MATH 243 Test #2 Solutions

There were two versions of Test #2 written up, and this document provides solutions to both of them. If you are in 243A, you should consult the parts of the solutions labelled \[ \text{A} \] if you are in 243E, consult the solutions labelled \[ \text{E} \]. In each of the problems, version \[ \text{A} \] of the test had the wording in the upper part of the \{+\} constructions, and \[ \text{E} \] had the lower wording. The numbering is the same order as Version \[ \text{A} \] of the test.

The letter \[ \text{Z} \] denotes the set of integers.

(1) Let \( f(n) \) be given as below. Find a function \( g(n) \) for which \( f(n) \) is \( O(g(n)) \). Your answer \( g(n) \) should be a simple function of the smallest order. (15 points)

\[ \text{A} \]
\[ (n^3 + 30n^2 \log n + 7^7(\log n)^{10} + n \log n) \left( 5 \cdot 2^n + 3^n + \frac{1}{12} n! \right) \left( 5^5 n + 23n^n + 3n \log n \right) \]

\[ \text{E} \]
\[ (4 \cdot 3^n + 8n^{10} + 8^8) \left( 8n^4 + 6n^3 \log n + 13n^2(\log n)^2 + n \right) \left( 15n! + 21n^n + \frac{30}{181} \cdot 8^2n \right) \]

Solution: Since \( f(n) \) is the product of each function, a big-O estimate needs to be found for each factor, then these are multiplied together. Each factor is the sum of several functions, so a big-O estimate is the function which grows the fastest when \( n \) is big.

For \[ \text{A} \], the fastest growing function in the first factor is \( n^3 \) (since any power of \( n \) grows faster than any power of \( \log n \)), the fastest growing function in the second factor is \( n! \), and the fastest growing function in the third factor is \( 23n^n \). Thus \( f(n) \) is \( O(n^3 \cdot n! \cdot 23n^n) \) and \( O(n^3 \cdot n! \cdot n^n) \).

For \[ \text{E} \], the fastest growing function in the first factor is \( 4 \cdot 3^n \), the fastest growing function in the second factor is \( 8n^4 \), and the fastest growing function in the third factor is \( 21n^n \). Thus \( f(n) \) is \( O(4 \cdot 3^n \cdot 8 \cdot n^4 \cdot 21 \cdot n^n) \) and \( O(n^3 \cdot 3^n \cdot n^n) \).

Grading: +5 points for choosing the largest term, +5 points for the addition rule, and +5 points for the product rule. Grading for common mistakes: −5 points if the answer was not of the smallest order; −2 points for each factor with a wrong big-O estimate; −1 point if a number was given in front of the function \( g(n) \).
(2) Let $S$ be a subset of \( \{ 1, 2, \ldots, 100 \} \) which contains at least \( \{ 51 \} \) elements.

a. Prove that there are two elements of $S$ whose difference is \( \{ 50 \} \). (10 points)

Solution: This is a Pigeonhole Principle problem, so you need to set up the pigeonholes. This needs to be done so that if any pigeonhole has two numbers in it, then these numbers differ by 50 (for $A$). So we can set up the first pigeonhole to only allow the numbers 1 and 51 in it, the second to only allow 2 and 52 in it, etc., with a final pigeonhole that only allows 50 and 100 in it.

The way that the pigeonholes are set up has nothing to do with the Pigeonhole Principle. The Pigeonhole Principle is true for any distribution of pigeons, and so is true for any distribution according to the rules of the pigeonholes. The only thing the Pigeonhole Principle says is that if there are more pigeons than pigeonholes, then some pigeonhole has at least two pigeons in it.

The number of pigeons is the same as the size of $S$, or 51, and the number of pigeonholes is 50 (obtained by looking at the lower number which is allowed in each pigeonhole). Thus some pigeonhole has at least \( \lceil \frac{51}{50} \rceil = 2 \) elements of $S$ in it, and these two numbers must differ by 51.

The only changes needed for $E$ are that we set up the pigeonholes as follows: Only allow the numbers 1 and 501 in the first pigeonhole, only allow the numbers 2 and 502 in the second pigeonhole, etc. Then there are 501 pigeons and 500 pigeonholes, so some pigeonhole has at least two elements of $S$ in it, and these numbers differ by 500.

