

GAIRE ANANTA PRASAD

M24W0272

CODE FOR MAX-HEAP

```
MaxHeap.java > MaxHeap > MaxHeap(int)
1  /*
2   * GAIRE ANANTA PRASAD
3   * M24W0272
4   */
5  public class MaxHeap
6  {
7      private int[] Heap;
8      private int size;
9      private int maxsize;
10
11     private static final int FRONT = 1;
12
13     public MaxHeap(int maxsize)
14     {
15         this.maxsize = maxsize;
16         this.size = 0;
17         Heap = new int[this.maxsize + 1];
18         Heap[0] = Integer.MAX_VALUE;
19     }
20
21     private int parent(int pos)
22     {
23         return pos / 2;
24     }
25
26     private int leftChild(int pos)
27     {
28         return (2 * pos);
29     }
30
31     private int rightChild(int pos)
32     {
33         return (2 * pos) + 1;
34     }
35
36     private boolean isLeaf(int pos)
37     {
38
39     }
```

```
MaxHeap.java > MaxHeap > MaxHeap(int)
40     public class MaxHeap
41     {
42         private boolean isLeaf(int pos)
43         {
44             if (pos >= (size / 2) && pos <= size)
45             {
46                 return true;
47             }
48             return false;
49         }
50
51         private void swap(int fpos,int spos)
52         {
53             int tmp;
54             tmp = Heap[fpos];
55             Heap[fpos] = Heap[spos];
56             Heap[spos] = tmp;
57         }
58
59         private void maxHeapify(int pos)
60         {
61             if (!isLeaf(pos))
62             {
63                 if (Heap[pos] < Heap[leftChild(pos)] || Heap[pos] < Heap[rightChild(pos)])
64                 {
65                     if (Heap[leftChild(pos)] > Heap[rightChild(pos)])
66                     {
67                         swap(pos, leftChild(pos));
68                         maxHeapify(leftChild(pos));
69                     }else
70                     {
71                         swap(pos, rightChild(pos));
72                         maxHeapify(rightChild(pos));
73                     }
74                 }
75             }
76         }
77     }
```

Gaire Ananta Prasad

M24W0272

```

NormalDistributionTable.java KnuthShellSort.java HibbardShellSort.java SedgewickShellSort.java MinHeap.java U MaxHea
MaxHeap.java > MaxHeap > MaxHeap(int)
public class MaxHeap

    public void insert(int element)
    {
        Heap[++size] = element;
        int current = size;

        while(Heap[current] > Heap[parent(current)])
        {
            swap(current,parent(current));
            current = parent(current);
        }
    }

    public void print()
    {
        for (int i = 1; i <= size / 2; i++ )
        {
            System.out.print(" PARENT : " + Heap[i] + " LEFT CHILD : " + Heap[2*i]
            | + " RIGHT CHILD : " + Heap[2 * i + 1]);
            System.out.println();
        }
    }

    public void maxHeap()
    {
        for (int pos = (size / 2); pos >= 1; pos--)
        {
            maxHeapify(pos);
        }
    }

    public int remove()
    {
        int popped = Heap[FRONT];
        Heap[FRONT] = Heap[size--];
        maxHeapify(FRONT);
    }

```

Ln 16, Col 23 Spaces: 4 UTF-8

```

NormalDistributionTable.java KnuthShellSort.java HibbardShellSort.java SedgewickShellSort.java MinHeap.java U
MaxHeap.java > MaxHeap > main(String...)
5 public class MaxHeap
102 public int remove()
103 {
104     int popped = Heap[FRONT];
105     Heap[FRONT] = Heap[size--];
106     maxHeapify(FRONT);
107     return popped;
108 }
109
Run | Debug
110 public static void main(String...arg)
111 {
112     System.out.println("The Max Heap is ");
113     MaxHeap maxHeap = new MaxHeap(maxsize:15);
114     maxHeap.insert(element:5);
115     maxHeap.insert(element:3);
116     maxHeap.insert(element:17);
117     maxHeap.insert(element:10);
118     maxHeap.insert(element:84);
119     maxHeap.insert(element:19);
120     maxHeap.insert(element:6);
121     maxHeap.insert(element:22);
122     maxHeap.insert(element:9);
123     maxHeap.maxHeap();
124
125     maxHeap.print();
126     System.out.println("The max val is " + maxHeap.remove());
127 }
128
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
PARENT : 84 LEFT CHILD : 22 RIGHT CHILD :19
PARENT : 22 LEFT CHILD : 17 RIGHT CHILD :10
PARENT : 19 LEFT CHILD : 5 RIGHT CHILD :6
PARENT : 17 LEFT CHILD : 3 RIGHT CHILD :9
The max val is 84
PS D:\kcg\Java>

