Jack Vestberg

Professor Das

CS-222-01: Programming II

May 18 2024

**Assignment #3 report**

For this assignment we were asked to implement a binary heap using Java. There was an option to either do an array-based implementation or a linked-list implementation and the same goes for if the project was a max or min binary heap. For my project I chose to do an array-based max heap. The reason for this choice was because we have in class focused more on linked list implementations, so I wanted to try using an array instead and see if I could do it. For a max heap it must maintain its max heap property, which is that the root should always be the maximum element. The binary heap also needs to maintain a balanced binary tree structure, so that the tree doesn’t become unbalanced. This report outlines the design, implementation, testing process, and challenges faced during the creation of an array-based implementation of a max binary heap data structure.

For my array-based max binary heap I used the indexes of the array to represent the binary tree, I used some functions to find the parent, left and right nodes. This was the formula for the parent node of i: (i1)+1, this is for the left child: 2\*i+1 and finally this is the formula for the right child: 2\*i+1. I also created a method isLeaf which returns a boolean whether the node is a leaf node or not. I have all the functions that are supposed to be callable by the user as public and the other “helper” functions set as private because they are only meant to be used in other functions. In order to keep the max heap properties I created a swap and maxHeapify method.The swap method saps positions of two nodes in the array, the maxHeapify method is a recursive function. That function is utilizing earlier methods such as swap and isLeaf to properly keep the max heap property. This recursion continues either until we hit a leaf node or that we satisfy the max heap property. For the insert method it takes an element as a parameter to be inserted and it puts the element in the last position in the tree to keep it balanced by the help of the variable size. The method later traverses up the tree in order to check that the max heap property isn’t violated, if the va;ue is greater than its parent then it swaps it and keeps going, when its done it increases the value of size in order to show the adding of an element. For the exctracMax method it removes the root from the binary tree and returns that value. Because the maximum value is the root it will remove the root of the tree, it will do this and replace the root with the last element in the heap in order to keep the tree balanced. When that is done it calls the maxHeapify method that corrects the array to keep the max array property. I also added a function printArray just to get the array printed in the console for testing purposes. The last methods I added were the display and displayRecursive methods. DisplayRecursive is a recursive helper method for displaying the heap, when implementing this I looked at previous display methods we used for our unbalanced tree. The functions visualize the tree from left to right.

For testing I often used print statements in methods in order to see if the method worked as I wanted or if not how it was working. I also used the function printArray in order to see that the array was updated correctly. The biggest problem that I encountered for this project was when I wanted to implement a visualization that printed the tree top down instead of left to right. I started by creating a function to calculate the height of the heap to now how many tab spaces there should be before the root so that the tree looks fine. Then after that I wanted to kind of for every element or value in the heap also keep track of the height it should be inputted and at what level. Height in this context is how far up or down vertically, and level is left to right where the root is in the middle with 0 and every element to the left or right is +1 or -1 to that level variable. I tried to use mapping in order to do this, but in the end I failed.

In conclusion, after several hours I was able to finish the code for an array-based implementation of max heaps, unfortunately not with the visualize function to print the tree vertically. It was however very helpful to try and look at problems differently (other approaches our trying to visualize it yourself by drawing) and not just sit staring at the code hoping for an epiphany.

**Github Repo**

https://github.com/JakkV/MaxHeap