Sorting

Eugeniy E. Mikhailov

The College of William & Mary

Lecture 27

Bubble sort method

Some one give us a vector of unsorted numbers.
We want to obtain the vector sorted in ascending order.

- assign `IndexOfTheLastToCheck` be the *index* of the vector end

1. start sweeping from the beginning of the vector
2. Compare the 2 consequent elements till we reach the `IndexOfTheLastToCheck`
3. if the left element is larger we swap these 2 elements
4. move to the next pair to the right i.e. move to the item 2
   - notice that at the end of the sweep the *index* of the last element to check holds the largest element
   - so next sweep does not have to be that long.
   - it is shorter by one element
   - i.e. the *index* of the last element to check should be decreased by 1
5. decrease `IndexOfTheLastToCheck` by 1
6. if `IndexOfTheLastToCheck` $>$ 1 repeat from the item 1

$x = [3, 1, 4, 5, 2]$
first sweep
$x = [\widehat{3, 1}, 4, 5, 2]$ swap
$x = [1, 3, 4, 5, 2]$ after swap
$x = [1, \widehat{3, 4}, 5, 2]$ no swap
$x = [1, 3, \widehat{4, 5}, 2]$ no swap
$x = [1, 3, 4, \widehat{5, 2}]$ swap
$x = [1, 3, 4, 2, 5]$ sweep done
new sweep
$x = [\widehat{1, 3}, 4, 2, 5]$ no swap
$x = [1, \widehat{3, 4}, 2, 5]$ no swap
$x = [1, 3, \widehat{4, 2}, 5]$ swap
$x = [1, 3, 2, 4, 5]$ sweep done
new sweep
$x = [\widehat{1, 3}, 2, 4, 5]$ no swap
$x = [1, \widehat{3, 2}, 4, 5]$ swap
$x = [1, 2, 3, 4, 5]$ sweep done
last sweep
$x = [\widehat{1, 2}, 3, 4, 5]$ no sweep
$x = [1, 2, 3, 4, 5]$ sweep done

Bubble sort properties

- This is the worst of all working algorithm!
- The execution time of this algorithm is $\mathcal{O}(N^2)$
- Never use it in the real life!
- However it is very simple to program, and does not require extra memory for execution.

Quick sort method

Much better yet simple algorithm
Let's discuss recursive realization
We will name our sorting function as `qsort`.

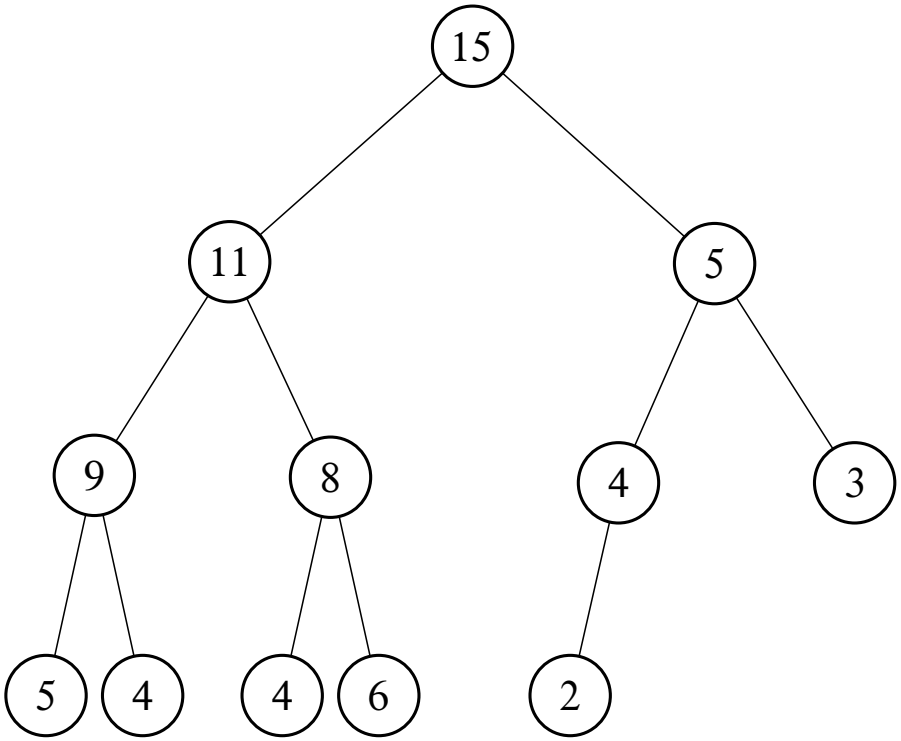
- choose a pivot point value
  - let's choose the pivot at the middle of the vector
  - pivotIndex=floor(N/2)
  - pivotValue=x(pivotIndex)
- create two vectors which hold lesser and larger than pivotValue elements of the input vector.
- now concatenate the result of
  xs=[`qsort`(lesser), pivotValue, `qsort`(larger)]
- done

Quick sort summary

- usually fast
- typical execution time $\mathcal{O}(N \log_2 N)$
- but it is not guaranteed
  - However for certain input vectors execution time could be as long as $\mathcal{O}(N^2)$

Heap

Heap is a structure where parent element is larger or equal to its children.



