Week 05 Quiz

Plagiarism declaration

By submitting work for this quiz I hereby declare that I understand the University's policy on **academic integrity (https://academicintegrity.unimelb.edu.au/)** 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 Apr 11 at 23:59 **Points** 8 **Questions** 8
Available Mar 31 at 10:00 - Apr 11 at 23:59 12 days **Time Limit** None
Allowed Attempts Unlimited

Instructions

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

- 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.
- The quiz might not display equations correctly in some browsers. If you experience problems, we recommend that you use Firefox.

Note: you must complete at least eight of the weekly quizzes to meet one of the hurdle requirements in this subject

This quiz was locked Apr 11 at 23:59.

Attempt History

	Attempt	Time	Score

	Attempt	Time	Score
LATEST	Attempt 1	77 minutes	8 out of 8

Score for this attempt: **8** out of 8
Submitted Apr 11 at 1:55
This attempt took 77 minutes.

Question 1

1 / 1 pts

A sorting algorithms is considered stable if it

Correct!

- (selected) preserves the relative order of any two equal elements in its input
- preserves the relative order of most two equal elements in its input
- perform its operations mostly in the same memory used by its input elements
- can change the relative order of equal elements in its input
- its worst case efficiency is in O(n log n)

Think back to the examples used in the lectures.

Create an array with duplicate values (perhaps labelled with a subscript). Sort the array. What happens to the relative ordering of the array elements?

Question 2

1 / 1 pts

Assume we want to sort integers into ascending order. To sort a small array that contains 42, 17, selection sort will perform one swap, or three assignments. How many assignments will it perform to sort an array that contains these 10 elements: 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 ?

- 15
- 18
- 24

Correct!

- (•) 27
- 63

Yes, that's right. First five swaps sort the array, but four more swaps are still performed. Selection sort does very little data movement.

Question 3

1 / 1 pts

Which of the following statements about **selection sort** are **true**? Note that you may select multiple statements.

- Selection sort is a stable sorting algorithm.

Correct!

- [x] Selection sort is an in-place sorting algorithm.
- Selection sort has a worst case time complexity of O(nlogn).

Correct!

- [x] The default implementation of selection sort is not stable.

Correct!

- [x] Selection sort has a worst case time complexity of $O(n^2)$.

Statements 1 and 3 are false. Selection sort is not a stable sorting algorithm by default, although it could be modified to become stable. A sorting algorithm is stable if the relative order of items with the same key in the resulting sorted list is the same as in the original, unsorted list. Let's consider an example.

Consider the following array of numbers: 5 3 4 5 0.

In our first pass of selection sort, we swap 0 (in the last position of our array) with 5 (sitting in the first position of our array). The result is the array: 0 3 4 5 5. Note here that the relative order of the two 5 keys has changed!

Selection sort is an in-place algorithm, as is does not need to create auxilliary storage in which to build the sorted version of the original array.

Let's now consider the worst case time complexity of selection sort. The number of comparisons performed by selection sort is always n(n-1)/2, irrespective of how the items are initially distributed in the array. For each element in our array, excluding the last, we must find the smallest element in the remainder of the array and perform a swap. The number of comparisons performed is consequently (n-1) + (n-2) + ... + 2 + 1 = n(n-1)/2.

Question 4

1 / 1 pts

Assume we want to sort integers into ascending order. To sort a small array that contains 42, 17, insertion sort will perform three assignments. How many assignments will insertion sort perform to sort an array that contains these 10 elements: 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 ?

- [ ] 15

- 18
- 24
- 55

Correct!

- (●) 63

Yes, that's right. The number of assignments is 3 + 4 + ... + 11 = 63.

Question 5

1 / 1 pts

What is the **best case** time complexity of **insertion sort**? Please select the tightest Big-O bound of those listed below.

Correct!

- (●) O(n)
- O(nlogn)
- $O(n^2)$
- $O(2^n)$

Let's consider the best case scenario for insertion sort -- an array that is already in sorted order. In this case, each element in the array need only be compared with the element to the left of it. The best case time complexity of insertion sort is consequently O(n).

Question 6

1 / 1 pts

Assume we want to use shellsort to sort integers into ascending order. We want to apply 4-sorting, followed by 1-sorting, to an array that contains 9, 8, 7, 6, 5, 4, 3, 2, 1, 0. Just before the last round of sorting (which is insertion sort), what does the array look like?

- 0, 2, 4, 6, 1, 3, 5, 7, 8, 9
- 6, 7, 8, 9, 2, 3, 4, 5, 0, 1
- 4, 3, 9, 8, 2, 1, 7, 6, 0, 5

Correct!

- (selected) 1, 0, 3, 2, 5, 4, 7, 6, 9, 8
- 1, 5, 0, 4, 3, 7, 2, 6, 8, 9

Yes, well done. Before the last round, the array is "almost-sorted".

Question 7

1 / 1 pts

If the DFS tree does not have any back edges, then there are no cycles in the graph.

Correct!

- (selected) True
- False

Question 8

1 / 1 pts

The time complexity of DFS on an undirected graph using adjacent list representation is:

Correct!

- (●) $O(|V| + |E|)$
- ( ) $O(|V|)$
- ( ) $O(|V|^2)$
- ( ) $O(\log V)$

Think about the number of nodes that have to be processed. Does a given node have a neighbour?

Quiz Score: **8** out of 8