Noter til DM11

Skrevet af
Joan Boyar

Modificeret af
Jørgen Bang-Jensen, Anders Yeo og Lene Monrad Favrholdt

December 2, 1998

Contents

1 Structural Induction

In computer science, recursively defined sets occur in many places. The well-formed formulae and strings defined in section 3.3 of the textbook by Rosen are good examples, but binary trees are perhaps an even better known example. When a set X is defined recursively, a set of basis elements S_0 is defined, along with a finite set of operators, and together these define an infinite sequence of sets $\{S_i \mid i \geq 0\}$. S_{i+1} will contain all of the objects in S_i, plus all of the objects which can be obtained from objects in S_i by applying one of the operators. Then, the set X being defined is the union of all of the sets S_i. For example, the set of strings Σ^* over an alphabet Σ contains the basis elements $S_0 = \{\lambda\}$, where λ is the empty string, and the operations are concatenation on the right by the different elements of Σ. Thus, if $x \in \Sigma^*$ and $a \in \Sigma$, then $xa \in \Sigma^*$. So S_n is the set of all strings of length at most n, and Σ^* contains all strings of finite length. When Σ^* is defined in this recursive manner, then the concatenation of strings should also be defined recursively. It can be done as follows: Suppose $x, y \in \Sigma^*$ and $a \in \Sigma$. Then $x \odot \lambda = x$, and $x \odot ya = (x \odot y)a$.

Example 12 in section 3.3 is an example of what is called *structural induction*, which is often used for proving properties of recursively defined sets. It is no different from the mathematical induction defined in the text, with the inductive hypothesis being that something holds for every element in the set S_n, and the inductive step being to show that then it must also hold for S_{n+1}.

Structural induction. Suppose X is a recursively defined set with basis elements S_0. Let $\{S_i \mid i \geq 0\}$ be the sequence of sets defined by the basis elements and operators. Let P be a propositional function defined on the elements of X (so X is a subset of the universe of discourse). To prove $P(x)$ holds for all $x \in X$:

- **Basis step:** Prove that $P(y)$ holds for all $y \in S_0$.
- **Inductive step:** Show that if $P(y)$ holds for all $y \in S_n$, then $P(z)$ also holds for all $z \in S_{n+1}$.

Proving the inductive step is the same as proving, for each operator $\oplus$ in the recursive definition, that if P holds for some elements x_1, x_2,..., x_k (where $k = 1$ if the operator is unary), then P holds for the element $\oplus(x_1, x_2, ..., x_k)$.

When using structural induction to prove a proposition P concerning strings over an alphabet Σ, one proves that P holds for the empty string as the basis step. Then, one shows that if it holds for some string y, it holds for every string ya where $a \in \Sigma$.

Example with strings: Let $[a]$ denote the string consisting only of a. The reverse of a string can be defined as follows:
$\lambda^{\mathrm{R}} = \lambda$, $(xa)^{\mathrm{R}} = [a] \odot (x^{\mathrm{R}})$, if $x \in \Sigma^*$ and $a \in \Sigma$.

Note that the notation $[a]$ is used to make it clear when a is being used as an element of Σ and when it is being used as a string. In xa, it is the last element of a string. In $[a] \odot (x^{\mathrm{R}})$, it must be a string since it is the first operand in a concatenation operation.

To prove that applying the reverse operation twice to a string gives the original string, we first prove the following claim:

Claim: For all strings x over Σ and all $a \in \Sigma$, $([a] \odot x)^{\mathrm{R}} = x^{\mathrm{R}}a$.
Pf. (By structural induction on x.)
Basis step: $x = \lambda$.
$([a] \odot x)^{\mathrm{R}} = ([a] \odot \lambda)^{\mathrm{R}} = (\lambda a)^{\mathrm{R}} = [a] \odot (\lambda)^{\mathrm{R}} = [a] \odot \lambda = \lambda a = \lambda^{\mathrm{R}}a = x^{\mathrm{R}}a$. $\surd$
Inductive step: Assume that for some string y, for all $a \in \Sigma$, $([a] \odot y)^{\mathrm{R}} = y^{\mathrm{R}}a$.
Consider any string $x = yb$ where $b \in \Sigma$. Then
$([a] \odot x)^{\mathrm{R}} = ([a] \odot yb)^{\mathrm{R}} = (([a] \odot y)b)^{\mathrm{R}} = [b] \odot ([a] \odot y)^{\mathrm{R}} = [b] \odot (y^{\mathrm{R}}a)$
(the last equality is by the induction hypothesis). But, by the definitions of concatenation and reverse, $[b] \odot (y^{\mathrm{R}}a) = ([b] \odot y^{\mathrm{R}})a = (yb)^{\mathrm{R}}a = x^{\mathrm{R}}a$. $\surd$
The claim follows by structural induction. $\square$

Thm. For all strings x over Σ, $(x^{\mathrm{R}})^{\mathrm{R}} = x$.
Pf. (By structural induction on x.)
Basis step: $x = \lambda$.
$(x^{\mathrm{R}})^{\mathrm{R}} = (\lambda^{\mathrm{R}})^{\mathrm{R}} = \lambda^{\mathrm{R}} = \lambda$. $\surd$
Inductive step: Assume that for some string y, $(y^{\mathrm{R}})^{\mathrm{R}} = y$.
Consider any string $x = ya$ where $a \in \Sigma$. Then
$(x^{\mathrm{R}})^{\mathrm{R}} = ((ya)^{\mathrm{R}})^{\mathrm{R}} = ([a] \odot (y^{\mathrm{R}}))^{\mathrm{R}} = (y^{\mathrm{R}})^{\mathrm{R}}a = ya = x$. $\surd$
The theorem follows by structural induction. $\square$

As mentioned earlier, the binary tree is probably the best known example of a set defined recursively.

Def. A *binary tree* is either empty or it consists of a node called the *root* together with two binary trees called the *left subtree* and the *right subtree* of the root. A node is a *leaf* if its left and right subtree are empty; otherwise it is an *internal node*.

When using structural induction to prove a proposition P concerning binary trees, one proves that P holds for the empty tree as the basis step. Then one shows that if P holds for two subtrees L and R, it holds for the tree with root r, left subtree L, and right subtree R. As an example, consider the following:

Thm. A binary tree with n internal nodes has at most $n + 1$ leaves.
Pf. (By structural induction.)
Basis step: The empty tree has zero internal nodes and zero leaves. $\surd$
Inductive step: Suppose L is a binary tree with n_1 internal nodes and at most $n_1 + 1$ leaves and R is a binary tree with n_2 internal nodes and at most $n_2 + 1$ leaves. Consider the tree T with root r, left subtree L and right subtree R. If both L and R are empty, then r is a leaf, so there are zero internal nodes and one leaf. $\surd$
Otherwise, r is an internal node of T, the internal nodes of L and R are also internal nodes of T, and the leaves of L and R are the leaves of T. Thus T has $1 + n_1 + n_2$ internal nodes and no more than $(n_1 + 1) + (n_2 + 1) = 2 + n_1 + n_2$ (i.e. at most one more than the number of internal nodes) leaves. $\surd$
The theorem follows by structural induction. $\square$

2 The Extended Euclidean Algorithm

This is a supplement to the textbook's method for finding the integers s and t such that $\gcd(a, b) = sa + tb$.

Thm. If $a, b \in \mathbb{N}$, there exist $s, t \in \mathbb{Z}$ s.t. $sa + tb = \gcd(a, b)$.