The top most element of the heap is called root.

Heap sorting method

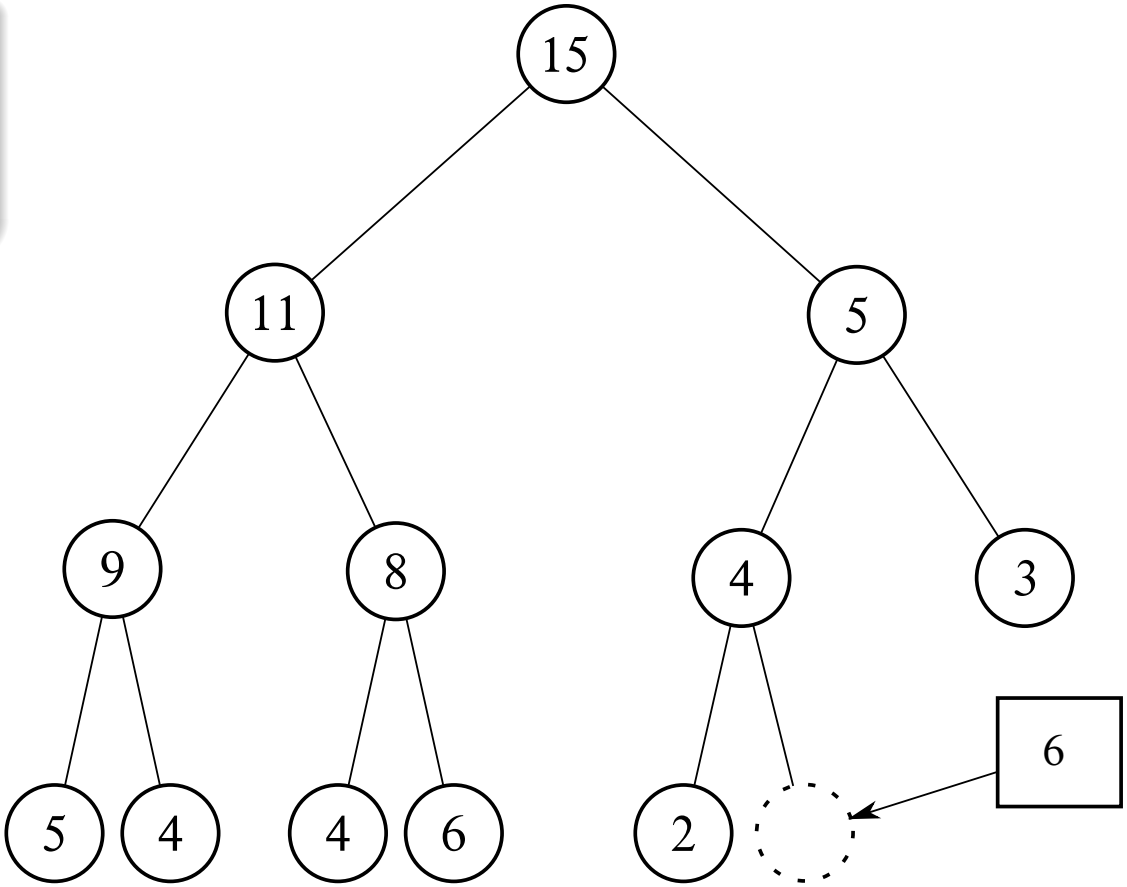
1. Fill the heap from the input vector elements
   1. take the element and place it at the bottom of the heap
   2. sift-up (bubble up) this element
   3. do the same with the next element
2. remove the root element since it is the largest
3. rearrange the heap i.e. sift-down
   1. take the last bottom element
   2. place it at the root
   3. check if parent is larger then children
      1. find the largest child element
      2. if the largest child is larger then parent swap them and repeat the check
4. repeat step 2 until no elements left in the heap

