### Week 02 Quiz

# Plagiarism declaration

By submitting work for this quiz I hereby declare that I understand the University's policy on <u>academic integrity</u> (<a href="https://academicintegrity.unimelb.edu.au/">https://academicintegrity.unimelb.edu.au/</a>) and that the work submitted is original and solely my work, and that I have not been assisted by any other person (collusion) apart from where the submitted work is for a designated collaborative task, in which case the individual contributions are indicated. I also declare that I have not used any sources without proper acknowledgment (plagiarism). Where the submitted work is a computer program or code, I further declare that any copied code is declared in comments identifying the source at the start of the program or in a header file, that comments inline identify the start and end of the copied code, and that any modifications to code sources elsewhere are commented upon as to the nature of the modification.

**Due** Mar 19 at 23:59

Points 10

**Questions** 10

**Available** Mar 10 at 10:00 - Mar 19 at 23:59 10 days

Time Limit None

Allowed Attempts Unlimited

### Instructions

You should attempt the quiz after the lectures and your tutorial.

You must complete the weekly quiz by the end of the 'next week' -- ie Week 2 quiz must be completed by the end of Week 3

- You may attempt the quiz multiple times (if you happen to get a question wrong, you can do it again)
- Your score on the quiz will be recorded in the grade book -- there are 10 weekly quizzes, each worth
   1%
- The quiz might not display equations correctly in some browsers. If you experience problems, we recommend that you try a different browser.

This quiz was locked Mar 19 at 23:59.

# **Attempt History**

**Attempt** 

Time

**Score** 

	Attempt	Time	Score
LATEST	Attempt 1	43 minutes	9 out of 10

Score for this attempt: 9 out of 10

Submitted Mar 14 at 7:01

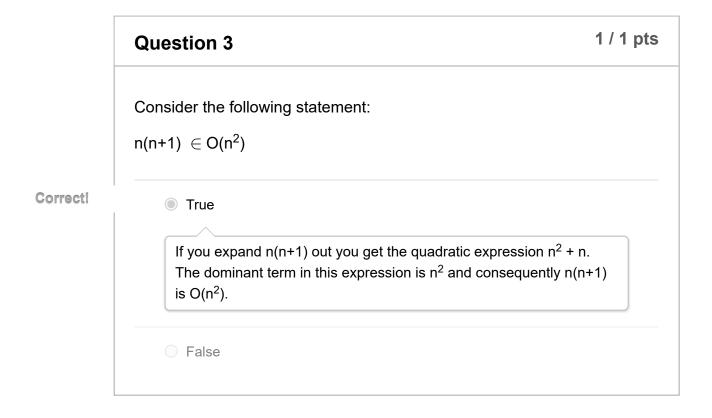
This attempt took 43 minutes.

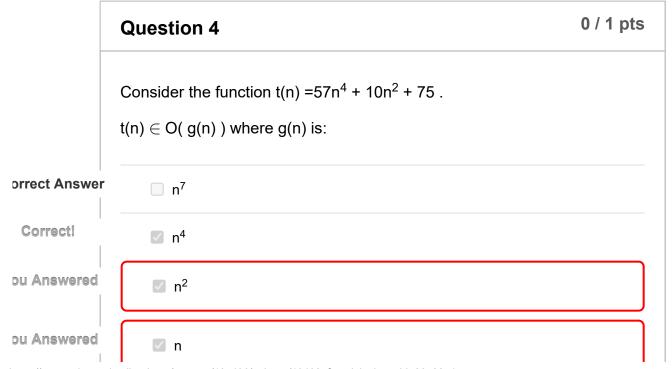
my machine, a certain O(n <sup>2</sup> ) sorting algorithm takes one second at 1000 random items. Sorting 100,000 random items can be expetake:  1-2 minutes	
10.15 minutes	
10-15 minutes	
2-3 hours	
about one day	
<ul> <li>almost one year</li> </ul>	
That's right. We would expect 100 times as many elements to to $100 \times 100 = 10,000$ times as long to be sorted.	ake
	about one day almost one year  That's right. We would expect 100 times as many elements to ta

# Question 2 If f(n) is O(g(n)) then f(n) grows asymptotically no faster than g(n). Correct! True

A function f(n) is O(g(n)) if g(n) defines an upper bound on the growth of f(n) as n becomes large

False





#### **Question 5**

1 / 1 pts

Which of the following claims about growth rate are correct:

- $\bigcirc (2n \log_2 n)^2 \in O(n^2)$
- $\sqrt{n} \epsilon O(\log_{10} n)$
- $\bigcirc \log_2 n \ \epsilon \ O\left(\frac{1}{\sqrt{n}}\right)$

Correct!

- $\bigcirc$   $3n^3 + n\sqrt{n} \epsilon O(n^3)$
- $\sqrt{n} \epsilon O(\sqrt[3]{n})$

That's right, only one of the statements is correct.

## **Question 6**

1 / 1 pts

Which of the following claims are correct:

Correct!

- $lacksquare 2^{n+1} \; \epsilon \; \; \Theta \left( 2^n 
  ight)$

Correct!

$$extbf{2} 3n^3 + n\sqrt{n} \epsilon O\left(n^3\right)$$

That's right, only one of the statements is correct.

**Question 7** 1 / 1 pts Which of the following claims are correct: Correct! lacksquare  $\sum_{i=1}^{n} 3^i \ \epsilon \ \Theta\left(3^{n-1}
ight)$ Correct!  $lacksquare \sum_{i=1}^n 3^i \; \epsilon \; \Theta\left(3^n
ight)$ Correct!  $lacksquare \sum_{i=1}^n 3^i \; \epsilon \; \Theta\left(3^{n+1}
ight)$  $\square \sum_{i=1}^n 2^i \ \epsilon \ \Theta(3^n)$ That's right.



That's correct!

 $2^{2n}$  is equivalent to  $2^n$   $2^n$ . Consequently,  $2^n$  is not an upper bound on  $2^{2n}$ !

Question 9 1 / 1	pts
Given an array of n items, A, what is the Big-⊖ complexity of retrieving element at index k in the array (A[k])?	the
Θ(n)	
⊕(1)	
Θ(kn)	
Θ(logkn)	
Correct!  Accessing the element at index k in an array is a constant time operation it does not depend on the number of items in the array.	
	Given an array of n items, A, what is the Big- $\ominus$ complexity of retrieving element at index k in the array (A[k])? $\ominus$ (n) $\ominus$ (kn) $\ominus$ (logkn)  Correct!  Accessing the element at index k in an array is a constant time operation it does

# 

Correct!

	n <sup>2</sup>
	n
	Both n and n²
Corre	ect!
Reca	all that a function $f(n)$ is $\Theta(g(n))$ if and only if it is both $\Omega(g(n))$ and $O(g(n))$ .

Quiz Score: 9 out of 10