Notes 5

1 Induction and Recursive Algorithms

We've spent a lot of time studying induction. This is because induction comes up all the time in analyzing computation. Why? Because induction is a "one step at a time" proof method. Computations also evolve "one step at a time."

In fact, many of our inductive proofs can be seen as *algorithms* for solving a problem.

- We proved any 2^n by 2^n board can be tiled with L shapes. If you look at the inductive proof, we actually gave a *procedure* for carrying out that tiling. Break the board into four, tile each piece, and merge the tilings. What kind of algorithm is this? *recursive:* it calls itself on smaller subproblems.
- The induction for existence of a fair ordering in a tournament gives a recursive algorithm for finding one.
- The definition of the Fibonacci numbers is also expressing a recursive algorithm for computing them.
- And the inductive proof for Nim gave a game-winning algorithm.

Just about any inductive proof that a problem has a solution is in fact providing a recursive algorithm for solving the problem. Conversely, when you have a recursive algorithm that you think solves some problem, induction is the usual way to prove that it works. In fact, an inductive proof can be thought of as a *recursive algorithm* for writing down a proof:

Procedure Write-Proof-For(n)

if n is 0
    Write-the-base-case
else
    Write-Proof-For($n-1$)
    Write-Inductive-Step(n)

Assuming we have rule for writing down the base case $P(0)$, and writing down the inductive step $P(n-1) \implies P(n)$ for any n, we can write down a proof of $P(k)$ recursively: we recursively prove $P(k-1)$, then we use the inductive step proof to prove $P(k-1) \implies P(k)$.

1.1 RSA

RSA is at the core of many of the secure communication protocols we use today (SSH, SSL, etc). RSA works by treating any given message (sequence of bits) as a big number x and applying a mathematical function that "hides" the number x unless you know a secret key.

More precisely, it computes the remainder r of x^a on division by b, where a and b are two specially constructed *large* numbers. It turns out that if someone is given r and knows the right secret about the origin of a and b, they can "invert" the exponentiation and get x.

So, how can we compute the exponentiation? The most obvious way is to work from the *recursive definition*:

- $x^0 = 1$
- $x^n = x \cdot x^{n-1}$

This is a recursive definition of x^n, but also gives a recursive algorithm for computing it:

Procedure $Exp(x, n)$

if $n = 0$
    return 1
else
    return $x * Exp(x, n-1)$

Now it's pretty "obvious", but how would you *prove* that algorithm *Exp* returns the right answer? Well, it doesn't always: what happens if you call $Exp(5, -2)$? $Exp(8, 2.5)$? (Trying to prove correctness can reveal implicit assumptions. We are assuming n is a natural number).

Theorem 1.1 For all $x, n \in \mathbb{N}$, a call to $Exp(x, n)$ returns x^n

Note: the procedure *Exp* above can be thought of as (i) a computational process or (ii) a recursive function definition (defined as the output of the computational process). We need to prove that it defines the *same* function as x^n. We are actually claiming two distinct things in the theorem:

- The algorithm returns something (i.e., doesn't run forever)
- The algorithm returns the *right* something.

Proof. By induction on n. The base case, that $Exp(x, 0)$ returns $x^0 = 1$, follows by inspection: on that call, the "if" condition is satisfied, so we return 1. For the inductive step, assume $Exp(x, n)$ returns x^n and prove for $n+1$. Consider the call to $Exp(x, n+1)$. $n+1 > 0$, so the "else" clause is invoked. We call $Exp(x, n)$ which, by assumption, returns x^n. We multiply that quantity by x, yielding x^{n+1}, and return it. This proves that $Exp(x, n+1)$ returns the right value. ∎

OK, we've shown the procedure returns the right answer. Is it a good implementation? How many multiplications are done to compute x^n? By induction, there are n multiplications. Remember those "128 bits keys" we aren't allowed to export because cryptography is considered a weapon? That means the number n has about 128 bits. So we will be doing about 2^{128} multiplications. We'll still be working on it when 6.042 finishes...

1.2 Fast Exponentiation

Let's cut down on the number of multiplies. Suppose we want to compute x^a and a is even, say $2c$. So we want to compute $x^a = x^{2c}$. Is there any obvious way to beat a multiplies? Note $x^{2c} = (x^c)^2$ So we can compute x^c (c multiplies) and then square it (1 multiply). This totals $c + 1 = 1 + a/2$ multiplies. What if a is odd?

We can generalize this idea.

