Question 51 states the following:

What is the time complexity of this function? Assume the initial value of i is one and j is zero.

```java
function f(i,j,n)
{
    if (i < n)
    {
        if (j < n)
            f(i,j+1,n);
        else
            f(i*2,0,n);
    }
    println(i,j);
}
```

I've boiled the question down to be either n*log(n) or n*sqrt(n). I am unsure how i*2 would result in, but I feel it would result in a sqrt(n) amount of runtime.

What would make code result in log(n) and what would make it result in sqrt(n)?

When I was working through this problem I came to the conclusion that it would run in n*log(n) with a base of 2. Please correct me if I am incorrect though.

When I look at this problem I see two loops, one inside the other:

```java
for(i = 1; i < n; i *= 2) {
    for(j = 0; j < n; j++) {
        /* print correctly */
```
Subject: Re: log(n) vs sqrt(n), and question 51 (and 52)
Posted by sbcarp on Tue, 24 Jan 2017 06:30:36 GMT

Testing Result:
n: 100
log(n): 6.643856
n*log(n): 664.385619
n^2: 10000
n^n: 9999999999999969733122212510361659474503275455023626482417
509503468484355540755341963384047062518680275124159738824081
821357343682784846393850410472398778710235910667899818111818
13306167128854888448.000000

Total time: 708 (Matches n*log(n))

Subject: Re: log(n) vs sqrt(n), and question 51 (and 52)
Posted by sbcarp on Tue, 24 Jan 2017 06:33:07 GMT

100*sqrt(100)=1000