Module 6 Assignment: MinHeap

Bookmark this page

Coding Assignment due Jan 29, 2022 19:51 HKT

MinHeap

For this assignment, you will be coding a **MinHeap** that is backed by an array of contiguous elements. Here is a tree and array representation of the same MinHeap:

IMPORTANT:

- **You will be given 5 attempts on this assignment, with a 30 minute cooldown between submissions.**
- **Please run your code before each submission to ensure that there are no formatting errors! If there are formatting errors in your code, your code will not be graded and a submission attempt will be logged. For more information, please review the Vocareum overview below.**

Properties

A MinHeap is a type of binary tree with two main properties:

- **Shape Property**: The tree must be complete. All levels of the tree must be full except the bottom-most level. If the bottom-most level is not full, it must be filled from left to right with no gaps.
- **Order Property**: Each node's data is smaller than the data in its children. There is no explicit relationship between sibling nodes.

These properties guarantee that the smallest element in the heap will be at the root of the heap.

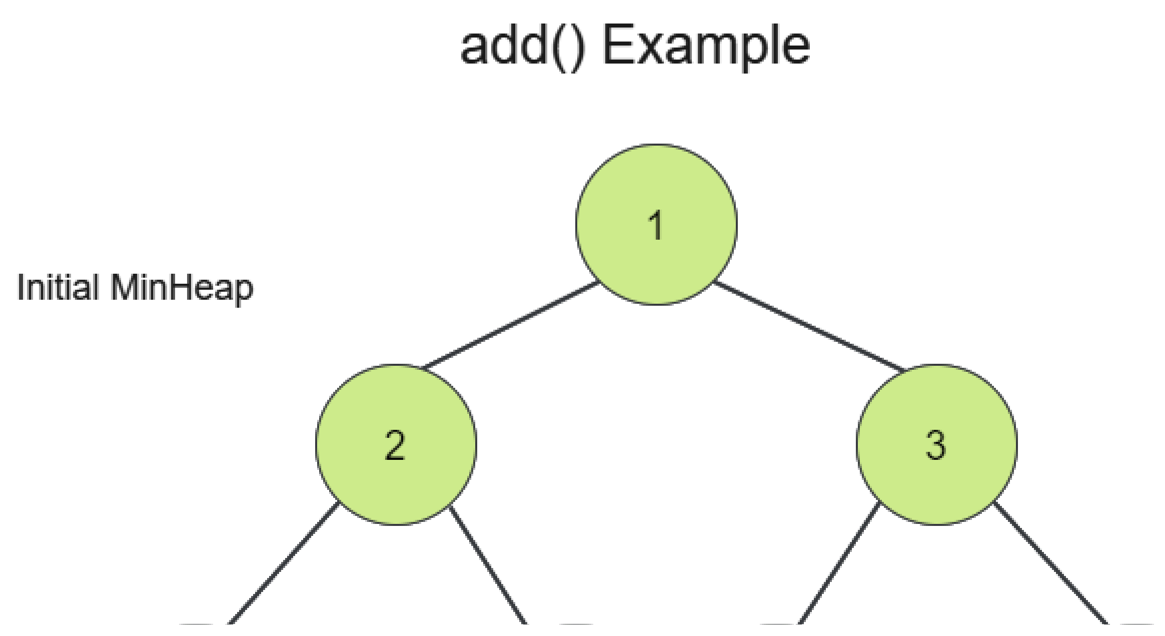
Implementation Details

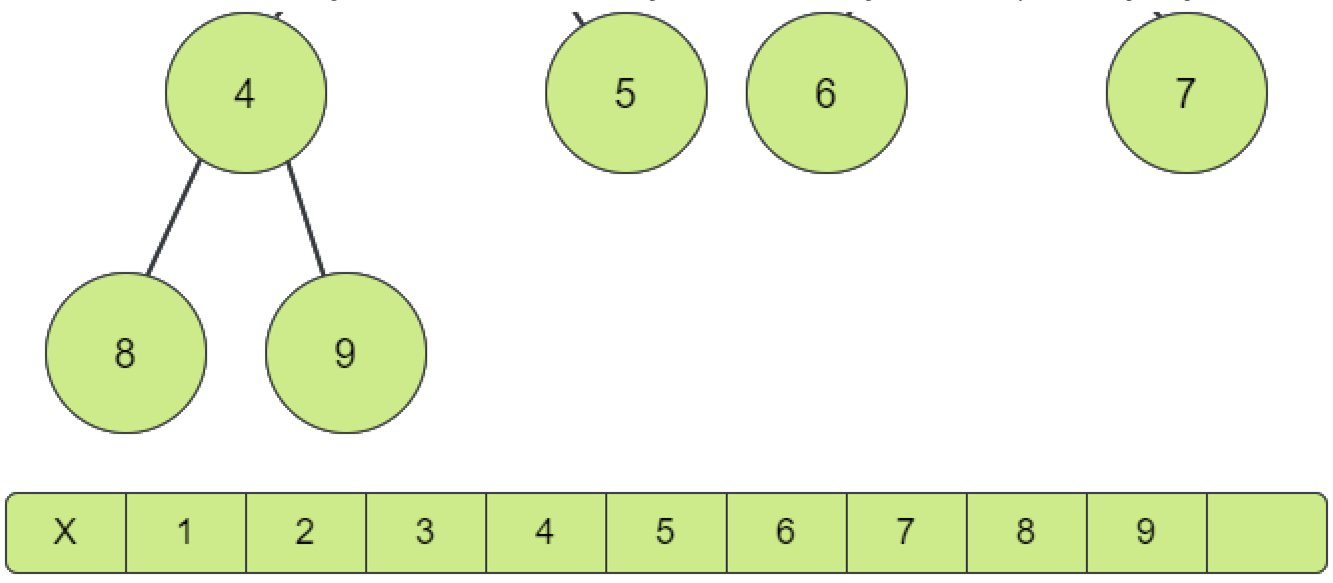
Although heaps are usually classified as a type of tree, they are commonly implemented using an array due to their completeness. In your implementation, you should **leave index 0 empty and begin your heap at index 1**. This will make the arithmetic for finding parent and children indices simpler.

When resizing your backingArray, double the current capacity. Therefore, if the initial capacity is 13, the first resize should bring the total backingArray capacity to 26. Note that this includes the 0th index!