Grading: +6 points for setting up the pigeonholes; +2 points for calculating the number of pigeons and pigeonholes, +2 points for mentioning the consequence of the Pigeonhole Principle.

b. Must there be two elements of $S$ with a difference of \( \{ 50 \} \), if $S$ only contains \( \{ 50 \} \) elements? (10 points)

Solution: In either case, the answer is no. If there are only 50 pigeons (for $A$), then they might be distributed one per pigeonhole, in which case no two numbers differ by 50. Some counterexamples were given; for instance $S = \{ 1, 2, \ldots, 50 \}$ has 50 elements but no two differ by exactly 50. Any other example constructed by taking exactly one element from each pigeonhole would also work.

Grading: +5 points for the answer “No”; +5 points for the explanation (a counterexample or an explanation of what can go wrong). Grading for common mistakes: −2 points for “$S$ might not contain two elements whose difference is 50 (or 500)”; −2 points for “\( \lceil \frac{50}{50} \rceil = 1 \)” without any elaboration.
A function $f : A \rightarrow B$ is onto if it satisfies the condition that

$$(\forall y \in B)(\exists x \in A)[f(x) = y].$$

(a) Give a condition (a boolean formula with quantifiers) which is equivalent to the function $f$ not being onto. (10 points)

Solution: The answer can be found by negating the symbolic formula given above, and simplifying. The answer is then the last line of the following (since no quantifiers have a $\neg$ to the left of them):

$$\neg(\forall y \in B)(\exists x \in A)[f(x) = y]$$

$$\exists y \in B)\neg(\exists x \in A)[f(x) = y]$$

$$\exists y \in B)(\forall x \in A)\neg[f(x) = y]$$

$$\exists y \in B)(\forall x \in A)[f(x) \neq y]$$

Grading: +5 points for the initial negation, +5 points for simplifying. Grading for common mistakes: -3 points if the order of the quantifiers was swapped; -2 points for each mistake; -1 point for each minor mistake made.

(b) Let $f : Z \rightarrow Z$ be the function defined by

$$f(x) = \left\lceil \frac{x}{5} \right\rceil.$$

Is $f$ onto? Prove your answer, using the definition, or your answer to (a), whichever is appropriate. (10 points)

Solution: This function is onto, and this can be shown by proving the definition holds for $f(x)$; that is, we must prove

$$(\forall y \in Z)(\exists x \in Z)\left\lceil \frac{x}{5} \right\rceil = y.$$

Since we have to prove a statement that begins with $(\forall y)$, we assume that $y$ is any integer; we do not necessarily know which one it is. Then what we need to show is that there is an $x$ with a certain property (i.e., $x$ is an integer and $\left\lceil \frac{x}{5} \right\rceil = y$). We can find an explicit formula for $x$; the simplest one is $x = 5y$. Then $x$ is an integer and

$$\left\lceil \frac{x}{5} \right\rceil = \left\lceil \frac{5y}{5} \right\rceil = \left\lceil y \right\rceil = y,$$

the last equality being true because $y$ is an integer. This $x$ works for any $y$, and so the proof is complete.

There are other choices for $x$ that work; any one of $5y - 1, 5y - 2, 5y - 3, \text{ or } 5y - 4$ will also work, although the proof is a bit more difficult in this case.

Grading: +3 points for “Yes”, +2 points for the outline of the proof, +5 points for the formula for $x$. Grading for common mistakes: +5 points total for an example (i.e., one value of $y$); +2 points total for “No” only.
A function \( f : A \to B \) is one-to-one if it satisfies the condition that

\[
(\forall x \in A)(\forall y \in A)[f(x) = f(y) \rightarrow x = y].
\]

(a) Give a condition (a boolean formula with quantifiers) which is equivalent to the function \( f \) not being one-to-one. (10 points)

Solution: The answer can be found by negating the symbolic formula given above, and simplifying. The answer is then the last line of the following (since no quantifiers have a \( \neg \) to the left of them):

\[
\neg(\forall x \in A)(\forall y \in A)[f(x) = f(y) \rightarrow x = y]
\]

\[
(\exists x \in A)\neg(\forall y \in A)[f(x) = f(y) \rightarrow x = y]
\]

\[
(\exists x \in A)(\exists y \in A)\neg[f(x) = f(y) \rightarrow x = y]
\]

\[
(\exists x \in A)(\exists y \in A)[f(x) = f(y) \land x \neq y]
\]

Grading: See A.

