09/08/2024

CMPS 6610 Problem Set 2

Name: Nazmun Nahar Khanom

1. Prove that $\log n! \in \Theta(n \log n)$.

Solution: Stirling's approximation provides:

$$\log n! \approx n \log n - n$$

Using the definition of big-Theta, the subtracted term n is much smaller than $n \log n$ for large n. Therefore,

$$\log n! = n \log n - n = \Theta(n \log n)$$

Here, $\log n!$ has the same asymptotic behavior as $n \log n$.

2. Derive asymptotic upper bounds for each recurrence below, using a method of your choice.

- $T(n) = 2T(n/6) + 1$

Solution: Applying the Master Theorem,

$$a = 2, b = 6, f(n) = O(1)$$

Since $n^{\log_b a} = n^{\log_6 2}$, and $f(n) = O(1)$, this falls under Case 1 of the Master Theorem. So,

$$T(n) = O(\log n).$$

- $T(n) = 6T(n/4) + n$

Solution: Applying the Master Theorem,

$$a = 6, b = 4, f(n) = n$$

Since $n^{\log_b a} = n^{\log_4 6}$, this falls under the Case 1 of the Master Theorem. So,

$$T(n) = O(n^{\log_4 6}).$$

• $T(n) = 7T(n/7) + n$.

Solution: Applying the Master Theorem,

$$a = 7,\ b = 7,\ f(n) = n$$

Since $n^{\log_b a} = n^1$, this falls under Case 2 of the Master Theorem.

So,

$$T(n) = O(n \log n).$$

• $T(n) = 9T(n/4) + n^2$.

Solution: Applying the Master Theorem,

$$a = 9,\ b = 4,\ f(n) = n^2$$

This falls under Case 3 of the Master Theorem.

So,

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

• $T(n) = 4T(n/2) + n^3$

Solution: Applying the Master Theorem,

$$a = 4,\ b = 2,\ f(n) = n^3$$

This falls under Case 3 of the Master Theorem.

So,

$$T(n) = O(n^3).$$

• $T(n) = 49T(n/25) + n^{3/2} \log n$

Solution: Applying the Master Theorem,

$$a = 49,\ b = 25,\ f(n) = n^{3/2} \log n$$

This falls under Case 3 of the Master Theorem.

So,

$$T(n) = O(n^{3/2} \log n).$$

• $T(n) = T(n-1) + 2$

Solution: This is a simple recurrence. It can be solved directly by using expansion,

$$T(n) = O(n).$$

• $T(n) = T(n-1) + n^c$, with $c \geq 1$.

Solution: It can be solved by using expansion,

$$T(n) = O(n^{c+1}).$$

• $T(n) = T(\sqrt{n}) + 1$

Solution: It can be solved ~~using~~ by using expansion on the recurrence,

$$T(n) = O(\log \log n).$$

3. Suppose that for a given task you are choosing between the following three algorithms:

- Algorithm A solves problems by dividing them into two subproblems of one fifth of the input size, recursively solving each subproblem, and then combining the solutions in quadratic time.
- Algorithm B solves problems of size n by recursively one subproblems of size $n-1$ and then combining the solutions in logarithmic time.
- Algorithm C solves problems of size n by dividing them into a subproblems of size $n/3$ and a subproblem of size $2n/3$, recursively solving each subproblem, and then combining the solutions in $O(n^{1.1})$ time.

What is the work and span of these algorithms? For the span, just assume that it is the same as the work to combine solutions. Which algorithm would you choose? Why?

Solution:

For Algorithm A,

Breaking the problem into 2 subproblems of size n/5.

The recurrence is, $W(n) = 2W(n/5) + O(n^2)$

Applying the Master Theorem, yielding

$$W(n) = O(n^2)$$

Span is the same as work for combining, so the span is $O(n^2)$.

For Algorithm B,

The recurrence is, $W(n) = W(n-1) + O(\log n)$

This solves to $W(n) = O(n \log n)$.

For Algorithm C,

Dividing into subproblem of size n/3 and 2n/3.

The recurrence is, $W(n) = W(n/3) + W(2n/3) + O(n^{1.1})$, which solves to $W(n) = O(n^{1.1})$.

So, we are going to choose Algorithm C because of the lower work and span.

4. Suppose that for a given task you are choosing between the following three algorithms:

- Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.

- Algorithm B solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.

- Algorithm C solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

What is the work and span of these algorithms? For the span, just assume that it is the same as the work to combine solutions. Which algorithm would you choose? why?

Solution: For Algorithm A,

The recurrence is: $W(n) = 5W(n/2) + O(n)$

This results to $W(n) = O(n^2)$.

For Algorithm B,

The recurrence is : $W(n) = 2W(n-1) + O(1)$, which solves to

$W(n) = O(2^n)$.

For Algorithm C,

The recurrence is: $W(n) = 9W(n/3) + O(n^2)$, which yields

$W(n) = O(n^2)$.

So, we are choosing Algorithm A as it has the best span.

5. In Module 2 we discussed two algorithms for integer multiplication. The first algorithm was simply a recapitulation of the "grade school" algorithm for integer multiplication, while the second was the Karatsaba-Ofman algorithm. For this problem, you will use the stub functions in the main.py to implement these two algorithms for integer multiplication. Once you've correctly implemented them, test the empirical running times across a variety of inputs to test whether your code scales in the manner predicted by our analyses of the asymptotic work.

Solution: Please check "main.py".