20171248 JaeHyung Ahn

Algorithms – Assignment 1

(Complexity)

Prof. Eunwoo Kim

Due: 19th March

1) Show directly that ① $f(n) = n^2 + 3n^3 \in O(n^3)$ and ② $f(n) = n^2 + 3n^3 \in \Omega(n^3)$.

① let $g(x) = n^3$. when $n \geq 1$, $f(n) = n^2 + 3n^3 \leq 4g(x)$. So $f(n) = n^2 + 3n^3 \in O(\underbrace{g(n)}_{n^3})$

② let $g(x) = n^3$. when $n \geq 1$, $f(n) = n^2 + 3n^3 \geq g(x)$. So $f(n) = n^2 + 3n^3 \in \Omega(\underbrace{g(n)}_{n^3})$

2) Using the definitions of O and Ω, show that

$$6n^2 + 20n \in O(n^3), \text{ but } 6n^2 + 20n \notin \Omega(n^3).$$

$f(x) \in O(g(x))$ if there are positive integers C and k such that $|f(x)| \leq C|g(x)|$ whenever $k \leq n$.

When $n \geq 1$, $6n^2 + 20n \leq 26n^3$. In this case, $C = 26$ and $k = 1$. So $6n^2 + 20n \in O(n^3)$.

And there's no C and k that satisfies $6n^2 + 20n \leq Cn^3$ when $n \geq k$. So $6n^2 + 20n \notin \Omega(n^3)$

3) The function $f(n) = 3n^2 + 10n\log n + 1000n + 4\log n + 9999$ belongs in which of the following complexity categories:

(a) $\Theta(\lg n)$ (b) $\Theta(n^2 \log n)$ (c) $\Theta(n)$ (d) $\Theta(n \lg n)$ (e) $\Theta(n^2)$ ✓ (f) None of these

As $0 \leq \log n \leq n \leq n\log n \leq n^2$ for $n \geq 16$, $f(n)$ is $O(n^2)$

And clearly $f(n)$ is $\Omega(n^2)$

So $f(n)$ is $\Theta(n^2)$

4) The function $f(n) = (\log n)^2 + 2n + 4n + \log n + 50$ belongs in which of the following complexity categories:

(a) $\Theta(\lg n)$ (b) $\Theta((\log n)^2)$ (c) $\Theta(n)$ ✓ (d) $\Theta(n \lg n)$ (e) $\Theta(n(\lg n)^2)$ (f) None of these

As $0 \leq \log n \leq (\log n)^2 \leq n$ for $n \geq 16$ (because: let $n = 2^m$, $(\log n)^2 = m^2 \leq 2^m = n$),

$f(n)$ is $O(n)$. And clearly $f(n)$ is $\Omega(n)$

So $f(n)$ is $\Theta(n)$

5) The function $f(n) = n + n^2 + 2^n + n^4$ belongs in which of the following complexity categories:

(a) $\Theta(n)$ (b) $\Theta(n^2)$ (c) $\Theta(n^3)$ (d) $\Theta(n \lg n)$ (e) $\Theta(n^4)$ (f) ✓ None of these

As $0 \leq n \leq n^2 \leq n^4 \leq 2^n$ for $n \geq 16$, $f(n)$ is $O(2^n)$

And clearly $f(n)$ is $\Omega(2^n)$

So $f(n)$ belongs none of these : $\Theta(n)$, $\Theta(n^2)$, $\Theta(n^4)$, $\underbrace{\Theta(n\lg n)}$

because $n \leq n\lg n \leq n^2$ for $n \geq 16$

6) What is the runtime (time complexity) of the below code? $O((len(array))^2)$, or can be written as $\boxed{O(n^2)}$

def printUnorderedPairs(array):	s/e	frequency	total Steps
for i in range(0,len(array)):	1	len(array)	same as frequency
for j in range(i+1,len(array)):	1	$len(array) \cdot \frac{1}{2}(len(array)-1)$	same as frequency
print(array[i] + "," + array[j])	1	$len(array) \cdot \frac{1}{2}(len(array)-1)$	same as frequency
		Total	$(len(array))^2$

7) What is the runtime of the below code? $O(len(arrayA) \cdot len(arrayB))$, or can be written as $\boxed{O(n^2)}$

def printUnorderedPairs(arrayA, arrayB):	s/e	frequency	total steps
for i in range(len(arrayA)):	1	len(arrayA)	〃
for j in range(len(arrayB)):	1	$len(arrayA) \cdot len(arrayB)$	〃
for k in range(0,100000):	1	$len(arrayA) \cdot len(arrayB) \cdot 100000$	〃
print(str(arrayA[i]) + "," + str(arrayB[j]))	1	$len(arrayA) \cdot len(arrayB) \cdot 100000$	〃
			$len(arrayA) + 200001 \cdot len(arrayA) \cdot len(arrayB)$

8) What is the runtime of the below code?

```
def powersOf2(n):
    # print("n:"+str(n))
    if n < 1:
        return 0
    elif n == 1:
        print(1)
        return 1
    else:
        prev = powersOf2(int(n/2))
        # print("prev:"+str(prev))
        print(prev)
        curr = prev*2
        print(curr)
        return curr
```

$T(n) = T(n/2) + c$

$T(n) \quad = \quad c + T(n/2) \quad = \quad c + c + T(n/4) \quad = \quad \underbrace{c + c + \cdots + c}_{c \log n}$ → Total Cost

$\boxed{O(\log n)}$