Discrete Structures Review

Computational Structures builds on the basic theory of Discrete Structures (CS250/251) to explore core theoretical concepts and issues in Computer Science. This class gives you a theoretical foundation that will be very beneficial as you study programming languages, algorithms, databases, cryptography, and much more.

We will approach the subject from three directions, where each reinforces the others. Our text ([3]) uses a particular building-block approach that we will adopt; namely, Automata and Languages, followed by Computability Theory, and finally Complexity Theory. We will use *models* of computing so that we may reason about computing without worrying about unnecessary details.

This subject is hard. Please take the class seriously and **DO NOT FALL BEHIND**. Study groups are *very strongly encouraged*, so please use the course D2L site or some other mechanism to network and find a group.

1 Sets

A *set* is an unordered collection of objects. Objects in a set are referred to as *members* or *elements*.

A set can be defined in several ways. The simplest way, when possible, is to list the members of the set. For example, $S = \{12, 8, -3, 0\}$ defines S as a set with four elements. Since a set is *unordered*, the sets $\{12, 8, -3, 0\}$ and $\{-3, 0, 8, 12\}$ are identical. Repetition is ignored in sets; the sets $\{7, 8\}$ and $\{7, 7, 8\}$ are the same. We write $8 \in \{12, 8, -3, 0\}$ and $7 \notin \{12, 8, -3, 0\}$ to indicate that a value is or is not a member of a set.

For two sets A and B, A is a *subset* of B, written $A \subseteq B$, if every member of A is also a member of B. If every member of B is also a member of A, so that $A \subseteq B$ and $B \subseteq A$, then $A = B$. If A is a subset of B but $A \neq B$ then A is a *proper subset* of B, written $A \subset B$.[1]

Normally, the order of the elements within a set or the repetition of members within a set does not matter. If we wish to take in to account the number of occurrences of individual members within a set, we call this grouping a *multiset*. Thus, {7} and {7, 7} are the same *set* but different *multisets*.

When we cannot define a set by listing all its members, we do so by formula or by rule. For example, the natural numbers, $\mathcal{N}$, cannot be listed exhaustively, so it is convention to use the ellipse, "...", as in $\mathcal{N} = \{1, 2, 3, \ldots\}$. The set of integers is similarly defined as $\mathcal{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$. When we wish to express a set according to some rule, we write $\{n|$ rule about $n\}$. Thus, $\{n|n = m^2$ for $m \in \mathcal{N}\}$ means the set of perfect squares. The *empty set* has no members and is written $\emptyset$.

Three common set operations are *union*, *intersection*, and *complement*. The union of two sets A and B, written $A \cup B$, is the set containing *all* members of both sets A and B. The intersection of two sets A and B, written $A \cap B$, is the set containing only those members which sets A and B have in common. The complement of A, written $\bar{A}$ or A', is the set containing all elements not in

[1] I prefer $\subset$ over $\subsetneq$, which is used in [3].

the set A. For example, if $A \subseteq B$, then $A' = B - A$. If $\mathcal{Z}^+$ is defined as $\{0, 1, 2, \cdots\}$ with $A_{pe} = 2n$ and $A_{po} = 2n+1$ for $n \in \mathcal{Z}^+$, then $A_{po} + A_{pe} = \mathcal{Z}^+$, $\mathcal{Z}^+ - A_{po} = A_{pe}$ and $\mathcal{Z}^+ - A_{pe} = A_{po}$. Further, $A_{pe} = A'_{po}$ and $A_{po} = A'_{pe}$.

The *Cartesian product* or *cross product* of two sets A and B, written $A \times B$, is the set of all ordered pairs where the first element is a member of A and the second element is a member of B. The *power set* of A is the set of all subsets of A. If $A = \{0, 1\}$, then the power set of A, written $\mathcal{P}(\mathcal{A})$ or $\wp(A)$, is $\{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$.[2] Any set A is a subset of itself, $A \subseteq A$, and while the empty set is not a member of every set ($\emptyset \notin A$ for arbitrary A), it is a member of each power set ($\emptyset \in \wp(A)$). The cross product of a set A with itself is written A^2. The cross product of A with itself k times is written A^k.

