Quiz 1 Solutions v3

Quiz 1 Prep - There are no solutions to these they are hints for the test. Work together or with the me or the TAs to solve these.
no solutions?!?1? ummm live miku reaction

anyways heres the quiz 1 solutions v3 by the one and only real hatsune miku

patch notes:

v2 - fixed up problem 6 code stuff and added comments
v3 - i asked questions abt stuff i wasnt sure about after lecture today oct 3rd so i hope this doc is correct now. question 4 i added base cases. question 5 not having a base case was in fact an oversight. oh and for question 1 i asked about log bases, we can use any base to our convenience if not specified by her.

Big-0, Omega and Theta by definition

1.) Prove that T(n)=5(n^2)logn+4n^2+3 is Big-Theta of g(n)=2(n^2)logn

To prove this, we must show that $T(n) = O(g(n))$ as well as $T(n) = \Omega(g(n))$.

First, let's prove that $T(n) = O(g(n))$. We know the following is true for $n \geq 10$:

$$5n^2log(n) + 4n^2 + 3 \leq 5n^2log(n) + 4n^2log(n) + 3n^2log(n)$$

We choose $n_0 = 10$ because $log(10) = 1$, and to ensure the new expression is greater or equal, terms must be multiplied by a value ≥ 1. The above expression can then be simplified as

$$= 12n^2log(n)$$

Thus, for $c = 6$ and $n_0 = 10$ we know that $5n^2log(n) + 4n^2 + 3 \leq c(2n^2log(n))$, and therefore $T(n) = O(g(n))$.

Next, let's prove that $T(n) = \Omega(g(n))$. Because $4n^2 + 3 \geq 0$ for any real n, we know the following is true for $n > 0$:

$$5n^2log(n) + 4n^2 + 3 \geq 5n^2log(n) \geq 2n^2log(n)$$

Therefore, for $c = \frac{5}{2}$ and $n_0 = 1$ we know that $5n^2log(n) + 4n^2 + 3 \geq c(2n^2log(n))$, and therefore $T(n) = \Omega(g(n))$.

As we have proven that $T(n) = O(g(n))$ and $T(n) = \Omega(g(n))$, we have also proved $T(n) = \Theta(g(n))$.

2.)Prove T(n)=O(g(n)) via limit lemma.

To prove this, we must show that $\lim_{n\to\infty} \frac{T(n)}{g(n)} \leq c$.

$$\lim_{n\to\infty} \frac{T(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{d}{dn}(5n^2log(n) + 4n^2 + 3)}{\frac{d}{dn}(2n^2log(n))}$$

$$= \lim_{n\to\infty} \frac{10nlog(n) + 13n}{4nlog(n) + 2n}$$

ok i give up on typing well ima do the rest of this doc as if im texting. here we will treat log as base e because it makes derivatives easier. time for round 2 of l'hopitals

$$= \lim_{n\to\infty} \frac{\frac{d}{dn}(10nlog(n) + 13n)}{\frac{d}{dn}(4nlog(n) + 2n)}$$

$$= \lim_{n\to\infty} \frac{10log(n) + 23}{4log(n) + 7}$$

ok ngl i dont know how many times she wants us to take the derivative like its very obviously 10/4 here so im not gonna bother to do it again

$$= \frac{5}{2}$$

yay boom its equal to constant, thus $T(n) = O(g(n))$

3.) Limit Lemma

f(n)=nlogn g(n)=n^2 +3n prove via limit lemma that f(n)=O(g(n))

same as before, we gotta show $\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq c$

$$\lim_{n\to\infty}\frac{T(n)}{g(n)} = \lim_{n\to\infty}\frac{\frac{d}{dn}(nlog(n))}{\frac{d}{dn}(n^2+3n)}$$

$$= \lim_{n\to\infty}\frac{log(n)+1}{2n+3}$$

run it back babyyyyy

$$= \lim_{n\to\infty}\frac{\frac{d}{dn}(log(n)+1)}{\frac{d}{dn}(2n+3)}$$

$$= \lim_{n\to\infty}\frac{\frac{1}{n}}{2}$$

$$= 0$$

once again, we got a constant and have thus proved $f(n) = O(g(n))$

Can I prove log(n)=O(n^x) for some x? prove yes or explain why no.