Heap sorting complexity $\mathcal{O}(N \log_2 N)$

Filling (sift-up) the heap

Step 1

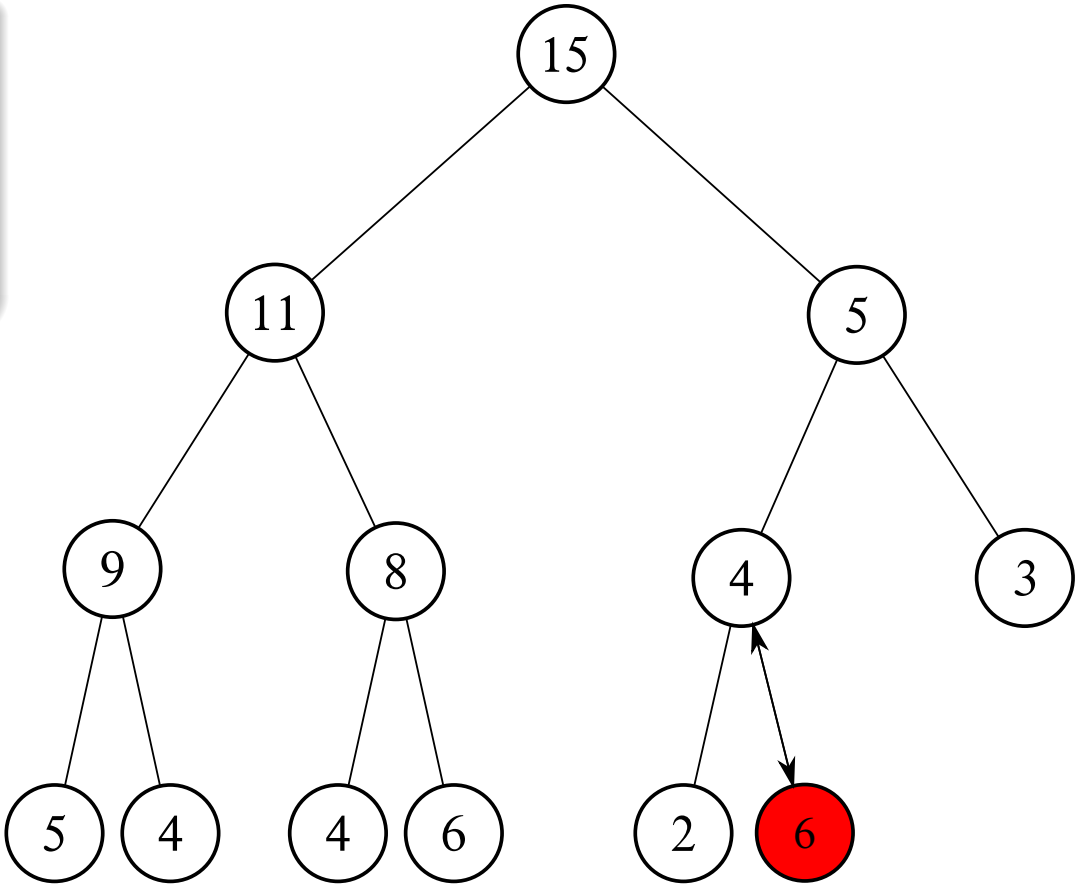
Place new element at the bottom of the heap



Filling (sift-up) the heap

Step 2

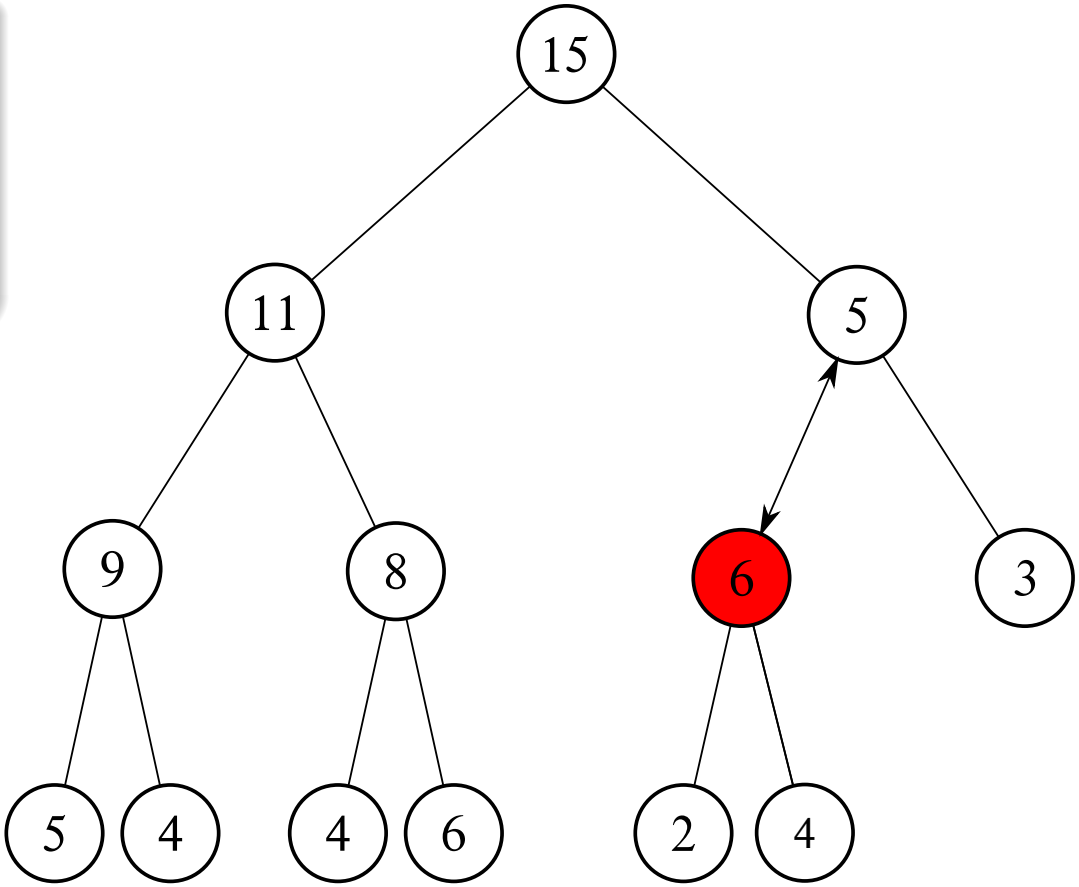
Check if parent is larger then child. If so swap them and repeat step 2.