Procedure *FastExp*(x, n)

if $n = 0$
    return 1
else
    if n is even
        let $y =$ *FastExp*$(x, n/2)$
        return y^2
    else
        let $y =$ *FastExp*$(x, (n-1)/2)$
        return $x * y^2$

Theorem 1.2 For all $x, n \in \mathbb{N}$, a call to *FastExp*(x, n) returns x^n

Proof. Induction again. The base case ($n = 0$) follows by inspection as before. So let's prove the inductive step. What kinds of subproblems are we going to use to prove it works for n? **Not** just $n - 1$, so let's use strong induction. Suppose the theorem is true for every $k < n$, let's prove it for n.

Suppose we call *FastExp*(x, n). We can assume $n \neq 0$ (we already did a proof for 0) so the **else** clause is selected in the algorithm. Now there are two cases. If n is even (say $n = 2m$) then we call *FastExp*(x, m). By strong induction this returns x^m, which we assign to y. Then we return $y = (x^m)^2 = x^n$ as required, so the proof works for this case. If n is odd (say $n = 2m + 1$) then we call *FastExp*(x, m) and set y to the result. Then we return $x * y^2 = x * (x^m)^2 = x^{2m+1} = x^n$. So the proof works in this case too. These are all the cases. QED. ∎

How many multiplications does **this** algorithm do?

Theorem 1.3 If n has b bits, then *FastExp*(x, n) performs at most $2b$ multiplications.

Proof. Induction on b. Base case: If $b = 0$, then $n = 0$ and we return without doing any multiplications. So the base case is true. Inductive step: assuming it true for b bits, consider a $b + 1$-bit number n. We execute a recursive call to *FastExp* on either $n/2$ or $(n - 1)/2$, depending whether n is odd or even. Both of these numbers have at most b bits. So by induction, the recursion does at most $2b$ multiplications. Then we do either one more (even case) or two more (odd case). So at most two more. Overall, at most $2b + 2 = 2(b + 1)$ multiplications. QED. ∎

Using this algorithm, a 128-bit key means only 256 multiplications: quite an improvement!

1.3 Greatest Common Divisor

Given 2 positive natural numbers, $a, b \in N^+$, their *greatest common divisor* is the largest number that divides both without remainder.

(Recall notation: $a \mid b$ means a divides b.)

Finding the GCD is a problem that must be solved to generate the "secret" used to decrypt RSA messages.

One of the first recorded algorithms—Euclid's algorithm—finds greatest common divisors. It uses a key lemma for "simplifying" the problem:

Lemma 1.4 Given $x > y$, let $r = x - y$. Then any common divisor of x and y is also a common divisor of y and r, and vice versa.

Corollary 1.5 GCD(x, y) =GCD(y, r)

Proof. (of the corollary). (x, y) and (y, r) have exactly the same set of divisors (by the lemma). Thus, the biggest number in each set, namely the greatest common divisor, is the same. ∎

Proof. (of the lemma). Suppose $d \mid x$ and $d \mid y$. So $x = zd$ and $y = wd$. Thus $r = x - y = d(z - w)$ so $d \mid r$, meaning d is a common divisor of y (by assumption) and r (by deduction). The other direction of proof is similar. ∎

From this lemma we get Euclid's algorithm:

Procedure *Euclid*(x, y)

if $x = 0$ return y
else if $y = 0$ return x
else if $x > y$ return *Euclid*$(y, x - y)$
else return *Euclid*$(x, y - x)$)

Example:
Find $GCD(112, 84) = ?$
Step 1: $112 - 84 = 28$, reduce to $(28, 84)$.
Step 2: $84 - 28 = 56$, reduce to $(28, 56)$.
Step 3: $56 - 28 = 28$, reduce to $(28, 28)$.
Step 4: $28 - 28 = 0$, reduce to $28, 0$
Now we are done; the answer is 28.

Let's prove this algorithm does the right thing. A first faint worry: maybe we've got an infinite recursion? This indicates the first thing to prove for any algorithm: it *terminates*.

Lemma 1.6 For any natural number inputs x and y, *Euclid*(x, y) returns a value.

Why do we think this is true intuitively? Because the arguments "shrink" on each recursive call and eventually reach 0. Induction lets us formalize this.

