Nicholas Nguyen

CS 3310

10/07/2020

Runtime of Heap Sort:

Array Size: 10

Time to sort the array in nanoseconds: 21900 ns

Time to sort the array in milliseconds: 0 ms

Array Size: 100 (DEMO)

Time to sort the array in nanoseconds: 339099 ns

Time to sort the array in milliseconds: 1 ms

Array Size: 1,000

Time to sort the array in nanoseconds: 371500 ns

Time to sort the array in milliseconds: 1 ms

Array Size: 10,000

Time to sort the array in nanoseconds: 3007000 ns

Time to sort the array in milliseconds: 4 ms

Array Size: 100,000

Time to sort the array in nanoseconds: 20290900 ns

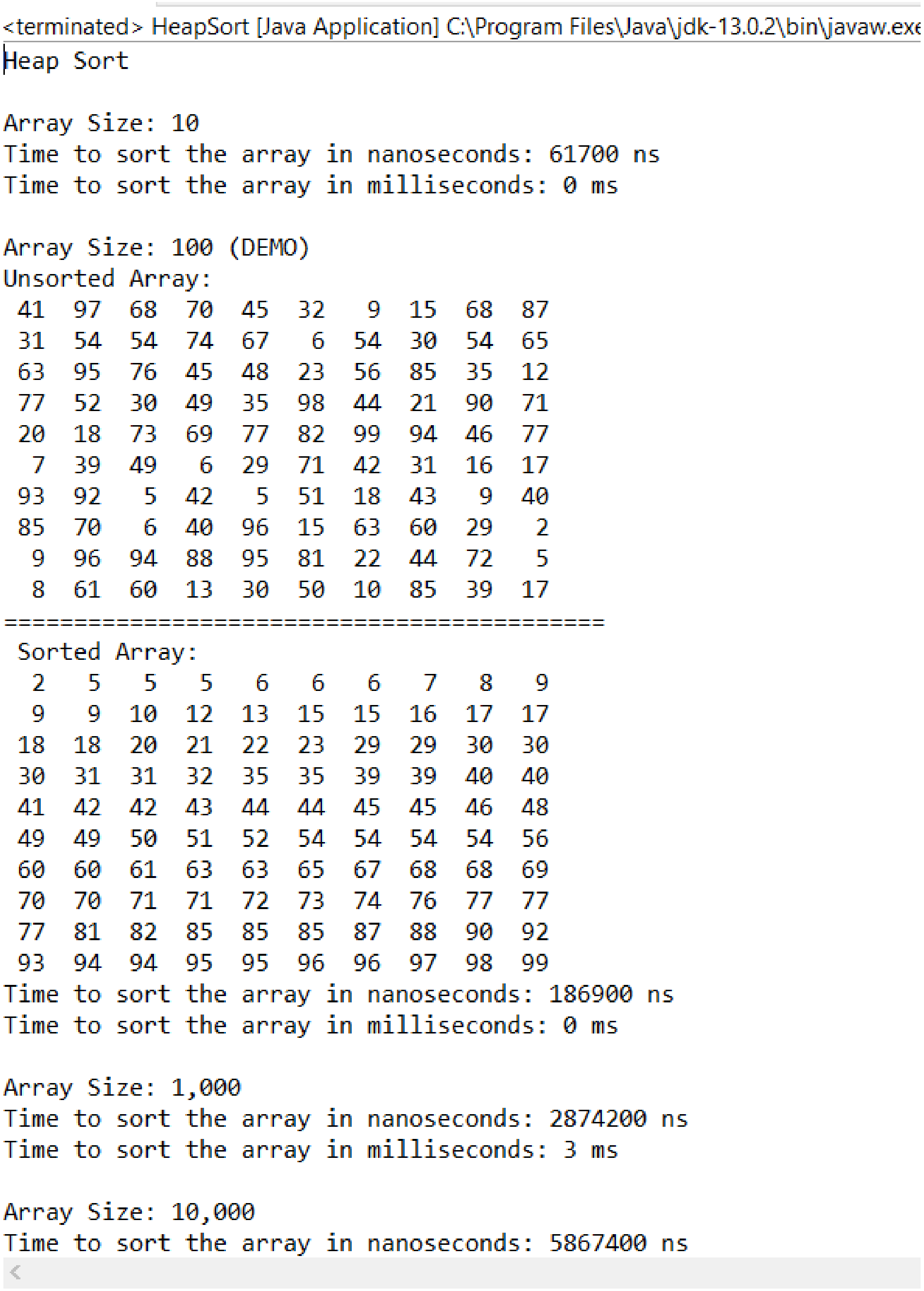
Time to sort the array in milliseconds: 20 ms

