ESO207A: Data Structures and Algorithms
Theoretical Assignment 2

Deadline: Jan 25, 2018

January 20, 2018

Instruction

- You are advised to solve these without use of the internet.
- All submission must be in the pdf format.
- Please write precise answers, don't write unnecessary details.
- Marks of subparts of questions may not be evenly distributed.

Problem 1. (20 + 10 marks) Given the head of a Linked-List of unknown length propose an algorithm to find whether the linked list has a cycle or not.
For this algorithm find time complexities big Oh(O), big omega(Ω) and big theta(Θ).

Problem 2. (20 marks) Find time complexity of following four procedures.

```
int sum = 0;
for ( int i = 0; i < N; i++ ) {
    for ( int j = 0; j < i; j++ ) {
        sum += j;
    }
}
```

```
int sum = 0;
for ( int i = 1; i < N; i*=2 ) {
    for ( int j = 0; j < i; j++ ) {
        sum += j;
    }
}
```

```
int sum = 0;
for ( int i = 1; i < N; i*=2 ) {
    for ( int j = 1; j < i; j*=2 ) {
        sum += j;
    }
}
```

```
real sum = 0;
for ( real i = N; i > 0; i/=2 ) {
    sum += i;
}
```

Problem 3. (20 marks) For each of the following functions, f(x) and g(x), indicate whether f(x) = O(g(x). Give a formal proof in each case.

1. $a \log n$ and $\log_a n$
2. 7^{4n} and $2^{n/11}$
3. $2^{\sqrt{\log n}}$ and $\sqrt{n}$
4. $\sum_{i=1}^{n} \frac{1}{i}$ and $\log n$

Problem 4. (40 marks) Solve the following recursion to get a tight upper bound with proper explanation. Assume T(x) = c, $\forall x < 2$ in all subparts, where c is a constant.

1. T(n) = c*n + T(n/5) + T(n/3) also find condition on a,b such that $T^{'}(n) = c * n + T^{'}(a * n) + T^{'}(b * n)$ such that T and $T^{'}$ have same tight upper bound.
2. T(n) = c + T(n/k), $k > 1$
3. T(n) = c + T($\sqrt[\alpha]{n}$), $\alpha > 1$
4. T(n) = c*n + T(n/k), $k > 1$

Problem 5. (10 marks) State true or false with proper mathematical proofs.

1. 4n + 7 is $o(n)$. *(Little-o)*
2. 4n + 7 is $o(n^2)$.
3. 4n + 7 is $\omega(n)$. *(Little-omega)*
4. 4n + 7 is $\omega(log(n))$.

Problem 6. (10 marks) Choose **all** correct answer and explain.

1. f(n) = c^n, g(n) = n^k, where c,k are constants.

(a) f(n) in O(g(n))
(b) f(n) in Ω(g(n))
(c) f(n) in Θ(g(n))
(d) None

2. f(n) $= \log_2(n)$, g(n) $=$ ln(n)

   (a) f(n) in O(g(n))
   (b) f(n) in Ω(g(n))
   (c) f(n) in Θ(g(n))
   (d) None

3. f(n) $= n^2 \log_2(n)$, g(n) $= n \log_2(n^3)$

   (a) f(n) in O(g(n))
   (b) f(n) in Ω(g(n))
   (c) f(n) in Θ(g(n))
   (d) None

Problem 7 . (50 marks) **Define** an inversion in an array **A** as the ordered pair (A_i, A_j) where

$$A_i > A_j \text{ and } j > i$$

For example: if $A = [1, 42, 3, 14, 2]$ then the inversions are $\{(42, 3), (42, 14), (42, 2), (3, 2), (14, 2)\}$

Devise an algorithm to count the number of inversions in a given array of length n. There exists a trivial algorithm of time complexity $O(n^2)$. However, this complexity can be improved. Your task is to find an algorithm that performs better than $O(n^2)$. Derive the time complexity for the efficient algorithm.

Please look at the hints (on the next page) **iff** you are stuck at some point.

Hint Can you divide the array so that the problem becomes simpler?
Another Hint Can you relate this to merge sort?