Proof. Induction on $x+y$. More formally, we prove by strong induction on n that for all inputs x and y such that $x+y=n$, the algorithm terminates. Base case: if $x+y=0$, then both x and y are 0 so we return immediately. Inductive step: Assume the claim for $x+y<n$ and prove it for $x+y=n$. Now consider two cases. Either $x \geq y$ or $y \geq x$. Suppose first that $x \geq y$. So consider two subcases. If $y=0$, then we return immediately and the inductive step is proved. If $y>0$, then we make a recursive call with arguments y and $x-y<x-0=x$. So the sum of the two new arguments is less than $x+y=n$. So the inductive hypothesis holds: the recursive call returns a value. So our original call returns (the same) value.

What about the second case ($y \geq x$)? Well, I would have to write the same proof all over again, swapping x and y everywhere. To avoid such tedium, we say "without loss of generality, assume $x \geq y$. This is code for "the other case works exactly the same way" and completes the proof. ∎

Lemma 1.7 For any natural number inputs x and y, *Euclid*(x, y) returns GCD(x, y)

Proof. Induction on $x+y$ as before. The base case is trivial (assuming we define GCD$(0,0)$ to be 0, which seems fair).

For the inductive step where $x+y=n>0$, assume without loss of generality that $x \geq y$. Then we return the result of a recursive call on *Euclid*$(y, x-y)$. This means (by strong induction) that we return GCD$(y, x-y)$. But we proved a lemma above that GCD(x, y) =GCD$(y, x-y)$ so we are returning the right value. ∎

Wait a minute, what's with the GCD$(0,0)$ definition? Well, technically every number is a common divisor of 0 and 0, so it should be infinite?

Why doesn't this matter? Because we never actually make a recursive call to $(0,0)$—once one number is 0, we terminate.

So even though we still need a base case for the induction pattern, we never actually apply that base case. Instead our base case is implicit in the "subcase" of the inductive step where one argument is 0.

2 Sets

This section was not covered in lecture, but summarized material from your background reading

2.1 Definitions

A *set* is a collection of objects (Collection, object are undefined terms).

The order of elements is not important. I.e., the sets $\{1,2\}$ and $\{2,1\}$ are considered the same set. Each element is included only once. So, if we write $\{a, b\}$ or $\{a, a, b\}$ they both denote the same set and the element a is considered to be present in the set only once. (repeating an element in the description of a set may be useful sometimes).

Here are some examples of sets:

- $\{1, 2, 3\}$.

  This is a simple finite set, obtained by enumerating all the elements.

- $\emptyset$, the *empty set*.

  A special case of a finite set, with *no* elements.

- $\{\emptyset\}$.

  This funny set is different from $\emptyset$. It has one element, which is a set.

- $\{1, \{1, 2\}\}$.

  This one contains two elements, one of which is a set itself. The set happens to contain the other element 1, but that does not contradict the condition that an element is included only once.

- $\{n \in \mathbb{N} \mid n \text{ is even }\}$.

  This is an infinite set, specified by giving a universal set (domain of discourse) and a property. In general, given a domain of discourse D and a predicate P over D, $\{x \in D \mid P(x)\}$ denotes the set of all x in D that satisfy predicate P.

We write that $a \in A$ ("a is in A" or "a is an element of A") if a is an object in the set A. The empty set satisfies the proposition $(\forall x) x \notin \emptyset$.

Sets can be defined by axioms and deduction rules, giving a very formal *axiomatic set theory*. But usually, people reason about sets less formally, using *naive set theory*. Either way, set theory can be used as the basis for defining almost everything else in mathematics and is therefore regarded as the most fundamental part of math.

In fact natural numbers can be defined as certain kinds of sets (!). For example we can represent the number 0 by the empty set $\emptyset$, number 1 by the set $\{\emptyset\}$, number 2 by $\{\emptyset, \{\emptyset\}\}$, and so on. In general, if S_n is the set representing the number n, the successor of n can be represented by the set $S_{n+1} = \{\emptyset\} \cup \{\{x\} \mid x \in S_n\}$. Then the properties natural numbers can be *proved as theorems* of set theory, rather than being assumed.

The axioms of set theory seem ridiculously obvious but aren't. Here are some examples:

- The *axiom of extensionality* says that two sets are equal if and only if they have the same elements.

  For example, the two sets $\{n \in \mathbb{N} \mid 2 \text{ divides } n\}$ and $\{2n \mid n \in \mathbb{N}\}$ are defined in different ways. But they are "equal" as sets because they have exactly the same elements.

- The *axiom of comprehension* says that we can form a new set by including all objects of a given set that share a common property.

  For example, $\{x \in \mathbb{R} \mid P(x)\}$ where $P(x) =$ "x is rational" denotes the set of all rational numbers. The axioms says that this is a set.

