Q5:

1.Worst-case analysis:

In computer science, the worst-case analysis measures the running time memory that an algorithm requires given an input of arbitrary size (commonly denoted as *n* or N). It gives an upper bound on the resources required by the algorithm.

In the case of running time, the worst-case time-complexity indicates the longest running time performed by an algorithm given any input of size *n*, and thus guarantees that the algorithm will finish in the indicated period of time. The order of growth of the worst-case complexity is commonly used to compare the efficiency of two algorithms.

The worst-case complexity of an algorithm should be contrasted with its [average-case complexity](https://en.wikipedia.org/wiki/Average-case_complexity), which is an average measure of the number of resources the algorithm uses on a random input.

**Conclusion:  The behavior of the algorithm with respect to the worst possible case of the input instance. The worst-case running time of an algorithm is an upper bound on the running time for any input.**

2.Amortized analysis:

In [computer science](https://en.wikipedia.org/wiki/Computer_science), amortized analysis is a method for [analyzing](https://en.wikipedia.org/wiki/Analysis_of_algorithms) a given algorithm's [complexity](https://en.wikipedia.org/wiki/Computational_complexity_theory), or how much of a resource it takes to [execute](https://en.wikipedia.org/wiki/Execution_(computing)). The motivation for amortized analysis is, that looking at the worst-case run time can be too pessimistic. Instead, amortized analysis averages the running times of operations in a sequence over that sequence.

For a given operation of an algorithm, certain situations may imply a significant cost in resources, whereas other situations may not be as costly. The amortized analysis considers both the costly and less costly operations together over the whole sequence of operations.

**Conclusion: Here the time required to perform a sequence of (related) operations is averaged over all the operations performed. Amortized analysis can be used to show that the average cost of an operation is small, if one averages over a sequence of operations, even though a simple operation might be expensive. Amortized analysis guarantees the average performance of each operation in the worst case.**

3.The intuitive difference between the worst-case analysis and the amortized analysis.

According to the definition of worst-case analysis and amortized analysis, we can know that worst case analysis is base on the worst input and getting the worst situation in order to get the running time that a program can use most time to execute, while the amortized one more focusing on the average situation and get a normal execution time. From the slide, we can also get the formula that: Amortized running time of ADT operations = worst-case running time of N operations / N.

We should choose worst-case analysis when we are going to calculate which method is faster and more efficiency. For example, when calculating the efficiency of quicksort and heapsort, we should use that analysis to get the conclusion that quicksort is faster

We should choose amortized analysis when we are going to calculate what time a program usually needs to spend on a normal input. It’s better if we are going to calculate an average efficiency in a year.

4.Example of an Abstract Data Type implementation for which worst-case analysis is better suited, and one where amortized analysis is more suitable.

(1) For stacks and queues, using worst-case analysis is better suited.

(2) For array lists and notably vectors, using amortized analysis is better suited.