Homework #1

Instructor: Ali Sharifian

Fall 2021

For all questions, choose the **best** answer.

1. What is the worst case rate of growth for the following code:

```
for (i = 1; i <= n; i=2*i)

        System.out.println(i);
```

Solution:

We start with i = 1.

At iteration 1: i = 1*2 = 2

At iteration 2: i = 2*2 = 4

At iteration 3: i = 4*2 = 8

At iteration 4: i = 8*2 = 16

At iteration 5: i = 16*2 = 32

Let j be the iteration number. We see that i is increasing at a rate of 2^j.

How many iterations does the for loop take though? We see that the for loop exits once i reaches n (because as you can see, the for loop says i <= n).

Let x be the last iteration of the for loop.

Based on the above two points, we get the following equation: $2^x = n$

Now solve for x:

$\log 2^x = \log n$

$x \log 2 = \log n$

$x = \frac{\log n}{\log 2}$

Thus, worst case rate of growth is O(lg n)

a. $O(n)$
b. $O(n^2)$
c. $O(\lg n)$
d. $O(n \lg n)$
e. $O(2^n)$

2. What is the runtime of the following algorithm:

```
n = A.length
for j = 1 to n - 1
    smallest = j
    for i = j + 1 to n
        if A[i] < A[smallest]
            smallest = i
    exchange A[j] with A[smallest]
```

Solution:
Doubly nested for loop, with both for loops having weight n. So n*n = n^2.
Let's analyze the second for loop weight further:

$$\sum_{j=1}^{n-1} n-(j+1) = (n-2)+(n-3)+(n-4)+\cdots+1+0 = 1+2+\cdots+(n-3)+(n-2)$$

Note that:

Arithmetic series

The summation

$$\sum_{k=1}^{n} k = 1+2+\cdots+n\ ,$$

is an ***arithmetic series*** and has the value

$$\sum_{k=1}^{n} k = \frac{1}{2}n(n+1) \qquad \text{(A.1)}$$

$$= \Theta(n^2)\ . \qquad \text{(A.2)}$$

Thus $1+2+\cdots+n-2 = \frac{1}{2}n(n+1)-n-(n-1) = \frac{1}{2}n(n+1)-n-n+1 = \theta(n^2)$

Also see page 26 for same approach used for insertion sort.

a. $\theta(n^3)$
b. $\theta(n)$
c. $\theta(n^2)$
d. $\theta(n \lg n)$
e. $\theta(\lg n)$

3. Which of the following is true?

I. n = $\theta(\lg n)$
II. $n = O(n^2)$
III. $\log n^2 = O(\log n)$
IV. $\log^2 n = O(\log n)$

Solution:
Statement I: false because n is not upper bounded by lg n.
Statement II: True, n is upper bounded by n^2
Statement III: True, because log n^2 = 2 log n. Drop the constant and you have log n. So log n is upper bounded (big O) by log n.
Statement IV: False. Assume true and prove by contradiction:
log^2 n = O(log n)
log^2 n <= c log n, for some constant c
Divide both sides by log^2 n: (log^2 n)/(log^2 n) <= (c log n)/(log^2 n)
1 <= c/(log n)

As n goes to infinity, does c/(log n) >= 1? No. The above inequality doesn't hold true. So log^2 n doesn't equal O(log n).

a. II and III
b. I only
c. II only
d. I and II
e. IV only

For questions 4 and 5, use the master theorem:

4. T(n) = 16T(n/4) + 1

Solution:
a=16
b=4
f(n)=1

$$n^{log_4 16} = n^2$$

Case 1:

f(n) = 1 = O($n^{2-\epsilon}$) $for\ some\ \in > 0? Yes, if\ \in = 1. Thus, T(n) = \theta(n^2)$

a. $\theta(1)$
b. $\theta(n)$
c. $\theta(\log n)$
d. $\theta(n \log n)$
e. $\theta(n^2)$

5. T(n) = 2T(n/4) + n

Solution:

a =2
b=4
f(n) = n

$$n^{log_4 2} = \sqrt{n}$$

Case 1:
n =O($n^{.5-\epsilon}$) $for\ some\ \epsilon > 0? No$

Case 2:
n = $\theta\left(\sqrt{n}\right)? No.$ No, n is not upper bounded by n^.5

Case 3:
n = $\Omega(n^{.5+\epsilon}) for\ some\ \epsilon > 0? Yes, if\ \epsilon = .5.$ Next, check:
a*f(n/b) <= c*f(n) for c < 1
2*f(n/4) <= c*f(n)
2*(n/4) <= c*n
Divide both sides by n:
2/4 <= c
This is true if c = 9/10.
2/4 <= 9/10

We just need to choose a c that is less than 1 and allows the inequality to be true. Thus, c is not unique. In this example, as long as c < 1 and c >= 2/4, that'd work. For instance, if c = 8/10, that would work too.

Based on the above, Case 3 holds true and T(n) = $\theta\big(f(n)\big) = \theta(n)$

a. $\theta(1)$
b. $\theta(n)$
c. $\theta(\log n)$
d. $\theta(n \log n)$
e. $\theta(n^2)$