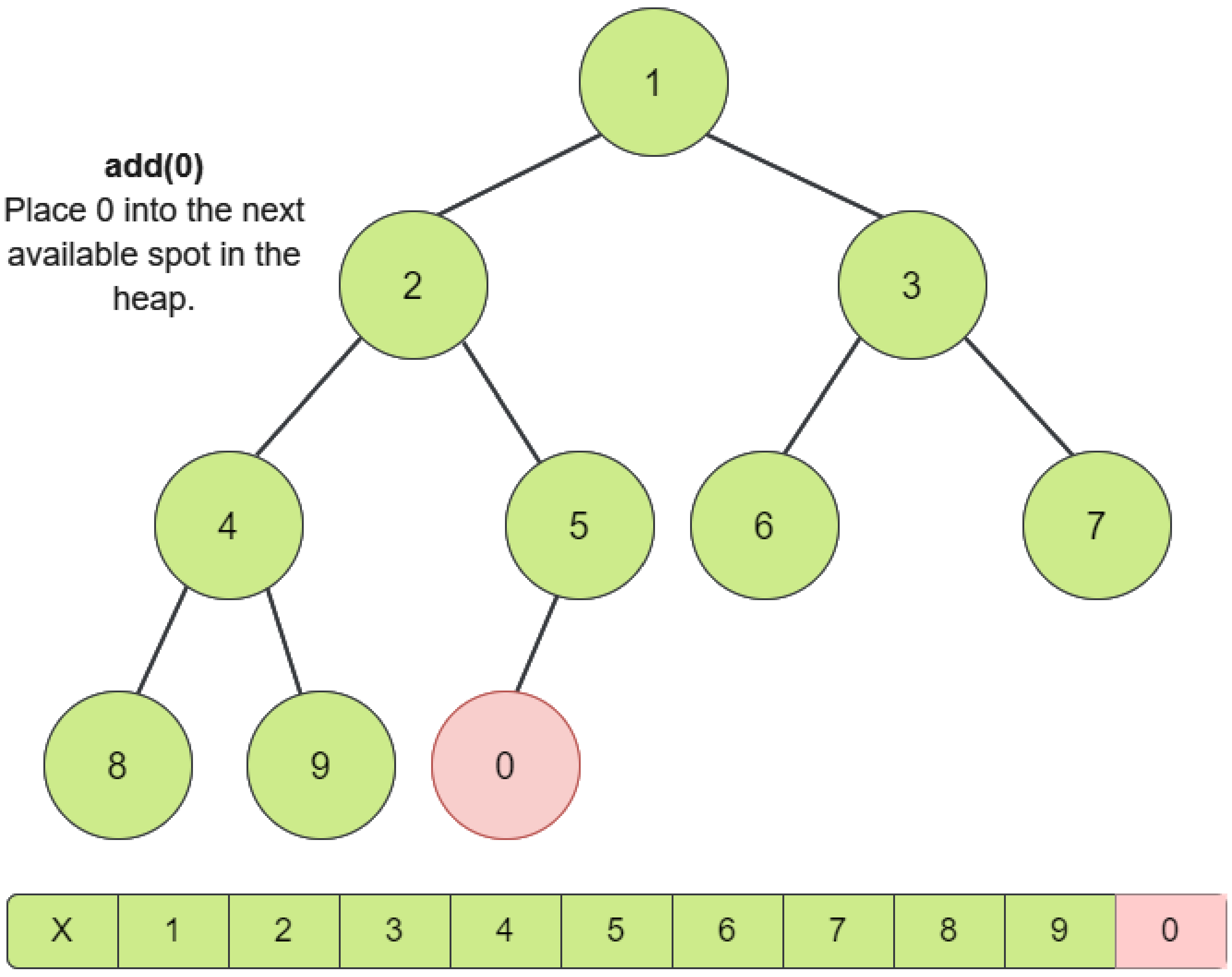
You may assume that your implementation does not need to handle duplicate elements. That is, the add method will never be passed duplicates and the remove method will never have to deal with the heap having duplicates. To be clear, your implementation would most likely work even if we were to test for duplicates; however, this will help remove ambiguity surrounding grading and testing your implementation.

Unlike your BST homework, you are **not required to use recursion** in this assignment. Use whatever you find most intuitive - recursion, iteration, or both. However, regardless of the technique you use, make sure to meet the efficiency requirements as discussed in this course.

