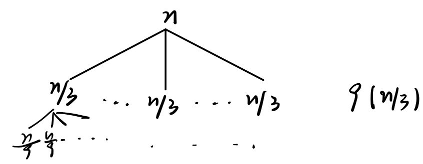
Jiayu Li HW 2

Prof. Shinko

Due. 10/14

Problem 1. Find the time complexity of $T(n)$ satisfying $T(n) = 9T(n/3) + O(n^2)$.



Solve: According the Master method

At the k-th level, the input is size $\frac{n}{3^k}$

so the number of steps there is $\frac{n^2}{9^k}$

At the kth level, there are 9^k nodes,

So in total, the number of steps

at the k th level is $\frac{9^k n^2}{9^k} = n^2$

Then we have the series $n^2 + n^2 + n^2 + \cdots$ with $\log_3(n)$ terms.

So the total number of steps is $n^2 \log n$

Therefore, $T(n) = \Theta(n^2 \log n)$

Problem 2. Find the time complexity of $T(n)$ satisfying $T(n) = 50T(n/3) + O(n^3)$.

Solve: At the kth level, the input size is $\frac{n}{3^k}$, so the number of steps is $\frac{n^3}{27^k}$

At the kth level, there are 50^k nodes, so in total, the number of steps at the kth level is $\frac{50^k n^3}{27^k}$

Then we have the series

$$n^3 + \frac{50}{27} n^3 + \left(\frac{50}{27}\right)^2 n^3 + \cdots \quad \text{with } \log_3(n)$$

This is a geometric series, so its sum is

$$\frac{\left(\frac{50}{27}\right)^{\log_3(n)} - 1}{\frac{50}{27} - 1} n^3$$

which is big-O of $n^{\log_3\left(\frac{50}{27}\right)} n^3 = n^{\log_3 50}$

Therefore, $T(n) = O(n^{\log_3 50})$.

$\blacksquare$

Problem 3. Find the time complexity of $T(n)$ satisfying $T(n) = 10T(n/4) + O(n^2)$.

Solve: At the kth level, the input size is $\frac{n}{4^k}$,
so the number of steps is $\frac{n^2}{16^k}$.

At the kth level, there are 10^k nodes,
So in total, the number of steps at the kth level is $\frac{10^k n^2}{16^k} = \left(\frac{5}{8}\right)^k n^2$

Then we have the series

$$n^2 + \frac{5}{8}n^2 + \left(\frac{5}{8}\right)^2 n^2 + \cdots \text{ with } \log_4(n)$$

The infinite series $n^2\left(1 + \frac{5}{8} + \left(\frac{5}{8}\right)^2 + \cdots\right)$ converges.

So the above series is just a constant times n^2

Therefore, $T(n) = O(n^2)$.

Problem 4. Find the time complexity of the following function.

```
# A is an integer array
def foo(A):
    n = len(A)
    if n <= 1:
        print("hello")
    else:
        for i in range(n):          n
            for j in range(i):      n^2
                print(j)
                print(i)
        foo(A[: n//2])
        foo(A[n//4 : 3*n//4])
        foo(A[n//2:])
```

We have 3 recursion

We can get $T(n) = 3T(\frac{n}{2}) + O(n^2)$

At the k^{th} level, the input size is $\frac{n}{2^k}$, so the number of steps is $\frac{n^2}{4^k}$

At the k^{th} level, there are 3^k nodes. so in total, the number of steps at the k^{th} level is $\frac{3^k n^2}{4^k}$

Then we have the series $n^2 + \frac{3}{4}n^2 + (\frac{3}{4})^2 n^2 + \cdots$ with $\log_2(n)$.

The infinite series $n^2(1 + \frac{3}{4} + (\frac{3}{4})^2 + \cdots)$ converges.

So the above series is just a constant times n^2

Therefore, $T(n) = O(n^2)$.

∎