

ARS DIGITA UNIVERSITY
MONTH 2: DISCRETE MATHEMATICS
PROFESSOR SHAI SIMONSON
PROBLEM SET 2 SOLUTIONS — SET, FUNCTIONS, BIG-O, RATES OF GROWTH

(1) **Prove by formal logic:**

(a) **The complement of the union of two sets equals the intersection of the complements.**

Let $x \in \overline{(A \cup B)}$. Then x is not in A , and x is not in B . Therefore $x \in \overline{A} \cap \overline{B}$. Since x was an arbitrary element of $\overline{(A \cup B)}$, we conclude that $\overline{(A \cup B)} \subseteq \overline{A} \cap \overline{B}$. Now, suppose that $x \in \overline{A} \cap \overline{B}$. We can conclude that $x \in \overline{A}$, and that $x \in \overline{B}$. This implies that $x \notin A$, and $x \notin B$, which implies $x \notin (A \cup B)$, which implies that $x \in \overline{(A \cup B)}$. Again, since x was arbitrary, we conclude that $\overline{(A \cup B)} \subseteq \overline{A} \cap \overline{B}$. Two sets which are subsets of each other are equal (see below), so we can conclude that $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$.

(b) **The complement of the intersection of two sets equals the union of the complements.**

Let $x \in \overline{(A \cap B)}$. Then x cannot be in both A and B — at least one of the statements “ $x \in \overline{A}$ ”, “ $x \in \overline{B}$ ” is true. Therefore $x \in \overline{A} \cup \overline{B}$, and $\overline{(A \cap B)} \subseteq \overline{A} \cup \overline{B}$. Now let $x \in \overline{A} \cup \overline{B}$. Again, x cannot be in both A and B , so $x \notin A \cap B$, so $x \in \overline{(A \cap B)}$, and $\overline{A} \cup \overline{B} \subseteq \overline{(A \cap B)}$. The two sets are equal.

(c) $(B - A) \cup (C - A) = (B \cup C) - A$.

Let $x \in (B - A) \cup (C - A)$. Then $x \in (B - A) \vee x \in (C - A)$. Assume without loss of generality that $x \in (B - A)$. This implies that $x \in B$ and $x \notin A$. Which implies that $x \in (B \cup C)$, and, combining this with the fact that $x \notin A$, we conclude that $x \in (B \cup C) - A$, and $(B - A) \cup (C - A) \subseteq (B \cup C) - A$. If $x \in (B \cup C) - A$, then $x \notin A$, and $x \in B \vee x \in C$. Therefore $x \in (B - A) \vee x \in (C - A)$, $x \in (B - A) \cup (C - A)$, and $(B \cup C) - A \subseteq (B - A) \cup (C - A)$, completing the proof.

(d) **If two sets are subsets of each other than they are equal.**

If two sets A and B are not equal, then there exists some element that is in one set that is not in the other. Without loss of generality, denote such an element x and assume that $x \in A$. Since $x \in A$ and $x \notin B$, we conclude that $A \not\subseteq B$.

(2) **Generalize De Morgan’s laws for n sets and prove the laws by induction.**

The n -version generalization of De Morgan's laws are as follows:

$$\begin{aligned}\overline{\bigcup_{i=1}^n A_i} &= \bigcap_{i=1}^n \overline{A_i} \\ \overline{\bigcap_{i=1}^n A_i} &= \bigcup_{i=1}^n \overline{A_i}\end{aligned}$$

