Heap User Manual

Team T:
Jurij Afanasjev
Martin Bevc
Richard Fleming
Craig McLaughlin
Zinonas Pilakouris

February 11, 2013

Contents

1 Interacting with the heap application

- Launch the application using the command: java -jar Heap.jar

You will see a screen similar to that in Figure 1. Use the input field to specify an integer in the range 0 to 99 to insert into the heap. Pressing the "Insert" button will result in a node, with the specified value, being displayed in both views of the data structure. At this stage, the animation is in the *up-heap bubbling* phase. Comparisons and swaps between the inserted node and it's parent node will continue until the node no longer violates the heap property.

To single step through this process ensure the checkbox to left of the the "Step" button is ticked as shown in the figure. To go to the next step in the animation push, the now activated, step button.

To delete the element at the top of the heap press the "Delete" button. This operation causes the last element of the heap (leaf node furthest to the right) to be swapped with the top element. The leaf is then deleted and the *down-heap bubbling* operation is performed starting at the root of the tree. Successive comparisons and swaps between the parent and it's largest child will continue until the node no longer violates the heap property.

Lastly, you can clear the contents of the heap using the "Clear Contents" button.

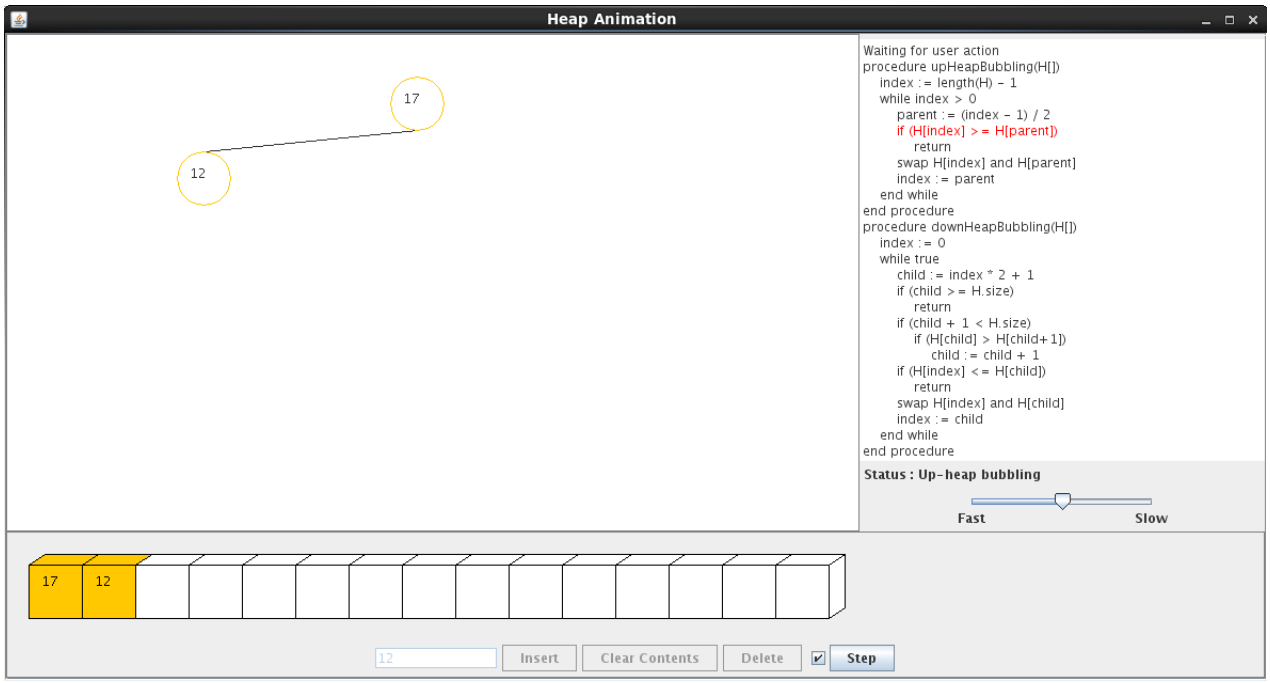

Figure 1: Program comparing two nodes

2 Heap Algorithm

The algorithms in this section describe the heap and it's two main operations for maintaining the heap property: up-heap bubbling upon an insertion and down-heap bubbling upon a deletion. Algorithm 1 describes the up-heap bubbling procedure and the auxillary procedure to exchange values in the heap. Algorithm 2 describes the down-heap bubbling operation.

Algorithm 1: up-heap bubbling operation: Restoring the heap property after an insertion.

```
1  void upHeapBubbling(Array H)
2  begin
3      index ← H.size − 1
4      while index > 0 do
5          parent ← (index − 1) ÷ 2
6          if H[index] >= H[parent] then return
7          swap(H, index, parent)
8          index = parent
9  void swap(Heap H, int i, int j)
10 begin
11     temp ← H[i]
12     H[i] ← H[j]
13     H[j] ← temp
```

Algorithm 2: down-heap bubbling operation: Restoring the heap property after a deletion.

```
1  void downHeapBubbling(Array H)
2  begin
3      index ← 0
4      while true do
5          child ← index × 2 + 1
6          if child >= H.size then return
7          if child + 1 < H.size then
8              if H[child] > H[child + 1] then child ← child + 1
9          if H[index] ≤ H[child] then return
10         swap(H, index, child)
11         index ← child
```