The set building notation that we used above has given the universe of discourse explicitly. Sometimes, when this seems obvious, we will leave this out. Just like what we did for predicate quantifiers. E.g., we might write: $\{n \mid n \text{ is even }\}$ if the universe $\mathbb{N}$ seems clear. (This Could be confusing – the universe could alternatively be the integers, positive and negative.)

Definition 2.1 We say that $A \subseteq B$ ("A is a subset of B" or "A is contained in B") if every element of A is also an element of B (that is, $(\forall x)(x \in A \implies x \in B)$).

Definition 2.2 We say that A is a *proper* subset of B ($A \subset B$) if $A \subseteq B$ and $A \neq B$.

Note that $(\forall X)\emptyset \subseteq X$, i.e. the empty set is contained in any other set.
Q: Can we have both $A \in B$ and $A \subseteq B$?

A: Yes, consider $\{1, \{1\}\}$.

Theorem 2.3 $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$.

The proof is by the axiom of extensionality. But we won't bother giving it.

This observation is very important as it provides a simple way to prove that 2 sets are equal.

Example 2.4 Prove that the two sets $\{n \in \mathbb{N} \mid 2 \text{ divides } n\}$ and $\{2n \mid n \in \mathbb{N}\}$ are equal.

We Must show containment in both directions.

$\subseteq$: We assume $x \in \mathbb{N}$ and 2 divides n, and show that there is some $y \in \mathbb{N}$ such that $x = 2y$. This is true by definition of divisibility.

$\supseteq$: Assume $x = 2y$. Then x is divisible by 2, again by the definition of divisibility.

Here's a slightly more interesting one:

Example 2.5 The set of evens ≥ 2, i.e., $\{2n \mid n \in \mathbb{N}, n \geq 1\}$,
is equal to $\{x \in \mathbb{N} \mid \exists y, z \text{ odd }, x = y + z\}$.

$\subseteq$: If $x = 2n$, then write $x = 1 + (x - 1)$, sum of two odds.

$\supseteq$: If $x = y+z$, where both y and z are odd, then write $y = 2a+1$, $z = 2b+1$. Then $x = 2(a+b+1)$, which is even.

The size of a set A (the number of elements it contains) is written $|A|$. For example, the set $\{a, b, c\}$ has size $|\{a, b, c\}| = 3$. The empty set has size $|\emptyset| = 0$. The size of a finite set is a natural number. If A is an infinite set, we write $|A| = \infty$.

Notice that for any two finite sets A, B, we have $|A \cup B| \leq |A| + |B|$. Equality holds if and only if $A \cap B = \emptyset$. If we allow A and B to be infinite sets, we still have $|A \cup B| \leq |A| + |B|$ if we define $x + \infty = \infty + x = \infty$ and $x \leq \infty$ for all $x \in \mathbb{N}$ and $x = \infty$.

2.2 Russell's paradox

We might think that set theory is obvious, but this is not the case. Consider the set $S = \{x \mid x \text{ is a set}\}$. This seems like a perfectly reasonable set (well, maybe a bit strange—note $S \in S$). If our set theory allows for such an entity, we get into trouble as follows.

Consider the set

$$Q = \{x \in S \mid x \text{ is a set and } x \notin x\}.$$

If S is a set, then so is Q by standard axioms like comprehension. Note $Q \subseteq S$.

Now note that some sets are in Q and some aren't. For example, the set $A = \{1, 2, 3\}$ is not an element of itself, so $A \in Q$. On the other hand, The set $P = \{$ all infinite sets$\}$ is a set (by comprehension from S). Therefore, $P \in P$ as the set P itself is infinite. So $P \notin Q$.

The troublesome question is: is $Q \in Q$?

1. If $Q \in Q$ then $Q \notin Q$, a contradiction.
2. If $Q \notin Q$ then $Q \in Q$ also a contradiction.

This is clearly a paradox. In fact, this paradox, at the time it was discovered by Russell (~ 1900), was a blow to the then current attempts to formalize all of mathematics axiomatically.

Mathematicians had to work very hard to find a consistent set of axioms (note that a consistent set of axioms does not allow this paradox to arise).

The problem with naive set theory is that you can form a set by describing its members, as we did for Q above. Axiomatic set theory has a specific set of rules for how you can form sets—rules that don't allow the set Q above to be specified, and therefore avoid the paradox.

Fortunately, we'll be working naively, so this is a matter of only theoretical interest.