Filling (sift-up) the heap

Step 2

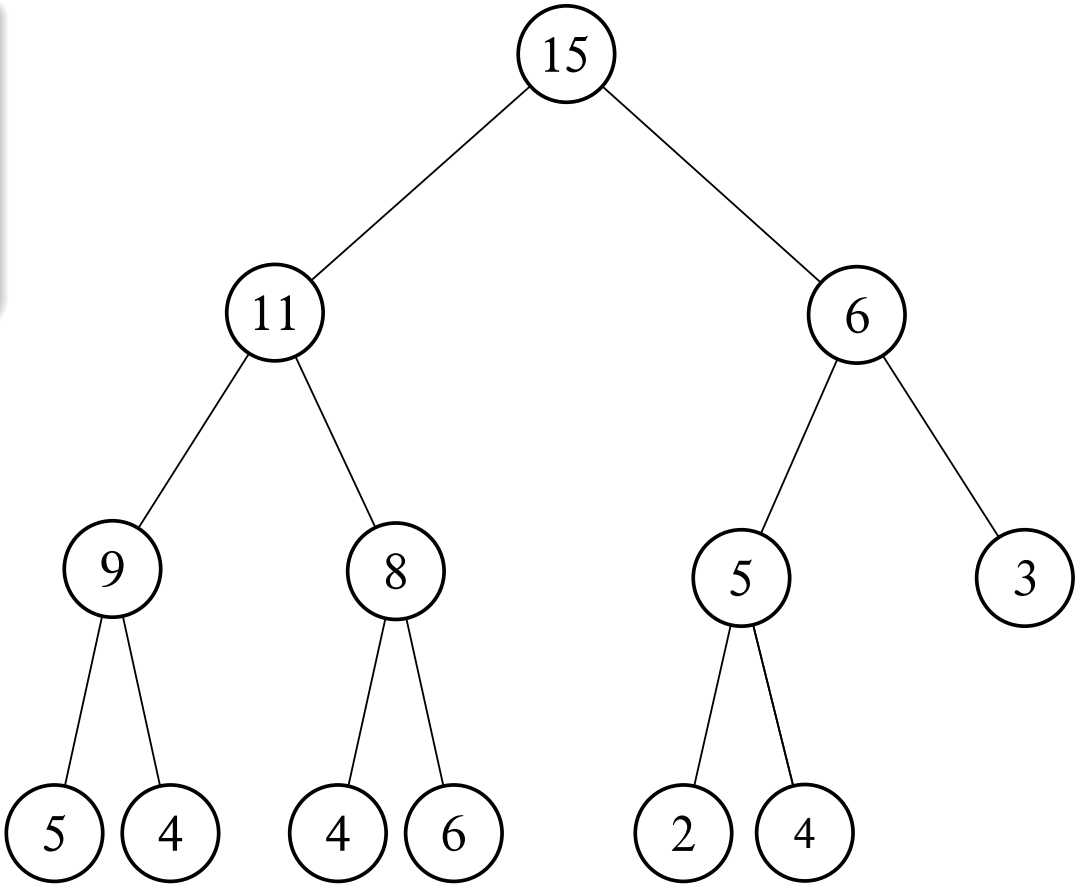
Check if parent is larger then child. If so swap them and repeat step 2.



