Foundation of Algorithms

605.421.83

Programming Assignment 2

Name: Guan Yue Wang

ID: gwang39

E-mail: gwang39@jhu.edu

Phone: +16479888206

Date: Oct. 30th, 2017

(a) [15 points] Write pseudocode for median-of-three partitioning.

```
int M3Partition (A, i, j)
{
    if (i < j −1)
    {
        k = floor((i + j)/2)

        if (A[i] <= A [k])
        {

            if (A[k] <= A[j])
            {
                Swap (A[i], A[k])
            }

            else if (A[i] <= A[j])
            {
                Swap (A[i], A[j])
            }

        else if (A[i] > A[j])
        {
            if (A[k] <= A[j])
            {
                Swap(A[i], A[j])
            }
            else
            {
                Swap(A[i], A[k])
            }
        }
    }

    return Partition (A, i, j)
}
```

(b) [15 points] What is the running time of median-of-three partitioning? Justify your answer.

The running time of mediam-of-three partitioning is θ(n) because determining the median-of-three can be done in constant time, and Partition is θ(n).

(c) [20 points] What is the running time of Quicksort if you use median-of-three partitioning on an input set that is already sorted? Justify your answer.

The running time of medium-of-three partitioning Quicksort is O(nlgn), because A[k] is always the median, thus leading to bisecting the array during every partitioning.

(d) [50 points] Implement Quicksort using a normal pivot process and the median-of-three process described above. Test your run time analysis of medium-of-three, and then compare the average and worst case run times of Quicksort with the two pivot processes. Note that you must implement all of these algorithms from scratch. Also remember that CPU time is not a valid measure for testing run time. You must use something such as the number of comparisons.

- Runtime can be seen based on number of comparisons and exchanges in the output file (if you run the program on data, you will get one output for quick sort using normal partitioning and another output for quick sort using medium-of-three partitioning with count of exchanges included)

- Below table includes the number of exchanges happens when we use quick sort on 100 ascending, descending, random data

Quick sort	100 ascending integers	100 random integers	100 descending integers
Normal partitioning	5049 exchanges	324 exchanges	2549 exchanges
Medium-of-three partitioning	3873 exchanges	372 exchanges	586 exchanges

- From above table we can see medium-of-three partitioning quick sort does better when data is sorted (either ascending or descending) but doesn't show a significant difference on random data.

- Therefore, by using medium-of-three partitioning, quick sort has a better run time on sorted data (ascending or descending) . For random ordered data, it's hard to tell if one is better than the other because it really depends how good/bad the normal pivot is for random data.