Jonathan White

Algorithm-1

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	n
3	1	$\sum_{i=0}^{n} \sum_{j=i}^{n} (n-1) = n(n+1)$
4	1	$n(n)$
5	1	$\sum_{i=0}^{n} \sum_{j=i}^{n} \sum_{k=1}^{n} n = n^3$
6	6	n^3
7	5	n^3
8	2	1

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

$T_1(n) = 1 + n + n(n+1) + n^2 + n^3 + 6n^3 + 5n^3 + 2 = 3 + 12n^3 + 2n^2 + 2n = O(n^3)$

Algorithm-2

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

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

$T_2(n) = 1 + n + 2 + n + 1 + (n(n+1)) + 6(n(n+1)) + 4(n(n+1)) = (2n+4) + 11(n(n+1))$

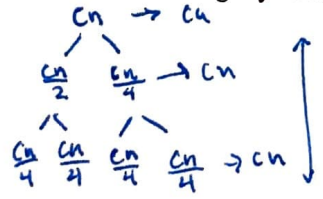
$* \; 11(n^2 + n) = 11n^2 + 11n \rightarrow 11n^2 + 13n + 4 = O(n^2)$

Algorithm-3

Step	Cost of each execution	Total # of times executed in any single recursive call
1	5	1
2		1
Steps executed when the input is a base case:		
First recurrence relation: T(n=1 or n=0) =		
3	5	1
4	2	1
5	1	$n/2$
6	5	$n/2 - 1$
7	5	$n/2 - 1$
8	2	1
9	1	$n/2$
10	5	$n/2 - 1$
11	5	$n/2 - 1$
12	4	1
13	1	(cost excluding the recursive call) 1
14	1	(cost excluding the recursive call) 1
15	4	1
Steps executed when input is NOT a base case: 3 - 15		
Second recurrence relation: T(n>1) = $T(n/2) + T(n/2)$		
Simplified second recurrence relation (ignore the constant term): T(n>1) = $2T(n/2)$		

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

$T_3(n) =$



$5 + 2 + n/2 + (n/2 - 1) + 5 + 5(n/2 - 1)$

$2 + n/2 + 5(n/2 - 1) + 5(n/2 - 1) + 4$

$= 13 + n + 25(\frac{n}{2} - 1)$

$13 + n + \frac{25n - 50}{2} + (n \lg n)$

$= O(n \lg n)$

Algorithm-4

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

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

$T_4(n)$ =

$n + 6n - 6 + 4n - 4 + 2 + 1 + 1 = 11n - 6$

$O(n)$