

(66 points)

The “limit rules” (p. 57) may not be used to solve any of these homework problems.

1. Exercise 4, p. 51.

(a) (2) Use the Pidgeon Hole Principle to solve this problem and to justify your answer.

(b) (4) For the average case analysis, see the example from the first day of class and the example on pp. 48-49.

2. (5) Exercise 8, p. 52.

This question asks you to determine the ratio of $f(4n)$ to $f(n)$ for each of the functions in the problem. For example, if the function is $f(n) = n$ (which is part c), then the ratio is $4n$ to n , meaning the function’s value increases by a factor of 4.

In each case, find the simplest way to express each factor.

3. (10) Exercise 9, p. 52.

Disregard the directions for this problem given in the text. Instead, either:

Find a function $g(n)$ such that both of the functions in the problem are in $\Theta(g(n))$ and prove it.

or:

If the two functions are not in the same order, find two functions, $g_1(n)$ and $g_2(n)$ such that the first function is in $\Theta(g_1(n))$ and the second is in $\Theta(g_2(n))$ and prove it.

(a) *Example:* We see that $n^2 \leq n(n+1) = n^2 + n \leq n^2 + n^2 = 2n^2 \leq 2000n^2$ for all $n \geq 0$. Thus we have found constants $c_1 = 1$ and $c_2 = 2$ such that $c_1n^2 \leq n(n+1) \leq c_2n^2$ for all n greater than the threshold $n_0 = 0$, so $n(n+1) \in \Theta(n^2)$. The same goes for $2000n^2$ (but the constants are different), so $2000n^2$ is also in $\Theta(n^2)$.

4. (15) For each of the following statements, tell whether it is “true” or “false”. When the correct answer is “false”, explain why. Do not give an explanation when the correct answer is “true”. Evaluate each statement independently of the others.

Example: It is possible for an algorithm to have a worst case cost in $O(n)$ and a best case cost in $\Omega(n^2)$.

False: The worst case cost of an algorithm is never lower than its best case cost. Since the best case cost is *at least* n^2 , the worst case cost must also be at least n^2 , so the worst case cannot be bounded from above by $O(n)$.

- (a) It is possible for an algorithm to have a worst case cost in $\Omega(n^3)$ and a best case cost in $O(n^3)$.
- (b) It is possible for an algorithm to have a worst case cost that is in both $\Theta(n^2)$ and $O(n^3)$.
- (c) It is possible for an algorithm to have a best case cost that is in both $\Theta(n^2)$ and $\Omega(n^3)$.
- (d) It is possible for an algorithm to have a worst case cost in $\Theta(n^2)$ and a best case cost in $\Theta(n^3)$.
- (e) If the worst case cost function $t(n)$ of an algorithm is in $O(n^2)$, and c and n_0 are the constants associated with this fact, then it is possible that $t(n) > n^3$ for some $n > n_0$.
5. (14) Exercise 3, p. 60. Do as the problem directs except that you do not have to prove your answers for parts (c), (d), and (e).
6. It would be interesting if we could find an algorithm with tight best, average, and worst case bounds that are all different. This problem yields such an algorithm.
- (a)(3) First, explain how any sorting algorithm can be modified to guarantee that its best case cost is in $\Theta(n)$, if it is not already.
- (b)(3) Now consider the following sorting algorithms given in the text. Find at least one with tight average and worst case bounds that are different than each other and greater than $\Theta(n)$. That's all you have to do.
- (i) selection sort
 - (ii) bubble sort
 - (iii) merge sort
 - (iv) quicksort
 - (v) insertion sort
 - (vi) heap sort

We can now observe that applying the solution to part (a) to the algorithm you selected in part (b) yields an algorithm with tight best, average, and worst case bounds that are all different.

7. (10) Find an expression in terms of k for the total number of non-leading zeros in the binary representations of the numbers $0, 1, \dots, 2^k - 1$. Your solution must be correct for all $k \geq 0$.

For example, for $k = 4$ there are 17 non-leading zeros, as you can count in the table given for the solution to Hw 1.5. (Note that the binary number 0 is considered to consist entirely of leading zeros.)

Hint: First find an expression for the total number of *leading* zeros and use that along with the idea behind the solution to Hw 1.5 to solve this problem.