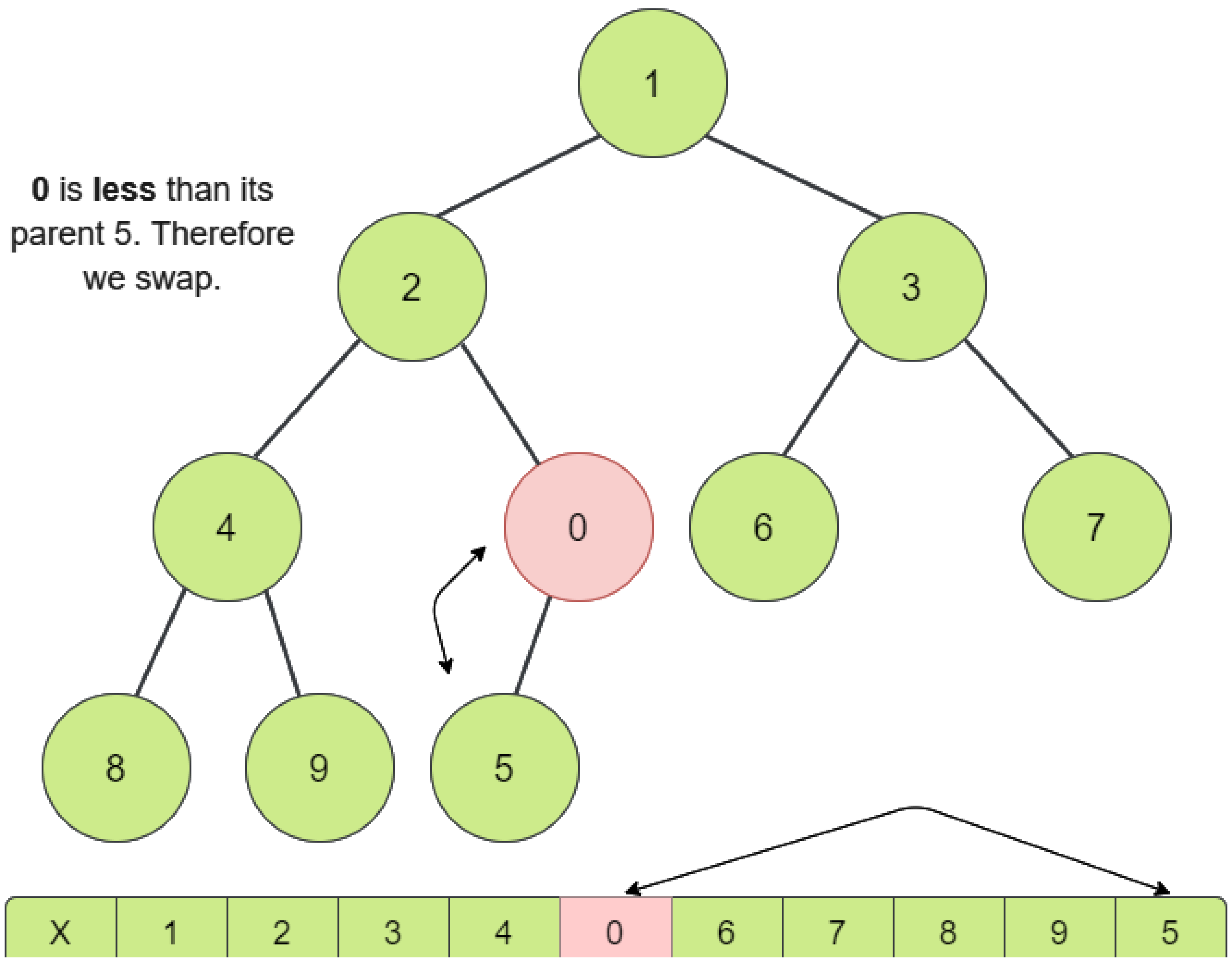


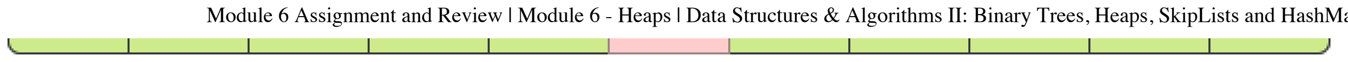


add(0)
Place 0 into the next available spot in the heap.

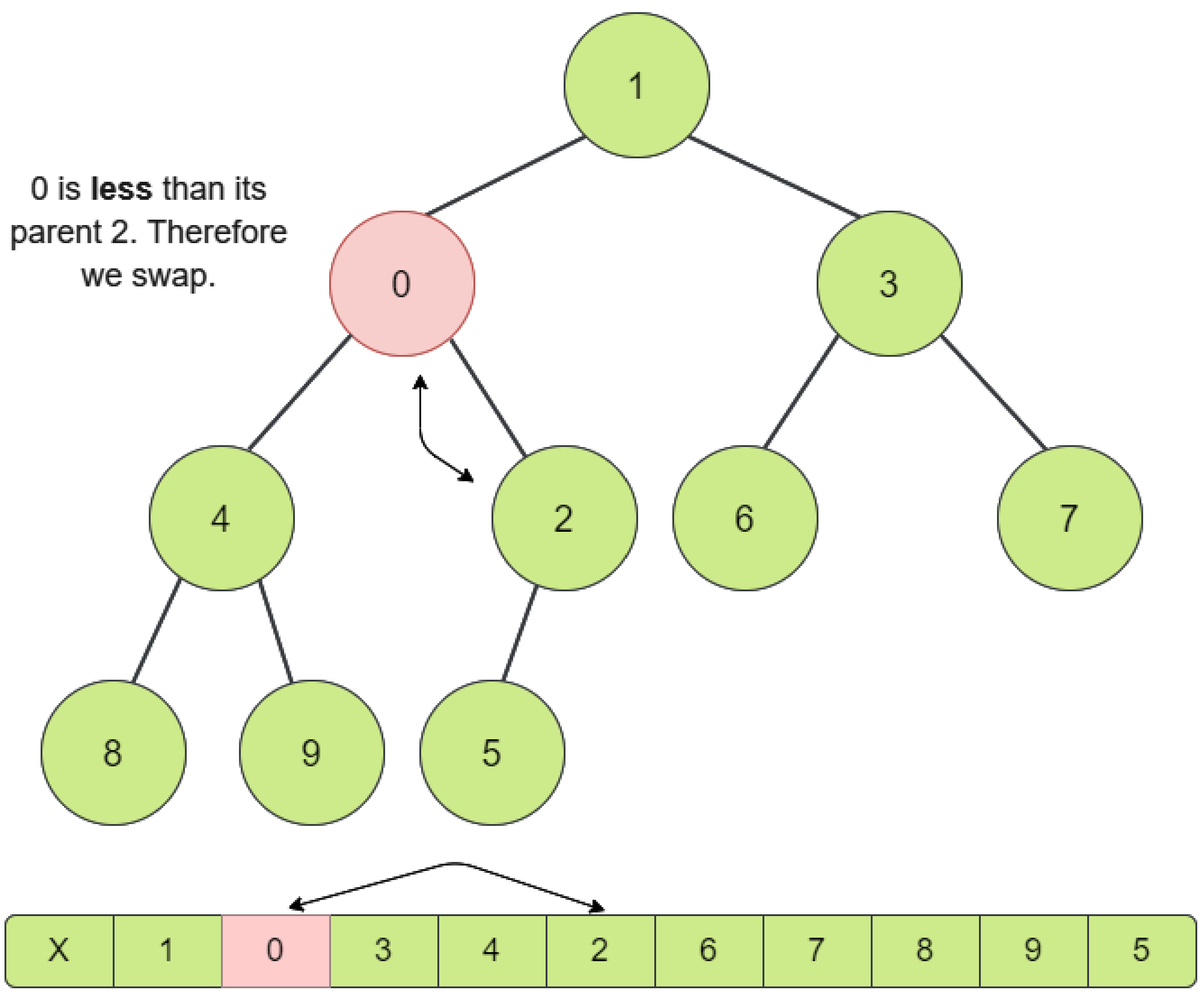


0 is **less** than its parent 5. Therefore we swap.

