Sidharth Sudarsan
Email: lensman@my.yorku.ca
Course: EECS 2011 - Fundamentals of Data Structures
Instructor: Uyen Trang Nguyen

Assignment 1
ID: 216697120
Term: Winter 2020
Due Date: 29^{th} January, 2020

Question 1a
Perform an analysis of each of the following fragments and give a θ bound for the running time:

```
sum = 0;
for (int i = 0; i < n; i++) {
    for (int j = i; j >= i; j--) {
        sum++;
    }
}
```

Solution:

Doing a detailed running time analysis,

```
sum = 0;                                        #1
for (int i = 0; i < n; i++) {
    for (int j = i; j >= 0; j--) {              #Solved by untangling below
        sum++;                                  #1
    }
}
```

i	**j**
0	0
1	1, 0
2	2, 1, 0
3	3, 2, 1,0
.	
.	
.	
$n-1$	$n-1, n-2.....0$

The nested **for loop** runs n times, that evaluates to $\frac{n(n+1)}{2}$.

$Total : \frac{n(n+1)}{2} + 1.$

$\therefore \theta(n^2)$

Question 1b
Perform an analysis of each of the following fragments and give a θ bound for the running time:

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

Solution:

Doing a detailed running time analysis,

```
sum = 0;                                              #1
for (int i = 0; i < n; i++) {                         
    for (int j = 0; j < i * i; j++) {                 #Solved by untangling below
        sum++;                                        #1
    }
}
```

i	**j**
0	does not run
1	0
2	0, 1, 2, 3
3	0, 1, 2, 3, 4, 5, 6, 7, 8
.	
.	
.	
$n-1$	0, 1, 2...$((n-1)^2-1)$

The nested **for loop** runs n times, that evaluates to $\dfrac{n(n+1)}{2}$.

$Total : \dfrac{(n-1)(n)(2n-1)}{6} + 1.$

$\therefore \theta(n^3)$

Question 2a
Show that:
2^{n+1} is $O(2^n)$

Solution:

$f(n)$ is $O(g(n))$ if
$\exists c, n_0 > 0$
$f(N) \leq c(N)$ when $n \geq n_0$

we have,
$2^{n+1} \leq c \cdot 2^n$ when $n \geq 1$
$2 \cdot 2^n \leq 2 \cdot 2^n$ when $n \geq 1$

for $c = 2, n_0 = 1$

$\therefore 2^{n+1}$ is $O(2^n)$

Question 2b
Show that:
$\frac{n^2}{2} + n + 10$ is $\Omega(n^2)$

Solution:

$f(n)$ is $\Omega(g(n))$ if
$\exists c, n_0 > 0$
$f(n) \geq c(n)$ when $n \geq n_0$

we have,
$\frac{n^2}{2} + n + 10 \geq c^2$ when $n \geq 1$
$\frac{n^2}{2} + n + 10 \geq \frac{n^2}{2} + n^2 + 10n^2 \geq \frac{23}{2} \cdot n^2$ when $n \geq 1$

for $c = \frac{23}{2}, n_0 = 1$

$\therefore \frac{n^2}{2} + n + 10$ is $\Omega(n^2)$

Question 2c
Show that:
$n^{1.5}$ grows faster than $n log\, n$ using L'Hopital's rule

Solution:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^{1.5}}{nlog\, n} = \lim_{n\to\infty} \frac{n \cdot \sqrt{n}}{nlog\, n}$$

Using L'Hopital's rule,

$$\lim_{n\to\infty} \frac{n \cdot \sqrt{n}}{nlog\, n} = \lim_{n\to\infty} \frac{n}{2\sqrt{n}} = \lim_{n\to\infty} \frac{\sqrt{n}}{2} = \infty$$

Since the limit evaluates to ∞,
$g(n)$ is $o(f(n))$
$\implies f(n)$ grows faster than $g(n)$

$\therefore n^{1.5}$ grows faster than $nlog\, n$

Question 2d
Show that:
$log^k\, n$ is $o(n)$ for any constant k

Solution:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{(log\, n)^k}{n}$$

Using L'Hopital's rule,

$$\lim_{n\to\infty} \frac{(log\, n)^k}{n} = \lim_{n\to\infty} \frac{k(log\, n)^{k-1}}{n} = \lim_{n\to\infty} \frac{k(k-1)(log\, n)^{k-2}}{n} = \lim_{n\to\infty} \frac{k(k-1)(k-2)(log\, n)^{k-3}}{n} .....$$

Differentiating up to,

$$\lim_{n\to\infty} \frac{(log\, n)^{(k-(k-1))}}{n} = \lim_{n\to\infty} \frac{(log\, n)}{n} = \lim_{n\to\infty} \frac{1}{n} = 0$$

Since the limit evaluates to 0,
$f(n)$ is $o(g(n))$

$\therefore\, log^k n$ is $o(n)$

Question 2e
Show that:
2^{10n} is not $O(2^n)$

Solution:

This is a proof by *contradiction*.
Let us assume 2^{10n} is $O(2^n)$

$\exists c, n_0 > 0$
$2^{10n} \leq c \cdot 2^n$ when $n \geq n_0$

Dividing both sides by 2^n,
$2^{9n} \leq c$

There is a contradiction.
2^{9n} is not bounded by some constant c.
There are no values for c and n_0 that can make the above condition true.

