

	There is no particular relationship between the two concepts										
	Best Case, Worst Case and Expected Case actually describe the big O or big Theta time for particular scenarios whereas these asymptotic notations describe the upper, lower and tight bounds for the runtime										
	Space complexity										
	Memory or space required by an algorithm	to create an array - if it is unidimensional, $O(N)$ space complexity; for a 2-D array, $O(N^2)$									
	Stack space in recursive calls counts too. Each call adds a level to the stack and takes up actual memory.	However, just because you have N calls does not mean it will take $O(N)$ time: check the example on Page 41 for more details									
	Drop the constants										
	$O(2N)$ is actually $O(N)$										
	Drop the non-dominant terms										
	$O(N^2 + N)$ becomes $O(N^2)$										
	$O(N + \log N)$ becomes $O(N)$										
	$O(5 \cdot 2^N + 1000N^{100})$ becomes $O(2^N)$										
	$O(x!) > O(2^x) > O(x^2) > O(x \log x) \dots > O(x)$										
	Multi-Parts algorithms: add versus multiply										
	Add:	Non-nested chunk of work A and B	$O(A + B)$	"DO THIS THEN WHEN YOU ARE ALL DONE, DO THAT"							

	Multiply	Nested A and B	$O(AB)$	"DO THIS FOR EACH TIME YOU DO THAT"							
--	----------	----------------	---------	---	--	--	--	--	--	--	--