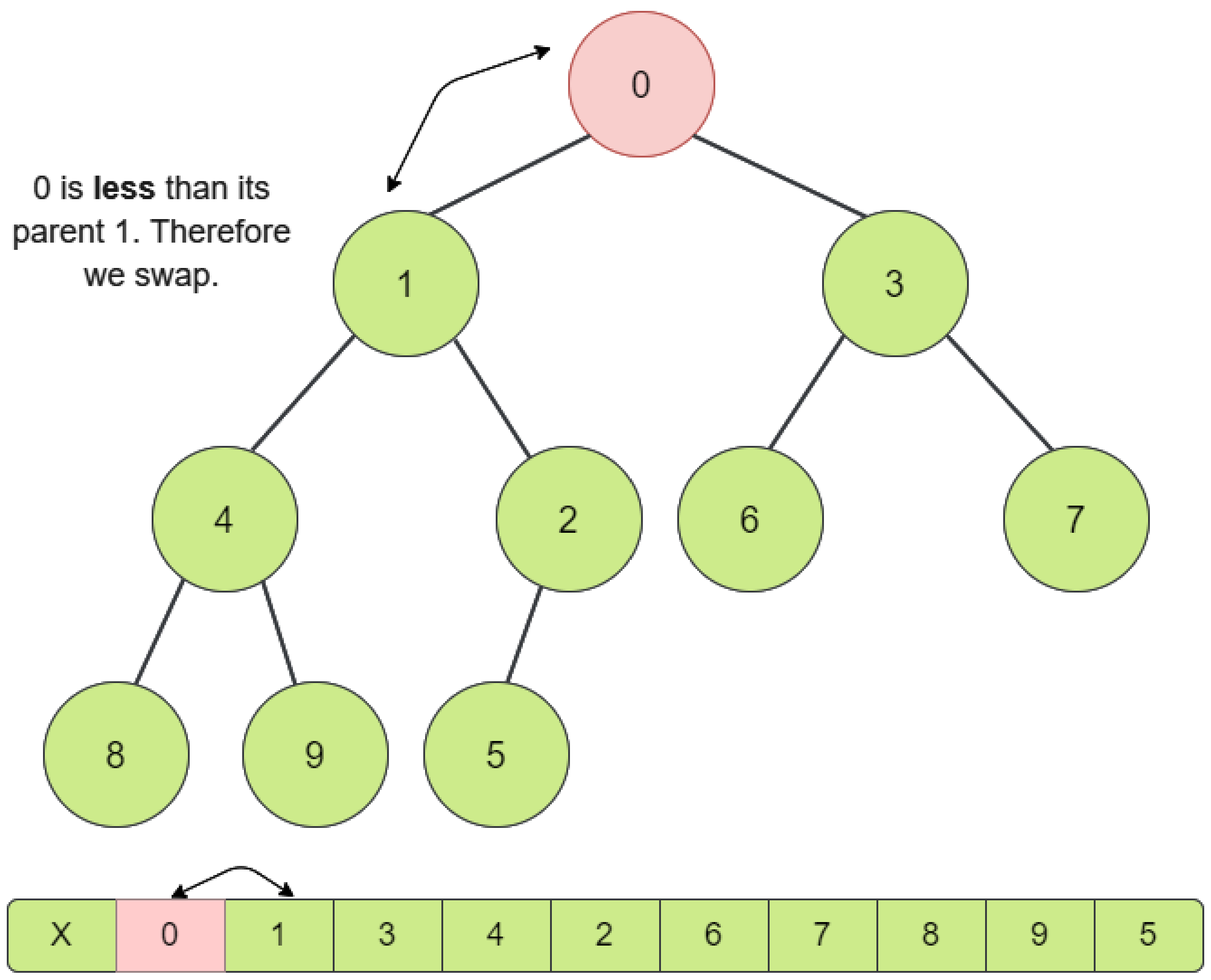




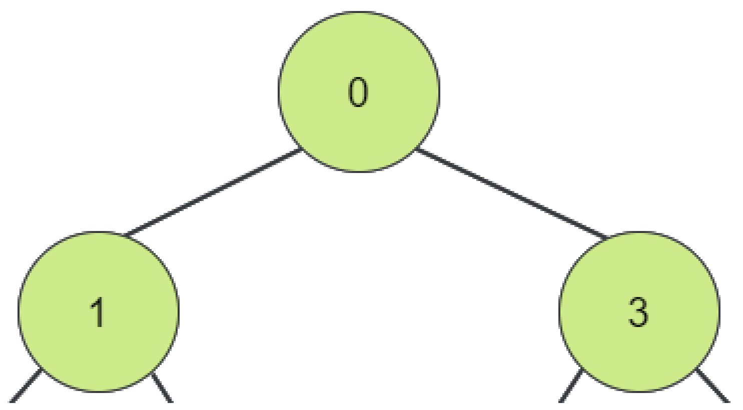
0 is **less** than its parent 2. Therefore we swap.



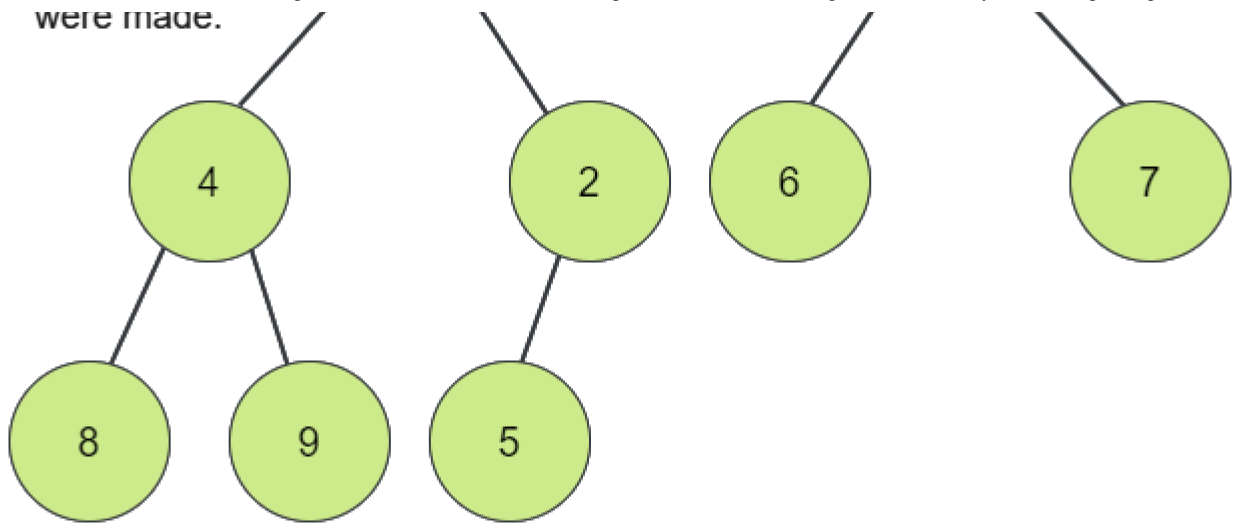
0 is **less** than its parent 1. Therefore we swap.



