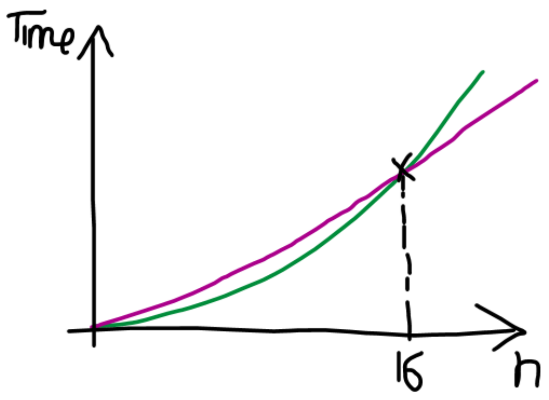
Question 1

Time Complexity

The following graph shows the time it takes for Merge Sort (Green) and Insertion Sort (Purple) to sort an array containing n elements. Assume that this time corresponds to the worst-case scenario of both algorithms.



For an array of size = 8, which of the two sorting algorithms would you prefer and why?

What if the array was of size = 20?

Array of size = 8
Array of size = 15

Question 2

We are given an array, and using it we want to compare the time performance of various sorting algorithms with each other.

For each comparison, does the array being sorted correspond to the same case scenario for both (i.e. both best case, both worst case or both neither), or different case scenarios (i.e. it is best case for one, worst case for other, etc.)?

Arr1 = [1, 2, 3, 4, 5]

First comparison
I run quick sort and merge sort on arr1.

Second comparison
I run insertion sort and merge sort on arr1.

Third comparison
I run insertion sort and quick sort on arr1.

Question 3

The Unsorted Map ADT is one that we have covered in the course. Its implementation is called different things in different languages, for e.g. it is referred to as a `Dictionary` in Python, `unsorted_map` in C++, etc.

Its key defining behaviors are as follows. M is the unsorted map, and (k, v) is the key-value pair.

- add/update(M, k, v)
- search(M, k)
- delete(M, k)

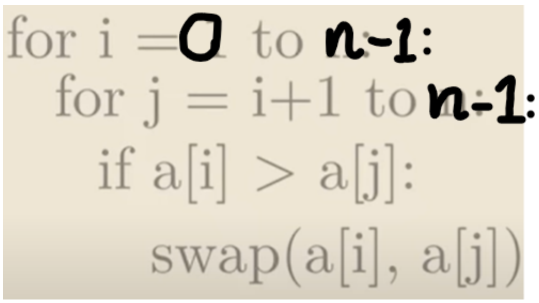
Generally, unsorted maps are implemented using hash tables.
Can you implement them using simple arrays? Why or why not? Explain in detail.

Question 4 (difficult, hence not included in the exam, but can serve for good practice).

Hint: for this question you would need to understand the underlying logic of sorting algorithms.

The sorting algorithm shown below is Selection Sort. The difference here is that swapping is occurring every time a minimum value is found in the unsorted part of the array, instead of waiting till the end.

Note: when watching the video, be vary of the indices. They assume that array indices start from 1.



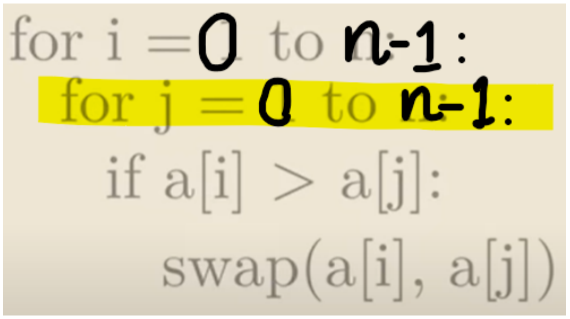
```
for i = 0 to n-1:
    for j = i+1 to n-1:
        if a[i] > a[j]:
            swap(a[i], a[j])
```

Here is an unsorted array. Dry run through it to visualize the sorting process.

Arr1 = [4, 10, 3, 5, 9, 8, 1]

Dry run through the same array, if the following change is made to the inner loop, to visualize the process.
Mark the sorted part of the array.

Hint: it would be easier to see the pattern after the first few outer passes (especially after the third or fourth pass). It would probably become even more clearer towards the last few outer passes.



```
for i = 0 to n-1:
  for j = 0 to n-1:
    if a[i] > a[j]:
      swap(a[i], a[j])
```

What do you find this process most similar to - selection sort, insertion sort, bubble sort?

*Check this reference **AFTER** you have tried solving it yourself.*
The Simplest Sorting Algorithm (You've Never Heard Of) - YouTube