- chegg.com
- algorist.com

William Kelley

(1) $$\sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{n=j+0}^{j+i}$$

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

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

$$= \frac{n(n+1)}{6}\left(\frac{2n+1+3}{2n+4}\right) = \frac{n(n+1)(n+2)}{3} = \frac{n^3+3n^2+2n}{3}$$

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

(2) $n^2 \leq n^3$

$n^2 + 3n^3 \leq 4n^3, n > 1$

$\therefore n^2 + 3n^3 = O(n^3)$ with $c = 4, n_0 = 1$

Big-O of $f(n) = O(n^3)$

Given $f(n) = n^2 + 3n^3$

$0 \leq cn^2 \leq 3n^3$

$cn^2 \leq 3n^3 \Rightarrow c = 1$ and $n_0 = 1$

Theta, Ω of $f(n) = \Omega(n^3)$

(3) $a^{n+1} \rightarrow \Theta(a^n)$

Once the threshold is reached, this becomes similar to the argument for $\infty = \infty + 1$, the addition of 1 to such a large number the 1 becomes virtually insignificant therefore the argument stands that the upper bound of a^{n+1} would yield $\Theta(a^n)$ at the middle bound.

(4) $O(n^2)$

The algorithm leads to worst case efficiency when it runs for a long time if the undirected graph is connected linearly as per the given node.

+ For example, if "n" is the # of vertices then the algorithm runs $1+2+3+\ldots n-1$ times.
+ So, according to big-oh notation the running time of the algorithm leads to $O(n^2)$

(5)