Practice 03:

Implementation of Heap Sort

Code:

**package** test;

**public** **class** HeapSort {

**public** **void** sort(**int** arr[])

{

**int** n = arr.length;

// Build heap (rearrange array)

**for** (**int** i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

// One by one extract an element from heap

**for** (**int** i = n - 1; i > 0; i--) {

// Move current root to end

**int** temp = arr[0];

arr[0] = arr[i];

arr[i] = temp;

// call max heapify on the reduced heap

heapify(arr, i, 0);

}

}

// To heapify a subtree rooted with node i which is an index in arr[].

**void** heapify(**int** arr[], **int** n, **int** i)

{

**int** largest = i;

**int** l = 2 \* i + 1;

**int** r = 2 \* i + 2;

// If left child is larger than root

**if** (l < n && arr[l] > arr[largest])

largest = l;

// If right child is larger than largest so far

**if** (r < n && arr[r] > arr[largest])

largest = r;

// If largest is not root

**if** (largest != i) {

**int** swap = arr[i];

arr[i] = arr[largest];

arr[largest] = swap;

// Recursively heapify the affected sub-tree

heapify(arr, n, largest);

}

}

// A function to print array

**static** **void** printArray(**int** arr[])

{

**int** n = arr.length;

**for** (**int** i = 0; i < n; ++i)

System.***out***.print(arr[i] + " ");

System.***out***.println();

}

// Main function

**public** **static** **void** main(String args[])

{

**int** arr[] = { 12, 11, 13, 5, 6, 7 };

**int** n = arr.length;

HeapSort ob = **new** HeapSort();

ob.sort(arr);

System.***out***.println("Sorted array is");

*printArray*(arr);

}

}