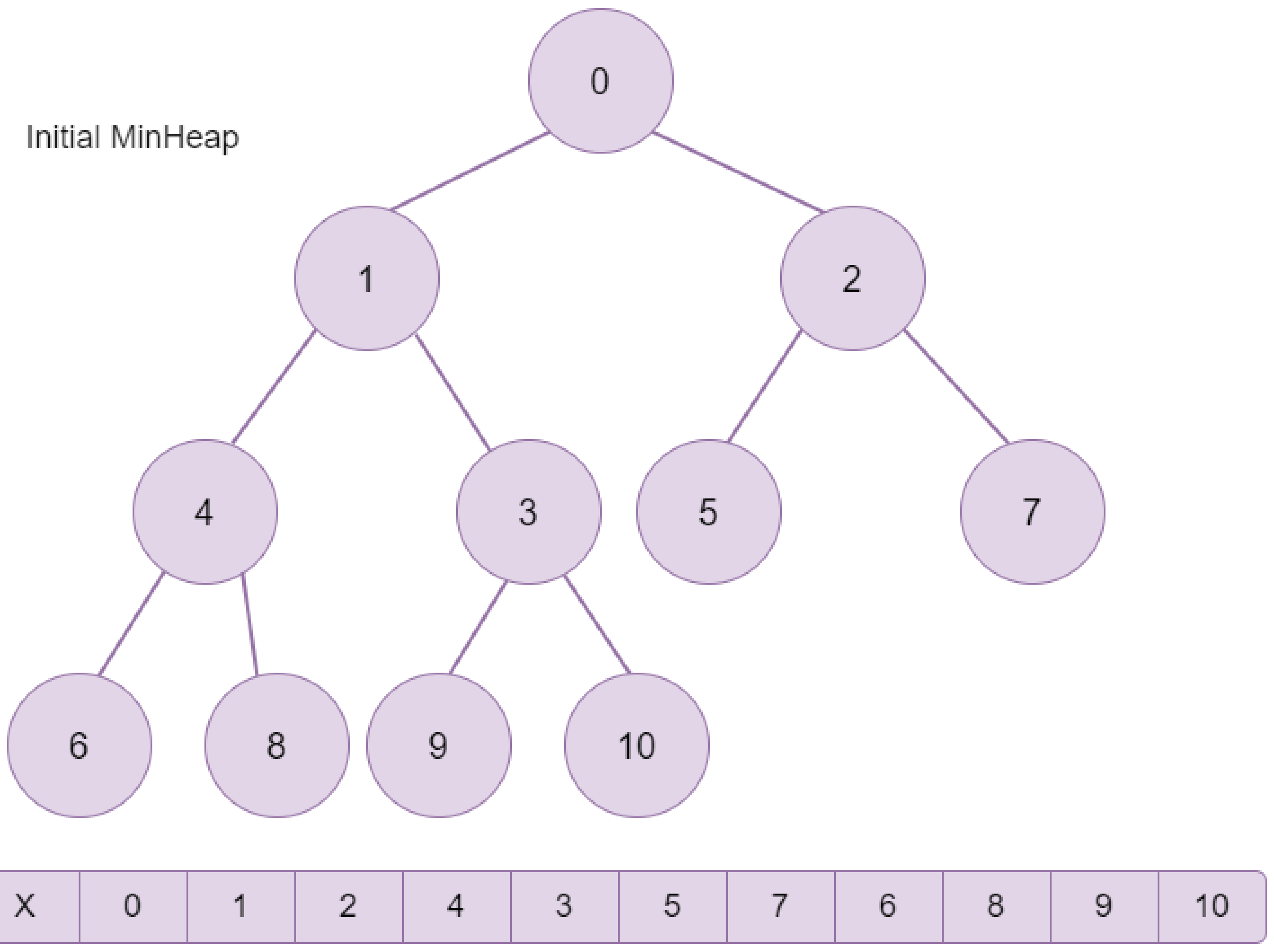
Since we have reached the top of the MinHeap, we stop upheaping. Note that we would have stopped upheaping at any point if no swaps were made.



were made.

