Solution

CMPS 340: Design and Analysis of Algorithms
Test 1

Contributes 15% to the final grade **Time: 75 min.**
Name: **ULID:**

1. **(10 marks)**

a) (4 marks) Is $5n^3 + 2n^2 + 5n + 3 \in O(n^3)$? Prove.

Yes.
let $f(n) = 5n^3+2n^2+5n+3$
$f(n) \in O(n^3)$ because
for all $n \geq 1$
$f(n) \leq cn^3$
where $c = 15$

b) (4 marks) Is $5n^3 + 2n^2 + 5n + 3 \in \Omega(n^3)$? Prove.

Yes. $f(n) \in \Omega(n^3)$ because
for all $n \geq 1$
$f(n) \geq cn^3$
where $c = 1$

c) (2 marks) Is $5n^3 + 2n^2 + 5n + 3 \in \Theta(n^3)$? Prove.

Yes.
As $f(n) \in O(n^3)$ AND $f(n) \in \Omega(n^3)$
$\therefore f(n) \in \Theta(n^3)$

2. **(15 marks)**

a) (10 marks) Solve the following recurrence relation and give a Θ bound. Assume $T(0) = 0$. (You do not need to show/prove how the Θ bound is obtained)
$T(n) = 2T(n-1) + 5$

$T(n) = 2T(n-1)+5$
$= 2[2T(n-2)+5]+5$
$= 2.2T(n-2)+2.5+5$
$= 2.2[2T(n-3)+5]+2.5+5$
$= 2.2.2T(n-3)+2.2.5+2.5+5$
$= 2^3 T(n-3)+5(1+2+2^2)$
$\vdots$
$= 2^i T(n-i)+5(1+2+2^2+\dots 2^{i-1})$
$= 2^i T(n-i)+5(2^{i-1+1}-1)$
$= 2^i T(n-i)+5(2^i-1)$
$= 2^n T(0)+5(2^n-1)$
$T(n) = 5\times 2^n - 5$
$T(n) \in \Theta(2^n)$

$T(n-1) = 2T(n-2)+5$
$T(n-2) = 2T(n-3)+5$

Given $T(0) = 0$
let $n-i = 0$
$i = n$

b) (5 marks) Is $16n^3 + 3n^2 + 5 \in o(n^4)$? Prove your answer.

Yes.

$\lim_{n\to\infty} \frac{16n^3+3n^2+5}{n^4}$
$= \lim_{n\to\infty} \left(\frac{16}{n}+\frac{3}{n^2}+\frac{5}{n^4}\right)$
$= 0$

$\therefore 16n^3+3n^2+5 \in o(n^4)$

3. **(10 marks)** which of the following algorithms is time efficient? Find their time complexity and order of complexity to answer this question. If there is any recursiveness in time complexity relation, show the steps how you remove the recursiveness. You do not need to show/prove how the order of complexity is obtained.

Algorithm 1	Algorithm 2

`int algo1 (int n)`
`{`
`  if (n<=1)`
`    return n;`
`  else`
`    return algo1(n-1)+algo1(n-2);`
`}`
	`int algo2 (int n)`
	`{`
	`  if (n<=1)`
	`    return n;`
	`  else`
	`    return algo2(n/2)+algo2(n/2);`
	`}`

Algorithm 2 is time efficient.

Algo 1

$$T(n) = T(n-1) + T(n-2)$$
$$T(n) \approx 2T(n-1)$$
$$= 2[2T(n-2)]$$
$$= 2 \cdot 2T(n-2)$$
$$= 2 \cdot 2 \cdot 2\, T(n-3)$$
$$= 2^3\, T(n-3)$$
$$\vdots$$
$$= 2^i\, T(n-i)$$
$$= 2^n\, T(0)$$
$$T(n) = 2^n$$
$$T(n) \in \Theta(2^n)$$

$T(n-1) = 2T(n-2)$

$T(n-2) = 2T(n-3)$

$T(0) = 1$

let $n-i=0$

$i = n$

Algo 2

$$T(n) = 2T\left(\frac{n}{2}\right)$$
$$= 2 \cdot 2T\left(\frac{n}{4}\right)$$
$$= 2 \cdot 2 \cdot 2T\left(\frac{n}{8}\right)$$
$$= 2^3\, T\left(\frac{n}{2^3}\right)$$
$$\vdots$$
$$= 2^i\, T\left(\frac{n}{2^i}\right)$$
$$= n\,T(1)$$
$$T(n) = n$$
$$T(n) \in \Theta(n)$$

$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right)$

$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right)$

$T(1) = 1$

let $\frac{n}{2^i} = 1$

$2^i = n$

$i = \log n$.

4. **(10 marks)** For the following pseudo-codes, what is the time complexity function ($T(n)$) and the order (Θ)? You can ignore the overhead operations and just count the basic operations. You do not need to show/prove how the Θ bound is obtained.

   a) (5 marks)

```
for ( i = 0 ; i <= n ; i++) {
    for ( j = 1 ; j <= n ; j++) {
        for ( k = 2 ; k <= n ; k++) {
            for ( l = 3 ; l <= n ; l++) {
                cout << i << j << k << l ;

            }
        }
    }
}
```

$T(n) = (n+1) * n * (n-1) * (n-2)$

$T(n) \in \Theta(n^4)$

   b) (5 marks)

```
for ( i = 1 ; i <= n ; i++) {
    for ( j = 0 ; j <= n) {
        cout << i << j ;
        j = j + floor(n/2) ;
    }
}
```

$T(n) = n * 3$

$T(n) \in \Theta(n)$