Homework assignment 4:

Suggested due date: Friday, October 27 2017 at 03:30pm

1- What is the worst-case running time of Quicksort if the pivot is randomly chosen as the first element in the array. Explain.

2- Show that the average size of a_{left} is *(n-1)/2* when the input to Quicksort is *n* distinct elements and the median M is randomly chosen from one of the elements.

3- Calculate the running time of a divide-and-conquer algorithm that requires three recursive calls (each with input-size *n/2*) and $3n^2$ steps that include dividing the input, and using the three solutions to obtain the final solution.

4- Use a recursion tree for the following algorithms to find the running time.

a. T(n) = $T(n/2)+1$
b. T(n) = $T(n-1)+1$
c. T(n) = $2T(n-1)+1$
d. T(n) = $3T(n/4)+n$
e. T(n) = $3T(n/3)+\sqrt{n}$
f. T(n) = $T(n/2)+n^2$
g. T(n) = $T(n/3)+T(2n/3)+n$
h. T(n) = $4T(\lfloor n/2 \rfloor+2)+n^2$

5- Use the formula to determine the asymptotic growth of T(n).

a. T(n) = $T(n/2)+1$
b. T(n) = $3T(n/3)+n$
c. T(n) = $4T(n/3)+n$
d. T(n) = $3T(n/4)+\sqrt{n}$
e. T(n) = $5T(n/7)+n^2$
f. T(n) = $6T(n/5)+n^3$

6- Find the contiguous sub-array with the maximum sum ($\sum_{k=i}^{j} a_k$) of the below array ($a_1,a_2,\ldots,a_n$)

a. 7, 2, -14, 3, 5 , -3 , 4

b. 1, -5, 6, -3, 10, -4, 2, 0, 3

7- Suppose a student wrote the below code for the previous question (**Maximum Subsequence Sum problem (MSS)**). What is the running time of this algorithm?

```
max_sum = 0;
for(i=0; i< n; i++)
        {
                for(j=i; j< n; j++)
                        {
                                sum=0;

                                for(k=i; k <= j; k++)
                                        sum += a[k];

                                if(sum > max_sum)
                                        max_sum = sum;
                        }
}
return max_sum;
```

8- Suppose another student who is a better programmer wrote the below code for **Maximum Subsequence Sum problem (MSS)**. What is the running time of this algorithm?

```
max_sum = 0;
for(i=0; i< n; i++)
        {
                sum= 0;
                for(j=i; j< n; j++)
                        {
                                sum += a[j];

                                if(sum > max_sum)
                                        max_sum = sum;
                        }
        }
return max_sum;
```

9- Suppose another student decides to solve the **Maximum Subsequence Sum problem (MSS)** using divide and conquer technique:

Divide: Divide the array into two halves.

Conqure: Recursively find MSS of the two sub-arrays, each of size n/2,

Combine: Now in order to combine the sub-arrays you need to know that MSS can lie in one of the following places:

1- Entirely in the left sub-array

2- Entirely in the right sub-array

3- Intersects both halves:
(We can easily find a maximum sub-array crossing the midpoint in time **linear** in the size of the sub-array. This problem is *not* a smaller instance of our original problem, because it has the added restriction that the sub-array it chooses must cross the midpoint. Any sub-array crossing the midpoint is itself made of two sub-arrays a[i…mid] and a[mid+1..j], where *i* is an *index* in the left sub-array, and *j* is an *index* in the right sub-array. Therefore, we just need to find maximum sub-arrays of the form a[i..mid] and a[mid+1..j] and then combine them.)

Therefore, to choose MSS as the final answer you need to find the max between these three.

What is the running time of this algorithm? (Note: You can read the complete description of MSS from your textbook: page 70-73)

10- Suppose we have a student who is extraordinarily smart, and decides to solve the **Maximum Subsequence Sum problem (MSS)** using the below code. What is the running time of this algorithm?

```
max_sum = 0;
sum = 0;
for(i=0; i< n; i++)
        {
                sum += a[i];

                if(sum > maxSum)
                        max_sum = sum;
                else if(sum < 0)
                        sum = 0;
        }
return max_sum;
```

11- Find the MSS of the below array using solutions for Q.10 and Q.9:

a. 7, 2, -14, 38, 52, -37, 4, 12, -4, 6, 3, 2

b. 3, 6, -20, 11, -15, 26, -43, 10, -14, 27, 0, 39

12- Find the
a. 2^{nd} least element
b. 4^{th} least element
using the random find statistics algorithm.

- 2, -1, 3, 8, 9, 0, 19, 6, 35, 17, 20
- 10, 11, 12, 13, 14, 15, 16, 17, 18

13- Calculate the running time of the find statistics algorithm when you are looking for the n^{th} least element in an sorted array with the size of n: Assume that in each iteration the pivot is chosen as the first element in the array.

14- Calculate the average-case running time of the find statistics algorithm. Explain.

15- Explain an algorithm to return the max k numbers from an unsorted array. (The average running time of your algorithm should be O(n))