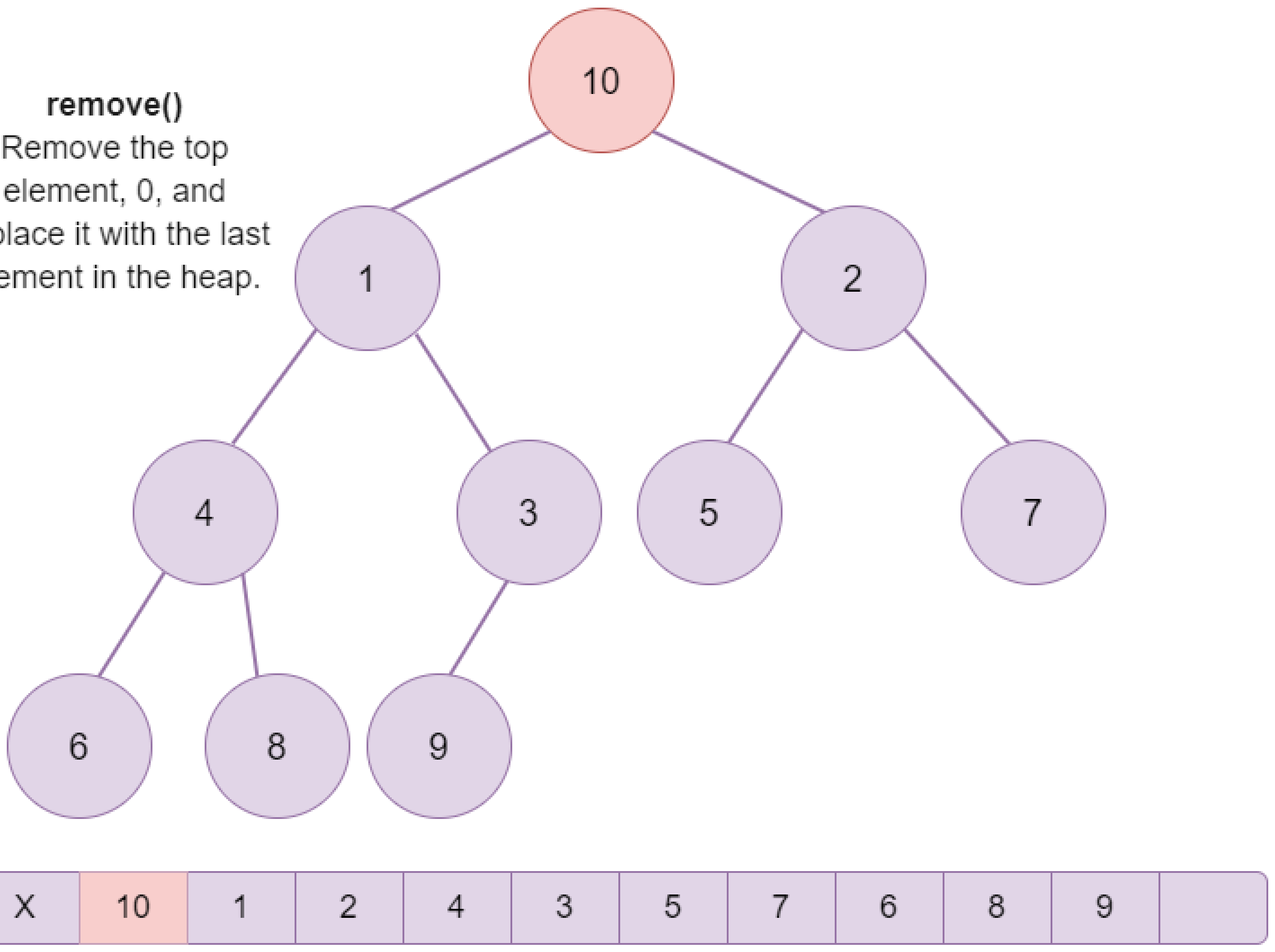


remove() Example



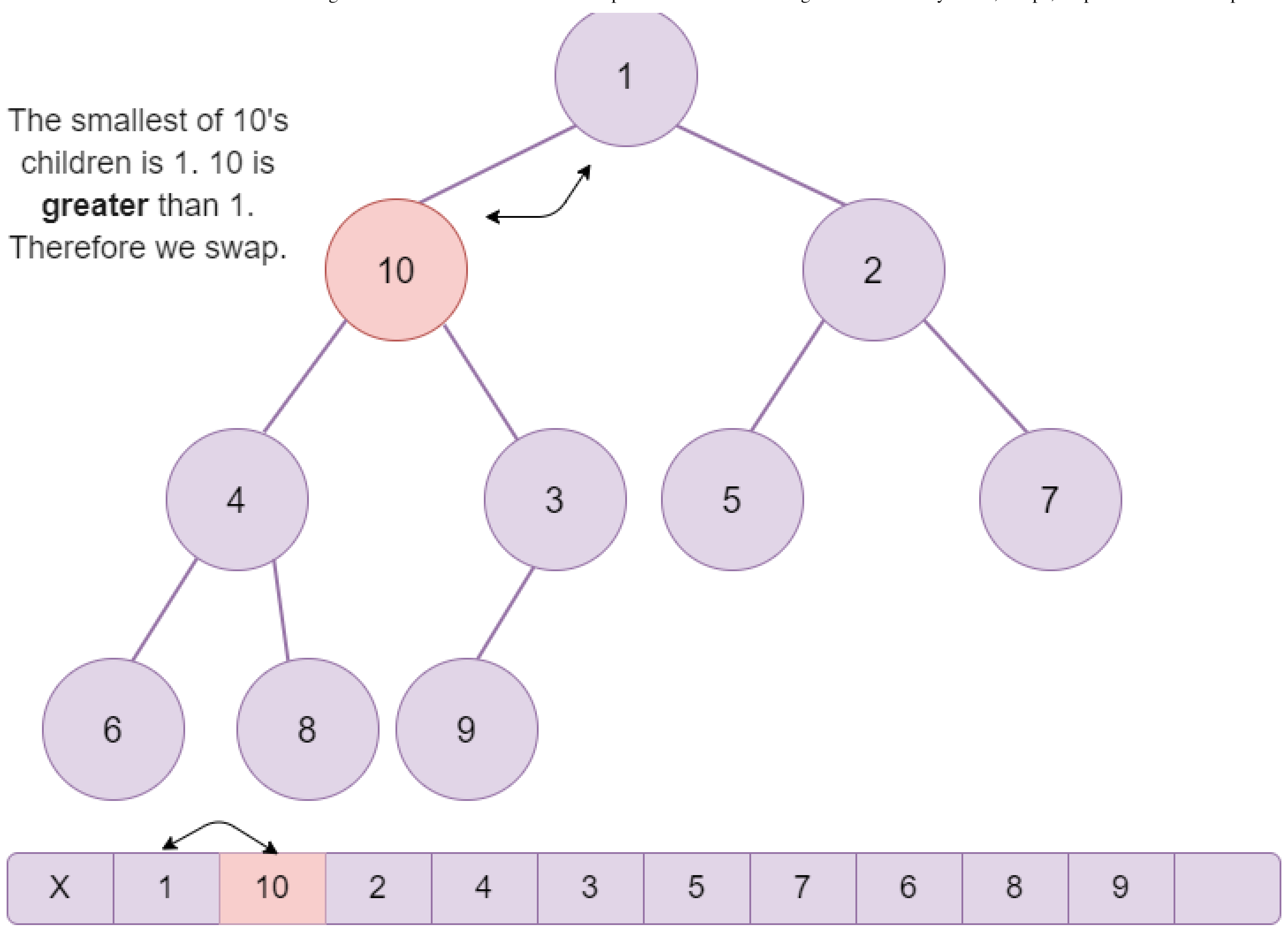
X	0	1	2	4	3	5	7	6	8	9	10

remove()
Remove the top element, 0, and replace it with the last element in the heap.

