Quick Assignment 3: Solutions
Total: 100

CS 2500: Algorithms

Due Date: September 6, 2024 at 11.59 PM

Solution

1. **Step 1: Base Case**

   First, let's check the base case when $n = 1$.

   For $n = 1$, the left-hand side is:

   $$t_1 = 1^2 = 1$$

   Now, check the right-hand side of the guessed formula:

   $$t_1 = \frac{1(1+1)(2(1)+1)}{6} = \frac{1 \times 2 \times 3}{6} = \frac{6}{6} = 1$$

   Thus, the formula holds for $n = 1$.

   Step 2: Induction Hypothesis

   Assume that the formula holds for $n = k$, i.e., assume that:

   $$t_k = 1^2 + 2^2 + 3^2 + \cdots + k^2 = \frac{k(k+1)(2k+1)}{6}$$

   We will now show that the formula also holds for $n = k+1$, i.e., that:

   $$t_{k+1} = 1^2 + 2^2 + 3^2 + \cdots + k^2 + (k+1)^2 = \frac{(k+1)(k+2)(2(k+1)+1)}{6}$$

   Step 3: Inductive Step

   By the recurrence relation, we know:

   $$t_{k+1} = t_k + (k+1)^2$$

   Using the induction hypothesis $t_k = \frac{k(k+1)(2k+1)}{6}$, substitute this into the equation:

   $$t_{k+1} = \frac{k(k+1)(2k+1)}{6} + (k+1)^2$$

   Factor out $(k+1)$ from the right-hand side:

   $$t_{k+1} = (k+1)\left(\frac{k(2k+1)}{6} + (k+1)\right)$$

Simplify the expression inside the parentheses:

$$t_{k+1} = (k+1)\left(\frac{k(2k+1)+6(k+1)}{6}\right)$$

$$t_{k+1} = (k+1)\left(\frac{2k^2+k+6k+6}{6}\right)$$

$$t_{k+1} = (k+1)\left(\frac{2k^2+7k+6}{6}\right)$$

Now, factor the quadratic expression in the numerator:

$$t_{k+1} = (k+1)\left(\frac{(k+2)(2k+3)}{6}\right)$$

Finally, distribute $(k+1)$ to get:

$$t_{k+1} = \frac{(k+1)(k+2)(2k+3)}{6}$$

This matches the form of the solution for $n = k+1$. Thus, the formula holds for $n = k+1$.

Step 4: Conclusion

Since the base case holds for $n = 1$, and the inductive step shows that if the formula holds for $n = k$, it also holds for $n = k+1$, by mathematical induction, we conclude that the formula:

$$t_n = 1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

is the solution to the recurrence relation $t_n = t_{n-1} + n^2$ for all $n \geq 1$.

2. **Step 1: Base Case**

For $k = 0$, we have $n = 2^0 = 1$.

The recurrence gives us:

$$T(1) = 3T(1 \div 2) + 1 = 3T(0) + 1$$

From the recurrence, we know $T(0) = 0$, so:

$$T(1) = 3 \times 0 + 1 = 1$$

Now, check the solution for $k = 0$ from the formula $T(2^k) = 3^{k+1} - 2^{k+1}$:

$$T(2^0) = 3^{0+1} - 2^{0+1} = 3^1 - 2^1 = 3 - 2 = 1$$

Thus, the base case holds: $T(1) = 1$.

Step 2: Induction Hypothesis

Assume that the formula holds for some $k = n$, i.e., assume:

$$T(2^n) = 3^{n+1} - 2^{n+1}$$

We will now show that the formula holds for $k = n+1$.

Step 3: Inductive Step

We need to prove that:

$$T(2^{n+1}) = 3^{(n+1)+1} - 2^{(n+1)+1} = 3^{n+2} - 2^{n+2}$$

Start by applying the recurrence relation for $n = 2^{n+1}$:

$$T(2^{n+1}) = 3T(2^n) + 2^{n+1}$$

Using the induction hypothesis $T(2^n) = 3^{n+1} - 2^{n+1}$, substitute this into the equation:

$$T(2^{n+1}) = 3 \times \left(3^{n+1} - 2^{n+1}\right) + 2^{n+1}$$

Now, simplify the right-hand side:

$$T(2^{n+1}) = 3^{n+2} - 3 \times 2^{n+1} + 2^{n+1}$$

Factor out 2^{n+1} from the last two terms:

$$T(2^{n+1}) = 3^{n+2} - 2^{n+1}(3-1)$$

$$T(2^{n+1}) = 3^{n+2} - 2^{n+2}$$

This matches the desired solution $T(2^{n+1}) = 3^{n+2} - 2^{n+2}$.

Step 4: Conclusion

Since both the base case and the inductive step hold, by the principle of mathematical induction, we conclude that the solution:

$$T(2^k) = 3^{k+1} - 2^{k+1}$$

holds for all $k \geq 0$.

3. **Step 1: Base Case:**

First, check the base case for $n = 1$ and $n = 2$:

$$T(1) = 3 \times 1^{\log_2 3} - 2 \times 1 = 3 - 2 = 1$$

$$T(2) = 3 \times 2^{\log_2 3} - 2 \times 2 = 3 \times 2^{\log_2 3} - 4 = 6 - 4 = 2$$

Clearly, $T(2) = 2 \geq T(1) = 1$. Thus, the base case holds.