With sets and set notation, we can readily visualize and understand the principles of set *inclusion* and *exclusion*. The *Pigeonhole Principle* is a well-known way of reasoning about certain such principles. The classic statement goes something like

> If more than k pigeons fly in to k pigeonholes, then at least one hole will wind up with more than one pigeon.

The more formal version leaves out the pigeons.

> If more than k items are placed into k bins, then at least one bin contains more than one item.

It is important to note that if k bins each contain at most one item, then there can be at most k items in total. A few examples will illustrate this concept.[3]

> How many people must be in a room to guarantee that two people will have last names that begin with the same initial letter?
>
> If each letter of the alphabet represents a *bin*, then there are 26 bins. By the Pigeonhole principle, there can be at most 26 people in the room who do not share a last name with the same initial letter. Thus, 27 is the minimum number of people who need to be in the room to *guarantee* that two people will have names beginning with the same initial letter.

While it is well known that the results of rolling a fair die is random, there are some non-random things that we can say about this random process.

> How many times must a single die be rolled in order to guarantee the same value twice?

Note that we do not say "twice in a row". Here is a slightly more difficult die problem.

> Suppose that you have two dice, each with 20 sides containing the numbers 1-20. What is the maximum number of times that you would have to roll the dice in order to roll either a result of 20-20 or duplicate a previous roll?

2 Sequences and Tuples

A *sequence* is a list of objects in some order. The order listed is characteristic of the sequence, so the sequence $(7, 15, 32)$ is not the same as the sequence $(15, 32, 7)$. A sequence is indicated by

[2] In general, for $A = \{a, b, c\}$, $\mathcal{P}(\mathcal{A}) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$.

[3] [2] p. 269.

enclosing its members in parenthesis. Repetition and order matter in a sequence, so $(3, 14, 116, 14)$ is unique from the sequences $(3, 14, 14, 116)$ and $(3, 14, 116)$.

A finite sequence is called a *tuple* with sequences of k elements being called a *k-tuple*. The sequence $(12, m, 47.2)$ is a 3-tuple, or *triplet*. The set of all ordered pairs whose elements are 0s and 1s is $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$.

3 Predicate Logic

A *proposition* is a statement whose range is {TRUE, FALSE}. A *predicate* is a statement that contains a *variable* and denotes a *property*. For any specific value of the variable, the statement is a proposition; has a truth value. If the predicate P(x) is $x \geq 0$, then $P(13)$ is a proposition with the value TRUE and P(-1) is FALSE.

The *domain of interpretation* for a predicate is the collection of objects over which the predicate will be interpreted. The *universal quantifier* is used to indicate "for all", "for any", etc. The statement $(\forall x)(x > 0)$ reads, "for all x greater than 0". The *existential quantifier* is used to indicate "there exists at least one". The statement $(\exists y)(y > 0)$ reads, "there exists at least one y greater than 0".

Students should review the derivation rules for predicate logic; both the equivalence and the inference rules.

4 Functions and Relations

A *function* takes an input and produces an output. A function is *deterministic* in that, for a given input, a specific function will always produce the same output. Thus, $f(a) = b$ mean that the function f operating on the specific value a will always produce the specific value b. Functions are also called *mappings* in that for $f(a) = b$, the function f maps the input a to the output b.

The set of possible inputs for a function is called its *domain* and the set of possible outputs for a function is called its *range*. We say that a function f has a domain D and a range R with the notation $f : D \to R$. The notation $f : D \mapsto R$ is also common. A function that maps to all elements of the range is said to be *onto* the range. $f(x) = x + 1$, where $x \in \mathcal{Z}$ and f maps $\mathcal{Z} \to \mathcal{Z}$, is onto. $f(x) = x^2$, where $x \in \mathcal{Z}$ and f maps $\mathcal{Z} \to \mathcal{Z}$, is not onto.[4]

A common question regarding functions, but one that is often misunderstood is: *How many functions*[5]? Our discussion is limited to functions without special properties and functions with the special property of being one-to-one. Suppose that we have two finite sets, S and T, where $m = |S|$ and $n = |T|$, and $f : S \to T$. The multiplication principle can be used in this case. Each function on S maps to an image (possibly distinct) in T. That is, there are m different ways to map to n images. Therefore, there are m functions, each of which map to at most n elements. This results in the simple formula for the number of function without special properties, namely, n^m.

