

Solutions to Practice Problems for Homework 1

1. Here is a summary of the correct answers. The justifications are provided after the table:

	$T_1(n)$	$T_2(n)$	Is $T_1(n) = O(T_2(n))$?	Is $T_1(n) = \Omega(T_2(n))$?	Is $T_1(n) = \Theta(T_2(n))$?	Which is best?
a	$25n \ln n + 5n$	$\frac{1}{2}n \log_2 n$	yes	yes	yes	A_2
b	$\frac{1}{2}n^2 + n \log_2 n$	$5n \log_2 n$	no	yes	no	A_2
c	$\sqrt{n}(\log_2 n)$	n	yes	no	no	A_1
d	$2^{(2 \log_2 n)}$	$2n^2$	yes	yes	yes	A_1
e	$n\sqrt{n}$	$n^{1.4}$	no	yes	no	A_2

(a) Recall that $\log_2 n = \frac{\ln n}{\ln 2}$. Hence $\lim_{n \rightarrow \infty} \frac{25n \ln n + 5n}{\frac{1}{2}n \log_2 n} = \lim_{n \rightarrow \infty} \left(\frac{25n \ln n}{n \ln n / (2 \ln 2)} + \frac{5n}{n \ln n / (2 \ln 2)} \right) = \lim_{n \rightarrow \infty} \left(50 \ln 2 + \frac{10 \ln 2}{\ln n} \right) = 50 \ln 2 \approx 35$. They grow at the same asymptotic growth rate, however A_2 is almost 35 times faster and hence is the preferred algorithm.

(b) $\lim_{n \rightarrow \infty} \frac{1/2n^2 + n \log_2 n}{5n \log_2 n} = \lim_{n \rightarrow \infty} \left(\frac{n}{10n \log_2 n} + \frac{1}{5} \right) = \infty$ and hence T_1 is asymptotically faster growing and so A_2 is the preferred algorithm.

(c) $\lim_{n \rightarrow \infty} \frac{\sqrt{n} \log_2 n}{n} = \lim_{n \rightarrow \infty} \frac{\log_2 n}{\sqrt{n}} = 0$ and hence T_2 is asymptotically faster growing and so A_1 is the preferred algorithm.

(d) $2^{(2 \log_2 n)} = 2^{\log_2 n^2} = n^2$. Hence $\lim_{n \rightarrow \infty} T_1(n)/T_2(n) = 1/2$. Hence $T_1(n) = T_2(n)$ grow at the same asymptotic growth rate. However, A_1 is twice as fast as A_2 , so A_1 is the preferred algorithm.

(e) Note that $n\sqrt{n} = n^{1.5}$. So $\lim_{n \rightarrow \infty} \frac{n^{1.5}}{n^{1.4}} = \infty$ and hence T_1 is asymptotically faster growing.

2. (a) The outer loop is executed n times and the inner loop is executed 10 times for each execution of the outer loop. So the loop body is executed $10n$ times. (As another way to think about it the number of times the loop body is executed is given by $\sum_{i=0}^{n-1} \sum_{j=1}^{10} 1 = \sum_{i=0}^{n-1} 10 = 10n$.) Thus the asymptotic time complexity is $\Theta(n)$.

(b) The first loop has asymptotic time complexity $\Theta(n)$. For the pair of nested loops, the loop body is executed $\sum_{i=0}^{n-1} \sum_{j=i}^{n-1} 1 = \sum_{i=0}^{n-1} (n-i) = (n + (n-1) + \dots + 1) = \frac{n(n+1)}{2} = n(n+1)/2$ times. (The first step above follows since there are $(n-1) - i + 1 = n - i$ values of j for each value of i .) So the second set of loops has asymptotic time complexity $\Theta(n^2)$. Thus the overall asymptotic time complexity of the entire program segment is $\Theta(n) + \Theta(n^2) = \Theta(n^2)$.

3. **Algorithm A:** $T(n) = T\left(\frac{3n}{4}\right) + 3n^2 + n$

We apply the master method. Here $a = 1, b = 4/3$ (and so $\log_b a = 0$). Since $3n^2 + n = \Theta(n^2), \ell = 2, k = 0$. Thus $\ell > \log_b a$, so we have case 3 which gives that $T(n) = \Theta(n^2)$.

Algorithm B: $T(n) = 4T(n/4) + 3$

We apply the master method. Here $a = 4, b = 4$ (and so $\log_b a = 1$). Since $3 = \Theta(1), \ell = 0, k = 0$. Thus $\ell < \log_b a$, so we have case 1 which gives that $T(n) = \Theta(n)$.

