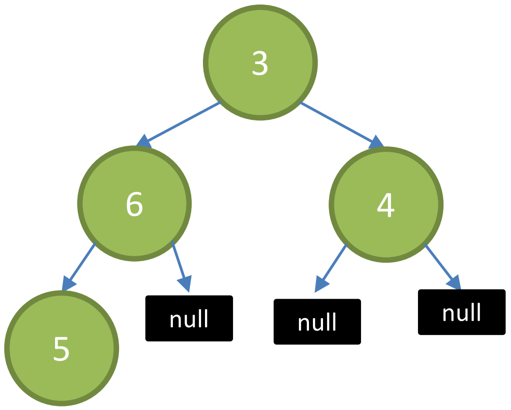
Returned Values: 1, 2

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



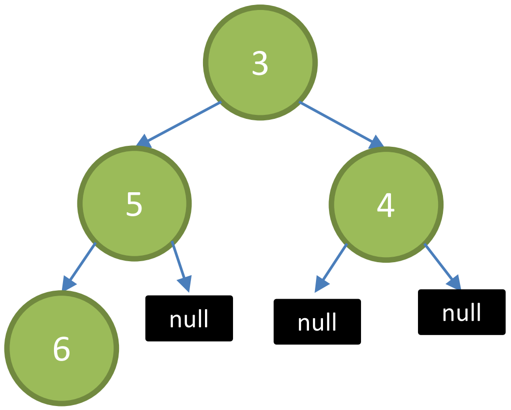
Returned Values: 1, 2

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



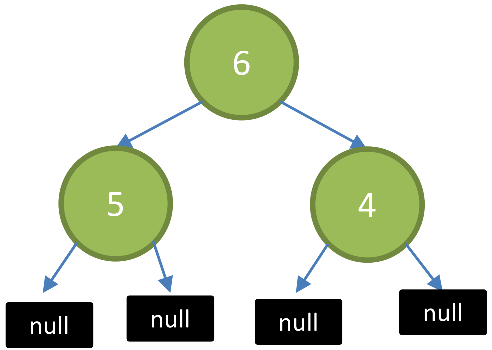
Returned Values: 1, 2

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



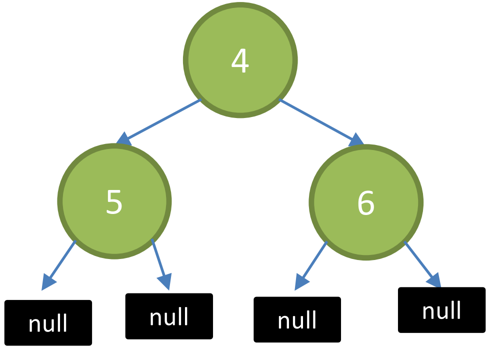
Returned Values: 1, 2, 3

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



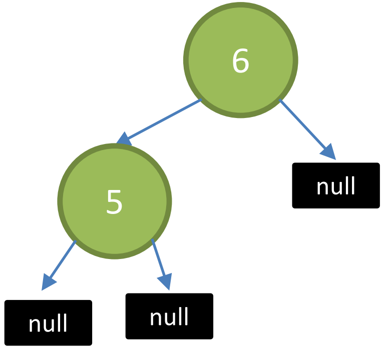
Returned Values: 1, 2, 3

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



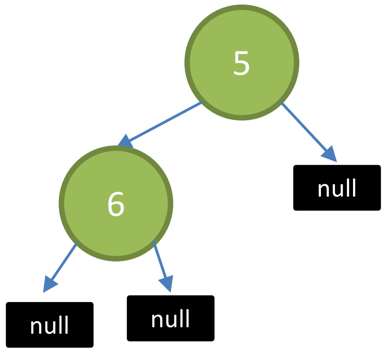
Returned Values: 1, 2, 3, 4

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

<u>Heap Sort Values {5,1,4,3,6,2}</u>



Returned Values: 1, 2, 3, 4

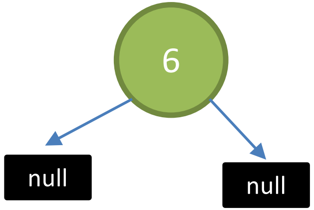
File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}



Returned Values: 1, 2, 3, 4, 5

File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}

Returned Values: 1, 2, 3, 4, 5, 6

File Edit View Help

Heaps

- Heap Sort

1. Add all Values to the Heap
2. Remove All Values from the Heap
3. DONE!

Heap Sort Values {5,1,4,3,6,2}

DONE!

Returned Values: 1, 2, 3, 4, 5, 6

File Edit View Help

Heap Sort Complexity

- Worst Case
  - Sorted in Descending Order
- Operations
  - Add all n values
  - Remove all n values
- Special Consideration
  - Heaps are always balanced trees

Complexity

?

File Edit View Help

Heap Sort Complexity

- Worst Case
  - Sorted in Descending Order
- Operations
  - Add all n values
  - Remove all n values
- Special Consideration
  - Heaps are always balanced trees

<u>Complexity</u>

$$O(n(\lg n))$$