Nama : Andyan Yogawardhana

NIM : 21/482180/PA/21030

Kelas : KOMB1

Tugas 5 – Heap Sort

Source Code

1. public class Main {
2. public static void main(String[] args) {
3. int[] key = {78, 3, 9, 10, 23, 77, 34, 86, 90, 100, 20, 66, 94, 63, 97};
4. Heap heap = new Heap(15);
5. for(int i = 0; i < key.length; i++) {
6. heap.insert(key[i]);
7. }
8. heap.heapSort();
9. }
10. }
11. class Node {
12. private int data;
13. public Node(int d) {
14. this.data = d;
15. }
16. public int getData() {
17. return this.data;
18. }
19. public void setData(int d) {
20. this.data = d;
21. }
22. }
23. class Heap {
24. private Node[] array;
25. private int maxSize, currentSize;
26. public Heap(int max) {
27. maxSize = max;
28. currentSize = 0;
29. array = new Node[maxSize + 1];
30. }
31. public boolean isFull() {
32. return currentSize == maxSize;
33. }
34. public boolean hasLeftChild(int i) {
35. return 2 \* i <= currentSize;
36. }
37. public boolean hasRightChild(int i) {
38. return 2 \* i + 1 <= currentSize;
39. }
40. public boolean insert(int data) {
41. if(isFull()) {
42. return false;
43. }
44. array[++currentSize] = new Node(data);
45. trickleUp(currentSize);
47. return true;
48. }
49. public void trickleUp(int i) {
50. int parent = i / 2;
51. Node bottom = array[i];
52. while(i > 1 && array[parent].getData() < bottom.getData()) {
53. array[i] = array[parent];
54. i = parent;
55. parent = i / 2;
56. }
57. array[i] = bottom;
58. }
59. public Node remove() {
60. Node root = array[1];
61. array[1] = array[currentSize--];
62. trickleDown(1);
64. return root;
65. }
66. public void trickleDown(int i) {
67. Node  top = array[i];
68. int largerChild;
70. while (hasLeftChild(i)) {
71. int leftChild = 2 \* i;
72. int rightChild = leftChild + 1;
74. if (hasRightChild(i) && array[rightChild].getData() > array[leftChild].getData()) {
75. largerChild = rightChild;
76. }
77. else {
78. largerChild = leftChild;
79. }
81. if (top.getData() >= array[largerChild].getData()) {
82. break;
83. }
85. array[i] = array[largerChild];
86. i = largerChild;
87. }
89. array[i] = top;
90. }
91. public void heapSort() {
92. for(int i = currentSize; i >= 1; i--) {
93. Node max = remove();
94. array[currentSize + 1] = max;
95. }
96. displayHeap();
97. }
99. public void displayHeap() {
100. System.out.println();
101. System.out.println("Sorted Heap Tree (Ascending): ");
102. for(int i = 1; i <= maxSize; i++) {
103. System.out.print(array[i].getData() + " ");
104. }
105. System.out.println("\n");
107. System.out.println("Sorted Heap Tree (Descending): ");
108. for(int i = maxSize; i >= 1; i--) {
109. System.out.print(array[i].getData() + " ");
110. }
111. System.out.println("\n");
112. }
113. }

Output Terminal

