Programming Assignment 3

Joni Vrapi

11/13/2022

Statement of Integrity: I, Joni Vrapi, attempted to answer each question honestly and to the best of my abilities. I cited any, and all, help that I received in completing this assignment.

Problem (a).

```
PARTITION(A, p, r)
 1  i = p − 1
 2
 3  (pivotValue, pivotIndex) = MEDIAN-OF-THREE(A, p, r)
 4  swap A[r], A[pivotIndex]
 5
 6  for j to [p, r)
 7       if A[j] ≤ pivotValue
 8            i = i + 1
 9            swap A[i] and A[j]
10
11  swap A[i + 1] and A[r]
12
13  return i + 1
```

```
MEDIAN-OF-THREE(A, p, r)
 1  k = ⌊(i + j)/2⌋
 2
 3  tempArray = [(A[p], p), (A[k], k), (A[r], r)]
 4  tempArray.sort(tuple => tuple[0])
 5
 6  medianIndex = 1
 7
 8  if r − p >= 2
 9       return tempArray[medianIndex]
10  else
11       return (A[r], r)
```

Problem (b). The worst case asymptotic behavior of QUICKSORT with median of three partitioning occurs when every element in the array is the same number. This negates the effect of the median choosing, as every element is the same, resulting in an arbitrary choice of a pivot (producing unbalanced partitions), requiring $\Theta(n)$ recursive calls giving us

$$T(n) = T(n-1) + T(0) + \Theta(n) \tag{1}$$

Which by substitution we know resolves to $\Theta(n^2)$. For normal QUICKSORT, this same thing happens both in this case, as well as when the array is in reverse sorted order, resulting in

maximally unbalanced partitions and a $\Theta(n^2)$ run time. The *expected* worst case run time with median of three partitioning, however, is $O(nlog(n))$ due to the fact that a median is chosen which results in balanced partitioning. The median can be chosen in $O(n)$ time while $log(n)$ partitions are made resulting in an $O(nlog(n))$ *expected* worst case run time. In this balanced partitioning case, we have

$$T(n) = 2T(n/2) + \Theta(n) \tag{2}$$

which by the Master theorem we know is $O(nlog(n))$.

Problem (c). The worst case asymptotic behavior of QUICKSORT with median of three partitioning on an input set that is already sorted is, as mentioned in the previous answer, $O(nlog(n))$. The median can be chosen in $O(n)$ time while $log(n)$ partitions are made resulting in an $O(nlog(n))$ *expected* worst case run time. In this balanced partitioning case, we have

$$T(n) = 2T(n/2) + \Theta(n) \tag{3}$$

which by the Master theorem we know is $O(nlog(n))$.

Analysis. After running each of the *quicksort.py* and *median-of-three.py* files as described in the README.md file, you will see the produced output in terms of a worst case scenario where each algorithm receives an input array (of increasing size from 0 to 10) in reverse sorted order. The normal implementation of quicksort takes the maximal amount of iterations on each run, growing exponentially. The median of three implementation, however, takes far fewer iterations, and in fact grows linearithmically.