

Master Method: An Alternate Formulation

January 28, 2003

Handout 1

Alternate formulation of the Master Method

The formulation presented here will solve recurrences of the form:

$$\begin{aligned} T(n) &= \Theta(1) \text{ for all } n \leq c \\ T(n) &= aT(n/b) + f(n) \text{ for all } n > c \end{aligned}$$

where $f(n) = \Theta(n^\ell(\log n)^k)$ for constants, $c \geq 0$, $a \geq 1$, $b > 1$, $\ell \geq 0$, and $k \geq 0$.

Case 1: For $\ell < \log_b a$, $T(n) = \Theta(n^{\log_b a})$

Case 2: For $\ell = \log_b a$, $T(n) = \Theta(f(n) \log n) = \Theta(n^{\log_b a}(\log n)^{k+1})$

Case 3: For $\ell > \log_b a$, $T(n) = \Theta(f(n)) = \Theta(n^\ell(\log n)^k)$

To get some practice with this, solve the problems in the text book on pages 74 and 75. Check that you get the same answers as given in the text.

Some Things to Remember

For recurrences of the form $T(n) = aT(n/b) + f(n)$, when the base case occurs for inputs of size 1, there are exactly $a^{\log_b n} = n^{\log_b a}$ applications of the base case of the recurrence. So $\Theta(n^{\log_b a})$ time is spent at the base of the recursive calls. In Case 1, the time spent here is as much (asymptotically) as all the time spent dividing and combining.

Just as we saw when analyzing the time complexity of the divide-and-conquer algorithm for finding the closest pair of points, in Case 2, there are $\log_b n$ levels of recursion, across each of which $f(n)$ time is spent splitting and combining.

Finally, in Case 3, the $f(n)$ time spent splitting and combining for the top-level call is as much (asymptotically) as all the rest of the computation.

Reason for Presenting This Formulation

I have selected this formulation to present since students generally find it easier to use than the one in the text book. Also, the version in the text book only considers when $k = 0$. The formulation in the textbook is more general in that it can apply for recurrences of the form $T(n) = aT(n/b) + f(n)$ where $f(n)$ is not of the form $\Theta(n^\ell(\log n)^k)$. However, it is rare to find any divide-and-conquer algorithms where the cost to split and then combine is not of the form $\Theta(n^\ell(\log n)^k)$.

Analyzing Alternate Closest Pair Algorithms

We now use the master method to determine the worst-case asymptotic time complexities for the following two variations of the divide-and-conquer algorithm to find the closest pair of points that were suggested in class. The first proposal considers making four recursive calls, one for the left half, one for the right half, one for the bottom half, and one for the top half. Let d be the distance of the closest pair among the four recursive calls. Observe then that the closest pair of points is either the pair of distance d or otherwise must be in a $2d$ by $2d$ square centered around where the left/right and top/bottom dividing lines meet. It can be proven that there are at most 16 points in this square.

So now let's derive a recurrence relation for this algorithm. As in the divide-and-conquer algorithm you implemented in lab 1, the dividing above can be done in $\Theta(n)$ time where n is the number of points. There are 4 subproblems each of size roughly $n/2$ that must be solved recursively. Finally, using the observation that square that must be considered when combining can have at most 16 points, we can actually do the combining in $\Theta(1)$ time by just computing the distance between each pair of points in the $2d$ by $2d$ square. So we get the recurrence:

$$T(n) = 4T(n/2) + \Theta(n).$$

In this recurrence, $a = 4, b = 2$ and so $\log_b a = 2$. Finally, $\ell = 1$ and $k = 0$. Thus case 1 applies and so $T(n) = \Theta(n^2)$. Suppose you thought that you could do the divide step in $\Theta(1)$. Observe there is no need to even pursue this direction since it would yield the same asymptotic solution.

The other variation proposed was to replace the division of the points into a left and right half, by dividing the points into 3 strips (left, middle, right) each with either $\lceil n/3 \rceil$ or $\lfloor n/3 \rfloor$ points. Then recursively find the closest pair in each of the 3 strips. Finally, use the combining procedure presented in class (and implemented in lab 1) around each of the 2 dividing lines. Here $f_1(n) = \Theta(n), f_2(n) = \Theta(n)$ and so

$$T(n) = 3T(n/3) + \Theta(n).$$

So we have $a = b = 3$ and so $\log_b a = 1$. Also $\ell = 1$ and $k = 0$. So case 2 applies and we get that $T(n) = \Theta(n \log n)$. So this algorithm has the same worst-case asymptotic complexity as the one you implemented in Lab 1.