CS 5800: Algorithms (Spring 2020) **Ahmed Ezzat**
Student: Samuel Steiner Due Feb. 7, 2020

Homework 3

1 Question 1 [1pt]: Chapter-7(page 180): Exercise 7.3.1

Question:

Why do we analyze the expected running time of a randomized algorithm and not its worst-case running time?

Answer:

Expected run time represents a more realistic and typical cost for an algorithm. Expected run time also allows us to take into account randomness used during computation.

2 Question 1 [1pt]: Chapter-7(page 186): Exercise 7-2

Question:

The analysis of the expected runtime of randomized quicksort in Section 7.4.2 assumes that all element values are distinct.In this problem, we examine what happens when they are not?
a)Suppose that all elements values are equal. What would be randomized quicksort's running time in this case?
d)Using Quicksort', how would you adjust the analysis in Section 7.4.2 to avoid the assumption that all elements are distinct?

Answer:

a) randomized quicksort will always return $q = r$, $T(n) = T(n-1) + \Theta(n) + \Theta(m^2)$
d) The subproblem size of quicksort' is no larger than the subproblem of normal quicksort when all elements are distinct since we don't recurse on elements equal to the pivot.

3 Question 1 [1pt]: Chapter-8(page 200): Exercise 8.3.2

Question:

Which of the following sorting algorithms are stable? Insertion sort, merge sort, and quicksort? Give a simple scheme that makes any comparison sort stable? How much additional time and space your scheme entails?

Answer:

Insertion and merge sort are stable the others are not.
A stable sorting algorithm we are able to preprocess, replacing each element of the array we are sorting with an ordered pair. The order pair will consist of 0 the value of the element and 1 the value of the index of the element.
ex) the array $[2,4,5,3,6,1]$ would be $[(2,1),(4,2),(5,3),(3,4),(6,5),(1,6)]$, we can now look at $(i,j) < (k,m)$ if $i < k$ or $i = k$ and $j < m$. Using this definition of less than, the algorithm is stable. This is because each of the new elements is distinct and the index comparison ensures that if a repeat element appeared later in the original array it must appear later in the sorted array. This does double the space requirement of the sorting algorithm but run time is unchanged.

4 Question 1 [1pt]: Chapter-8(page 204): Exercise 8.4.1

Question:

Using Figure 8.4 as a model, illustrate the operations of Bucket-Sort on the array:
$A = 0.79, 0.13, 0.16, 0.64, 0.39, 0.20, 0.89, 0.53, 0.71, 0.42$

Answer:

R		
0		
1	.13	.16
2	.20	
3	.39	
4	.42	
5	.53	
6	.64	
7		
8	.79	.71
9	.89	