Pf. Let d be the smallest positive integer in $D = \{xa + yb \mid x, y \in \mathbb{Z}\}$.
$d \in D \Rightarrow d = x'a + y'b$ for some $x', y' \in \mathbb{Z}$. Now, $\gcd(a, b) \mid a$ and $\gcd(a, b) \mid b$, so $\gcd(a, b) \mid x'a$, $\gcd(a, b) \mid y'b$, and $\gcd(a, b) \mid (x'a + y'b) = d$. Now, we will show that $d \mid \gcd(a, b)$, so $d = \gcd(a, b)$. Note $a \in D$, since $a = 1 \cdot a + 0 \cdot b$, so $0 < d \leq a$. Suppose $a = dq + r$ with $0 \leq r < d$.

$$\begin{aligned} r &= a - dq \\ &= a - q(x'a + y'b) \\ &= (1 - qx')a - (qy')b \end{aligned}$$

$\Rightarrow r \in D$
$r < d \Rightarrow r = 0 \Rightarrow d \mid a$. Similarly, one can show that $d \mid b$.
Therefore, $d \mid \gcd(a, b)$. $\square$

How do you find d, s and t?

Write b as $b = aq + r$ with $0 \leq r < a$.
Then $\gcd(a, b) = \gcd(a, r)$:

Let $d = \gcd(a, b)$.
Then, $d \mid b \Rightarrow d \mid (aq + r)$.
Also, $d \mid a \Rightarrow d \mid (aq) \Rightarrow d \mid ((aq + r) - aq) \Rightarrow d \mid r$.
Since d divides both a and r, $\gcd(a, r) \geq d$ — i.e. $\gcd(a, r) \geq \gcd(a, b)$.

Let $d' = \gcd(a, b - aq)$.
Then, $d' \mid a \Rightarrow d' \mid (aq)$
Also, $d' \mid (b - aq) \Rightarrow d' \mid ((b - aq) + aq) \Rightarrow d' \mid b$.
Since d' divides both a and b, $\gcd(a, b) \geq d'$ — i.e. $\gcd(a, b) \geq \gcd(a, r)$.

Thus, $\gcd(a, b) = \gcd(a, b \pmod a) = \gcd(a \pmod b, b)$. Thus, if we start out with two large numbers for which we want to find the greatest common divisor, we have a method for reducing to a problem where we have reduced the size of one of the numbers. This is the idea behind the Euclidean algorithm for finding the gcd. While executing the Euclidean algorithm, we can keep track of some of the quotients we compute and use them to calculate the s and t we want. This gives us the Extended Euclidean Algorithm:

{ Initialize}

$$\begin{array}{lll} d_0 \leftarrow b & s_0 \leftarrow 0 & t_0 \leftarrow 1 \\ d_1 \leftarrow a & s_1 \leftarrow 1 & t_1 \leftarrow 0 \\ n \leftarrow 1 & & \end{array}$$

{ Compute next d}

while $d_n > 0$ **do**

begin

$n \leftarrow n + 1$

$q_n \leftarrow \lfloor d_{n-2}/d_{n-1} \rfloor$

$d_n \leftarrow d_{n-2} - q_n d_{n-1}$ $\qquad$ $\{ d_{n-2} \pmod{d_{n-1}}\}$

$s_n \leftarrow q_n s_{n-1} + s_{n-2}$

$t_n \leftarrow q_n t_{n-1} + t_{n-2}$

end

$s \leftarrow (-1)^n s_{n-1}$ $\qquad$ $t \leftarrow (-1)^{n-1} t_{n-1}$

$\gcd(a, b) \leftarrow d_{n-1}$

Note that $\lfloor x \rfloor$ is the floor function, also called the greatest integer function, defined on page 68 of the textbook. It is simply truncation, and in this case it gives the result of an integer division.

The proof of correctness is by induction and the analysis of the running time uses Fibonacci numbers. We get to this later in the course.

How can we find multiplicative inverses modulo m? Given a and m, we want to find x such that $a \cdot x = 1 \pmod{m}$. This means we also want a k, so it all satisfies $ax = 1 + km$. Thus, we want to solve for an s in an equation $sq + tm = 1$. This can be done if a and m are relatively prime. Just use the Extended Euclidean algorithm.