(b) Let \( f : Z \to Z \) be the function defined by

\[
f(x) = 3x + 2.
\]

Is \( f \) one-to-one? Prove your answer, using the definition, or your answer to (a), whichever is appropriate. (10 points)

Solution: This function is one-to-one, and we prove it using the definition. Specifically, we need to show that

\[
(\forall x \in Z)(\forall y \in Z)[3x + 2 = 3y + 2 \rightarrow x = y].
\]

Since the statement to be proven is a \( (\forall x) \), we let \( x \) be any integer. Since then we need to show that for every \( y \), some condition is true, we let \( y \) be any integer, not necessarily equal to \( x \). Then we need to show that

\[
3x + 2 = 3y + 2 \rightarrow x = y,
\]

so let us assume that \( 3x + 2 = 3y + 2 \) (the first part). The goal is then to show that \( x = y \).

We can subtract 2 from both sides of \( 3x + 2 = 3y + 2 \) to obtain \( 3x = 3y \). Dividing by 3 yields \( x = y \), which is what we wanted to show. Hence the result is true, and the proof is complete.

Grading: See A. Proofs based on the fact that the graph of the function \( y = 3x + 2 \) is a straight line which is not horizontal were also accepted as correct.
(4) Let \( f(x) \) and \( g(x) \) be functions such that \( f(x) \) is \( \{ O(x^3) \} \) and \( g(x) \) is \( \{ O(x \log x) \} \). Using the definition of big-O notation, show that \( f(x)g(x) \) is \( \{ O(x^4 \log x) \} \). (15 points)

Solution: The definition of big-O notation is the following: A function \( F(x) \) is \( O(G(x)) \) if

\[
(\exists k)(\exists N)[x \geq N \rightarrow F(x) \leq k \cdot G(x)].
\]

Thus, the assumptions about \( f(x) \) and \( g(x) \) say that

\[
(\exists k)(\exists N)[x \geq N \rightarrow f(x) \leq k \cdot x^3] \quad \text{and} \quad (\exists k)(\exists N)[x \geq N \rightarrow g(x) \leq k \cdot x \log x],
\]

for \( A \). What we want to show is that

\[
(\exists k)(\exists N)[x \geq N \rightarrow f(x)g(x) \leq k \cdot x^4 \log x].
\]

Note that the product rule given in class proves this result, but you were asked to use the definition here.

So we need to find \( k \) and \( N \) which make the statement above (about \( f(x)g(x) \)) true. What we can assume is that there are constants which make the statements above (about \( f(x) \) and \( g(x) \)) true. **Note that the values of \( k \) and \( N \) might be different for \( f(x) \) from those which work for \( g(x) \).** Thus, we can only say that there are constants \( k_1 \) and \( N_1 \) such that \( x \geq N_1 \rightarrow f(x) \leq k_1 \cdot x^3 \) and there are constants \( k_2 \) and \( N_2 \) such that \( x \geq N_2 \rightarrow g(x) \leq k_2 \cdot x \log x \).

If \( x \geq N_1 \) and \( x \geq N_2 \), then we can access the statements that say that \( f(x) \leq k_1 \cdot x^3 \) and \( g(x) \leq k_2 \cdot x \log x \). Multiplying these inequalities together (left-hand sides together and right-hand sides together) yields

\[
f(x)g(x) \leq (k_1 \cdot x^3)(k_2 \cdot x \log x) = (k_1k_2) \cdot x^3 \cdot x \log x = (k_1k_2) \cdot x^4 \log x.
\]

Thus we can let the \( k \) we are looking for be \( k_1k_2 \). To choose a suitable \( N \), we need \( N \geq N_1 \) and \( N \geq N_2 \), so we can let \( N = \max(N_1, N_2) \). These are the values of \( k \) and \( N \) we need to show that \( f(x)g(x) \) is \( O(x^4 \log x) \).