Filling (sift-up) the heap

Step 2

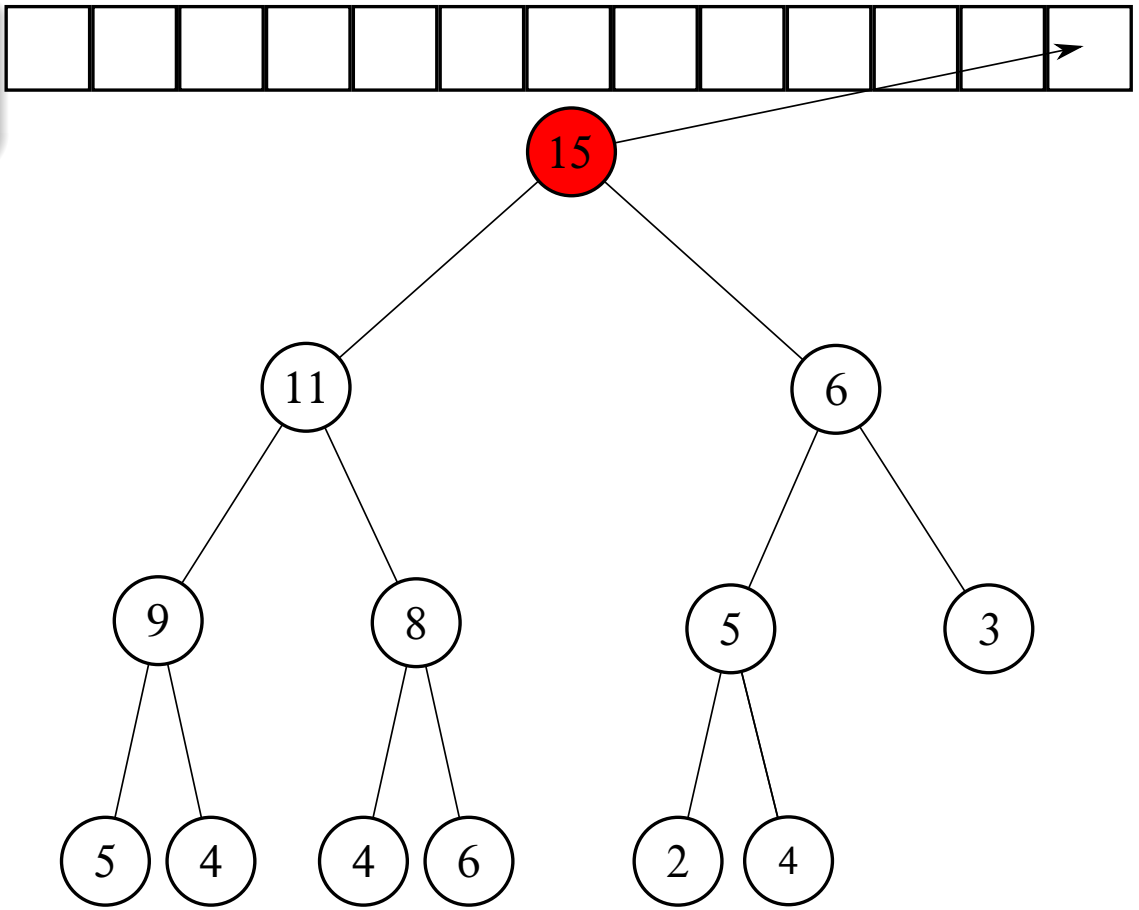
Check if parent is larger then child. If so swap them and repeat step 2.