Proof of correctness of the Extended Euclidean Algorithm

It is easiest to first show that the greatest common divisor is itself computed correctly and then show that the values s and t are correct. These two steps toward proving the correctness each use a claim which is proven by induction.

Claim: $\gcd(a, b) = \gcd(d_k, d_{k-1})$ for $1 \le k \le n$.
Pf. (by induction on k)
Basis step: $k = 1$. $d_1 = a$, $d_0 = b$, so $\gcd(a, b) = \gcd(d_1, d_{1-1})$. $\surd$
Inductive step: Assume that $\gcd(a, b) = \gcd(d_k, d_{k-1})$ and $1 \le k \le n - 1$. Then $d_{k+1} = d_{k-1} - q_{k+1} d_k$, so, as argued before, $\gcd(d_k, d_{k-1}) = \gcd(d_{k+1}, d_k)$. By the inductive hypothesis, this is equal to $\gcd(a, b)$. $\surd$
Thus, $\gcd(a, b) = \gcd(d_k, d_{k-1})$ for $1 \le k \le n$. $\qquad \square$

Since $d_n = 0$, $\gcd(d_n, d_{n-1}) = d_{n-1}$, which is the result produced by the algorithm. By the previous claim, this is the correct greatest common divisor.

Claim: $(-1)^{k-1}s_k a + (-1)^k t_k b = d_k$ for $0 \leq k \leq n-1$.
Pf. (by induction on k)
Basis step: $k = 0$. $d_0 = b$, $s_0 = 0$, and $t_0 = 1$, so $(-1) \cdot 0 \cdot a + 1 \cdot 1 \cdot b = d_0$. $\surd$
$k = 1$. $d_1 = a$, $s_1 = 1$, and $t_1 = 0$, so $1 \cdot 1 \cdot a + (-1) \cdot 0 \cdot b = d_1$. $\surd$
Inductive step: Assume $(-1)^{k-1}s_k a + (-1)^k t_k b = d_k$ for $0 \leq k < k'$, where $2 \leq k' \leq n-1$.

$$\begin{aligned}
d_{k'} &= d_{k'-2} - q_{k'} d_{k'-1} \\
&= ((-1)^{k'-3} s_{k'-2} a + (-1)^{k'-2} t_{k'-2} b) - q_{k'}((-1)^{k'-2} s_{k'-1} a + (-1)^{k'-1} t_{k'-1} b) \\
&= (-1)^{k'-1}(s_{k'-2} + q_{k'} s_{k'-1}) a + (-1)^{k'}(t_{k'-2} + q_{k'} t_{k'-1}) b \\
&= (-1)^{k'-1} s_{k'} a + (-1)^{k'} t_{k'} b \ \surd
\end{aligned}$$

Thus, $(-1)^{k-1}s_k a + (-1)^k t_k b = d_k$ for $0 \leq k \leq n-1$. $\square$

Since the algorithm sets $s = (-1)^n s_{n-1}$ and $t = (-1)^{n-1} t_{n-1}$, the final values satisfy $sa + tb = d_{n-1} = \gcd(a, b)$. Thus, the algorithm is correct.

3 Algebra

3.1 The definition of a group

Def. A *group* is a set G closed under a binary operation $\odot$ such that

- *Associative Law:* For all $x, y, z \in G$, $x \odot (y \odot z) = (x \odot y) \odot z$.
- *Identity:* There is an element $e \in G$, called the *identity*, such that for all $x \in G$, $e \odot x = x \odot e = x$.
- *Inverse:* For every $x \in G$, there exists a $y \in G$ such that $x \odot y = y \odot x = e$. This y is the *inverse* of x. We write $y = x^{-1}$.