2.3 Basic set operations

There are several standard operations on sets. In axiomatic set theory, the claims that these operations yield new sets are axioms.

Union

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}.$$

Intersection

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$$

Difference

$$A - B = \{x \mid x \in A \text{ and } x \notin B\}$$

This is also written as $A \setminus B$.

Complement

$$A^c = \{x \mid x \notin A\}.$$

This is also written as $\bar{A}$.

Symmetric difference

$$A \oplus B = A \cup B - A \cap B.$$

Sets and operations on sets can be represented using Venn diagrams (see Rosen, sections 1.4, 1.5).

Union and intersection can be extended to apply to more than two sets. Given a finite collection of sets A_i, $i = 1, \ldots, n$,

$$\bigcup_{i=1}^{n} A_i$$

is the set of elements in at least one of the A_i, and

$$\bigcap_{i=1}^{n} A_i$$

is the set of elements in all of the A_i. Both of these make sense if $i = 1$ – they reduce to just A_1. In fact, for $i = 0$, the appropriate thing is to say the union of no sets is $\emptyset$ and intersection of no sets is universal set. This definition of union and intersection of no sets might seem arbitrary, but in fact it can be motivated in several ways. For example, if we want to define the union of a finite family of sets $A_1, A_2, \ldots, A_n$, we can define $\bigcup_{i=1}^{1} A_1 = A_1$ and $\bigcup_{i=1}^{n+1} A_i = (\bigcup_{i=1}^{n} A_i) \cup A_{n+1}$. If we want this recursive definition to work also for base case $n = 0$ we must set $\bigcup \emptyset = \bigcup_{i=1}^{0} A_i = \emptyset$. The analogue holds for intersection. The recursive definition of $\bigcap_{i=1}^{n+1} A_i = (\bigcap_{i=1}^{n} A_i) \cap A_{n+1}$ must have base case $\bigcap \emptyset = \bigcap_{i=1}^{0} A_i = U$ if we want $\bigcap \{A_1\} = (\bigcap \emptyset) \cap A_1 = A_1$.

Instead of using an explicit index, we can just consider a set of sets. If $\mathcal{S}$ is a *set of sets*, then

$$\bigcup \mathcal{S}$$

denotes the union of all the sets in $\mathcal{S}$, in any order. (One can prove that the order doesn't matter.) Likewise,

$$\bigcap \mathcal{S}$$

denotes the intersection of all the sets in $\mathcal{S}$.

Complement is different from all the other operations, because it assumes some universe U from which we extract the things not in A. All sets in question are supposed to be subsets of U. Given a universal set U, the complement operation can be expressed as $A^c = U - A$. The universe U is often implicit, where there seems to be little chance of confusion.

There is a close correspondence between set operations and logical operations: consider sets A, B and C defined by $A = \{x \mid P(x)\}$, $B = \{x \mid Q(x)\}$, $C = \{x \mid R(x)\}$, where P, Q, R are predicates.

set union vs. predicate or: $A \cup B = C$ if and only if $P(x) \vee Q(x) \equiv R(x)$

set intersection vs. predicate and: $A \cap B = C$ if and only if $P(x) \wedge Q(x) \equiv R(x)$

set difference vs. predicate "non implication": $A - B = C$ if and only if $\neg(P(x) \rightarrow Q(x)) \equiv R(x)$

set complement vs. predicate not: $A^c = B$ if and only if $\neg P(x) \equiv Q(x)$

set symmetric difference vs. predicate exclusive or: $A \oplus B = C$ if and only if $P(x) \oplus Q(x) \equiv R(x)$.

Predicates and sets often form alternative ways of looking at the same notions. Given predicates p and q, define

$$P = \{x \mid p(x)\}$$
$$Q = \{x \mid q(x)\}$$

Then $P \cup Q = \{x \mid p(x) \vee q(x)\}$, and so on.

2.4 Power set (set of all subsets)

Another very important operation on sets is that of the *power set* of a given set A. It is denoted by $\mathcal{P}(A)$ and consists of the set of all subsets of A. In other words, $\mathcal{P}(A) = \{x \mid x \subseteq A\}$. Note that for any set A, $\emptyset \in \mathcal{P}(A)$ and $A \in \mathcal{P}(A)$.

Example 2.6 If $A = \{1, 2\}$ then $\mathcal{P}(A) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$.

Theorem 2.7 If A has n elements, then $P(A)$ has 2^n elements.