Algorithm C: $T(n) = 3T(n/2) + 3n$

We apply the master method. Here $a = 3$, $b = 2$ (and so $\log_b a = \log_2 3$). Since $3n = \Theta(n)$, $\ell = 1$, $k = 0$. Thus $\ell < \log_b a$, so we have case 1 which gives that $T(n) = \Theta(n^{\log_2 3})$.

Algorithm D: $T(n) = 4T(n/2) + \Theta(n^2 \log_2 n)$

We apply the master method. Here $a = 4$, $b = 2$ (and so $\log_b a = 2$), $\ell = 2$, $k = 1$. Thus $\ell = \log_b a$, so we have case 2 which gives that $T(n) = \Theta(n^2(\log_2 n)^2)$.

Algorithm E: $T(n) = T(n - 1) + 10$

For this recurrence the master method does not apply. Using the recurrence tree technique we find that there are recursive calls made for problem size of $n, n - 1, \dots, 2$ each of which does 10 statements. Finally, there is one leaf (for the problem size of 1) for which a single statement is executed. Hence $T(n) = 10(n - 1) + T(1) = 10n - 10 + c = \Theta(n)$ where c is the number of statements executed when $n = 1$.

Which is best? Suppose that the above recurrences describe the exact time complexities of different algorithms to solve the same problem. Which algorithm is the fastest? Answer this as carefully as you can.

Algorithms B and E are asymptotically better than the others. Further, for E we know that the time complexity is roughly $10n$. So now we need to analyze algorithm B more carefully. For n a power of 4, we can solve the recurrence exactly using the recurrence tree technique. Let c' be the number of statements executed for Algorithm B when $n = 1$. There are 4^i nodes at each level, with 3 statements executed for each node. There are $\log_4 n$ levels of recursion with n leaves. Hence we get that

$$T(n) = \left(\sum_{i=0}^{\log_4 n - 1} 3 \cdot 4^i \right) + n \cdot T(1) = 3 \left(\frac{4^{\log_4 n} - 1}{3} \right) + c'n = n - 1 + c'n = (c' + 1)n - 1$$

So for Algorithm B the time complexity is roughly $(c' + 1)n$. Hence, as long as $c' < 9$ (recall c' is the number of statements executed when $n = 1$ which would generally be less than 9) then Algorithm B is best. For example if $c' = 1$ then Algorithm B would be roughly 5 times faster than Algorithm E. If $c' > 9$ then Algorithm E is best. Of course, if c' is very close to 9 then you would want to implement both algorithms and do testing on sample data to really see which is fastest (though you know in this case they are very close).

4. (a) For the case where $n = 1$ clearly $\Theta(1)$ work is performed, so $T(1) = \Theta(1)$. For $n \geq 2$, there are 3 statements to compute the value of `half` and to allocate `A1` and `A2`. The majority of the splitting cost is in executing the pair of nested loops. Notice that `i` takes on `half` different values. For each value of `i` there are just two values of `j`. Thus the statement in the loop (there was a “;” after the `if` that of course didn’t belong), is executed $2 * \text{half}$ times. Since `half` = $\lfloor n/2 \rfloor$, the asymptotic time complexity of these two nested loops is $\Theta(n)$. After the nested loops there is one more statement. Thus, putting this all together the time spent splitting is asymptotically that of the nested loops (with an additional $\Theta(n)$ steps outside the loop). Thus $\Theta(n)$ time is spent splitting.

The time spent combining is $\Theta(1)$ (again I apologize for the extra “;” after the `if`).

There are 2 recursive calls each on a problem of size $n/2$. Thus we obtain the recurrence $T(n) = 2T(n/2) + \Theta(n) + \Theta(1) = 2T(n/2) + \Theta(n)$.

Now we can apply the master method. We have $a = b = 2$, $\ell = 1$, and $k = 0$. Since $\ell = \log_b a = 1$, the solution is $T(n) = \Theta(n \log n)$.

- (b) For the case when $n = 1$ clearly $\Theta(1)$ work is performed. For $n \geq 2$ notice that $\Theta(n^2)$ time is spent in splitting into the subproblems. (The loop body is executed $(n/2)^2 = n^2/4$ times).

There are 4 recursive calls on problems of size $n/2$. Finally, constant time is spent combining. So the total time spent in the splitting and combining (i.e. in everything besides the recursive calls) is $\Theta(n) + \Theta(1) = \Theta(n)$ since n is asymptotically faster growing than $\Theta(1)$ and thus dominates.

Hence, $T(1) = \Theta(1)$ and for $n \geq 2$, $T(n) = 4T(n/2) + \Theta(n^2)$ Now we can apply the master method. We have $a = 4$, $b = 2$, $\ell = 2$, and $k = 0$. Since $\ell = 2 = \log_b a$, the solution is $T(n) = \Theta(n^2 \log n)$.