When referring to groups in general, rather than to any specific example of a group, the binary operation is called multiplication and the result $x \odot y$ is called a product. The notation x^n is used to represent $x \odot x \odot ... \odot x$, where x appears n times. There are many examples of groups.

Example: The integers $\mathbb{Z}$ under addition. The associative law holds. The identity is 0. The inverse of $n \in \mathbb{Z}$ is $-n$. Note that the reals $\mathbb{R}$ are also a group under addition, as are the rationals $\mathbb{Q}$.

Example: $\mathbb{R} - \{0\}$ under multiplication. The associative law holds. The identity is 1. The inverse of $x \in \mathbb{R}$ is $1/x$. $\mathbb{Q} - \{0\}$ is also a group under multiplication, but the integers are not, since 2, for example does not have a multiplicative inverse.

These first examples were infinite groups. The next examples are finite groups.

Def. The number of elements in a group G, written $|G|$, is the *order* of the group.

Example: $\mathbb{Z}_n$, the integers modulo n, under addition. The set is the integers $\{0, 1, 2, ..., n-1\}$, and the addition operation is defined so that

$$a \odot b = a + b \pmod{n}.$$

Note that the result is always in $\mathbb{Z}_n$. A group must be closed under its operation, i.e. the result of applying the operation to two elements of the group must be in the group. It is easy to check that the associative law holds. The identity is 0. The inverse of $x \in \mathbb{Z}_n$ is $n - x$. The order of $\mathbb{Z}_n$ is n.

Example: $\mathbb{Z}_n^*$, the positive integers less than n which are relatively prime to n, under multiplication. The operation is defined so that

$$a \odot b = a \cdot b \pmod{n}.$$

The set is closed because if $x, y \in \mathbb{Z}_n^*$ and $x \odot y = z$, then there is an integer k such that $x \cdot y = z + kn$. If $d = gcd(z, n)$, then d divides $z + kn$, so d divides $x \cdot y$. But x and y are relatively prime to n, so d must be 1 and $z \in \mathbb{Z}_n^*$. It is easy to check that the operation is associative. The identity is 1. The inverse exists by Theorem 3 of section 2.5 in the textbook. The order of $\mathbb{Z}_n^*$ is $\phi(n)$, where ϕ is the Euler ϕ-function, and its value at n is the number of positive integers less than n which are relatively prime to n.

Def. A group G with the property that $x \odot y = y \odot x$ for all $x, y \in G$ is called *abelian.*

The above examples are all *abelian groups.* Not all groups are abelian.

Example: S_n, the symmetric group on n letters, is the set of permutations of the set $\{1, 2, ..., n\}$. Let $n = 5$ and consider the permutation $\sigma = (2, 1, 5, 3, 4)$. This can be written as $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 5 & 3 & 4 \end{pmatrix}$, and defines the function $\sigma(1) = 2$, $\sigma(2) = 1$, $\sigma(3) = 5$, $\sigma(4) = 3$, $\sigma(5) = 4$. Let $\eta = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 4 & 3 & 2 \end{pmatrix}$. Composing the functions σ and η gives $\sigma\eta(1) = \sigma(5) = 4$, $\sigma\eta(2) = \sigma(1) = 2$, $\sigma\eta(3) = \sigma(4) = 3$, $\sigma\eta(4) = \sigma(3) = 5$, and $\sigma\eta(5) = \sigma(2) = 1$. So $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 5 & 3 & 4 \end{pmatrix}\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 4 & 3 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 2 & 3 & 5 & 1 \end{pmatrix}$. Since function composition is associative, the composition of permutations is an associative operation. Note that $\eta\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 5 & 2 & 4 & 3 \end{pmatrix} \neq \sigma\eta$, so composition of permutations is not necessarily commutative. This composition of permutations is the operation in S_n. The identity is $\begin{pmatrix} 1 & 2 & ... & n \\ 1 & 2 & ... & n \end{pmatrix}$. If $\sigma = \begin{pmatrix} 1 & 2 & ... & n \\ i_1 & i_2 & ... & i_n \end{pmatrix}$ and $\eta = \begin{pmatrix} i_1 & i_2 & ... & i_n \\ 1 & 2 & ... & n \end{pmatrix}$, then $\sigma\eta = \eta\sigma = \begin{pmatrix} 1 & 2 & ... & n \\ 1 & 2 & ... & n \end{pmatrix}$, so every element has an inverse. Thus, S_n is a group, but it is not an abelian group.