```

Gaire Ananta Prasad
M24W0272

CODE FOR MIN-HEAP

```
NormalDistributionTable.java  KnuthShellSort.java  HibbardShellSort.java  SedgewickShellSort.java  MinHeap.java U  MaxHeap.java U
MinHeap.java > MinHeap > MinHeap(int)
1  /*
2   * GAIRE ANANTA PRASAD
3   * M24W0272
4   */
5  public class MinHeap
6  {
7      private int[] Heap;
8      private int size;
9      private int maxsize;
10
11      private static final int FRONT = 1;
12
13      public MinHeap(int maxsize)
14      {
15          this.maxsize = maxsize;
16          this.size = 0;
17          Heap = new int[this.maxsize + 1];
18          Heap[0] = Integer.MIN_VALUE;
19      }
20
21      private int parent(int pos)
22      {
23          return pos / 2;
24      }
25
26      private int leftChild(int pos)
27      {
28          return (2 * pos);
29      }
30
31      private int rightChild(int pos)
32      {
33          return (2 * pos) + 1;
34      }
35
36      private boolean isLeaf(int pos)
37      {
```

```
MinHeap.java > MinHeap > MinHeap(int)
5  public class MinHeap
36      private boolean isLeaf(int pos)
37      {
38          if (pos >= (size / 2) && pos <= size)
39          {
40              return true;
41          }
42          return false;
43      }
44
45      private void swap(int fpos, int spos)
46      {
47          int tmp;
48          tmp = Heap[fpos];
49          Heap[fpos] = Heap[spos];
50          Heap[spos] = tmp;
51      }
52
53      private void minHeapify(int pos)
54      {
55          if (!isLeaf(pos))
56          {
57              if (Heap[pos] > Heap[leftChild(pos)] || Heap[pos] > Heap[rightChild(pos)])
58              {
59                  if (Heap[leftChild(pos)] < Heap[rightChild(pos)])
60                  {
61                      swap(pos, leftChild(pos));
62                      minHeapify(leftChild(pos));
63                  }else
64                  {
65                      swap(pos, rightChild(pos));
66                      minHeapify(rightChild(pos));
67                  }
68              }
69          }
70      }
71  }
```

Gaire Ananta Prasad
M24W0272

```

MinHeap.java > MinHeap > MinHeap(int)
5 public class MinHeap
71
72     public void insert(int element)
73     {
74         Heap[++size] = element;
75         int current = size;
76
77         while (Heap[current] < Heap[parent(current)])
78         {
79             swap(current, parent(current));
80             current = parent(current);
81         }
82     }
83
84     public void print()
85     {
86         for (int i = 1; i <= size / 2; i++)
87         {
88             System.out.print(" PARENT : " + Heap[i] + " LEFT CHILD : " + Heap[2*i]
89                 + " RIGHT CHILD : " + Heap[2 * i + 1]);
90             System.out.println();
91         }
92     }
93
94     public void minHeap()
95     {
96         for (int pos = (size / 2); pos >= 1; pos--)
97         {
98             minHeapify(pos);
99         }
100     }
101
102     public int remove()
103     {
104         int popped = Heap[FRONT];
105         Heap[FRONT] = Heap[size--];
106         minHeapify(FRONT);

```

```

NormalDistributionTable.java  KnuthShellSort.java  HibbardShellSort.java  SedgewickShellSort.java  MinHeap.java
MinHeap.java > MinHeap > MinHeap(int)
5 public class MinHeap
02     public int remove()
04     {
05         int popped = Heap[FRONT];
06         Heap[FRONT] = Heap[size--];
07         minHeapify(FRONT);
08         return popped;
09     }
10
11     Run | Debug
12     public static void main(String...arg)
13     {
14         System.out.println("The Min Heap is ");
15         MinHeap minHeap = new MinHeap(maxsize:15);
16         minHeap.insert(element:5);
17         minHeap.insert(element:3);
18         minHeap.insert(element:17);
19         minHeap.insert(element:10);
20         minHeap.insert(element:84);
21         minHeap.insert(element:19);
22         minHeap.insert(element:6);
23         minHeap.insert(element:22);
24         minHeap.insert(element:9);
25         minHeap.minHeap();
26         minHeap.print();
27         System.out.println("The Min val is " + minHeap.remove());

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```

The Min Heap is
PARENT : 3 LEFT CHILD : 5 RIGHT CHILD :6
PARENT : 5 LEFT CHILD : 9 RIGHT CHILD :84
PARENT : 6 LEFT CHILD : 19 RIGHT CHILD :17
PARENT : 9 LEFT CHILD : 22 RIGHT CHILD :10
The Min val is 3
C:\D:\kcgi\Java>