The second case, one-to-one functions, is straightforward. The one-to-one property implies that $m \leq n$[6]. We now note that for each function, the elements of S must all be used and must only be used once per function with each having a unique image. Again using the multiplication principle,

[4] but if f maps $\mathcal{Z} \to \mathcal{Z}^+$ then f is onto.

[5] [2] p. 397.

[6] recall the Pigeonhole Principle

we have a product of m factors, which results in

$$\begin{aligned} n(n-1)(n-2)\cdots[n-(m-1)] &= n(n-1)(n-2)\cdots(n-m+1) \\ &= \frac{n!}{(n-m)!},\ 0! \text{ is defined to be } 1 \\ &= P(n,m) \end{aligned}$$

These two methods of counting functions are sufficient for this review.

Function composition can occur when the range of some function f is the domain for some other function g. That is, suppose that functions f and g are defined such that $f : S \to T$ and $g : T \to U$, then, for any $s \in S, f(s)$ is an element of T and thus the function g can be applied to $f(s)$. The result of $g(f(s))$ is a member of U. Thus, we have created a function $S \to U$, which we call the *composition* of f and g, denoted by $g \circ f$. Be careful to remember that $f \circ g$ is not the same as $g \circ f$, as we evaluate the composed functions right-to-left. For a function g such that $(g \circ f)(s) = s, \forall s \in S$, then g is called the *inverse* function for f and $g \circ f$ can be written as $f^{-1}(f(s)) = i_s$, the *identify function* on S.

A *binary relation* on a set S is a subset of $S \times S$; that is, a set of ordered pairs of elements of S. In general, we do not list the ordered pairs and instead define the relation through description. As the description gives a characterizing property of the elements of the relation, it is a binary predicate satisfied by certain ordered pairs. The binary relation ρ is defined by $x \,\rho\, y \leftrightarrow (x, y)$; the relation ρ is satisfied by the ordered pairs (x, y) and the ordered pairs (x, y) define the relation ρ. The definition of the binary relation for multiple sets should be obvious.

You should understand the concept of a closure of a relation and its usefulness. For example,[7] let $S = \{1, 2, 3\}$ and $\rho = \{(1, 1), (1, 2), (1, 3), (2, 3), (3, 1)\}$. In this case, ρ is not reflexive, not symmetric, and not transitive. The closure of ρ, ρ^*, will contain ρ; $\rho \in \rho^*$. The closure of ρ with respect to reflexivity is

$$\rho^* = \{(1, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 1), (3, 3)\} \text{ reflexive closure}$$

The closure of ρ with respect to symmetry is

$$\rho^* = \{(1, 1), (1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)\} \text{ symmetric closure}$$

For both reflexive and symmetric closure, we were able to add the required new ordered pairs through inspection. The transitive closure, however, may require several steps. Inspection shows that we need to add (3,2) because of (3,1) and (1,2); (3,3) because of (3,1) and (1,3); and (2,1) because of (2,3) and (3,1). This gives us

$$\{(1, 1), (1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2), (3, 3)\}$$

which, unfortunately, is still not transitive. We need to add (2,2) because of (2,1) and (1,2). This gives us the proper transitive closure

$$\rho^* = \{(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)\} \text{ transitive closure}$$

which is also the smallest transitive relation containing ρ. This method uses inspection to identify the missing parts of the transitive closure, which can be tedious. The concept of *reachability* helps in finding the transitive closure.

[7] [2] p. 334-335.