Proof. Ordinary Induction on n. Base case: $n = 0$. Then $P(A)$ has one element, $\emptyset$. Inductive step: We assume the theorem for n and prove it for $n + 1$. Pick one element, a. The subsets can be divided into two groups – those containing a and those not containing a. Moreover, the set of subsets of A not containing a is $\mathcal{P}(A - \{a\})$, and the set of subsets of A containing a is the set $\{\{a\} \cup B \mid B \in \mathcal{P}(A - \{a\})$ which has the same number of elements as $\mathcal{P}(A - \{a\})$. By our induction hypothesis $\mathcal{P}(A - \{a\})$ has 2^{n-1} elements, so the total is $2^{n-1} + 2^{n-1} = 2^n$. ∎

I did a base case of 0 here. Does the inductive step from 0 to 1 work correctly? Yes, it corresponds to the sets $\{\emptyset\}$ and $\{a\}$.

Because of this cardinality relationship, $\mathcal{P}(A)$ is sometimes denoted by 2^A.

2.5 Set identities

There are a large number of set identities involving relationships among sets built using various operations. See p. 50 of Rosen. Here we just list some sample properties, and show how they can be proved.

Theorem 2.8 (Distributivity: Union distributes over intersection) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

Proof. We show the two sets are equal by proving that each is contained in the other.

$A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C)$. Suppose $x \in A \cup (B \cap C)$. Then by definition, $x \in A$ or $x \in (B \cap C)$. We argue by cases.

If $x \in A$, then by definition $x \in A \cup B$ and similarly $x \in A \cup C$. It follows by definition of intersection that x is in the intersection of these two sets, which is the RHS.

On the other hand, if $x \in (B \cap C)$, then $x \in B$ and $x \in C$. Then by definition of union, $x \in A \cup B$ and $x \in A \cup C$. It again follows that x is in the intersection.

$A \cup (B \cap C) \supseteq (A \cup B) \cap (A \cup C)$. Suppose $x \in (A \cup B) \cap (A \cup C)$. It follows that $x \in A \cup B$ and $x \in A \cup C$. We argue by cases. If $x \in A$, then certainly $x \in A \cup (B \cap C)$. If $x \notin A$, then since $x \in A \cup B$, we know $x \in B$. Similarly, $x \in C$. Therefore, $x \in B \cap C$. It follows that $x \in A \cup (B \cap C)$.

■

Theorem 2.9 De Morgan's rule: $(A \cup B)^c = A^c \cap B^c$.

Proof. We have to show that $(A \cup B)^c \subseteq A^c \cap B^c$ and that $A^c \cap B^c \subseteq (A \cup B)^c$.

$(A \cup B)^c \subseteq A^c \cap B^c$. Let $x \in (A \cup B)^c$. Then by definition of complement, $x \notin A \cup B$. It follows (proof by contradiction) that $x \notin A$ and $x \notin B$. Therefore, again by definition of complement, $x \in A^c$ and $x \in B^c$. Then by definition of intersection, $x \in A^c \cap B^c$.

$A^c \cap B^c \subseteq (A \cup B)^c$. Let $x \in A^c \cap B^c$. Then x is in A^c and in B^c. Therefore, x is in neither A nor B and therefore not in $A \cup B$. We conclude that $x \in (A \cup B)^c$.

■

The above two theorems have *dual theorems* produced by swapping $\cap$ and $\cup$. This is a general phenomenon – in Rosen, the theorems are arranged in pairs. Duality can be applied to any theorem that is true for *all* sets. Each theorem in a pair can be obtained from the other by swapping the operations $\cap$ and $\cup$, and the sets $\emptyset$ and U. Notice that the complement operations are not changed.

Theorem 2.10 $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

Theorem 2.11 $(A \cap B)^c = A^c \cup B^c$.

2.6 Cartesian products

Another important set operation is the *Cartesian product.* For sets A and B define $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$. Here, (a, b) is considered an *ordered* pair so that $(a, b) \neq (b, a)$.

Example 2.12 If $A = \{0, 1\}$ and $B = \{x, y\}$ then $A \times B = \{(0, x), (0, y), (1, x), (1, y)\}$.

Order is important! In the above example, $(a, 0) \notin A \times B$, though $(a, 0) \in B \times A$.

We can generalize Cartesian products to apply to more than 2 sets and define $A_1 \times \cdots \times A_n = \prod_{i=1}^{n} A_i = \{(a_1, \cdots, a_n) \mid a_i \in A_i \text{ for } 1 \leq i \leq n\}$. Also, we write A^n for $\prod_{i=1}^{n} A$.

