A document with text and numbers

Description automatically generated

A screenshot of a computer

Description automatically generated

A white paper with black text

Description automatically generated

Heapsort:

public class Heapsort{

    public static void BuildMaxHeap(int[] A){

        int heapSize = A.length;

        for(int i = heapSize/2; i >= 0; i--){

            MaxHeapify(A, i, heapSize);

        }

    }

    public static void MaxHeapify(int[] A, int i, int heapSize){

        int l = 2\*i+1;

        int r = 2\*i+2;

        int largest;

        if(l < heapSize && A[l] > A[i]){

            largest = l;

        }else{

            largest = i;

        }

        if(r < heapSize && A[r] > A[largest]){

            largest = r;

        }

        if(largest != i){

            int temp = A[i];

            A[i] = A[largest];

            A[largest] = temp;

            MaxHeapify(A, largest, heapSize);

        }

    }

    public static void Heapsort(int[] A){

        int heapSize = A.length;

        BuildMaxHeap(A);

        for(int i = A.length-1; i >= 1; i--){

            int temp = A[0];

            A[0] = A[i];

            A[i] = temp;

            heapSize--;

            MaxHeapify(A, 0, heapSize);

        }

    }

    public static void main(String[] args){

        int[] A = {4, 1, 3, 2, 16, 9, 10, 14, 8, 7};

        Heapsort(A);

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

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

        }

    }

}

Output:

A screenshot of a computer program

Description automatically generated

Heap sort optimization:

public class HeapSortOptimization {

    public static void BuildMaxHeap(int[] A) {

        int heapSize = A.length;

        for (int i = heapSize / 2 - 1; i >= 0; i--) { // Iterate down from the last non-leaf node

            MaxHeapify(A, i, heapSize);

        }

    }

    public static void MaxHeapify(int[] A, int i, int heapSize) {

        int largest = i;

        int l = 2 \* i + 1;

        int r = 2 \* i + 2;

        if (l < heapSize && A[l] > A[largest]) {

            largest = l;

        }

        if (r < heapSize && A[r] > A[largest]) {

            largest = r;

        }

        if (largest != i) {

            int temp = A[i];

            A[i] = A[largest];

            A[largest] = temp;

            MaxHeapify(A, largest, heapSize);

        }

    }

    public static void Heapsort(int[] A) {

        int heapSize = A.length;

        BuildMaxHeap(A);

        for (int i = A.length - 1; i >= 1; i--) {

            int temp = A[0];

            A[0] = A[i];

            A[i] = temp;

            heapSize--;

            MaxHeapify(A, 0, heapSize);

        }

    }

    public static void main(String[] args) {

        int[] A = {4, 1, 3, 2, 16, 9, 10, 14, 8, 7};

        Heapsort(A);

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

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

        }

    }

}

Output:

A screenshot of a computer program

Description automatically generated

ALL CODE WORKS PROPERLY!!