Sean Morrissey

Professor Ramoza Ahsan

CS 2223 Algorithms

1 April 2019

Assignment 2

Question 1.

1. T(n) = 3T(3/5 n) + n

* A =3, B =5/3, α = 1, β =
* O()

1. T(n) = 3T(n/4) + n

* A =3, B =4, α = 1, β =
* O(n)

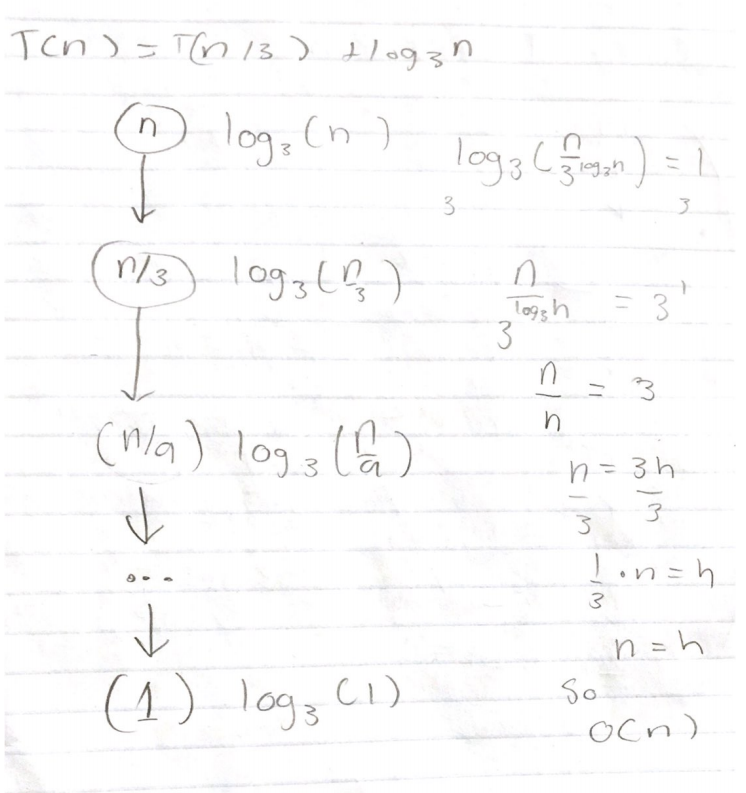
1. T(n) = 7T(n/2) +

* A =7, B =2, α = 2, β =
* O()

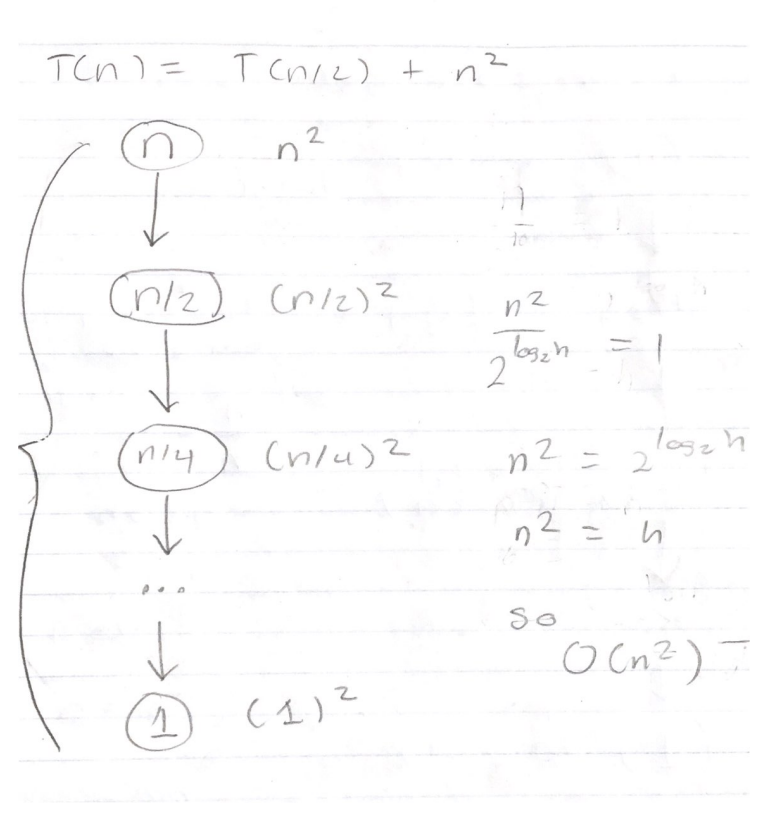
1. T(n) = T(3/4n) + 3 + n

* A =1, B =4/3, α = 2, β =
* O(

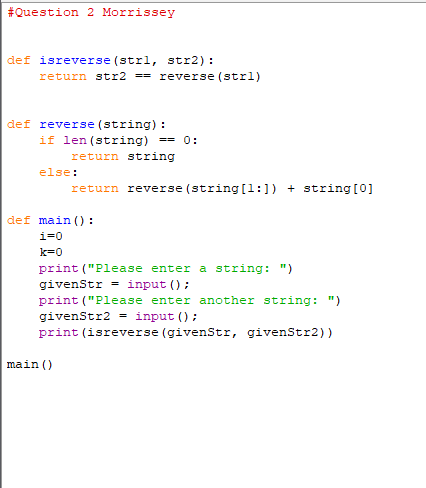
1. T(n) = T(n/3) +



1. T(n) = T(n/2) +



Question 2.