Examples:

- $\{0, 1\}^n$ is the set of all n-tuples of 0's and 1's. Or, length n strings of 0's and 1's.
- $\Re^2 = \Re \times \Re$ is the real plane.

3 Relations

A "relation" is a fundamental mathematical notion expressing a relationship between elements of sets. It's an abstract notion useful in practice for modeling many different sorts of relationships. It's the basis of the *relational data base* model, the standard data model for practical data processing systems.

3.1 Definitions

Definition 3.1 A *binary relation from* a set A *to* a set B is a subset $R \subseteq A \times B$.

Definition 3.2 A binary relation *on* a set A is a subset $R \subseteq A^2$ (I.e., a relation from A to A)

We often write $a \sim_R b$ or aRb instead of $(a, b) \in R$.

We can also define a *ternary* relation on A as a subset $R \subseteq A^3$ or, in general, an n-ary relation as a subset $R \subseteq A^n$, or $R \subseteq A_1 \times A_2 \times \cdots \times A_n$ if the sets A_i are different.

Relations are used to model many different things:

1. The relation "is taking class" as a subset of $\{$students at MIT$\} \times \{$classes at MIT$\}$. A relation from students to classes.
2. The relation "is living in the same room" as a subset of $\{$students at MIT$\} \times \{$students at MIT$\}$. A relation on students.
3. The relation "can drive from first to second city". (Not necessarily directly – just some way, on some roads.)
4. A relation on computers, "are connected (directly) by a wire"
5. "meet one another on a given day"
6. "likes"
7. Let $A = \mathbb{N}$ and define $a \sim_R b$ iff $a \leq b$.
8. Let $A = \mathcal{P}(\mathbb{N})$ and define $a \sim_R b$ iff $|a \cap b|$ is finite.
9. Let $A = \Re^2$ and define $a \sim_R b$ iff $d(a, b) = 1$.
10. Let $A = \mathcal{P}(\{1, \cdots, n\})$ and let $a \sim_R b$ if $a \subseteq b$.
11. In a tournament, the "who beats who" relation.
12. "Implies" is a relation on the set of boolean formulae.

3.2 Properties of Relations

There are several standard properties that are satisfied by (some) relations. These occur commonly enough that they are worth identifying. Sometimes interesting things can be deduced just from the abstract properties.

Definition 3.3 A binary relation R on A is:

1. *reflexive* if for every $a \in A$, $a \sim_R a$.
2. *irreflexive* if for every $a \in A$, $a \not\sim_R a$.
3. *symmetric* if for every $a, b \in A$, $a \sim_R b$ implies $b \sim_R a$.
4. *antisymmetric* if for every $a, b \in A$, $a \sim_R b$ and $b \sim_R a$ implies $a = b$.
5. *transitive* if for every $a, b, c \in A$, $a \sim_R b$ and $b \sim_R c$ implies $a \sim_R c$.
6. *asymmetric* if for every $a, b \in A$, $a \sim_R b$ implies $\neg(b \sim_R a)$.

There's no sense to the specific negative prefixes; they just need to be remembered. The difference between antisymmetric and asymmetric relations, is that antisymmetric relations may contain pairs (a, a), i.e., elements can be in relations with themselves, while in an asymmetric relation this is not allowed. Clearly, any asymmetric relation is also antisymmetric, but not vice versa.

Among our examples:

- Relation 2 is reflexive, symmetric, transitive.
- Relation 3 is reflexive, transitive. Not necessarily symmetric, since roads could be one-way (Cambridge story), but in actuality.... But definitely not antisymmetric.
- Relation 4 is symmetric but not transitive. Whether it is reflexive is open to interpretation.
- Relation 5 likewise.
- Relation 6 is (unfortunately) not symmetric. Not antisymmetric. Not transitive. Not even reflexive!
- Relation 7 is reflexive, antisymmetric, transitive.
- Relation 8 is not reflexive. It is symmetric. Not transitive. Evens$\cap$Odds is finite (empty), but not Evens$\cap$Evens.
- Relation 9 is only symmetric.
- Relation 10 is reflexive, antisymmetric and transitive.

3.3 Representation

There are different ways of representing relations, e.g., for handling in a computer program. We can describe by properties, as we did above. That's about all we can do, for infinite sets. But for finite sets, we usually use some method that explicitly enumerates all the elements of the relation. We'll discuss three alternatives: lists, matrices, and graphs.