Consider the permutation $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 5 & 3 & 4 \end{pmatrix}$. This is composed of two cycles $1 \to 2 \to 1$ and $3 \to 5 \to 4 \to 3$, so this permutation σ can also be written as a product of cycles: $(12)(354)$. In fact any permutation can be written similarly as a product of cycles. For a permutation $(1)(2)(34)(5)$, we often just write (34); any integers which form cycles of length 1 are fixed, and they are not listed when the permutation is written as a product of cycles. A *transposition* is a cycle of length 2, (ij). Note that one can compose transpositions, as one composes permutations, since they are permutations. Thus the product $(45) \odot (34)$ is the cycle (354), so $\sigma = (12) \odot (45) \odot (34)$. A permutation is *even* if and only if it can be expressed as a product of an even number of transpositions.

If the underlying set which gets permuted is some finite set X, not necessarily the same as $\{1, 2, ..., n\}$, then the set of permutations of the elements of X is clearly also a group under composition. This group is called the *symmetric group* of X and can be denoted $\mathrm{Sym}(X)$.

3.2 Subgroups

Def. Let G be a group and H be a subset of G. Then, H is a *subgroup* of G if H itself is a group with respect to the operation in G. The *order* of a subgroup is its cardinality.

Suppose G is a group and $H \subseteq G$. Then H is a subgroup of G if and only if the following hold:

- For all $x, y \in H$, $x \odot y \in H$. (H is closed under the group operation.)
- The identity is in H.
- For every $x \in H$, $x^{-1} \in H$.

Note that associativity was not mentioned here; H automatically inherits that from the group G. If H is known to be nonempty, it is unnecessary to mention that the identity must be in H, since that is implied by the other two properties.

Example: Any group is a subgroup of itself.

Example: The subset of any group, G, which consists of only the identity element is a subgroup of G.

Example: The even integers are a subgroup of the integers under addition.

Example: The set $\left\{ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 3 & 5 \end{pmatrix} \right\}$ is a subgroup of S_5.

Example: It can be shown that the subset A_n of the symmetric group S_n, which contains all even permutations on n letters, is a subgroup of S_n. It is called the *alternating group* on n letters.

Example: The set $\{0, 5, 10\}$ is a subgroup of $\mathbb{Z}_{15}$ under addition.

Example: The set $\{1, 2, 4\}$ is a subgroup of $\mathbb{Z}_7^*$.

Thm. Suppose S is a nonempty collection of subgroups of a group G. Let $H' = \bigcap_{H \in S} H$. Then H' is a subgroup of G.

Pf. Suppose $x, y \in H'$. Then $x, y \in H$ for all $H \in S$. Since each of the subgroups in S must themselves be closed under the group operation, $x \odot y \in H$ for all $H \in S$, so $x \odot y \in H'$. Thus H' is closed under the group operation. The identity is in H' since it must be in every subgroup $H \in S$. Suppose $x \in H'$. Then x^{-1} must also be in H' since it must be in each of the subgroups in S. $\square$

3.3 Generators

Def. Let H be a subset of a group G, and let S be the collection of all subgroups of G which contain H. Then, $<H>= \bigcap_{G' \in S} G'$ is the *subgroup generated by* H.

Suppose H is a subset of a group G. Consider all products $h_1 \odot h_2 \odot ... \odot h_n$, where h_i or h_i^{-1} is an element of H for all i (the h_i are not necessarily all distinct). The set of all these products is clearly a subgroup of G, and any subgroup of G containing H must contain all these elements, so this set is $<H>$, the subgroup generated by H.