Removing from the heap (sift-down) the heap

Step 1

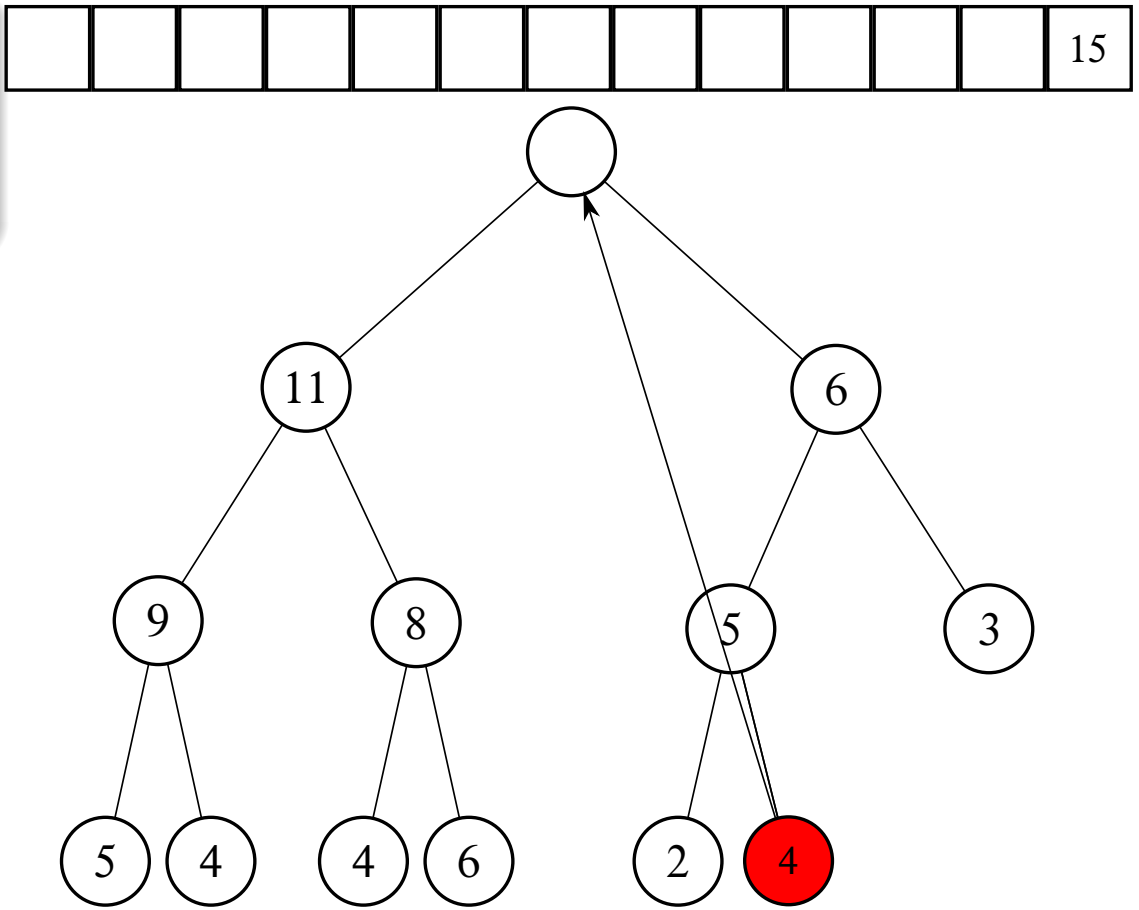
Remove the root element



Removing from the heap (sift-down) the heap

Step 2

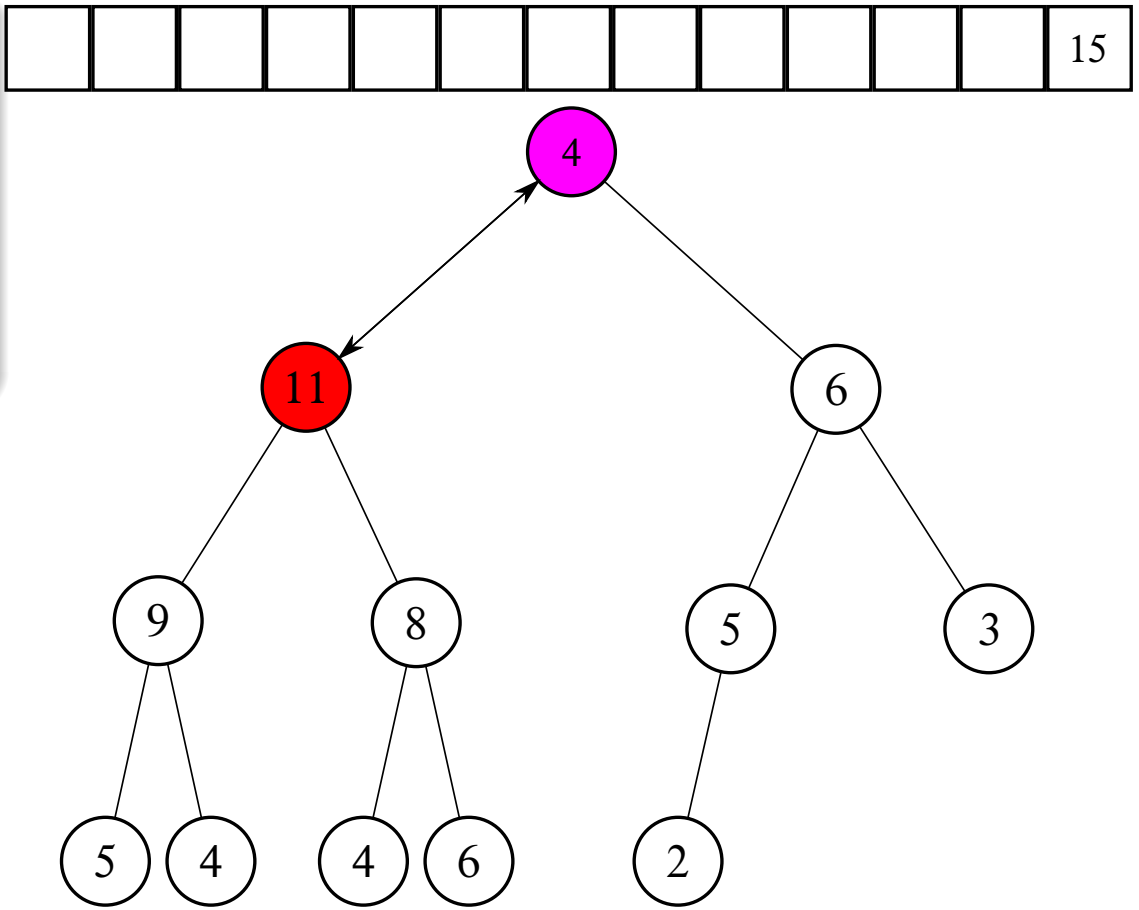
Place the last element of the heap to the root