For \( E \), the procedure is the same, except “\( x^3 \)” is replaced by “\( x^4 \)” and “\( x \log x \)” is replaced by “\( (\log x)^2 \)”.

**Grading:** +5 points for the definition of big-O notation, +5 points for the formula for \( N \), +5 points for the formula for \( k \). Grading for common mistakes: −5 points if it was assumed that \( k_1 = k_2 \) and \( N_1 = N_2 \); −2 points for \( N = N_1N_2 \) (which works if \( N_1 \) and \( N_2 \) are positive, but either of these could conceivably be negative); −7 points if the result from class was used.
(5) Let $F$ be the function which doubles each character of the string $x$; e.g., $F(abc) = aabbcc$. Give a recursive definition for $F(x)$, where the recursion is with respect to the length of $x$. (15 points)

Solution: First, a general note for both definitions: A recursive definition is one that relates $F(x)$ to $F$ evaluated at smaller objects which are the same (data)type as $x$. If $x$ is a positive integer, $F$ must be evaluated at some positive integer less than $x$ in order for $F$ to be recursive. In this case, $x$ is a string, so $F$ must be evaluated with a string shorter (having fewer characters) than $x$. So a natural thing to do is to consider $F(abc)$ and see how it relates to $F(ab)$ or $F(bc)$. This relationship can be turned into a definition.

For $F(abc) = aabb$, which is the same as $F(abc)$, except it is missing two copies of $c$ at the end. This can be remedied by gluing two copies of $c$ onto the end of $F(ab)$. The recursive part of the recursive definition would then be something like

$$F(u \cdot y) = F(u) \cdot y \cdot y,$$

where $u \cdot y = x$ and $y$ is the last character of $x$. $(u \text{ contains the rest of } x$. The dot $\cdot$ denotes concatenation.) The string $x$ could also be broken into two smaller strings, if it has at least two characters, and if $x = u \cdot v$ (the strings $u$ and $v$ glued together),

$$F(u \cdot v) = F(u) \cdot F(v).$$

Either of these formulas can be used for the recursive part of the definition.

Now for the base case. If $x$ only contains one character, it cannot be broken into two smaller strings, so this is a natural base case (strings which contain exactly one character). We would then want to define $F(x) = x \cdot x$ if $x$ only has one character. This base case goes with either of (1) or (2) above.

The base case could also be a string with no characters ($\varepsilon$). Then we would want to define $F(\varepsilon) = \varepsilon$.

Note that this would only work with the formula (1), unless $F$ was also defined for strings of length 1. We then need to combine a base case with a recursive formula, to get a formula such as

$$F(x) = \begin{cases} x, & \text{if } x \text{ consists of a single character;} \\ F(u) \cdot y \cdot y, & \text{if } x = u \cdot y, \text{ and } y \text{ is a single character.} \end{cases}$$

Now for $F(abc)$ again, the relationship between $F(abc)$ and $F(ab)$ or $F(bc)$ needs to be examined. It turns out that if $x = u \cdot y$ (notation used for gluing $y$ onto the end of $u$), then

$$F(x) = F(u) + F(y),$$

where the addition is ordinary addition. Once again, a base case is needed, so $F(\varepsilon)$ or $F(y)$ (where $y$ is a single character) needs to be defined non-recursively. In this case, $F(y) = 0$ if $y$ is any character other than $a$. So a definition that would work would be

$$F(x) = \begin{cases} 1, & \text{if } x \text{ consists of the character } a; \\ 0, & \text{if } x \text{ consists of a character other than } a; \\ F(u) + F(y), & \text{if } x = u \cdot y, \text{ and } u \text{ and } y \text{ are strings of positive length.} \end{cases}$$

In addition, programs written in pseudocode were accepted. In order for a routine to be recursive, it must call itself somewhere in its code; some answers lost 5 points because the routine was written non-recursively. Another problem that some routines had is that some variables need to be reset when the routine is called the second time; these routines lost 1 point in grading.

