WA3 - Assignment Group 16

Sravani Kirla - skirla
Matthew Heil - heilx069
Terrence Hanrahan - hanra028

1.*Initialization*:
Before start of first iteration i=A.length, A[1 .. A.length] is a max-heap due to the step build-max-heap at line 1 where array A is max-heapified and contains A.length smallest elements.
A[i+1 . . n] => A[n+1 . . n] contains (n-i) => (n-A.length) => (n-n) => 0 largest elements of A.

Maintenance:
Assume: Before the iteration i = k, A[1. . k] is max-heap with k smallest elements of A and A[k+1 . . n] contains (n-k) largest elements in sorted order.

After the iteration i= k-1 must Satisfy:

1. A[1 . . k-1] must be max-heap with k-1 smallest elements of A
   Proof:
   - During the iteration when i=k, A[1] is exchanged with A[k], which pushes maximum of A[1..k] to A[k]..which leaves k-1 smallest elements in subarray A[1 . . k-1].
   - A.heap-size is reduced by one which makes heap to A[1 . . k-1]
   - Max-Heapify operation is applied for new heap which is A[1 . . k-1]
2. A[k...n] contains (n-k-1) largest elements in sorted order.
   Proof:
   - Before the iteration i = k array satisfies A[2...k] < A[1] < A[k+1...n]. During the iteration i=k exchange operation between A[1] and A[k] makes A[1...k-1] < A[k . . n].

Termination:
Assuming maintenance holds, at termination becomes i=(2-1) => i=1 and the elements from A[1..1] => A[1] is the array of one element which is a max-heap and smallest value. And the elements from [1+1 . . n] => [2 . . n] contains the (n-i)=> (n-1) largest elements in sorted order. With A[1] being the smallest element and A[2...n] in sorted order entire array A[1...n] is in sorted order proving algorithm.

```
2. Heap-Delete(A,i):                        runtime
     If A.heap-size<1                         Θ(1)
           Error “heap underflow”             Θ(1)
     delete = A[i]                            Θ(1)
     A[i] = A[A.heap-size]                    Θ(1)
     A.heap-size = A.heap-size-1              Θ(1)
     Max-Heapify(A,i)                         Θ(lg n)
     return delete                            Θ(1)
```

Heap-Delete runtime complexity is Θ(lg n) because all of the lines run in Θ(1), so the call to Max-heapify to retain the heap properties once the node is deleted, dominates the run-time with Θ(lg n).

WA3 - Assignment Group 16

3. a. Array A and C after line 5
A = [2, 0, 1, 1, 4, 0, 4, 1]
C = [2, 3, 1, 0, 2]
b. Array C after line 8
C = [2, 5, 6, 6, 8]
c.- e. Array B and array C after one, two, three iterations of loop in lines 10-12 respectively
1st iteration
B = [-, -, -, -, 1, -, -, -]
C = [2, 4, 6, 6, 8]
2nd iteration
B = [-, -, -, -, 1, -, -, 4]
C = [2, 4, 6, 5, 8]
3rd iteration
B = [-, 0, -, -, 1, -, -, 4]
C = [1, 4, 6, 5, 8]
f. The final sorted array B
B = [0, 0, 1, 1, 1, 2, 4, 4]