Percolate UP

This code is an implementation of the "percolate up" operation for a max heap data structure. A max heap is a binary tree where the value of each node is greater than or equal to the value of its children, and the root node contains the largest value in the heap.

The "percolate up" operation is used when a new value is added to the heap. The new value is initially placed at the bottom of the heap, and then "percolated up" by swapping it with its parent node until it reaches a position where it is larger than its parent.

The input to the function is the index of the node to percolate up (nodeIndex) and the array that represents the heap (heapArray). The function starts by checking if the node is already at the root of the heap (nodeIndex > 0). If it is not, the function calculates the index of the parent node using the formula parentIndex = (nodeIndex - 1) / 2.

If the value at the current node is less than or equal to the value at its parent node (heapArray[nodeIndex] <= heapArray[parentIndex]), then the function returns without making any changes to the heap.

If the value at the current node is greater than the value at its parent node, then the function swaps the values at the current node and its parent node (swap heapArray[nodeIndex] and heapArray[parentIndex]) to move the larger value up the tree. After the swap, the function updates nodeIndex to be the index of the parent node, so that the process can continue until the new node is in its correct position in the heap.

Overall, this function implements the percolate up operation for a max heap, which is a crucial operation for maintaining the heap property and ensuring that the maximum value is always at the root of the heap.

Percolate DOWN

This code is an implementation of the "percolate down" operation for a max heap data structure. A max heap is a binary tree where the value of each node is greater than or equal to the value of its children, and the root node contains the largest value in the heap.

The "percolate down" operation is used when the root node is removed from the heap. In this case, the last node in the heap is moved to the root position, and then "percolated down" by swapping it with its larger child node until it reaches a position where it is larger than both of its children.

The input to the function is the index of the node to percolate down (nodeIndex), the array that represents the heap (heapArray), and the size of the heap array (arraySize). The function starts by calculating the index of the left child of the node (childIndex) using the formula childIndex = 2 \* nodeIndex + 1, and storing the value at the current node in a variable called value.

The function then enters a while loop that continues as long as the current node has at least one child node (childIndex < arraySize). Within the while loop, the function iterates over the two children nodes of the current node (if they exist), and finds the maximum value among the current node and its children nodes. This is done using a for loop that iterates over the two children nodes and checks if their values are greater than the value of the current node. If a child node has a larger value than the current node, then the maximum value and its index (maxValue and maxIndex) are updated accordingly.

After finding the maximum value among the current node and its children, the function checks if the maximum value is equal to the value at the current node. If it is, then the function returns without making any changes to the heap, because the heap property is already satisfied.

If the maximum value is greater than the value at the current node, then the function swaps the values at the current node and the maximum value node (swap heapArray[nodeIndex] and heapArray[maxIndex]) to move the larger value down the tree. After the swap, the function updates nodeIndex to be the index of the maximum value node, and recalculates the index of the left child of the new current node (childIndex = 2 \* nodeIndex + 1). The while loop continues until the current node no longer has any child nodes that are larger than it, and the heap property is restored.

Overall, this function implements the percolate down operation for a max heap, which is a crucial operation for maintaining the heap property and ensuring that the maximum value is always at the root of the heap.