mmmm lets see. for $x > 0$,

$$\lim_{n\to\infty}\frac{log(n)}{n^x} = \lim_{n\to\infty}\frac{\frac{d}{dn}(log(n))}{\frac{d}{dn}(n^x)}$$

$$= \lim_{n\to\infty}\frac{\frac{1}{n}}{xn^{x-1}}$$

$$= \lim_{n\to\infty}\frac{1}{xn^x}$$

$$= 0$$

The expression approaches a constant, so we have therefore proven that $log(n) = O(n^x)$ for some x.

4.)Recurrences.

v3 update - I asked and she said add base cases so I have updaded the pseudo code to include base cases.

a.)Provide a Sample code for the recurrences below.

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

```
foo(int n)
    if n ⩽ 1
        return
    foo(n/5);
    foo(3*n/5);
    for(i=1 to n)
           print "meow :3"
```

ii.)T(n)=T(2n/5)+T(3n/5)+n

```
foo(int n)
    if n ⩽ 1
        return
    foo(2*n/5);
    foo(3*n/5);
    for(i=1 to n)
           print "meow :3"
```

iii.) T(n)=4T(n/2)+n^3

```
foo(int n)
    if n ⩽ 1
        return
    for(i=1 to 4)
            foo(n/2);
    for(i=1 to n^3)
           print "meow :3"
```

Problem 5;

v3 update: the lack of a base case here was an error. treat the code as if there was an

Given code can you write the recurrence

```
foo( int n)
    foo(n/2);
    for( i=1 to n^2)
           print hello
```

fuck man just look at it i guess
$T(n) = T(n/2) + n^2$

Problem 6.)

Write pseudocode that lists the subsets of an array of ints.

```
# need 2^n iterations for the 2^n subsets
for (i = 0 to (2^arr.length - 1))
    subset = []
    # ok i feel like this will need some explanation bc i do some
    # funny bitwise stuff. basically for each value of i, we use
    # its binary value to get every possible combination of elements

    # here we iterate through each bit
    for (j = 0 to (arr.length - 1))
        # using bitshift operators, 1<<j we can check each bit in i
        # for example for 4 bits, (1<<j) resolves to 0001, then 0010,
        # then 0100, then 1000. using the bitwise AND operator, we can
        # check to see if that bit in i is equal to 1. if so, we add
        # the value to the subset. hope that makes sense :3
        if i & (1 << j)
            subset.push(arr[j])
    print subset
```

Write psuedocode finds the smallest number of an arrya of ints.

```
if arr.len = 0
    return "empty"

smallest = arr[0]
# iterate thru each element. if a smaller one is found, set it as smallest
for (i = 1 to (arr.length - 1))
    if arr[i] < smallest
        smallest = arr[i]
print smallest
```

Write psudedocode the list all possible substring of a string S.

```
# iterate thru each character the substring could start with
for (i = 0 to (S.length - 1))
    # iterate thru each character the substring could end with
    for (j = i to (S.length - 1))
        sub = S[i to j]
        print sub
```

Given you have two sort lists write pseudocode that returns a new merged sorted

```
mergeSortedLists(list1, list2)
    mergedList = []
    # these are basically iterators
    i = j = 0

    # we run thru the lists until one of them runs out of elements
    while (i < list1.length && j < list2.length)
        # compare the smallest elements of both lists, then add
        # it to mergedList. increment that list's iterator
        if list1[i] < list2[j]
            mergedList.push(list1[i])
            i++
        else
            mergedList.push(list2[j])
            j++

    # once one of the lists runs out of elements, add the rest
    # of the elements from the other list
    while (i < list1.length)
        mergedList.push(list1[i])
        i++
    while (j < list2.length)
        mergedList.push(list2[j])
        j++
    return mergedList
```