To prove these laws by induction, we use $n = 2$ as our base cases: these are the traditional two-set De Morgan's laws. Now, we assume that the laws hold for n sets, and show that they hold for $n + 1$ sets as well, using both our inductive hypothesis and the two-set De Morgan's laws:

$$\begin{aligned}\overline{\bigcup_{i=1}^{n+1} A_i} &= \overline{\left(\bigcup_{i=1}^n A_i\right) \cup A_{n+1}} \\ &= \overline{\bigcup_{i=1}^n A_i} \cap \overline{A_{n+1}} \\ &= \left(\bigcap_{i=1}^n \overline{A_i}\right) \cap \overline{A_{n+1}} \\ &= \bigcap_{i=1}^{n+1} \overline{A_i} \\ \overline{\bigcap_{i=1}^{n+1} A_i} &= \overline{\left(\bigcap_{i=1}^n A_i\right) \cap A_{n+1}} \\ &= \overline{\bigcap_{i=1}^n A_i} \cup \overline{A_{n+1}} \\ &= \left(\bigcup_{i=1}^n \overline{A_i}\right) \cup \overline{A_{n+1}} \\ &= \bigcup_{i=1}^{n+1} \overline{A_i}\end{aligned}$$

- (3) **Prove by induction on the size of the set, that the power set $P(A)$ has cardinality $2^{|A|}$.**

For a set S containing a single element x , the power set is $\{\emptyset, \{x\}\}$, which is of size 2. This is our base case. Now assume that a set A of size n has power set with cardinality $2^{|A|} = 2^n$. Consider adding a new element a to A to make the set A' , of size $n + 1$. The power set of A' will consist of all the sets in the power set of A , plus all those sets taken again, with the element a added. Every set in $P(A)$ gives rise to two sets in $P(A')$. We conclude that $|P(A')| = |2P(A)| = 2(2^n) = 2^{n+1} = 2^{|A'|}$.

(4) $A \oplus B$ is defined to be the set of all elements in A or B but not in both A and B .

(a) Determine whether or not \oplus is commutative. Prove your answer.

We check this by means of a membership table.

A	B	$A \oplus B$	$B \oplus A$
1	1	0	0
1	0	1	1
0	1	1	1
0	0	0	0

We conclude that \oplus is commutative.

(b) Determine whether \oplus is associative. Prove your answer.

A	B	C	$A \oplus B$	$B \oplus C$	$(A \oplus B) \oplus C$	$A \oplus (B \oplus C)$
1	1	1	0	0	1	1
1	1	0	0	1	0	0
1	0	1	1	1	0	0
1	0	0	1	0	1	1
0	1	1	1	0	0	0
0	1	0	1	1	1	1
0	0	1	0	1	1	1
0	0	0	0	0	0	0

We conclude that \oplus is associative; indeed, \oplus acts as a parity operator: if we “ \oplus ” n sets together, we will get a set consisting of all those elements that are in an odd number of our sets.

(c) Determine whether \oplus can be distributed over union. Prove your answer.

A	B	C	$B \cup C$	$A \oplus (B \cup C)$	$A \oplus B$	$A \oplus C$	$(A \oplus B) \cup (A \oplus C)$
1	1	1	1	0	0	0	0
1	1	0	1	0	0	1	1
1	0	1	1	0	1	0	1
1	0	0	0	1	1	1	1
0	1	1	1	1	1	1	1
0	1	0	1	1	1	0	1
0	0	1	1	1	0	1	1
0	0	0	0	0	0	0	0

Looking at the second and third lines, we see that \oplus does not distribute over union. In particular, if an element x is in a set A and is in *one* of B or C , *but not both*, then $x \in (A \oplus B) \cup (A \oplus C)$, but $x \notin A \oplus (B \cup C)$.

(d) Determine whether \oplus can be distributed over intersection. Prove your answer.

A	B	C	$B \cap C$	$A \oplus (B \cap C)$	$A \oplus B$	$A \oplus C$	$(A \oplus B) \cap (A \oplus C)$
1	1	1	1	0	0	0	0
1	1	0	0	1	0	1	0
1	0	1	0	1	1	0	0
1	0	0	0	1	1	1	1
0	1	1	1	1	1	1	1
0	1	0	0	0	1	0	0
0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0

Reasoning as above, we see that \oplus does not distribute over intersection. If an element x is in a set A and is in *one* of B or C , *but not both*, then $x \in A \oplus (B \cap C)$, but $x \notin (A \oplus B) \cap (A \oplus C)$.