3.3.1 Lists

To represent a finite relation from set A to set B, we can just list all the pairs.

Example 3.4 Consider for example the relation from $A = \{0, 1, 2, 3\}$ to $\{a, b, c\}$ defined by the list $\{(0, a), (0, c), (1, c), (2, b), (1, a)\}$.

Example 3.5 The divisibility relation on natural numbers $\{1, \cdots, 12\}$ is represented by the list:
$\{(1,1),(1,2),\ldots,(1,12),(2,2),(2,4),\ldots,(2,12),$
$(3,3),(3,6),(3,9),(3,12),(4,4),(4,8),(4,12),(5,5),(6,6),$
$(6,12),(7,7),(8,8),(9,9),(10,10),(11,11),(12,12)\}.$

A more compact representations lists the "partners" for each item:
$1 : 1, 2, 3, 4, \ldots$
$2 : 2, 4, 6, 8, \ldots$
$3 : 3, 6, 9, 12$
etc.

Reflexivity in this representation means the list contains all pairs (a, a). Symmetry means if the relation contains (a, b) then it contains (b, a). Transitivity: if it contains (a, b) and (b, c) then it contains (a, c).

3.3.2 Boolean matrices

These are used Used for finite sets A, B. We assign rows for elements of A, columns for elements of B. We put a 1 in row r, column c if the pair (r, c) is in the relation, 0 otherwise.

Example 3.6 The relation from $\{0, 1, 2, 3\}$ to $\{a, b, c\}$ of our first example is represented by the matrix

	a	b	c
0	1	0	1
1	1	0	1
2	0	1	0
3	0	0	0

Example 3.7 The divisibility relation over $\{1, 2, \ldots, 12\}$ is represented by the matrix

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	0	1	0	1	0	1	0	1	0	1
3	0	0	1	0	0	1	0	0	1	0	0	1
4	0	0	0	1	0	0	0	1	0	0	0	1
5	0	0	0	0	1	0	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0	0	1
7	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	1

Some properties of the relation can be seen in the matrix:

Reflexivity: major diagonal is all 1

Symmetry: the matrix is symmetric around major diagonal

Transitivity: unclear right now

3.3.3 Digraphs

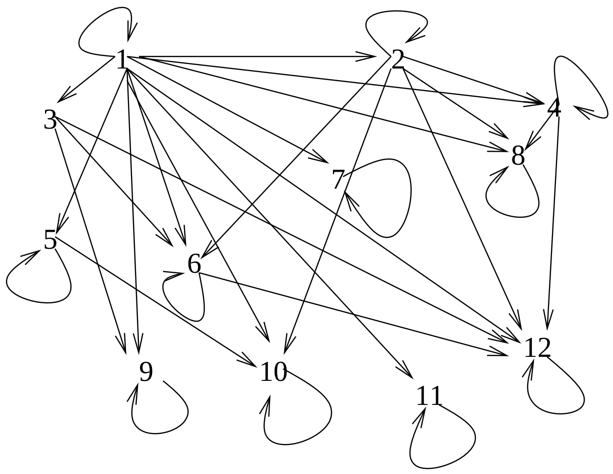
Digraphs are ways to make a *picture* of a relation. The picture can reveal lots of "shape" (topological) properties of the relation that are not clear from the other representations (the matrix is good for *algebraic* properties).

The real power of graphs is for relations on a single set A. To represent a relation on A as a digraph we draw a dot (vertex) for each element of A, and draw an arrow from first element to second element of each pair in the relation. The digraph may contain self-loops, i.e. arrows from a dot to itself, associated to the elements a such that (a, a) is in the relation.

For relations from A to $B \neq A$, you use two "columns" of vertices: A on left, B on right, and draw arrows from A vertices to B vertices. But we will focus on relations on A.

When we talk about we relations as graphs, we change terminology: elements of A are *vertices* and pairs in the relation are *edges*. (This is inherited from geometry, where graphs also play an important role.)

Example 3.8 The divisibility relation over $\{1, 2, \ldots, 12\}$ is represented by the digraph



Once again, relation properties can be seen in the graph:

Reflexivity: All nodes have self-loops.

Symmetry: all edges are bidirectional. In this case, instead of drawing lots of bidirectional arrows, we just draw a line with no arrowhead. This is called an *undirected graph.*

Transitivity: Short-circuits, for any pair of consecutive arrows, there is a single arrow from the first to the last node.

We will devote time later to the topological properties of graphs. For now, they are just a way to represent relations.