Algorithm Assignment Summer 19’ Tae Myles

Algorithm-1

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	n + 1
3	1	$\sum_{i=1}^{n}(i) + 1$
4	1	$\sum_{i=1}^{n}(i)$
5	1	$\sum_{i=1}^{n}\sum_{j=1}^{i}(j) + 1$
6	6	$\sum_{i=1}^{n}\sum_{j=1}^{i}(j)$
7	5	$\sum_{i=1}^{n}(i)$
8	2	1

Multiply col.1 with col.2, add across rows and simplify

$T_1(n) = 1 + n + 1 + \sum_{i=1}^{n}(i) + 1 + \sum_{i=1}^{n}(i) + \sum_{i=1}^{n}\sum_{j=1}^{i}(j) + 1 + 6\sum_{i=1}^{n}\sum_{j=1}^{i}(j) + 5\sum_{i=1}^{n}(i) + 2$

$= 1 + n + 1 + \frac{n(n+1)}{2} + 1 + \frac{n(n+1)}{2} + \frac{1}{6}n(n+1)(n+2) + 1 + 6\frac{1}{6}n(n+1)(n+2) +$
$5\frac{n(n+1)}{2} + 2$

$= 6 + n + \frac{n(n+1)}{2} + \frac{n(n+1)}{2} + \frac{1}{6}n(n+1)(n+2) + n(n+1)(n+2) + 5\frac{n(n+1)}{2}$

$= 6 + n + 7\frac{n(n+1)}{2} + \frac{7}{6}n(n+1)(n+2)$

$= 6 + n + 7\frac{n(n+1)}{2} + \frac{7}{6}\left(n(n^2 + 3n + 2)\right)$

$= 6 + n + 7\frac{n(n+1)}{2} + \frac{7}{6}(n^3 + 3n^2 + 2n)$

$= 6 + n + \frac{7}{2}n^2 + \frac{7}{2}n + \frac{7}{6}n^3 + \frac{21n^2}{6} + \frac{14n}{6}$

$= \frac{7}{6}n^3 + \frac{42n^2}{6} + \frac{41}{6}n + 6$

$= \frac{7}{6}n^3 + 7n^2 + \frac{41}{6}n + 6$

$T_1(n) = O(n^3)$

Algorithm-2

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	n + 1
3	1	n
4	1	$\sum_{i=1}^{n}(i) + 1$
5	6	$\sum_{i=1}^{n}(i)$
6	5	$\sum_{i=1}^{n}(i)$
7	2	1

Multiply col.1 with col.2, add across rows and simplify

$T_2(n) = 1 + n + 1 + n + \sum_{i=1}^{n}(i) + 1 + 6\sum_{i=1}^{n}(i) + 5\sum_{i=1}^{n}(i) + 2$

$= 5 + 2n + \frac{n(n+1)}{2} + 6\frac{n(n+1)}{2} + 5\frac{n(n+1)}{2}$

$= 5 + 2n + 12\frac{n(n+1)}{2}$

$= 5 + 2n + \frac{12n^2}{2} + \frac{12n}{2}$

$= 6n^2 + 8n + 5$

$T_2(n) = O(n^2)$

Algorithm-3

Step	Cost of each execution	Total # of times executed
1	4	1
2	11	1
Steps executed when the input is a base case: **Steps 1 or 2**		
First recurrence relation: T(n=1 or n=0) = T(n=1): 11, T(n=0):4,		
3	5	1
4	2	1
5	1	$\frac{n}{2} + 1$
6	6	$\frac{n}{2}$
7	5	$\frac{n}{2}$
8	2	1
9	1	$\frac{n}{2} + 1$

10	6	$\frac{n}{2}$
11	5	$\frac{n}{2}$
12	4	1
13	5	Lgn (cost excluding the recursive call)
14	6	Lgn (cost excluding the recursive call)
15	6	1
Steps executed when input is NOT a base case: Steps 1 - 15		
Second recurrence relation: T(n>1) = 2T(n/2) + c		
Simplified second recurrence relation (ignore the constant term): $T(n>1) = 2^i(T(n/2^i) + (2^i - 1)$		

Solve the two recurrence relations using any method (recommended method is the Recursion Tree). Show your work below:

c = 12n + 38

$T_3(n) = 2T(n/2) + c$
$= 2(2T(n/4) + c) + c$
$= 4(T(n/4) + 2c + c$
$= 4(2T(n/8) + c) + 2c + c$
$= 8(T(n/8) + 4c + 2c + c$

$= 2^i(T(n/2^i) + (2^i - 1)c$

$= 2^{lgn}T(1) + (2^{lgn} - 1)(12n + 38)$
$= 11n + (12n + 38)(n - 1)$
$= 11n + 12n^2 + 38n - 12n - 38$
$= 12n^2 + 37n - 38$

$T_3(n) = O(n^2)$

Algorithm-4

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	1
3	1	n + 1
4	8	n
5	5	n
6	2	1

Multiply col.1 with col.2, add across rows and simplify

$$
\begin{aligned}
T_4(n) &= 1 + 1 + n + 1 + 8n + 5n + 2 \\
&= 5 + n + 13n \\
&= 14n + 5
\end{aligned}
$$

$T_4(n) = O(n)$