CSE 301 - Algorithm Analysis: Assignment 2

Deadline: 1.12.2023 23:59
Rule#1: Individual work. Plagiarism is not accepted, and once notified, the grade for the assignment will be automatically zero.
Rule#2: Although it is good practice before the midterm to solve all of them, you are expected to solve a certain amount. Mod 7 of your entire studentID (e.g. 20220808222) will show the number of the questions you must have in your assignment!

> **0:** 1a - 2b - 4b - 5a - 7
> **1:** 1b - 3 - 4b - 5b - 6a
> **2:** 2a - 4a - 5b - 6b - 7
> **3:** 1a - 2a - 3 - 4a - 6b
> **4:** 1b - 4a - 5b - 6a - 7
> **5:** 1b - 2b - 3 - 5a - 6b
> **6:** 1b - 2b - 4b - 5b - 7

Rule#3: Please submit your file in a PDF format (-10 points when it is not in PDF) and include your student number in the beginning of your file name (-5 points when your studentID is not on anywhere of the document)
Suggestion: You can start a conversation on the questions on the team's general page and expect us to help. You can also message me or Gökhan Hoca if you can't proceed with your answer. However, we recommend you keep the questions the day before the deadline.

1. **Explain which of the following are True or False**
    a. $n\,logn = O(n\sqrt{n})$
    b. $\sqrt{n} = O(logn)$
    c. $logn + \sqrt{n} = O(n)$
    d. $n + \sqrt{n} = O(n)$
2. **Give the asymptotic relations for the following sets of f and g functions.**
    a. $f(n) = n + n\sqrt{n} \ | \ g(n) = 4nlog(n^2 + 1)$
    b. $f(n) = 2^n - n^2 \ | \ g(n) = n^4 + n^2$

3. **Write an algorithm that returns two elements from an n size of an integer array. The subtraction of these two elements results in the maximum value from any other pair in the array. This algorithm should be a divide-and-conquer algorithm with O(n) time complexity and *recursive* structure. For example:**
    $A = [4.5, 10, -2, \pi, -7.115]$
    >> **17.115.**

4. **Solve the following occurences:**
    - **a.** $T(1) \; = \; 3 \; and \; for \; all \; n \; \geq \; 2, \; T(n) \; = \; 4T(n/3) \; + \; 2n \; - \; 1$
    - **b.** $T(1) \; = \; 2 \; and \; for \; all \; n \; \geq \; 2, \; T(n) \; = \; 4T(n/3) \; + \; 2n \; - \; 5$

5. **What is the value returned by the following function? Express your answer as a function with "n" as a variable using big-O notation. (worst-case scenario)**

a.

```
function mystery(n)
    r := 0;
    for i := 1 to n − 1 do
      for j := i + 1 to n do
        for k := 1 to j do
          r := r + 1
    return(r)
```

b.

```
function pesky(n)
    r := 0;
    for i := 1 to n do
      for j := 1 to i do
        for k := j to i + j do
            r := r + 1
    return(r)
```

6. **The following is a less explicit version of Hoare's Quicksort algorithm. Suppose S is the input array of unsorted numbers.**

```
function quicksort(S)
  if |S| ≤ 1
    then return(S)
    else
      Choose an element a from S
      Let S1, S2, S3 be the elements of S that are respectively <, =, > a
      return(quicksort(S1),S2,quicksort(S3))
```

- **a.** The median of a set of n values is the $\lceil n/2 \rceil th$ smallest value. Suppose the quicksort always takes the median value as pivot. In the worst case, how many comparisons will be made (Show it as a function with parameter 'n')
- **b.** Suppose quicksort was to always pivot on the $\lceil n/3 \rceil th$ smallest value. In the worst case, how many comparisons will be made (Show it as a function with parameter 'n')

7. Given the following algorithm, how many comparison operations are done? How would you turn it into a recursive divide-and-conquer algorithm?

```
procedure maxmin2(S)
  comment computes maximum and minimum of S[1..n]
    in max and min resp.
  if n is odd then max:=S[n]; min:=S[n]
    else max:=−∞; min:=∞
  for i := 1 to ⌊n/2⌋ do
    if S[2i − 1] ≤ S[2i]
      then small:=S[2i − 1]; large:=S[2i]
      else small:=S[2i]; large:=S[2i − 1]
    if small < min then min:=small
    if large > max then min:=small
```