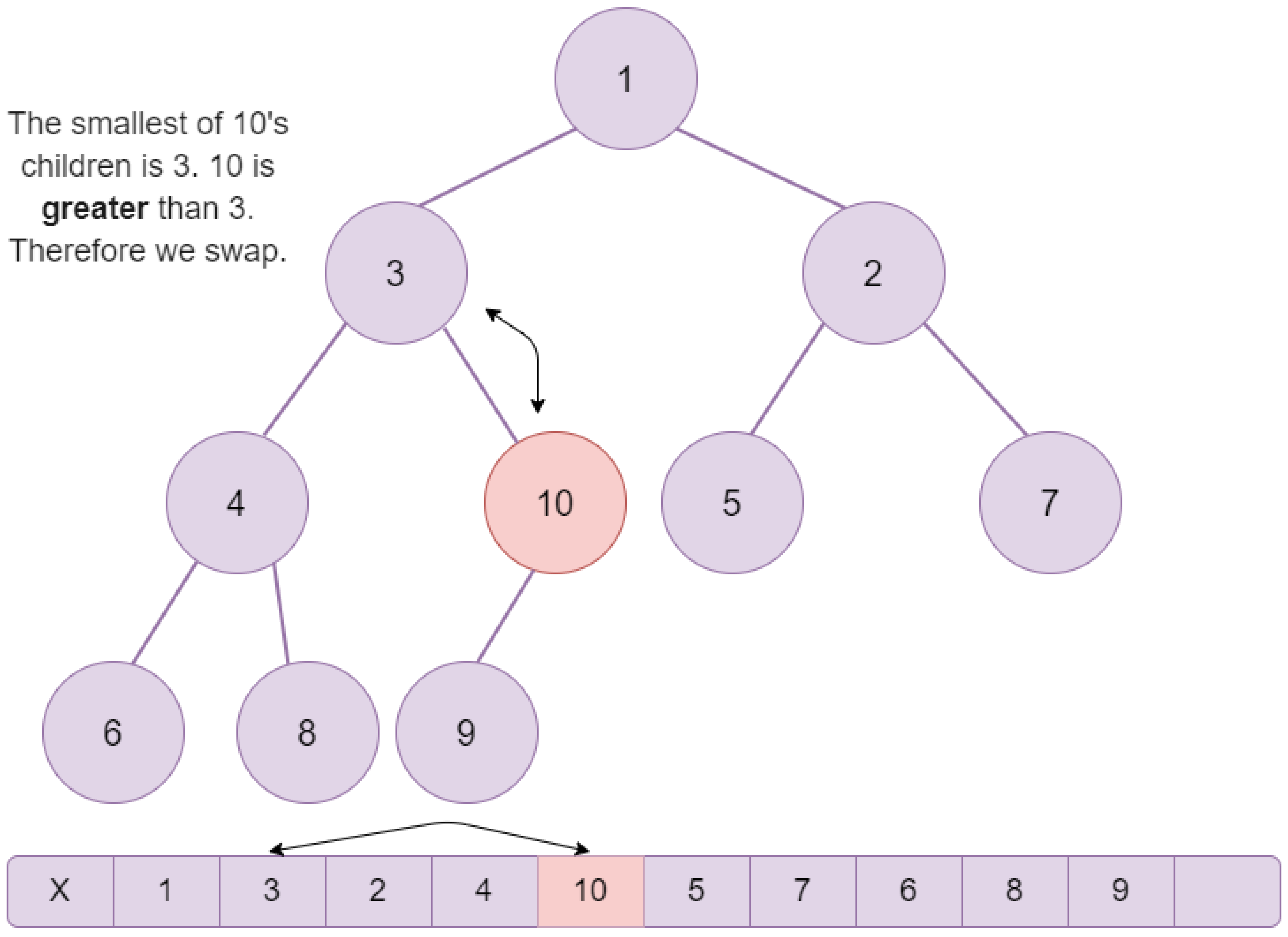


X	10	1	2	4	3	5	7	6	8	9	

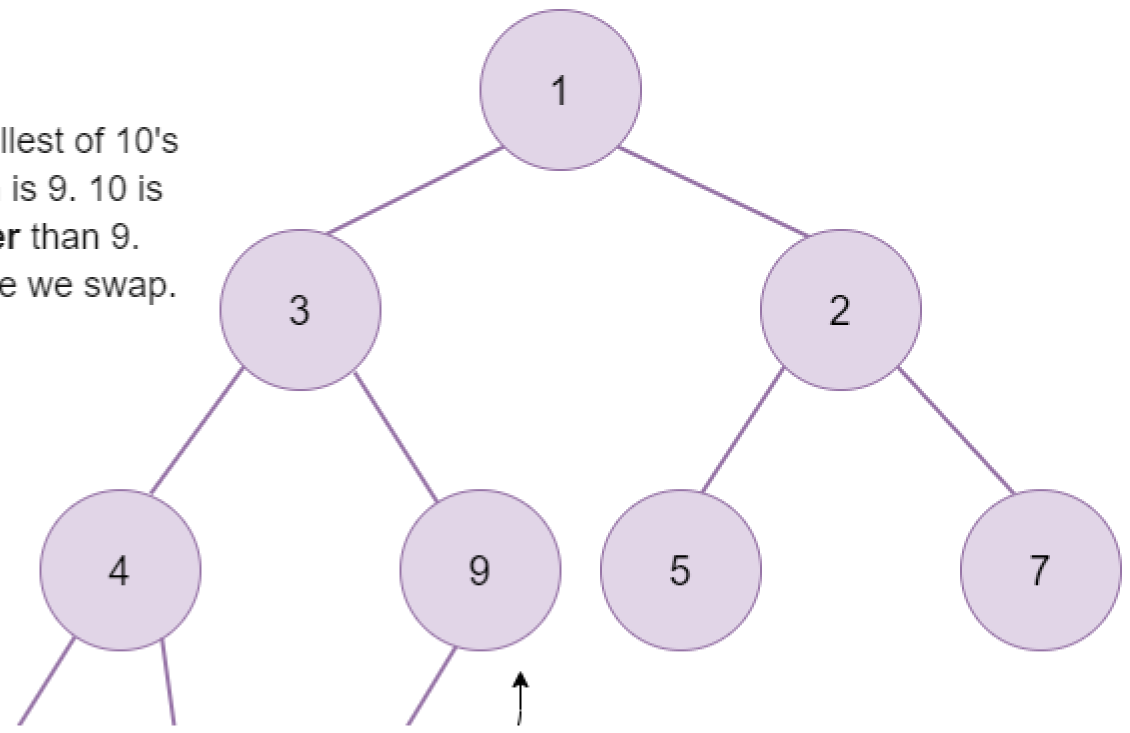
The smallest of 10's children is 1. 10 is **greater** than 1. Therefore we swap.