Removing from the heap (sift-down) the heap

Step 3

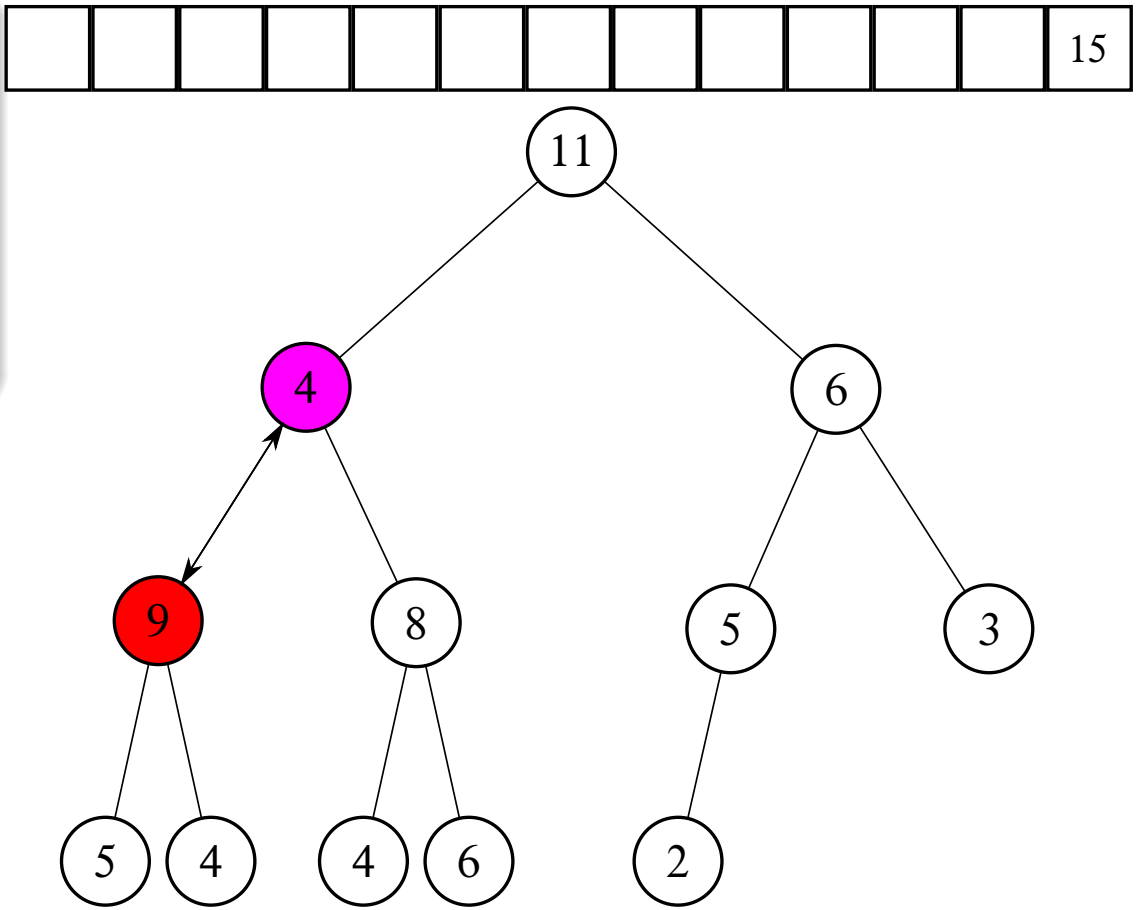
Check if parent is smaller than the largest child. If so swap and repeat step 3 else go to step 1



Removing from the heap (sift-down) the heap

Step 3

Check if parent is smaller than the largest child. If so swap and repeat step 3 else go to step 1