- (5) **Assume a universal set of 8 elements. Given a set $A = a_7a_6a_5a_4a_3a_2a_1a_0$, represented by 8 bits, explain how to use bitwise and/or/not operations, in order to:**

- (a) **Extract the rightmost bit of set A .**

To “extract” the rightmost bit of set A , we compute $A \wedge 00000001$.

- (b) **Extract the odd numbered bits.** To “extract” the odd numbered bits, we compute $A \wedge 10101010$.

- (c) **Make bits 4-6 equal to 1.** To set bits 4-6 equal to 1, without changing anything else, we compute $A \vee 01110000$.

Given another set B , explain how to:

- (a) **Determine if $A \subseteq B$.** $A \subseteq B$ iff $A - B = 0$ (and see below)

- (b) **Extract $A - B$.** $A - B$ can be computed as $A \wedge (\overline{A \wedge B})$.

- (6) **The Inclusion/Exclusion theorem for three sets says that if there are three sets A , B , and C , then**

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

Note that the converse of this theorem is not true.

Given a set of 29 students, where 8 need housing and financial aid; 12 need housing, financial aid, and an e-mail account; 17 need an e-mail account and financial aid; 23 need housing, 20 need financial aid, 19 need e-mail accounts and 4 students don't need anything:

- (a) **How many students need both housing and e-mail?**

Let H , F , and E denote the sets of students needing housing, financial aid, and an e-mail account, respectively. We are trying to compute $|H \cap E|$ from the Inclusion/Exclusion principal, given everything else:

$$\begin{aligned} |H \cap E| &= |H| + |F| + |E| - |H \cap F| - |F \cap E| + |H \cap F \cap E| - |H \cup F \cup E| \\ &= 23 + 20 + 19 - 8 - 17 + 12 - 25 \\ &= 24 \end{aligned}$$

- (b) **Change the numbers to 57,8,3,21,21,32,31,8. What is your answer now?**

$$\begin{aligned} |H \cap E| &= 21 + 32 + 31 - 8 - 21 + 3 - 49 \\ &= 3 \end{aligned}$$

- (c) **Which answer is bogus and why?**

The first answer is bogus. It is impossible to have 24 students requiring housing and an email account, when only 23 students require housing. Sets that satisfy the characteristics given in the first part of the problem do not exist.

- (7) **How many numbers are there between 1 and 10,000, which are either even, end in 0, or have the sum of their digits divisible by 9? Hint: Use inclusion/exclusion.**

Let E , Z and N denote the numbers between 1 and 10,000 which are even, end in 0, or have the sum of their digits divisible by 9, respectively (not that N consists of all *multiples of 9*. Then, (including the number 10,000 in our calculations), $|E| = 5000$, $|Z| = 1000$, $|N| = 1111$, $|E \cap Z| = 1000$, $|E \cap N| = 555$, $|Z \cap N| = 111$, $|E \cap Z \cap N| = 111$. Therefore, by the Inclusion/Exclusion principle:

$$\begin{aligned} |E \cup Z \cup N| &= 5000 + 1000 + 1111 - 1000 - 555 - 111 + 111 \\ &= 5556 \end{aligned}$$

- (8) **Prove that $f(x) = x^3 - 1000x^2 + x - 1$ is $\Omega(x^3)$ and $O(x^3)$.**

For all $x > 10,000$:

$$\begin{aligned} f(x) &= x^3 - 1000x^2 + x - 1 \\ &> x^3 - 1000x^2 \\ &= (x - 1000)x^2 \\ &> (.9x)x^2 \\ &= .9x^3 \end{aligned}$$

Therefore, $f(x)$ is $\Omega(x^3)$ with $C = .9$, $k = 10,000$.

Also, for all $x > 0$:

$$\begin{aligned} f(x) &= x^3 - 1000x^2 + x - 1 \\ &< x^3 + 1000x^2 + x^3 + x^3 \\ &= 1002x^3 \end{aligned}$$

Therefore, $f(x)$ is $O(x^3)$ with $C = 1002$, $k = 1$.

Note that these result are a direct consequence of Theorem 4 on page 90 of Rosen.