The function call to the helper function reverse() leads into creating the recurrence of the function isreversed(). In reverse(), there is a call to len(), which in python, is constant time; therefore, it has no effect on the recurrence time. In the recursive call to reverse(), I am calling it on a string that is of size n-1, if n is the size of the string passed into to reverse(). Thus, every call the string is shorter by n-1 times which means in the recurrence function T(n) is equal to T(n-1).

With that, the recurrence for the function is…

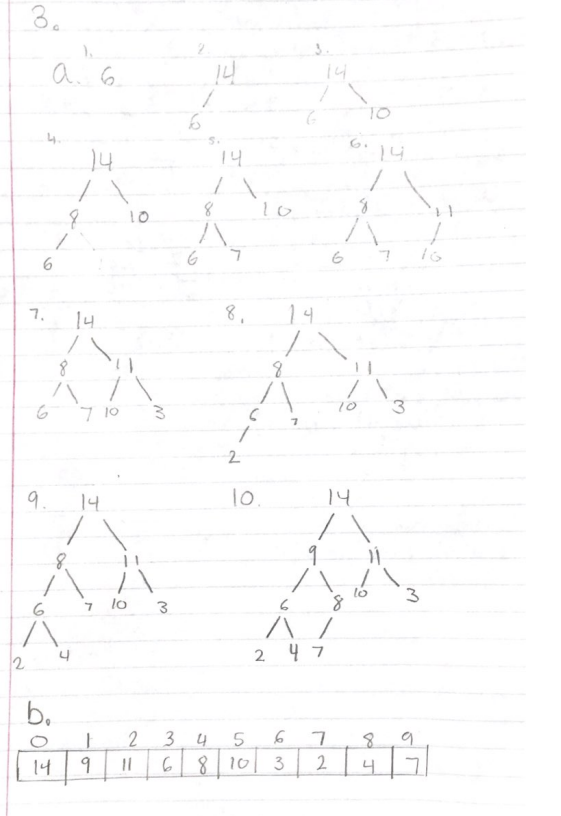
T(n) = T(n-1) + c

Or

T(n) = T(n-1) + O(1)

As a result, the big O notation of the function is O(n).

Question 3.



Question 4.

1. True. The recurrence function of T(n) = 3T(n/2) + n does imply O() because if one were to solve for the big O runtime with the Master Theorem, it will be O().

* A = 3, B = 2, α = 1, β = ; Case 3 applies

1. False. The recurrence equation for merge sort is T(n) 2T(n/2) + n. By using the Master Theorem, we know that the big O runtime is O(n log n). Because is a runtime that is greater than , it is impossible for Ω( to be true for merge sort.
2. False. If we expand the given recurrence function, we end up getting something that looks like the following:
   1. T(n-1) + log(n)
   2. T(n-2) + log(n-1) + log(n)
   3. T(n-3) + log(n-2) + log(n-1) + log(n)

If we assume n = 0, then

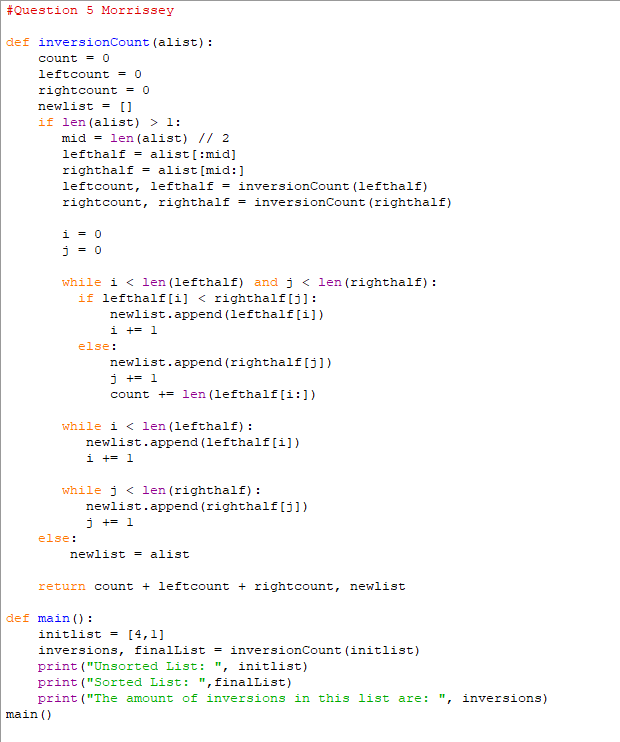
T(0) + log(1) + log(2) + log(3) … log(n-1) + log(n)

Which is the same as…

T(0) + log(n!)

Therefore, for the given recurrence function, the runtime is ϴ(log(n!)), but this does not work for quick sort. The worst case for quick sort is O(, which is when a sorted array is passed in. Thus, the statement is false because the runtime of the recurrence is not the same as quick sort’s worst case.

Question 5.



Because the function inversionCount() function is basically merge sort with a counter, the runtime and recurrence function are practically the same. Using a counter and modifying it are constant operations, so they do not have an affect on the recurrence function of merge sort because n has greater growth than a constant, O(1).

Therefore, the recurrence function of inversionCount() is…

T(n) = 2T(n/2) + n

Using the Master Theorem to find the big O runtime in the worst case…

A= 2, B = 2, α = 1, β = = 1

Case 2 -> ϴ(n log n)

Thus, the runtime in the worst case for inversionCount() is O(n log n).