CSCI 570 - Spring 2022 - HW2

Due January 26th

1 Graded Problems

1. What is the tight upper bound to the worst-case runtime performance of the procedure below?

   $c = 0$
   $i = n$
   while $i > 1$ **do**
       for $j = 1$ to i **do**
           $c = c + 1$
       end for
       $i = \text{floor}(i/2)$
   end while
   return c

2. Arrange these functions under the O notation using only $=$ (equivalent) or $\subset$ (strict subset of):

   (a) $2^{\log n}$
   (b) 2^{3n}
   (c) $n^{n \log n}$
   (d) $\log n$
   (e) $n \log \left(n^2\right)$
   (f) n^{n^2}
   (g) $\log(\log(n^n))$

   E.g. for the function n, $n+1$, n^2, the answer should be

$$O(n+1) = O(n) \subset O(n^2).$$

3. Given functions f_1, f_2, g_1, g_2 such that $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample.

   (a) $f_1(n) \cdot f_2(n) = O\left(g_1(n) \cdot g_2(n)\right)$

   (b) $f_1(n) + f_2(n) = O\left(\max\left(g_1(n), g_2(n)\right)\right)$

   (c) $f_1(n)^2 = O\left(g_1(n)^2\right)$

   (d) $\log_2 f_1(n) = O\left(\log_2 g_1(n)\right)$

4. Given an undirected graph G with n nodes and m edges, design an $O(m+n)$ algorithm to detect whether G contains a cycle. Your algorithm should output a cycle if G contains one.

2 Practice Problems

1. Solve Kleinberg and Tardos, **Chapter 2, Exercise 6.**

2. Solve Kleinberg and Tardos, **Chapter 3, Exercise 6.**

HW-2

PAGE No.
DATE / /

1.
```
c = 0
i = n
while i > 1 do
    for j = 1 to i do
        c = c + 1          - - -   (k)
    end for
    i = floor(i/2)
end while
return c
```

$$kn + \frac{kn}{2} + \frac{kn}{4} + \cdots$$

$$= kn\left(\frac{1}{1-\frac{1}{2}}\right)$$

$$= 2kn$$

$\therefore$ tight upper bound $= O(n)$

PAGE No.

DATE / /

2. (a) $2^{\log n} = n$ (assuming log with base 2)

(b) 2^{3n}

(c) $n^{n \log n}$

(d) $\log n$

(e) $n \log(n^2) = 2n \log n$

(f) n^{n^2}

(g) $\log(\log(n^n)) = \log(n \log n)$

logarithmic = $\log n$, $\log(n \log n)$

polynomial = n, $2n \log n$

exponential = 2^{3n}, $n^{n \log n}$, n^{n^2}

$$O(\log n) \subset O(\log(n \log n)) \subset O(n) \subset O(n \log n)$$

$$\subset O(n^{n \log n}) \subset O(n^{n^2}) \subset O(2^{3n})$$

3. (a) $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$ True

If $f_1(n) = O(g_1(n))$

$\therefore$ there exists c_1 & n_1 such that

$$0 \leq f_1(n) \leq c_1 g_1(n) \ \forall \ n \geq n_1 \quad -(1)$$

Similarly for $f_2(n) = O(g_2(n))$
there exists c_2 & n_2 such that

$$0 \leq f_2(n) \leq c_2 g_2(n) \ \forall \ n \geq n_2 \quad -(2)$$

Now, let's say, $c = c_1 \cdot c_2$.
By (1) & (2) we get
$$f_1(n) \times f_2(n) \leq c_1 g_1(n) \cdot c_2 g_2(n)$$

$$f_1(n) \cdot f_2(n) \leq c_1 \cdot c_2 \cdot g_1(n) g_2(n)$$

$$\therefore f_1(n) \cdot f_2(n) \leq c(g_1(n) \cdot g_2(n))$$

$$\forall n \geq \max(n_1, n_2)$$

$$\therefore \quad f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$$

PAGE No.

DATE / /

(6) $f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$ True

By $f_1(n) = O(g_1(n))$ & $f_2(n) = O(g_2(n))$

$f_1(n) \le c_1 g_1(n)$ & $f_2(n) \le c_2 g_2(n)$
$\forall n \geq n_1$ $\qquad$ $\forall n \geq n_2$

where $c_1, c_2 > 0$

$\therefore f_1(n) + f_2(n) \le c_1 g_1(n) + c_2 g_2(n)$

$\le c_1 \max(g_1(n), g_2(n)) + c_2 \max(g_1(n), g_2(n))$

$\le (c_1 + c_2) \max(g_1(n), g_2(n))$

let's say $c_3 = c_1 + c_2$

$\le c_3 \max(g_1(n), g_2(n))$
$\forall n \geq \max(n_1, n_2)$

$\therefore f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$

PAGE No.
DATE / /

(c) $f_1(n)^2 = O(g_1(n)^2)$ True

By $f_1(n) = O(g_1(n))$ $f_1(n) \le c_1 g_1(n) \ \forall n \ge n_1$
where $c_1 > 0$

$\therefore f_1(n) \cdot f_1(n) \le c_1 g_1(n) \cdot c_1 g_1(n)$

$f_1(n)^2 \le c_1^2 g_1(n)^2$

let $c = c_1^2$

$f_1(n)^2 \le c\, g_1(n)^2$

$\therefore f_1(n) = O(g_1(n)^2)$

(d) $\log_2 f_1(n) = O(\log_2 g_1(n))$ False

Will not work for $f_1(n) = 2n$ and
$g_1(n) = n$

as $\log_2 f_1(n) \le c_1 \log_2 g_1(n)$

doesn't exist

DATE / /

4. We can use DFS to detect cycles in an undirected graph.
For every visited vertex 'v', if there is an adjacent 'u' such that u is already visited and u is not parent of v, then there is a cycle in the graph.

Algorithm: (Graph of V vertices & E edges)

Function IsCyclic()

```
Maintain a boolean array 'visited' of all unvisited nodes
For i from 0 to V
    if not visited then
        if (call IsCyclicNode) then
            return true
        End if
    End if
End For
return false
```

Function IsCyclicNode()

```
visit the node
for all adjacent vertices
    if not visited then
        if (call IsCyclicNode) then
            return true
        End if
    Else if visited & not a parent then
        return true
    End if
    return false
End For
```