Array Size: 1,000,000

Time to sort the array in nanoseconds: 293266401 ns

Time to sort the array in milliseconds: 293 ms

OUTPUT:



<terminated> HeapSort [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe

```
Heap Sort

Array Size: 10
Time to sort the array in nanoseconds: 61700 ns
Time to sort the array in milliseconds: 0 ms

Array Size: 100 (DEMO)
Unsorted Array:
  41  97  68  70  45  32   9  15  68  87
  31  54  54  74  67   6  54  30  54  65
  63  95  76  45  48  23  56  85  35  12
  77  52  30  49  35  98  44  21  90  71
  20  18  73  69  77  82  99  94  46  77
   7  39  49   6  29  71  42  31  16  17
  93  92   5  42   5  51  18  43   9  40
  85  70   6  40  96  15  63  60  29   2
   9  96  94  88  95  81  22  44  72   5
   8  61  60  13  30  50  10  85  39  17
===========================================
 Sorted Array:
   2   5   5   5   6   6   6   7   8   9
   9   9  10  12  13  15  15  16  17  17
  18  18  20  21  22  23  29  29  30  30
  30  31  31  32  35  35  39  39  40  40
  41  42  42  43  44  44  45  45  46  48
  49  49  50  51  52  54  54  54  54  56
  60  60  61  63  63  65  67  68  68  69
  70  70  71  71  72  73  74  76  77  77
  77  81  82  85  85  85  87  88  90  92
  93  94  94  95  95  96  96  97  98  99
Time to sort the array in nanoseconds: 186900 ns
Time to sort the array in milliseconds: 0 ms

Array Size: 1,000
Time to sort the array in nanoseconds: 2874200 ns
Time to sort the array in milliseconds: 3 ms

Array Size: 10,000
Time to sort the array in nanoseconds: 5867400 ns
```

Time Complexity:

The heap sort algorithm will have O(n logn) time complexity for the best, average, and worst cases.

However, heap sort does not always use the same number of comparisons:

The worst case is a*n logn and the best case is b*n logn, where a > b

Analysis:

Let T(n) be the time to run Heapsort on an array of size n. Examination of the algorithms leads to the following formulation for runtime:

$$\mathrm{T(n) = TB(n) + \sum_{k=1}^{n-1} TH(k) + \theta(n-1)}$$

(equation 1)

Where, TB is the time complexity of building the heap and TH is the time complexity of heapify

Heapify is also used in the buildheap, so:

$$\mathrm{TH(n) = \ \theta(1) + \ TH(size\ of\ subtree)}$$

This can be deduced to

$$TH(n) = \ \theta(\log n)$$

Putting this formula back into equation 1, we get:

$$\begin{aligned} T(n) &= TB(n) + \sum\nolimits_{k=1}^{n-1} TH\ (k) + \ \theta(n-1) \\ &= \ \theta(n) + \textstyle\sum_{k=1}^{n-1} log\, k \ \ + \ \theta(n-1) \\ &= \theta(n\ \log n\ ) \end{aligned}$$

Thus, the time complexity of the algorithm is: O(n logn)

Memory:

Since, the data in the array is organized into a heap, in place – the data is actually not stored anywhere else, except during the swap step.

Thus, the memory complexity is O(1), constant time.