$\therefore\, 2^{10n}$ is not $O(2^n)$

Question 3a
Solve the following recurrences by obtaining a θ bound for $T(n)$ given that $T(1) = \theta(1)$:

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

Solution:

We have,

$$\begin{aligned}
T(n) &= 2n - 1 + T(n-1) \\
T(n-1) &= 2(n-1) - 1 + T(n-2) \\
&= 2n - 3 + T(n-2) \\
T(n-2) &= 2(n-2) - 1 + T(n-3) \\
&= 2n - 5 + T(n-3) \\
T(n-3) &= 2(n-3) - 1 + T(n-4) \\
&= 2n - 7 + T(n-4)
\end{aligned}$$

Using the $T(n-1)$ expansion,

$$\begin{aligned}
T(n) &= 2n - 1 + 2n - 3 + T(n-2) \\
&= 2(2n-2) + T(n-2)
\end{aligned}$$

Using the $T(n-2)$ expansion,

$$\begin{aligned}
T(n) &= 2(2n-2) + 2n - 5 + T(n-3) \\
&= 3(2n-3) + T(n-3)
\end{aligned}$$

Using the $T(n-3)$ expansion,

$$\begin{aligned}
T(n) &= 3(2n-3) + 2n - 7 + T(n-4) \\
&= 4(2n-4) + T(n-4)
\end{aligned}$$

$$\implies T(n) = k(2n-k) + T(n-k)$$

when $n = k + 1$,

$$\begin{aligned}
\implies T(n) &= k(2(k+1) - k) + T(k+1-k) \\
&= k(k+2) + T(1) \\
&= k(k+2) + \theta(1)
\end{aligned}$$

Substituting $k = n - 1$,

$$\begin{aligned}
\implies T(n) &= (n-1)(n-1-2) + \theta(1) \\
&= n^2 - 1 + \theta(1)
\end{aligned}$$

$\therefore \theta(n^2)$

Question 3b
Solve the following recurrences by obtaining a θ bound for $T(n)$ given that $T(1) = \theta(1)$:

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

Solution:

We have,

$$T(n) = n + T(n-3)$$
$$T(n-3) = n - 3 + T(n-6)$$
$$T(n-6) = n - 6 + T(n-9)$$
$$T(n-9) = n - 9 + T(n-12)$$

Using the $T(n-3)$ expansion,

$$T(n) = 2n - 3 + T(n-6)$$

Using the $T(n-6)$ expansion,

$$T(n) = 3n - 9 + T(n-9)$$

Using the $T(n-9)$ expansion,

$$T(n) = 4n - 18 + T(n-12)$$

There is a series in the pattern above:
3(1), 3(3), 3(6), 3(10), ...

It can be written as:
3(1), 3(1 + 2), 3(1 + 2 + 3), 3(1 + 2 + 3 + 4), ...

So the k^{th} element of this series is $\frac{3k(k-1)}{2}$

$$\implies T(n) = (k)(n) - \frac{3k(k-1)}{2} + T(n-k)$$

when $n = k + 1$,

$$\implies T(n) = k(k+1) - \frac{3k(k-1)}{2} + T(k+1-k)$$
$$= k(k+1) - \frac{3k(k-1)}{2} + \theta(1)$$

Substituting $k = n - 1$,

$$\implies T(n) = (n-1)(n) - \frac{3(n-1)(n-2)}{2} + \theta(1)$$

$\therefore \theta(n^2)$

Question 3c

Solve the following recurrences by obtaining a θ bound for $T(n)$ given that $T(1) = \theta(1)$:

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

Solution:

We have,

$$T(n) = n^2 + T(n-1)$$
$$T(n-1) = (n-1)^2 + T(n-2)$$
$$T(n-2) = (n-2)^2 + T(n-3)$$
$$T(n-3) = (n-3)^2 + T(n-4)$$

Using the $T(n-1)$ expansion,

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

Using the $T(n-2)$ expansion,

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

Using the $T(n-3)$ expansion,

$$T(n) = n^2 + (n-1)^2 + (n-2)^2 + (n-3)^2 + T(n-4)$$

$$\implies T(n) = T(n-k) + (n-k+1)^2 + (n-k+2)^2 + ... + (n-1)^2 + n^2$$

when $n = k+1$,

$$\begin{aligned}\implies T(n) &= T(1) + (2)^2 + (3)^2 + ... + n^2 \\ &= \theta(1) + (2)^2 + (3)^2 + ... + n^2 \\ &= \theta(1) + \frac{n(n+1)(2n+1)}{6} - 1^2\end{aligned}$$

$\therefore \theta(n^3)$

Question 4
Mergesort does have a worst-case time of $\theta(nlog\, n)$ but its overhead (hidden in the constant factors) is high and this is manifested near the bottom of the recursion tree where many merges are made. Someone proposed that we stop the recursion once the size reaches k and switch to insertion sort at that point. Analyze this proposal (by modifying the recurrence analysis of standard mergesort) and prove that its running time is $\theta(nk + nlog\,(\frac{n}{k}))$.

Solution:

From the standard mergesort analysis,

$$T(n) = 2^i T(\frac{n}{2^i}) + in$$

Since the algorithm switches to insertion sort once the size reaches k, the standard merge sort analysis has to be adjusted to account for the worse running time of insertion sort.

So $T(\frac{n}{2^i})$ should be replaced with $T(k)$ for size k.

$$\implies \frac{n}{2i} = k$$

$$2^i = \frac{n}{k}$$

$$i = log\,\frac{n}{k}$$

Substituting the value of i in the standard mergesort analysis,

$\implies T(n) = 2^{log\,\frac{n}{k}} T(k) + nlog\,\frac{n}{k}$

Since $T(k)$ is the running time for insertion sort for size k or lesser, $T(k) = k^2$

$T(n) = \frac{n}{k}(k^2) + nlog\,\frac{n}{k} = nk + nlog\,\frac{n}{k}$

$\implies T(n) = nk + nlog\,\frac{n}{k}$

$\therefore$ *The running time is* $\theta(nk + nlog\,\frac{n}{k})$