This thread is part of the proposed schedule for study for the first exam. Discuss the practice problems here.
View questions here: http://beastie.cs.ua.edu/concepts/cs/al/recurrences.html
This thread covers questions 1-35 of recurrences.

Work together on the proposed answers to questions on this shared Google Doc (comment reasoning/arguments behind answers)

Upcoming topics (threads start two days in advance): Recurrences 36-70, Feb 11 | SB trees 1-26, Feb 12 | SB trees 26-52, Feb 13 | Recurrences 71-105, Feb 14
Full schedule

---

Subject: Re: Concept Review: Recurrences (Part 1)
Posted by davidmccoy on Thu, 09 Feb 2017 19:46:25 GMT

Quote: 1. Stooge sort has the following algorithm. Recursively sort the lower two-thirds of an array, then recursively sort the upper two-thirds, then recursively sort the lower two-thirds again. The recursion stops when the array consists of two or fewer elements. If the array size is two, the elements are swapped if necessary. Which of the following recurrence equations describe stooge sort?

There are 3 recursive calls, each call operates on 2/3 of the array, and there's no work done to combine the results of each recursive step. Help?

---

Subject: Re: Concept Review: Recurrences (Part 1)
Posted by davidmccoy on Thu, 09 Feb 2017 21:17:13 GMT

Quote: Number 11. Given the recurrence T(n) = 4T(n/3) + logn, how much work is done at the bottom level?
So I thought this was how a recursion tree was drawn...

but then shouldn't the work done at the bottom level be (number of nodes) * log(n/3), which should be 4log(base 3)(n)? Not sure what the answer is.
davidmccoy wrote on Thu, 09 February 2017 13:46

Quote: 1. Stooge sort has the following algorithm. Recursively sort the lower two-thirds of an array, then recursively sort the upper two-thirds, then recursively sort the lower two-thirds again. The recursion stops when the array consists of two or fewer elements. If the array size is two, the elements are swapped if necessary. Which of the following recurrence equations describe stooge sort?

There are 3 recursive calls, each call operates on 2/3 of the array, and there's no work done to combine the results of each recursive step. Help?

Look closely at your recursive calls: 3T(3n/2). Does that make sense?

---

davidmccoy wrote on Thu, 09 February 2017 15:17

Quote: Number 11. Given the recurrence T(n) = 4T(n/3) + logn, how much work is done at the bottom level?

...but then shouldn't the work done at the bottom level be (number of nodes) * log(n/3), which should be 4log(base 3)(n)? Not sure what the answer is.

The work done by each node at the bottom level is usually theta(1).

---

davidmccoy wrote on Fri, 10 February 2017 21:06:55

I was putting 3/2 on the bottom of the fraction, but I see now that doesn't make sense. If I was operating on half the array, I would put 3T(n/2), where (1/2) is being multiplied by n. So the correct