Applied Programming
Fall 2017-18
Quiz # 1

Roll Number:

Q. 1. Suppose assignment statement takes 1 time unit to run, whereas reading from memory takes 3 time units, comparing two numbers takes 2 time units and addition takes 4 time units. If n is the size of the array A, then find the exact number of time units required to complete the following pseudocode. Then, find the best Big-Oh bound on the worst case running time.

```
sum = 0;
for i = 1 to n
    do
        if A[i] > 0 then
            sum = sum + A[i]
        end if
end for
```

Solution:

Sum = 0 makes an assignment to memory (1 time unit)
Loop initialization I = 0 is done once (1 time unit)
Every iteration of the loop reads the value of I from memory, adds one to it and stores it back to memory, performs a comparison (10 time units). This is incurred n+1 times
The if statement adds an offset to the starting address of array A, then reads from memory and performs a comparison (9 time units). This statement is repeated n times
The statement inside the if statement fetches A[i] in 7 time units, reads the value of sum in 3 time units, performs an addition in 4 time units and then assigns to sum in 3 time units. (17 time units). This is incurred a variable number of times. In the worst case, it is repeated n times.
So, the total time is : $2 + 10(n+1) + 9n + 17n = O(n)$

Applied Programming

Fall 2017-18

Quiz # 1

Roll Number:

Q. 1. Suppose we have a singly linked list with a head pointer that points to the first element in the list and a tail pointer that points to the last element in the list. If n is the number of nodes in the linked list, what is the time complexity, in Big-Oh notation, of deleting the last node in the linked list. Note: The obvious answer may not be the right answer.

Solution:

As discussed in class, you may delete the node in constant time through the tail pointer. However, we need the tail pointer pointing to the new last node so that every subsequent delete of last node can be performed just as easily. However, to get to the new last element and update the tail pointer, requires O(n) time.

Applied Programming
Fall 2017-18
Quiz # 1

Roll Number:

Q. 1. We wish to show that n lg n is $O(n^2)$ by using the definition of Big-Oh. Suppose we pick $n_0 = 2$. Pick an appropriate positive value for c to show that n lg n is $O(n^2)$ for all $n \geq n_0$.

Solution:

We require that $n \lg n \leq cn^2$ for $n \geq n_0$ where c and n_0 are positive constants. Divide both sides of the inequality by n^2.

$\lg n / n \leq c, n \geq 2$

The function on the left hand side is decreasing with n, so if we pick $c \geq \lg 2 / 2 = 1 / 2$, we should be fine. Any value of c greater than or equal to ½ is a correct answer.

Applied Programming
Fall 2017-18
Quiz # 1

Roll Number:

Q. 1. An algorithm has the exact running time of 4 n lg n + 9000. Is it $O(n^2)$? Prove or disprove using the definition of Big-Oh.

Solution:

We need to find positive constants c and n0 such that $4 n \lg n + 9000 \leq c n^2$, for all $n \geq n0$

Divide both sides of the inequality by n^2.

$4 \lg n / n + 9000 / n^2 \leq c$

The left hand side decreases with increasing n. So let's pick $n_0 = 64$, then $c \geq 4 \lg 64 / 6 + 9000 / 64^2 = 2.57$. So, the given function is $O(n^2)$ for c = 3 and $n_0 = 64$.

Applied Programming
Fall 2017-18
Quiz # 1

Roll Number:

Q. 1. What is the worst case, best case and average case time complexity of inserting an item into an array, in Big-Oh notation? Provide reasoning.

To insert an item at index i in an array, items at indices i+1, …, n must be moved over one place to make room for the new item before copying the new item into the array. The number of items to be moved depends on where the new item is being inserted, whereas the copy of the new value into the array is a constant time operation. In the best case, the item is being inserted at the end of the array, whereby no items need to be moved: O(1). In the worst case, the item is being inserted at the beginning of an array, so that every item needs to be moved. This requires O(n). On average, if all indices are equally likely, then the new item will be inserted at the middle of the array, which requires n / 2 items to be copied over: O(n).

Applied Programming
Fall 2017-18
Quiz # 1

Roll Number:

Q. 1. Give an algorithm to find the maximum from an array of integers A of size n. What is the worst case, best case and average case number of primitive operations performed by this algorithm?

Solution:

Input: An array A[1…n] of n elements
Output: An array item A[i], such that A[i] ≥A[j] for j = 1, 2, …, n.
Procedure:

```
Max ← A[1]
For I ← 2 to n
        If A[i] > Max then
                Max ← A[i]
```

In the worst case, the array is sorted in ascending order and the if condition is always true. Suppose all primitive operations have unit cost. Then, the total number of operations in the worst case is $1 + (n - 1 + 1) + (n - 1)*2$. In the best case, it is $1 + (n - 1 + 1) + (n - 1)*1$. On average, half of the time, the if statement is true, then $1 + (n - 1 + 1) + 1*(n - 1) + 2*(n - 1)/2$.