A *partial order* is a binary relation that is antisymmetric, transitive, and either reflexive or irreflexive. The common symbols used in discussing partial orders are $\prec$ (irreflexive) and $\preceq$ (reflexive). For the binary relation $x \prec y$, x is said to be a *predecessor* of y and y is said to be a *successor* of x. A partially ordered set, or *poset*, is the set S over which a relation R is a partial order. For a non-empty subset, U, of a poset, we have an $x \in U$, the *minimal element* of U, if x has no predecessor; and we have a $y \in U$, the *maximal element* of U, if y has no successor. The maximal and minimal elements do not have to be unique, and may not even exist.[8] A poset can be represented graphically by a *Hasse diagram*. If a poset has the added properties of having a greatest lower bound (*glb*) and a least upper bound (*lub*) for any two elements, it is called a *lattice*. The classic example for creating a totally ordered set from a partially ordered set is the topological sort.

5 Graphs and Trees

An *undirected graph*, or just *graph*, is a set of points with lines connecting some of the points. The points are called *nodes* or *vertices* and the lines are called *edges*. A *directed graph* is a graph in which edges have a specific direction associated. A *directed acyclic graph* or *DAG* is a directed graph in which there are no cycles. A graph is *connected* if for every two nodes there is a path between them. A *simple graph* is one with no loops or parallel arcs.

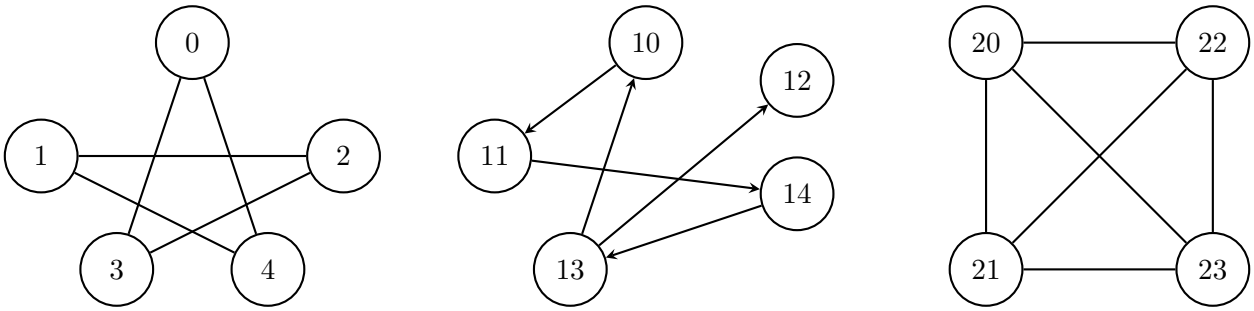

Figure 1: Graph examples

Note that the graph on the right in Figure 1 is complete[9] but not simple. By changing the arcs to be directed instead of undirected, the graph can be made simple and complete.

A graph is a *tree* if it is connected and has no cycles. A tree may have a *root* and all terminal nodes are called *leaves*.

In Figure 2, node A is the *root* node; nodes E, F, J, ,H, K, and L are *leaf* nodes; and nodes B, C, D, G, and I are *interior* nodes. When two nodes are connected by a line segment, the node closer to the root node is called the *parent* and the node further from the root node is called the *child*. Nodes that are at the same level of the tree are called *siblings*. A *subtree* is any portion of a tree T that it rooted at an interior node or leaf node. The subtree rooted at node D represents the tree comprised of nodes $D, H, I, K,$ and L. The *depth* of a node is the number of nodes along the path from root node to that node.

[8] Consider the minimal real number greater than 1 or the maximal real number less than 2.

[9] A complete graph is one in which any two nodes are adjacent. [2] p. 482.

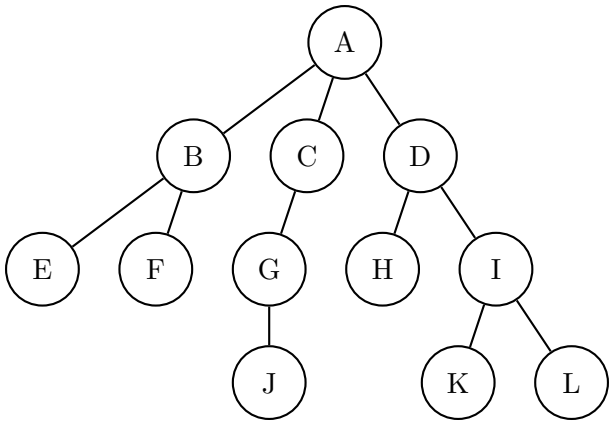

Figure 2: Tree example

6 Strings and Languages

An *alphabet* is a non-empty set of characters over which *strings* may be defined. The characters (*members*) of an alphabet are known as the *symbols* of the alphabet and the strings are often called *words*. An alphabet is typically indicated using the Greek letters Σ and Γ. See Figure 3.