(9) Prove that $1^k + 2^k + 3^k + \dots + n^k$ is $\Omega(n^{k+1})$ and $O(n^{k+1})$, for any constant k .

$$\begin{aligned}
 1^k + 2^k + 3^k + \dots + n^k &> \left(\frac{n}{2}\right)^k + \left(\frac{n}{2} + 1\right)^k + \dots + n^k \\
 &> \underbrace{\left(\frac{n}{2}\right)^k + \left(\frac{n}{2}\right)^k + \dots + \left(\frac{n}{2}\right)^k}_{\frac{n}{2} \text{ terms}} \\
 &= \frac{n}{2} \left(\frac{n}{2}\right)^k \\
 &= \left(\frac{n}{2}\right)^{k+1}
 \end{aligned}$$

Therefore, $f(n) = 1^k + 2^k + 3^k + \dots + n^k$ is $\Omega(n^{k+1})$: $f(n) > C(n^{k+1})$, where $C = \frac{1}{2^k}$. Note that this constant depends on k . $f(n)$ is also $O(n^{k+1})$:

$$\begin{aligned}
 1^k + 2^k + 3^k + \dots + n^k &< \underbrace{n^k + n^k + n^k + \dots + n^k}_{n \text{ terms}} \\
 &= n(n^k) \\
 &= n^{k+1}
 \end{aligned}$$

(10) Order the following functions in order of their growth rate. If $f(x)$ is $O(g(x))$, but $g(x)$ is not $O(f(x))$, then put $f(x)$ above $g(x)$. If they are each big- O of each other, then place them on the same level.

$$\begin{array}{cccccccc}
 x^2 + x^3 & 3^x & x! & \frac{x}{\log_2 x} & x^2 + 2^x & x \log_2 x & 2^{x \log x} & \log x^2 \\
 \log_2 x! & \log_2 x & \ln x & 2^x & x(1 + 2 + \dots + x) & \log \log x & 2^{x^2} & \log^2 x
 \end{array}$$

We list functions in increasing order of growth. Functions on the same line have identical orders of growth.

$$\begin{array}{cccc}
 & & & 2^{x^2} \\
 & & & x! \quad 2^{x \log x} \\
 & & & 3^x \\
 & & & x^2 + 2^x \quad 2^x \\
 x^2 + x^3 & x(1 + 2 + \dots + x) & & \\
 x \log_2 x & \log_2 x! & & \\
 & & & \frac{x}{\log_2 x} \\
 & & & \log^2 x \\
 \log x^2 & \log_2 x & \ln x & \\
 & \log \log x & &
 \end{array}$$

(11) Compute the sum of the infinite series below:

(a) $1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots$

Using the first formula in Table 2 on page 76 of the text (Rosen), we have a ratio series with $a = 1, r = \frac{1}{4}$:

$$\begin{aligned}
 1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots &= \sum_{k=0}^{\infty} \left(\frac{1}{4}\right)^k \\
 &= \frac{\left(\frac{1}{4}\right)^{\infty} - 1}{\frac{1}{4} - 1} \\
 &= \frac{-1}{-\frac{3}{4}} \\
 &= \frac{4}{3}
 \end{aligned}$$

(b) $1 + \frac{2}{4} + \frac{3}{16} + \frac{4}{64} + \dots$

Letting $S = 1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots = \frac{4}{3}$ (the sum we computes in the first part of this problem), we can rewrite the current series as follows:

$$\begin{aligned}
 1 + \frac{2}{4} + \frac{3}{16} + \frac{4}{64} + \dots &= \begin{array}{r} (1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots) \\ + (\frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots) \\ + (\frac{1}{16} + \frac{1}{64} + \dots) \\ + (\frac{1}{64} + \dots) \\ + \dots \end{array} \\
 &= \begin{array}{r} (1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots) \\ + \frac{1}{4}(1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots) \\ + \frac{1}{16}(1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots) \\ + \frac{1}{64}(1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots) \\ + \dots \end{array} \\
 &= S + \frac{1}{4}S + \frac{1}{16}S + \frac{1}{64}S + \dots \\
 &= S(1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots) \\
 &= S^2 \\
 &= \frac{16}{9}
 \end{aligned}$$

