Heap Lab.

Dr. Tanvir Ahmed

Attend the lab and complete it with the TA! No need to keep working after the lab if you can just follow and work actively during the lab.

Is it a Heap?

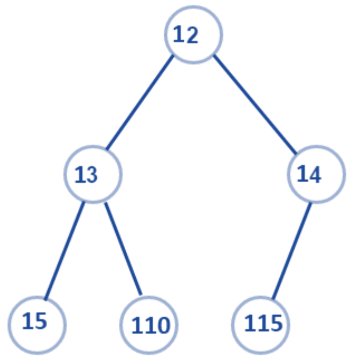
In this lab we will write a code that will check whether an array represents a minheap or not.

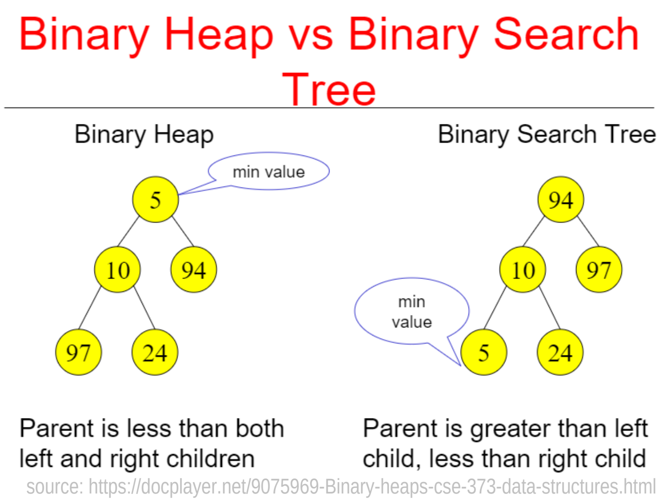
For example:

Is this array representing a minheap?

12	13	14	15	110	115

Difficult to say, without looking it like a tree. Let's see the tree:




Binary Heap vs Binary Search Tree
Binary Heap
Binary Search Tree
min value
5
10
94
97
24
94
10
97
min value
5
24
Parent is less than both left and right children
Parent is greater than left child, less than right child
source: https://docplayer.net/9075969-Binary-heaps-cse-373-data-structures.html

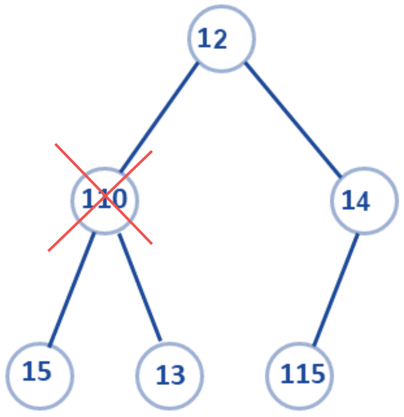
Yes, it seems the array represents a minheap as for any node, all the nodes of the sub tree is smaller than the data at that node! Additionally, it is also a complete binary tree

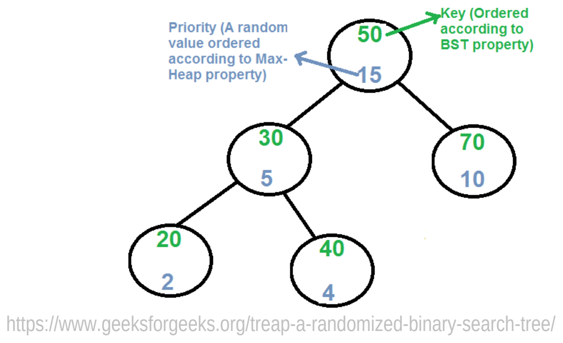
How about this one?

12	110	14	15	13	115

Let's see the tree:

For the curious:
there's a really neat structure that combines binary trees and heaps called TREAP
(will likely learn it in CS2)




Priority (A random value ordered according to Max-Heap property)
Key (Ordered according to BST property)
50
15
30
5
70
10
20
2
40
4
https://www.geeksforgeeks.org/treap-a-randomized-binary-search-tree/

However, this is not a minheap has 110 > its children (15 and 13)

Now if we want to write a code to check this, we have to keep checking from the root and compere it with the children. In our implementation of heap, we have started with the array index 1 and put kept our root in the index 1. However, a given array will start with 0 and we have to modify our formula to accommodate that.

So, in this case, as the first item is located at index 0 and we are considering that item as the root of the heap, it's left child will be at index 1 and right child will be at index 2.

So, generalizing, left child will be at 2*i+1 and right child will be at 2*i+2

For this type of implementation where root starts at index 0, the parent of a node at index i will be at

(i-1)/2

We will try to write a recursive version and one iterative version.

For a recursive version the function prototype would be:

Int isHeap_recursive(int arr[], int I, int n); //i is the starting point of the array

The main idea is to compare the left child with the parent and then right child with the parent and keep checking until it satisfies the properly. If it does not satisfy at any point, return 0. If it keeps satisfying the property and if you see you have reached to the nodes and your code has not returned false, it means it is really a min heap.

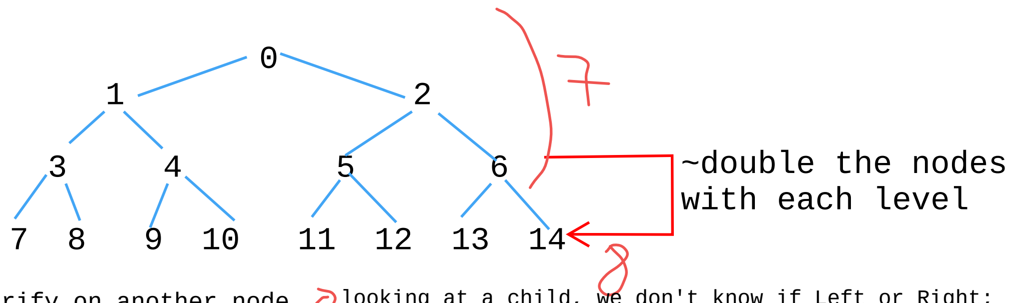
Try this!

Also try an iterative approach.

The function prototype would be:

```
int isHeap_iter(int arr[],  int n);
```

More hints: the loop will be like this: for (int i=0; i<=(n-2)/2; i++) //start from root and end at last parent node. Whey (n-2)/2? Because last index is n-1. So, parent is at (n-2)/2



```
try to find the pattern, then verify on another node
parent 5, left-child 11=2*5+1, right child 12=2*5+2
potential formula: left=parent*2+1, right=parent*2+2
verify with parent 4: left 4*2+1=9, right 4*2+2=10

now to go backwards, isolate parent in the formula:
left=parent*2+1  --> (left-1)/2 = parent
right=parent*2+2 --> (right-2)/2 = parent
```

```
looking at a child, we don't know if Left or Right;
only one child is even, other is odd, so either child-1 is even
or child-2 is even, producing an int when divided by 2

subtracting an extra 1 and dividing by 2 can shift us by 0.5;
we need int index so we'll take the floor of that (i.e. round
down): shift=1.0, and now we'd be at the wrong parent.

if we don't subtract enough, floor will fix it; so:
Left: floor((child-1)/2) gives correct int parent regardless
Right: floor((child-1)/2) gives correct int parent because floor
```