Step 2: Induction Hypothesis:

Assume that for some $n = k$, the function is non-decreasing, i.e.,

$$T(k+1) \geq T(k)$$

That is, assume the inequality holds for $n = k$:

$$3(k+1)^{\log_2 3} - 2(k+1) \geq 3k^{\log_2 3} - 2k$$

Step 3: Inductive Step:

We need to show that the inequality holds for $n = k + 1$, i.e.,

$$T(k+2) \geq T(k+1)$$

This means proving:

$$3(k+2)^{\log_2 3} - 2(k+2) \geq 3(k+1)^{\log_2 3} - 2(k+1)$$

Start by expanding both sides:

$$3(k+2)^{\log_2 3} - 2(k+2) = 3(k+2)^{\log_2 3} - 2k - 4$$

$$3(k+1)^{\log_2 3} - 2(k+1) = 3(k+1)^{\log_2 3} - 2k - 2$$

Subtract the right-hand side from the left-hand side:

$$3(k+2)^{\log_2 3} - 3(k+1)^{\log_2 3} \geq 2$$

Factor out 3 on the left-hand side:

$$3\left((k+2)^{\log_2 3} - (k+1)^{\log_2 3}\right) \geq 2$$

Now, we need to show that:

$$(k+2)^{\log_2 3} - (k+1)^{\log_2 3} \geq \frac{2}{3}$$

Since $\log_2 3 \approx 1.585$ (which is greater than 1), the function $n^{\log_2 3}$ grows faster than n. Thus, for large k, the difference between $(k+2)^{\log_2 3}$ and $(k+1)^{\log_2 3}$ is positive and grows large enough to satisfy the inequality $(k+2)^{\log_2 3} - (k+1)^{\log_2 3} \geq \frac{2}{3}$.

Therefore, the inductive step holds.

Step 4: Conclusion:

Since both the base case and the inductive step hold, by mathematical induction, we conclude that $T(n) = 3n^{\log_2 3} - 2n$ is a non-decreasing function for all $n \geq 1$.

4. Given the recurrence equation:

$$T(n) = 2T\left(\frac{n}{2} + 17\right) + n,$$

we will solve it using the guess-and-verify method.

Guess the Solution: The recurrence looks like a divide-and-conquer recurrence, so we make an educated guess that the solution is of the form:

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

Verify the Guess Using Induction: We will use induction to verify our guess.

Induction Hypothesis: Assume that for some constant $c > 0$, the following holds for all $k < n$:

$$T(k) \leq ck \log k.$$

Base Case: Since the recurrence involves $T\left(\frac{n}{2} + 17\right)$, we won't reach $T(0)$ directly. Instead, choose $n = 34$, where:

$$T(34) = c' \quad \text{(a constant)}.$$

Inductive Step: Now, substitute the inductive hypothesis into the recurrence:

$$T(n) = 2T\left(\frac{n}{2} + 17\right) + n.$$

Using the hypothesis:

$$T\left(\frac{n}{2} + 17\right) \leq c\left(\frac{n}{2} + 17\right) \log\left(\frac{n}{2} + 17\right).$$

Approximating for large n:

$$\log\left(\frac{n}{2} + 17\right) \approx \log n,$$

so:

$$T(n) \leq 2\left[c\left(\frac{n}{2} + 17\right) \log n\right] + n.$$

Simplifying this:

$$T(n) \leq cn \log n + 34c \log n + n.$$

For large n, the dominant term is $cn \log n$, so:

$$T(n) \leq (c + \epsilon) n \log n,$$

for some small $\epsilon > 0$.

Conclusion: The recurrence relation holds, and the complexity is $O(n \log n)$.

Making a Good Guess

Unfortunately, there is no general way to correctly guess the tightest asymptotic solution to an arbitrary recurrence. Making a good guess takes experience and, occasionally, creativity. Fortunately, learning some recurrence-solving heuristics, as well as playing around with recurrences to gain experience, can help you become a good guesser. You can also use recursion trees, which we'll see in Section 4.4, to help generate good guesses.

If a recurrence is similar to one you've seen before, then guessing a similar solution is reasonable. As an example, consider the recurrence:

$$T(n) = 2T\left(\frac{n}{2} + 17\right) + \Theta(n),$$

defined on the reals. This recurrence looks somewhat like the merge-sort recurrence, but it's more complicated because of the added $+17$ in the argument to T on the right-hand side. Intuitively, however, this additional term shouldn't substantially affect the solution to the recurrence. When n is large, the relative difference between $\frac{n}{2}$ and $\frac{n}{2} + 17$ is not that large: both cut n nearly in half. Consequently, it makes sense to guess that:

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

which you can verify is correct using the guess-and-verify method.

Another way to make a good guess is to determine loose upper and lower bounds on the recurrence and then reduce your range of uncertainty. For example, you might start with a lower bound of $T(n) = \Omega(n)$ for recurrence above, since the recurrence includes the term $\Theta(n)$, and you can prove an initial upper bound of:

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

Then split your time between trying to lower the upper bound and trying to raise the lower bound until you converge on the correct, asymptotically tight solution, which in this case is:

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