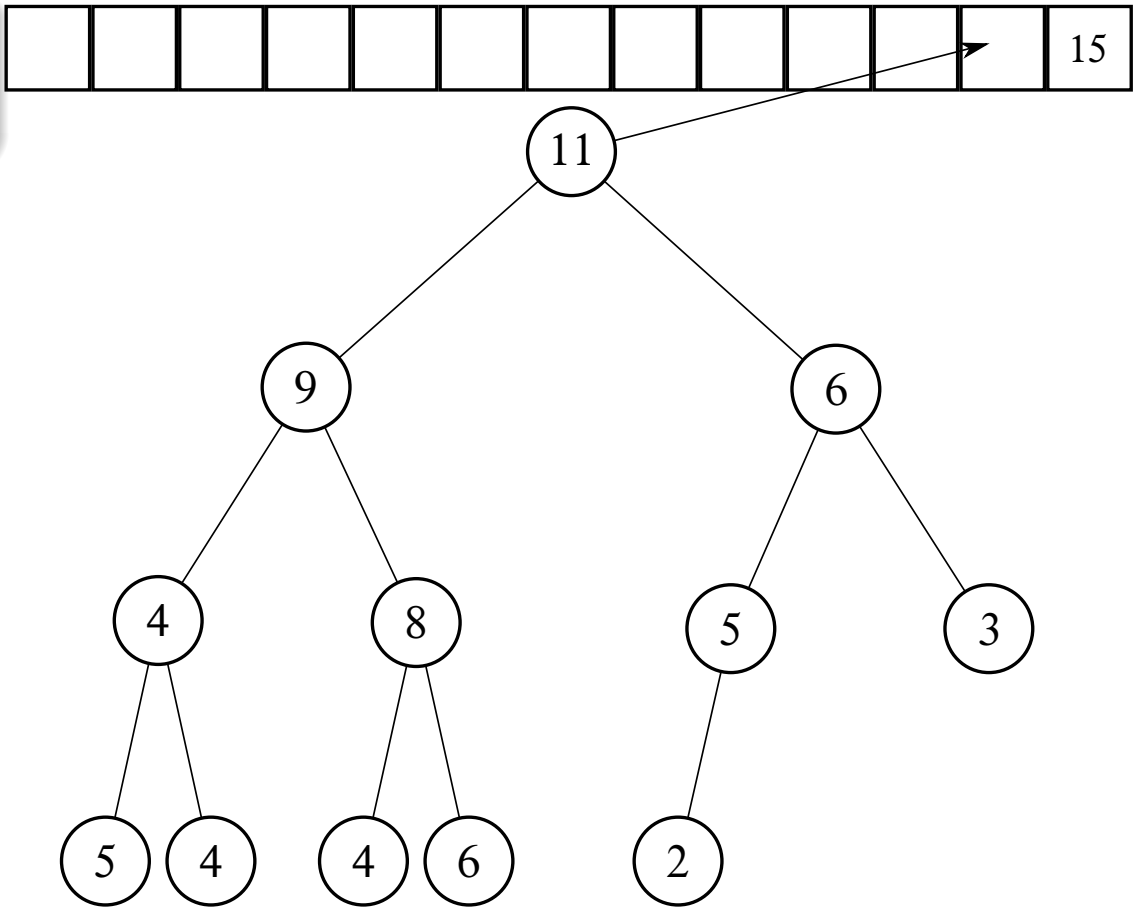


Removing from the heap (sift-down) the heap

Sequence repeats

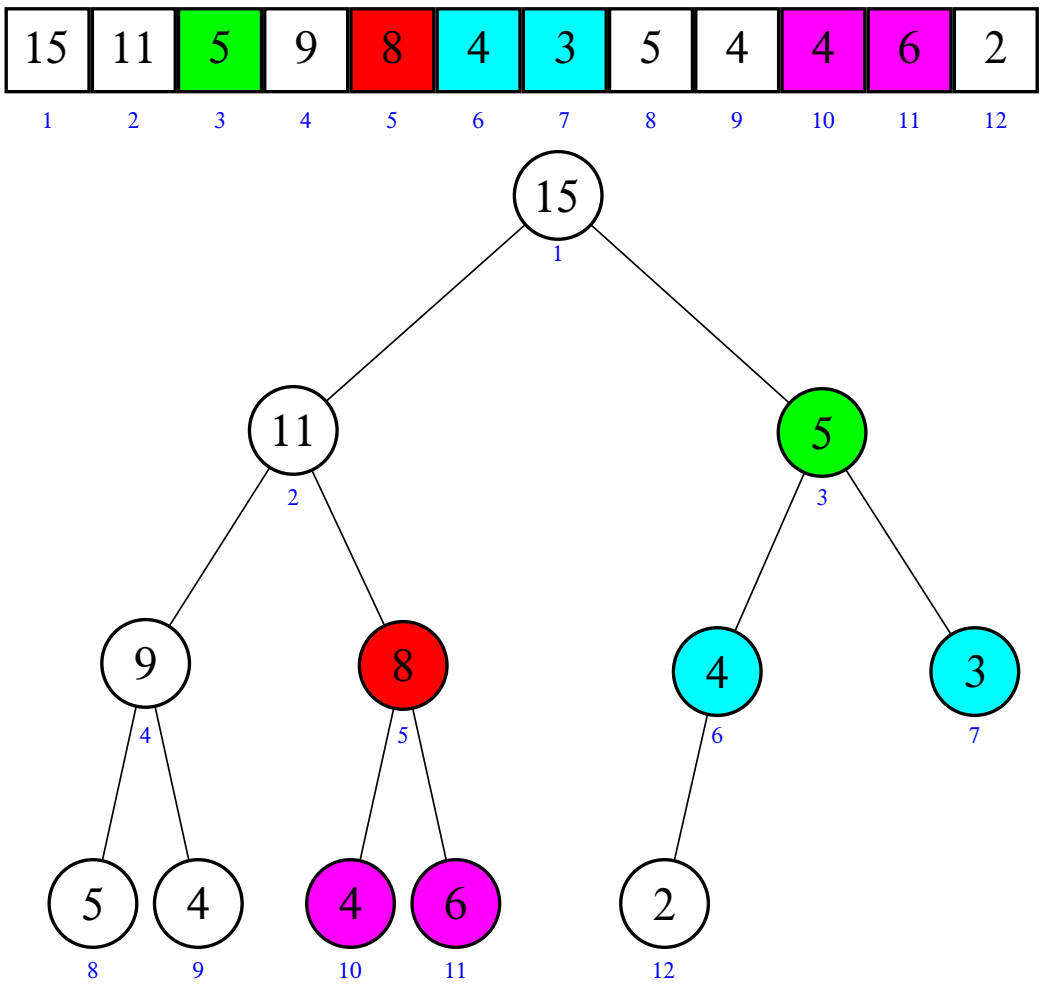
Step 1

Remove the root element



Vector heap representation

- Heap nodes are numbered consequently these numbers represent the node position in the vector.
- notice that parent and children have very simple relationship
  - if parent node index is i
    - child 1 index is $2i$
    - child 2 index is $2i + 1$
  - if we know child index (i) then
    - parent index is `floor`$(i/2)$



Matlab built in 'issorted'

Easy check if an array is sorted can be done with `issorted` which returns `true` or `false`.

```
>> x=[1,2,3];
>> issorted(x)
ans =
1
```

`issorted` checks only for ascending order, for example

```
>> x=[3,2,1];
>> issorted(x)
ans =
0
```

Recall that '0' is equivalent of `false` in Matlab