HEAP-SORT

import java.util.Arrays;

class GfG {

// To heapify a subtree rooted with node i

// which is an index in arr[].

static void heapify(int arr[], int n, int i) {

// Initialize largest as root

int largest = i;

// left index = 2\*i + 1

int l = 2 \* i + 1;

// right index = 2\*i + 2

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 temp = arr[i];

arr[i] = arr[largest];

arr[largest] = temp;

// Recursively heapify the affected sub-tree

heapify(arr, n, largest);

}

}

// Main function to do heap sort

static void heapSort(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);

}

}

// A utility function to print array of size n

static void printArray(int arr[]) {

for (int i = 0; i < arr.length; i++) {

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

}

System.out.println();

}

// Driver's code

public static void main(String args[]) {

int arr[] = {9, 4, 3, 8, 10, 2, 5};

heapSort(arr);

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

printArray(arr);

}

}