Grading: +5 points for the base case, +10 points for the recursive case. Grading for common mistakes: −5 points if the length of $F(x)$ was calculated; +8 points (total) for $F(x) = x^2$ (for $F(ab)$); (abc)$^2 = (abc) \cdot (abc) = abcabc$, not aabbcc); for pseudocode, see the previous paragraph.
Consider the sequence defined by

\[ s_n = \begin{cases} 
0, & \text{if } n = 0; \\
3s_{n-1} + 1, & \text{if } n > 0.
\end{cases} \]

(The first few terms of this sequence are 0, 1, 4, 13, 40, \ldots.) Use mathematical induction to show that

\[ s_n = \frac{1}{2}(3^n - 1), \]

for all nonnegative integers \( n \). (15 points)

**Solution:** To use mathematical induction to prove a proposition \( P(n) \), two things need to be done: (1) Prove \( P(0) \); (2) Prove that \( P(k) \) implies \( P(k + 1) \). The base case for this problem is \( n = 0 \), since the statement to be shown is supposed to be shown for all nonnegative integers.

The proposition \( P(n) \) here is \( s_n = \frac{1}{2}(3^n - 1) \), not just the right-hand side expression, and not \( s_1 + s_2 + \cdots + s_2 = \frac{1}{2}(3^n - 1) \).

To prove step (1), we show that \( P(0) \) is true: \( \frac{1}{2}(3^0 - 1) = \frac{1}{2} \cdot 0 = 0 = s_0 \).

To prove step (2), we assume that \( P(k) \) is true, namely that

\[ s_k = \frac{1}{2}(3^k - 1), \]

with the intention to prove \( P(k + 1) \), namely

\[ s_{k+1} = \frac{1}{2}(3^{k+1} - 1). \]

To do this, we use the definition of \( s_n \), given above. Note that \( s_{k+1} \neq s_k + 1 \).

\[ s_{k+1} = 3s_k + 1 = 3 \cdot \frac{1}{2}(3^k - 1) + 1 = \frac{3}{2} \cdot 3^k - \frac{3}{2} + \frac{2}{2} = \frac{3^{k+1}}{2} - \frac{1}{2} = \frac{1}{2}(3^{k+1} - 1). \]

Grading: +5 points for showing that \( P(0) \) is true; +5 points for setting up the proof of \( P(k) \rightarrow P(k + 1) \); +5 points for the actual proof. Grading for common mistakes: +5 points total for trying to prove that \( s_1 + s_2 + \cdots + s_2 = \frac{1}{2}(3^n - 1) \) (which is not true); -1 points for a base case of \( n = 1 \) or \( n = 2 \) instead of \( n = 0 \).
Consider the sequence defined by

\[ s_n = \begin{cases} 
0, & \text{if } n = 0; \\
4s_{n-1} + 1, & \text{if } n > 0. 
\end{cases} \]

(The first few terms of this sequence are 0, 1, 5, 21, 85, \ldots.) Use mathematical induction to show that

\[ s_n = \frac{1}{3}(4^n - 1), \]

for all nonnegative integers \( n \). (15 points)

**Solution:** See the first two paragraphs of version \( A \) of this problem for general comments, except that here \( P(n) \) denotes the proposition that

\[ s_n = \frac{1}{3}(4^n - 1). \]

First, show that \( P(0) \) is true: \( s_0 = 0 \), and \( \frac{1}{3}(4^0 - 1) = 0 \).

Now assume that \( P(k) \) is true, that

\[ s_k = \frac{1}{3}(4^k - 1); \]

then we need to show that \( P(k + 1) \) is also true, namely that

\[ s_{k+1} = \frac{1}{3}(4^{k+1} - 1). \]

We do this like in the previous problem:

\[ s_{k+1} = 4s_k + 1 = 4 \cdot \frac{1}{3}(4^k - 1) + 1 = \frac{4 \cdot 4^k}{3} - \frac{4}{3} + \frac{3}{3} = \frac{4^{k+1}}{3} - \frac{1}{3} = \frac{1}{3}(4^{k+1} - 1). \]

**Grading:** See the previous problem.

MAT 243 Website: http://math.la.asu.edu/~checkman/243/