(12) **Telescoping Series.**

(a) **Using the identity**

$$\frac{1}{k(k+1)} = \frac{1}{k} - \frac{1}{k+1}$$

find the value of the infinite sum $\frac{1}{1*2} + \frac{1}{2*3} + \frac{1}{3*4} + \dots$

$$\begin{aligned}
 \frac{1}{1*2} + \frac{1}{2*3} + \frac{1}{3*4} + \dots &= (1 - \frac{1}{2}) + (\frac{1}{2} - \frac{1}{3}) + (\frac{1}{3} - \frac{1}{4}) + \dots \\
 &= 1 - \frac{1}{2} + \frac{1}{2} - \frac{1}{3} + \frac{1}{3} - \frac{1}{4} + \dots \\
 &= 1
 \end{aligned}$$

(b) The n th triangle number is defined to be the sum of the first n positive integers. What is the value of the infinite sum of the reciprocals of the triangle numbers?

The n th triangle number is $\frac{n(n+1)}{2}$. Therefore, the infinite sum of the reciprocals of the triangle numbers is simply double the series above, or 2.

(13) **Countability Proofs.**

(a) **Prove that the positive odd numbers have the same cardinality as the positive even numbers.** We show that the positive odd numbers have the same cardinality as the positive even numbers simply by noting that the function $f(x) = x + 1$ is a one-to-one mapping from the positive odd numbers to the positive even numbers.

(b) **Prove that the set of all integer points in the positive quadrant of 3-dimensional space are countable.** This will follow immediately from the next part of the problem (below).

(c) **Prove by induction that the set of all vectors in k dimensions with positive integer values is countable.**

As our base case, we use $k = 1$: this is the natural numbers, which are countable. Now assume that the set V_k of all k dimensional vectors with positive integer values is countable. Then there exists some enumeration of this V_k ; denote the i th element in this enumeration by V_{k_i} . We now show how to construct an enumeration V_{k+1} . We consider a two dimensional table (infinite in both directions), with the enumeration V_k along one axis and the natural numbers along the other axis. We then proceed “triangularly” through this table, taking the upper-left most entry, then the two entries along the “diagonal” next to that entry, then the three entries along the next diagonal, and so forth. The first few elements of our enumeration for V_{k+1} are $(V_{k_1}, 1), (V_{k_2}, 1), (V_{k_1}, 2), (V_{k_3}, 1), (V_{k_2}, 2), (V_{k_3}, 1), \dots$

(d) **Prove that the number of scheme programs is countable. Hint: Order them by the number of characters each contains.**

We order the scheme programs by the number of characters each contains. Because scheme programs are composed from a finite alphabet, for any fixed length, there are a finite number of scheme programs of that length. The set of all scheme programs is the union of the set of scheme programs of length k , for all positive integers k . So the set of all scheme programs is the union of a countable number of finite sets, which is countable.

(14) **The following is a version of Russell’s Paradox for sets, described in your text (Rosen) in problem 26, on page 45. Consider any computer program that takes other programs as inputs, and outputs *yes* or *no* based on some criterion. (A compiler or interpreter, for example). It is possible for such a program to be fed back into itself, and depending on the program it might either say *yes* (a Scheme interpreter written in Scheme) or *no* (a Scheme interpreter written in Java). Call the programs that say *no* to themselves *self-hating* programs. Now assume that there is a computer program that takes other programs as inputs and says *yes* to all the *self-hating* programs, and no otherwise.**

(a) **Assuming this program never runs forever on an input, analyze what happens when this program is input to itself.**

Consider first the possibility that our hypothetical program P says *yes* when given itself as input. Then, since our program is defined to be a program that

says *yes* only to self-hating programs, we can conclude that P is self-hating. But the definition of self-hating programs is that they say *no* when given themselves as input. So we conclude that P cannot say *yes* to itself.

