MaxHeap

v2025-05-04

Content

heapifyDown

```C-like
void heapifyDown(int heap[], int size, int index) {
    int leftChild, rightChild, smallest, temp;

    while (index < size) {
        leftChild = 2 * index + 1;   // Index of left child
        rightChild = 2 * index + 2; // Index of right child
        smallest = index;           // Assume the current node is the smallest

        // Check if left child exists and is smaller
        if (leftChild < size && heap[leftChild] < heap[smallest]) {
            smallest = leftChild;
        }

        // Check if right child exists and is smaller
        if (rightChild < size && heap[rightChild] < heap[smallest]) {
            smallest = rightChild;
        }

        // If the current node is already in the correct position, break
        if (smallest == index) {
            break;
        }

        // Swap the current node with the smallest child
        temp = heap[index];
        heap[index] = heap[smallest];
        heap[smallest] = temp;

        // Move down to the smallest child's index
        index = smallest;
    }
}
```

where - heap[]: The array representing the heap. - size: The current size of the heap. - index: The index of the node to heapify down.

heapifyUp

C-like

```
void heapifyUp(int heap[], int index) {
    int parent, temp;

    while (index > 0) {
        parent = (index - 1) / 2; // Index of the parent node

        // If the current node is smaller than its parent, swap them
        if (heap[index] < heap[parent]) {
            temp = heap[index];
            heap[index] = heap[parent];
            heap[parent] = temp;

            // Move up to the parent's index
            index = parent;
        } else {
            // If the current node is not smaller, the heap property is restored
            break;
        }
    }
}
```