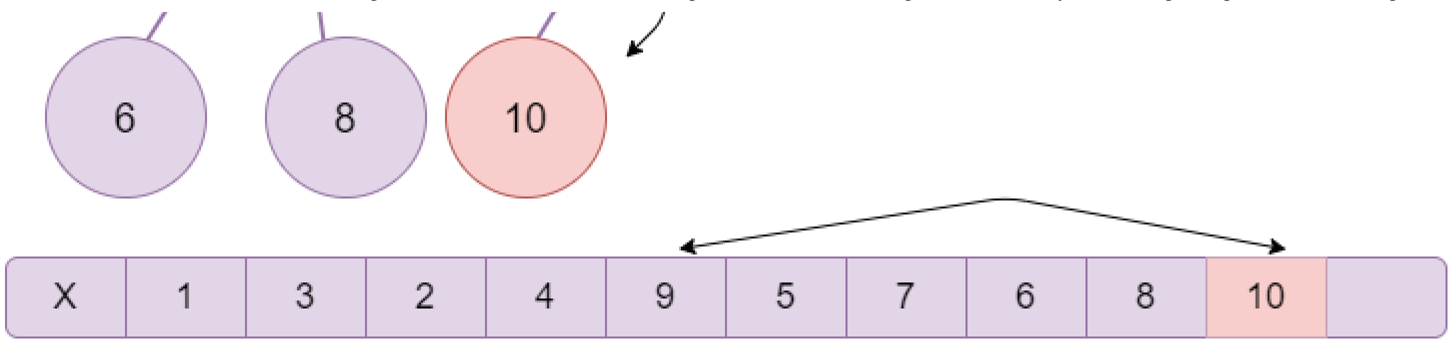


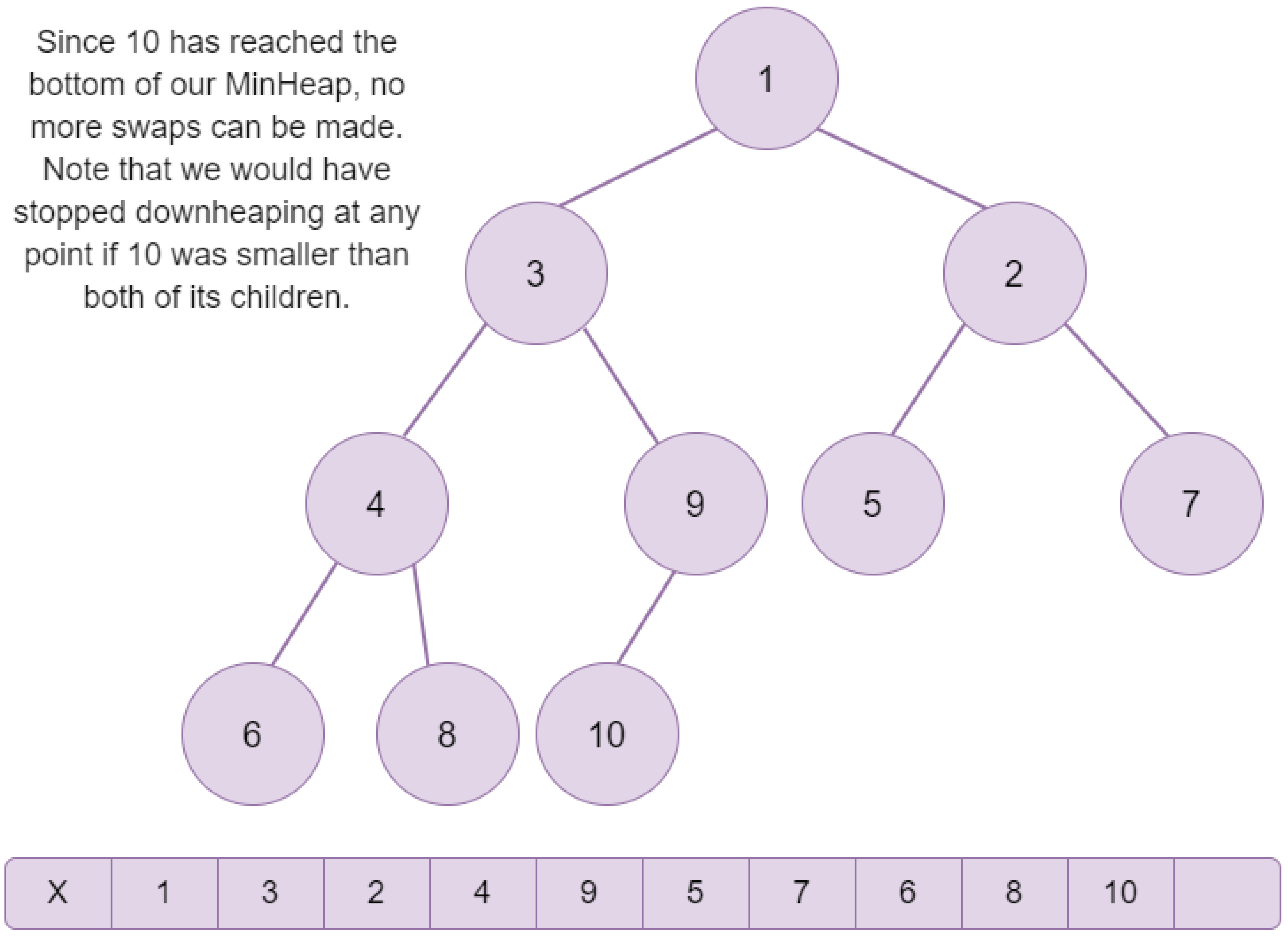
The smallest of 10's children is 3. 10 is **greater** than 3. Therefore we swap.



The smallest of 10's children is 9. 10 is **greater** than 9. Therefore we swap.






Since 10 has reached the bottom of our MinHeap, no more swaps can be made. Note that we would have stopped downheaping at any point if 10 was smaller than both of its children.

General Tips

- In a MinHeap containing *n* elements, only the first n / 2 elements will have children. How can you utilize this important property when upheaping and downheaping elements in your MinHeap?
- We highly recommend copying the starter code and working in your preferred IDE in order to have access to features such as code completion, auto-formatting, and much more!

Here are general assignment guidelines that should be followed.

- Do not include any package declarations in your classes.
- Do not change any existing class headers, constructors, instance/global variables, or method signatures. For example, do not add throws to the method headers since they are not necessary. Instead, exceptions should be thrown as follows: `throw new InsertExceptionHere("Error: some exception was thrown");`
- All helper methods you choose to write should be made private. Recall the idea of Encapsulation in Object-Oriented Programming!
- Do not use anything that would trivialize the assignment. (e.g. Don't import/use java.util.ArrayList for an ArrayList assignment.)
- Always be very conscious of efficiency. Even if your method is to be $O(n)$, traversing the structure multiple times is considered inefficient unless that is absolutely required (and that case is extremely rare).
- If applicable, use the generic type of the class; do not use the raw type of the class. For example, use `new LinkedList<Integer>()` instead of `new LinkedList()`.

Use of the following statements should be avoided at all times.

package	System.arraycopy()	clone()
assert()	Arrays class	Array class
Thread class	Collections class	Collection.toArray()
Reflection APIs	Inner or nested classes	Lambda Expressions

The Vocareum (code editor) interface has six main components:

- The **Drop-Down** in the top left. This lets you choose from multiple available files. Note that this drop-down will only be visible in assignments that require multiple files.
- The **Run** button. This will compile your code and run a file scan. Running your code will not count towards your total allowed submission attempts, therefore you are free to run as many times as needed.
- The **Submit** button. This will compile your code, run a file scan, grade your assignment, and output results to console. Note that for most assignments in this class, you will only be allowed a limited number of submissions. A submission is counted when the submit button is clicked, regardless of whether or not your code can compile or if there are any file issues. Therefore, we **highly recommend** that you run your code before submitting to ensure that there are no issues that will prevent your code from being graded and that every submission attempt will generate meaningful results.
- The **Reset** button. This will revert all your changes and reset your code to the default code template.
- The **Code Window**. This is where you will write your code. For large coding assignments, we highly recommend copying the starter code and working in your preferred IDE to have access to features such as code completion, auto-formatting, and much more!
- The **Output Window**. This window will appear whenever you run or submit your code and will display the output for you to view.

For additional help, please visit the Vocareum information page located in the course information module!

Module 6 Assignment: MinHeap (External resource) (10.0 points possible)

Submit Run Grades Reset

```java
import java.util.NoSuchElementException;

/**
 * Your implementation of a MinHeap.
 */
public class MinHeap<T extends Comparable<? super T>> {

    /**
     * The initial capacity of the MinHeap.
     *
     * DO NOT MODIFY THIS VARIABLE!
     */
    public static final int INITIAL_CAPACITY = 13;

     /*
     * Do not add new instance variables or modify existing ones.
     */
    private T[] backingArray;
    private int size;

    /**
     * This is the constructor that constructs a new MinHeap.
     *
     * Recall that Java does not allow for regular generic array creation,
     * so instead we cast a Comparable[] to a T[] to get the generic typing.
     */
    public MinHeap() {
        //DO NOT MODIFY THIS METHOD!
        backingArray = (T[]) new Comparable[INITIAL_CAPACITY];
    }

    /**
     * Adds an item to the heap. If the backing array is full (except for
     * index 0) and you're trying to add a new item, then double its capacity.
     *
     * Method should run in amortized O(log n) time.
```

Console output will be displayed here

< Previous Next >