```

Gaire Ananta Prasad
M24W0272

Pseudocode for MaxHeap

```
NormalDistributionTable.java  KnuthShellSort.java  HibbardShellSort.java  SedgewickShellSort.java  MinHeap.java
1  // Gaire Ananta Prasad
2  //M24W0272
3  Start MaxHeap:
4      Variables:
5          Array Heap
6          Integer size
7          Integer maxsize
8          Constant FRONT = 1
9
10     Method MaxHeap(maxsize):
11         Initialize maxsize, size, and Heap with size maxsize + 1
12         Set Heap[0] to Integer.MAX_VALUE
13
14     Method parent(pos) Returns Integer:
15         Return pos / 2
16
17     Method leftChild(pos) Returns Integer:
18         Return 2 * pos
19
20     Method rightChild(pos) Returns Integer:
21         Return 2 * pos + 1
22
23     Method isLeaf(pos) Returns Boolean:
24         Return pos >= size / 2 And pos <= size
25
26     Method swap(fpos, spos):
27         Swap Heap[fpos] and Heap[spos]
28
29     Method maxHeapify(pos):
30         If Not isLeaf(pos):
31             If Heap[pos] < Heap[leftChild(pos)] Or Heap[pos] < Heap[rightChild(pos)]:
32                 If Heap[leftChild(pos)] > Heap[rightChild(pos)]:
33                     Swap and recursively maxHeapify left child
34                 Else:
35                     Swap and recursively maxHeapify right child
36
37     Method insert(element):
38         Add element to Heap and increment size
```

```
NormalDistributionTable.java  KnuthShellSort.java  HibbardShellSort.java  SedgewickShellSort.java
3  Start MaxHeap:
29     Method maxHeapify(pos):
30         If Not isLeaf(pos):
31             If Heap[pos] < Heap[leftChild(pos)] Or Heap[pos] < Heap[rightChild(pos)]:
32                 If Heap[leftChild(pos)] > Heap[rightChild(pos)]:
33                     Swap and recursively maxHeapify left child
34                 Else:
35                     Swap and recursively maxHeapify right child
36
37     Method insert(element):
38         Add element to Heap and increment size
39         While Heap[current] > Heap[parent(current)]:
40             Swap and update current position
41
42     Method print():
43         For i from 1 to size / 2:
44             Print PARENT, LEFT CHILD, and RIGHT CHILD
45
46     Method maxHeap():
47         For pos from size / 2 down to 1:
48             maxHeapify(pos)
49
50     Method remove() Returns Integer:
51         Remove and return the root element, then maxHeapify
52
53     Method main():
54         Create MaxHeap with size 15
55         Insert elements 5, 3, 17, 10, 84, 19, 6, 22, 9
56         Build max heap
57         Print heap
58         Print and remove max value
59     End
60
```

Gaire Ananta Prasad
M24W0272

Pseudocode for MaxHeap

```
NormalDistributionTable.java KnuthShellSort.java HibbardShellSort.java SedgewickShellSort.java MinHeap.java U MaxHeap.java U
1 // Gaire Ananta Prasad
2 //M24W0272
3 Start MinHeap:
4   Variables:
5     Array Heap
6     Integer size
7     Integer maxsize
8     Constant FRONT = 1
9
10  Method MinHeap(maxsize):
11    Initialize maxsize, size, and Heap with size maxsize + 1
12    Set Heap[0] to Integer.MIN_VALUE
13
14  Method parent(pos) Returns Integer:
15    Return pos / 2
16
17  Method leftChild(pos) Returns Integer:
18    Return 2 * pos
19
20  Method rightChild(pos) Returns Integer:
21    Return 2 * pos + 1
22
23  Method isLeaf(pos) Returns Boolean:
24    Return pos >= size / 2 And pos <= size
25
26  Method swap(fpos, spos):
27    Swap Heap[fpos] and Heap[spos]
28
29  Method minHeapify(pos):
30    If Not isLeaf(pos):
31      If Heap[pos] > Heap[leftChild(pos)] Or Heap[pos] > Heap[rightChild(pos)]:
32        If Heap[leftChild(pos)] < Heap[rightChild(pos)]:
33          Swap and recursively minHeapify left child
34        Else:
35          Swap and recursively minHeapify right child
36
37  Method insert(element):
38    Add element to Heap and increment size
39
40  Live Share Java: Ready Ln 3, Col 7 Spaces: 4 UTF-8 CRLF
```

```
NormalDistributionTable.java X KnuthShellSort.java HibbardShellSort.java SedgewickShellSort.java MinHeap.java U
3 Start MinHeap:
29 Method minHeapify(pos):
30   If Not isLeaf(pos):
31     If Heap[pos] > Heap[leftChild(pos)] Or Heap[pos] > Heap[rightChild(pos)]:
32       If Heap[leftChild(pos)] < Heap[rightChild(pos)]:
33         Swap and recursively minHeapify left child
34       Else:
35         Swap and recursively minHeapify right child
36
37 Method insert(element):
38   Add element to Heap and increment size
39   While Heap[current] < Heap[parent(current)]:
40     Swap and update current position
41
42 Method print():
43   For i from 1 to size / 2:
44     Print PARENT, LEFT CHILD, and RIGHT CHILD
45
46 Method minHeap():
47   For pos from size / 2 down to 1:
48     minHeapify(pos)
49
50 Method remove() Returns Integer:
51   Remove and return the root element, then minHeapify
52
53 Method main():
54   Create MinHeap with size 15
55   Insert elements 5, 3, 17, 10, 84, 19, 6, 22, 9
56   Build min heap
57   Print heap
58   Print and remove min value
59
```

Gaire Ananta Prasad
M24W0272

Flowchart for Heap Sort