Now consider the possibility that P says *no* to itself. Then, by how P was defined, we can conclude that P is not self-hating. But if P is not self-hating, then it must say *yes* when given itself as input. So P cannot say *no* to itself either.

We conclude that there cannot exist a program which halts on all inputs, and says *yes* to all self-hating programs and *no* to all other programs.

- (b) **Consider the possibility that the program will run forever answering neither *yes* nor *no* on some input.**

This offers a way out of the dilemma posed above. The program P could run forever if given itself as input. It is possible to make a program that with the property that if it says *yes* to an input I , then I represents a self-hating program, and if it says *no* to an input I , then I does not represent a self-hating program, *if we allow that the machine may run forever on some instances*. A trivial example is a program that runs forever on *all* instances — it satisfies the requirements, although it is obviously not very useful.

- (15) **Counting each arithmetic calculation or comparison, extraction or exchange of a card as one operation, what is the worse-case order of growth of an algorithm that sorts numbered cards in the following way?**

- (a) **Find the largest valued card in the deck by shuffling through one card at a time extracting a card if it is the largest one seen so far, and swapping the previously largest card back into the deck. When the largest has been found, place this card face down in a new pile and repeat the previous process until no cards in the original pile are left. Explain your answer.**

Each time we make a pass through the deck, we extract the largest card. Each pass through the deck takes $O(k)$ time, where k is the number of cards remaining in the deck during that pass. The total number of times for all the passes is:

$$\begin{aligned} \Theta(n + (n - 1) + (n - 2) + (n - 3) + \dots + 2 + 1) &= \Theta\left(\frac{n(n + 1)}{2}\right) \\ &= \Theta(n^2) \end{aligned}$$

- (b) **This time we assume that the largest number on any of the n cards is n^2 . We sort the cards by placing a set of n^2 cards numbered from 1 to n^2 on a table. Then one by one, place each card on top of the number equal to it on the desk. The sorted list can be extracted by looking through all n^2 piles in order.**

To make the explanation clearer, we refer to the original cards we are trying to sort as *cards*, and will refer to the extra set of cards numbers 1 to n^2 that we place on the table as *slots*; it is not important that they are cards, they are merely “placeholders”.

It takes $\Theta(n^2)$ time to create the slots numbered 1 to n^2 on the table. We can then place the cards in their appropriate slot in a total time of $\Theta(n)$. To obtain

the final sorted list, we have to examine each of the n^2 slots, and pick up the cards in that slot (if any). It takes $\Theta(n^2)$ time to examine the slots, and a total of $\Theta(n)$ time to pick up the cards. The total time is therefore $\Theta(n^2)$, as above.

- (c) **This method can be improved to work in linear time. Explain how. Hint: This is not easy. Use division, to try to turn each number into a pair of numbers, each with a value between 1 and n .**

We begin by creating slots numbered 1 to n on the table. We next divide each of our numbers by n , saving both the quotient and the remainder. Then, we place each number in the slot that corresponds to its *remainder*. We pick up the cards, starting from slot 1 and proceeding to slot n . At this point, we have the cards sorted in order of increasing remainder — cards that have the same remainder are in no particular order. We call this step the *remainder pass*.

Next, we place the cards in the slots again, this time placing each card in the slot corresponding to its *quotient*. We proceed through the slots again, picking up the cards, *putting cards that were placed in the slot first before cards that were placed in the slot later*. We call this step the *quotient pass*. After the quotient pass, we are done.

It should be clear that this algorithm operates in linear time — we perform a constant number of steps, each of which take linear time. Does it work? Are the cards sorted after the second pass? Yes. Clearly, the second pass will order any two cards with different quotients properly (in fact, if we didn't care about getting a complete ordering, but only cared about getting cards with different quotients in the correct order, we could just do the second pass, ignoring the initial, remainder pass). Now, consider any two cards with the same quotient, but different remainders. The card with the smaller remainder will be earlier in the pile after the remainder pass, so it will be placed in the appropriate quotient pile *before* the card with the larger remainder.