1 /\*\*  
 2 \* @author Amar Bessedik   
 3 \* This class implements the heapSort algorithm. It is adapted to sort an array   
 4 \* of n Edges in increasing order of weight.  
 5 \*/  
 6 public class HeapSort  
 7 {  
 8 /\*\*  
 9 \* Constructor  
 10 \*/  
 11 public HeapSort(){}  
 12   
 13   
 14   
 15 /\*\*  
 16 \* @param E and array of graph edges  
 17 \* @param n length of the array E  
 18 \*/  
 19 public void heapSort(Edge[] E, int n)  
 20 {  
 21 //Build a heap where root is the largest  
 22 buildHeap(E, n);  
 23 //Extract edges from heap one by one  
 24 for (int i = n - 1; i >= 0; i--)  
 25 {  
 26 //Swap root (largest weight) to the end of array  
 27 swap(E, 0, i);  
 28 //Rebuild the reduced heap.  
 29 heapify(E, 0, i);  
 30 }  
 31 }//end heapSort  
 32   
 33 /\*\*  
 34 \* @param E array of a graph edges  
 35 \* @param n length of the array E  
 36 \*/  
 37 private void buildHeap(Edge[] E, int n)  
 38 {  
 39 for (int i = n/2; i >= 0; i--)  
 40 {  
 41 heapify(E, i, n);  
 42 }  
 43 }//end heapify  
 44

45 /\*\*  
 46 \* @param E array of a graph edges  
 47 \* @param i index from which start make a heap.  
 48 \* @param n length of the array E.  
 49 \*/  
 50 private void heapify(Edge[] E, int i, int n)  
 51 {  
 52 int largest = i;// Initialize largest as i  
 53 int l = 2\*i + 1; //left child  
 54 int r = 2\*i + 2; //right child  
 55   
 56 // If left child's weight is greater than root's weight  
 57 if (l < n && E[l].getWeight() > E[largest].getWeight())  
 58 {  
 59 largest = l;  
 60 }  
 61 // If right child's weight is greater than root's weight  
 62 if (r < n && E[r].getWeight() > E[largest].getWeight())  
 63 {  
 64 largest = r;  
 65 }  
 66 // If largest is not i  
 67 if (largest != i)  
 68 {  
 69 //swap i with largest  
 70 swap(E, i, largest);  
 71 //heapify the reduced heap recursively  
 72 heapify(E, largest, n);  
 73 }  
 74 }//end heapify  
 75   
 76 //=========== HELPER METHODS ===============  
 77 /\*\*  
 78 \* @param E array of a graph edges  
 79 \* @param i the edge at index i goes to index j.  
 80 \* @param j the edge at index j goes to index i.  
 81 \*/  
 82 private void swap(Edge[] E, int i, int j)  
 83 {  
 84 Edge tmp = E[i];  
 85 E[i] = E[j];  
 86 E[j] = tmp;  
 87 }//end swap  
 88 }//end class  
 89