$$\begin{aligned} \Sigma_1 &= \{0, 1\} \\ \Sigma_2 &= \{a, b, a, d, e, f, g, h, \ldots\} \\ \Gamma &= \{0, 1, x, y, z\} \end{aligned}$$

Figure 3: Alphabet examples

A *string* over an alphabet is a finite sequence of symbols from the alphabet, usually written without a comma separator. The set of all strings over the alphabet Σ is denoted Σ^*, called the Kleene closure.[10] The symbols 0101001 represents a string over Σ_1 and supercalifragilisticexpialidocious is a string over Σ_2. For a string w over Σ, $|w|$ represents the number of symbols in w, called the *length* of w. The *empty string* is a string of length zero and denoted ϵ. For any $w \in \Sigma^*$ of length n, $w = w_1 w_2 w_3 \cdots w_n$ where each $w_i \in \Sigma$. A string z is a *substring* of w if z appears consecutively within w. For example, califrag is a substring of supercalifragilisticexpialidocious.

A *language* is a set of strings from a particular alphabet, $\mathcal{L} \subseteq \Sigma^*$. Note that ϵ is a string in every language since $\epsilon w = w\epsilon = w$.

A string x is a *prefix* of a string y if there exists xz such that $xz = y$. If $x \neq y$ then x is a *proper prefix* for y. If $x = y$, then $z = \epsilon$. A language is *prefix free* if no member is a proper prefix for another member. Note that any string of a language $\mathcal{L}$ is a trivial prefix for itself.

7 Boolean Logic and Algebra

Suppose that a propositional wff P has n statement letters. The associated truth table for that wff associates a T or F value with an n-tuple of T-F values. The entire truth table defines a function f where $f : \{\mathsf{T}, \mathsf{F}\}^n \to \{\mathsf{T}, \mathsf{F}\}$. A tautology under f maps $\{\mathsf{T}, \mathsf{F}\}^n \to \{\mathsf{T}\}$ and a contradiction maps $\{\mathsf{T}, \mathsf{F}\}^n \to \{\mathsf{F}\}$.

[10] or Kleene star or Kleene operator, or sometimes just closure when it is clear we are discussing languages.

A *Boolean algebra*, denoted $[B, +, \cdot, ', 0, 1]$[11] is a set B on which are defined two binary operations $+$ and $\cdot$; one unary operation $'$; and in which there are two distinct elements 0 and 1 such that these five properties hold for all $x, y, z \in B$:

1a.	$x + y = y + x$	1b.	$x \cdot y = y \cdot x$	commutative properties
2a.	$(x + y) + z = x + (y + z)$	2b.	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	associative properties
3a.	$x + (y \cdot z) = (x + y) \cdot (x + z)$	3b.	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	distributive properties
4a.	$x + 0 = x$	4b.	$x \cdot 1 = x$	identity properties
5a.	$x + x' = 1$	5b.	$x \cdot x' = 0$	complement properties

Figure 4: Boolean Algebra Properties

For $S = \{a, b, c\}$, $\mathcal{P}(S)$ has 8, or $2^{|S|}$, elements: $\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}$. The elements of the power set together with the operations of union, intersection, and complementation gives us an 8-element Boolean algebra. The empty set is the 0 element and $\{a, b, c\}$ is the 1 element. Note that $x \cup x' = \{a, b, c\}$ for any $x \in S$.[12]

8 Theorems and Proofs

Much of the time in this course we will relax the usual rigorous proof in favor of a less formal *direct proof by argument*. The argument must be such that we can derive the rigorous version directly, but we will not get bogged down with dense mathematics and terminology.

