ADw3

Name : **Peter Garamvoelgyi** Student ID: **2018280070**

**CLRS 4.4-7. Draw the recursion tree for, whereis a constant, and provide a tight asymptotic bound on its solution.**

Following the method from CLRS page 88 (“*we know that floors and ceilings usually do not matter when solving recurrences*”), we create a recursion tree for the recurrence

… where we also assumed the base case to be.



We get a recursion tree withlevels.

Relying on the fact that, the cost at the leaves is

The sums on the other levels are:

Summing these up, we get

To get the total cost, we sum up the cost of the leaves and the cost of the other levels:

We can verify this using the *master method*:

In this case,grows polynomially slower than:

… i.e. this is case 1. Thus,

**CLRS 4-1. Give asymptotic upper and lower bounds forin each of the following recurrences. Assume thatis constant for. Make your bounds as tight as possible, and justify your answers.**

**b.**

Using the *master method*, our parameters are

In this case,grows polynomially faster than:

… andsatisfies the regularity condition

… i.e. this is case 3. Thus,

**c.**

Using the *master method*, our parameters are

In this case,grows at a similar rate as:

… i.e. this is case 2. Thus,

**d.**

Using the *master method*, our parameters are

In this case,grows polynomially faster than:

… andsatisfies the regularity condition

… i.e. this is case 3. Thus,

**CLRS 4-2. Consider the recursive binary search algorithm for finding a number in a sorted array. Give recurrences for the worst-case running times of binary search when arrays are passed using each of the three methods above[[1]](#footnote-1), and give good upper bounds on the solutions of the recurrences.**

The running time of binary search is characterized by the equation

… i.e. we half the size of the problem and we only have one single recursive call in every iteration.in every case. The difference is in the complexity of the divide step

1. **Pass by pointer**

Using the *master method*, our parameters are

In this case,grows at a similar rate as:

… i.e. this is case 2. Thus,

1. **Pass by copying**

The easiest way to prove this is to draw the recursion tree. We get a list with levels, each of them containing steps.

1. **Pass by copying subrange**

Using the *master method*, our parameters are

In this case,grows polynomially faster than:

… andsatisfies the regularity condition

… i.e. this is case 3. Thus,

1. Pass by pointer:. Pass by copying:. Pass by copyingsubrange:. [↑](#footnote-ref-1)