For example, we can prove that the product of two even integers is always an even integer using the argument technique. Let's say that we have two even integers x and y. Our goal is to show that for $z = x \times y$, that z will be even for any two x and y that are even.

$x = 2m$	by definition
$y = 2n$	by definition
$x \times y = 2m \times 2n$	
$2m \times 2n = 4mn$	
$4mn = 2(2mn)$	
$= 2k$	set $k = 2mn$ $\square$

Figure 5: Product of two even integers is even

To prove something true for all $n \geq$ some value we will often use *induction*.[13]

1.	$P(1)$ is true	basis step
2.	$\forall k[P(k)$ true $\rightarrow P(k + 1)$ true$]$	inductive hypothesis
3.	$\therefore P(n)$ true for all positive integers n	result

Figure 6: First Principle of Mathematical Induction

[11] An alternate notation is $[B, +, \cdot, ', I_+, I.]$ where I indicates the *identity element*, but $I.$ can be confusing and even misleading.

[12] [2] p. 622 (corrected).

[13] [2] p. 113-114.

Proof by Induction Prove that the equation

$$1 + 3 + 5 + \cdots + (2n - 1) = n^2 \tag{1}$$

is true for any positive integer n.

The basis step is to establish $P(1)$

$$P(1) : 1 = 1^2$$

which is most certainly true. For the inductive hypothesis, we assume $P(k)$ for an arbitrary integer k, which is equation (2) when n has the value k.

$$P(k) : 1 + 3 + 5 + \cdots + (2k - 1) = k^2 \tag{2}$$

We want to show that equation (2) holds for $P(k + 1)$.

$$P(k + 1) : 1 + 3 + 5 + \cdots + [2(k + 1) - 1] =? (k + 1)^2 \tag{3}$$

We rewrite equation (3) to show the next-to-last term.

$$1 + 3 + 5 + \cdots + (2k - 1) + [2(k + 1) - 1] \tag{4}$$

Since equation (4) contains the left side of equation (2), we can substitute.

$$\begin{aligned} &= k^2 + [2(k + 1) - 1] \\ &= k^2 + (2k + 2 - 1) \\ &= k^2 + 2k + 1 \\ &= (k + 1)^2, \text{ see (3)} \qquad \square \end{aligned}$$

The second principle of induction is similar.

Fundamental Theorem of Arithmetic An integer $p > 1$ is called a *prime number*, or simply *prime*, if its only positive divisors are 1 and p. An integer greater than 1 that is not prime is termed *composite*.[14] This representation is unique, apart from the order that the factors occur.[15]

$\forall x \in N > 1$, the factors are $p_1^{a_0} p_2^{a_1} \ldots p_n^{a_n}$, where p_k is a prime number $< x$ and $a_i \in Z^+$.

Proof by Contradiction There are an infinite number of primes.

Let $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, ... be the prime numbers in ascending order, and suppose that there is a last prime, called p_n. Now consider the positive integer

$$P = p_1 p_2 p_3 \ldots p_n + 1$$

Because $P > 1$, we know by the Fundamental Theorem of Arithmetic that P is divisible by some prime p. But $p_1, p_2, \ldots, p_n$ are the only prime numbers, so p must equal one of $p_1, p_2, \ldots, p_n$. Since $p|p_1 p_2 \ldots p_n$ and $p|P$, we have $p|(P - p_1 p_2 \ldots p_n)$, or equivalently, $p|1$. Since the only divisor of 1 is 1 itself and because $p > 1$, we arrive at a contradiction. Therefore, the initial assumption that there is a last prime, p_n, is false, thus proving the infinity of the prime numbers.[16]

[14] [1] p. 40.
[15] ibid. p. 42.
[16] ibid. p. 47.

References

[1] David M. Burton: *Elementary Number Theory, Fifth Edition*, ISBN 0-07-232569-0, 2001.

[2] Judith L. Gersting: *Mathematical Structures for Computer Science and Its Applications, Seventh Edition*, ISBN: 987-1-4292-1510-0, 2014.

[3] Michael Sipser: *Introduction to the Theory of Computation, Third Edition*, ISBN: 978-1-133-18779-0, 2013.