Def. A group or subgroup is said to be *cyclic* if it is generated by a single element. An element which generates a cyclic group or subgroup is called a *generator* or *primitive element.*

Note that a group G with identity e is cyclic if and only if it is of the form $\{..., g^{-2}, g^{-1}, g^0 = e, g^1, g^2, ...\}$, since if an element h of G is a product of t copies of g and s copies of g^{-1} in any order, the result is g^{t-s} if $t \geq s$ and $(g^{-1})^{s-t}$ otherwise.

Def. The *order* of an element of a group G is the order of the subgroup that element generates.

Thm. Suppose G is a group with identity e, and $g \in G$ has finite order m. Then m is the least positive integer such that $g^m = e$.

Pf. Since g generates a finite subgroup, there exist positive integers s, t with $s < t$ such that $g^s = g^t$. (Otherwise, the subgroup generated would be infinite.) Consider $g^{t-s} = g^t(g^{-1})^s = g^s(g^{-1})^s = e$. This shows that there exists at least one positive integer $t - s$ such that $g^{t-s} = e$. Let k be the smallest positive integer such that $g^k = e$. Suppose $r \geq k$. Then r can be written as $r = ik + j$, where i and j are nonnegative integers, such that $j \leq k - 1$. Since $g^r = g^{ik+j} = g^{ik}g^j = (g^k)^i g^j = e^i g^j = eg^j = g^j$, any element g^r where $r \geq k$ is in the set $H = \{e = g^0, g^1, g^2, ..., g^{k-1}\} \subseteq <g>$. Note that this set H contains k distinct elements, since if $g^i = g^j$, with $0 \leq i < j \leq k - 1$, then $r = j - i < k$ is a smaller positive integer such that $g^r = e$. But k was the smallest, so all these powers must be distinct. Suppose $x \in <g>$. Then there is some nonnegative integer r such that $x = g^r$ or $x = (g^{-1})^r$. We have already argued that if $x = g^r$, then $x \in H$, so suppose that $x = (g^{-1})^r$. Let d be the smallest integer such that $kd \geq r$. Then $g^{kd-r} \in H$, but $g^{kd-r} = (g^k)^d (g^{-1})^r = (g^{-1})^r$. Hence $<g> \subseteq H$, and so $H = <g>$. Since $|H| = k$, we have that $<g> = k$, which is the least positive integer such that $g^k = e$. $\square$

Example: The set $\{2\}$ generates the subgroup $\{1, 2, 4\}$ of $\mathbb{Z}_7^*$. Thus, it is a cyclic subgroup, and the order of the element 2 is 3. The set $\{3\}$ generates all of $\mathbb{Z}_7^*$, so it is a cyclic group. In fact, it can be shown that $\mathbb{Z}_p^*$ is cyclic whenever p is prime.

3.4 Lagrange's Theorem

Def. Let H be a subgroup of G and $x \in G$. Then, $Hx = \{h \odot x \mid h \in H\}$ is a *right coset* of H in G.

Lemma Let H be a subgroup of G. All right cosets of H contain $|H|$ elements.

Pf. Clearly for any x, Hx contains at most $|H|$ elements. If it contains fewer than $|H|$ elements, there must be two distinct elements $h_1, h_2 \in H$ such that $h_1 \odot x = h_2 \odot x$. But multiplying on the right by x^{-1} gives $h_1 = h_2$. Thus, every element of H is mapped onto a different element of G when multiplied on the right by $x \in G$, so $|Hx| = |H|$. $\square$

Consider the subgroup $H = \{1, 2, 4\}$ of $\mathbb{Z}_7^*$. The right cosets of H are $H \odot 1 = H$, $H \odot 2 = H$, $H \odot 3 = \{3, 5, 6\}$, $H \odot 4 = H$, $H \odot 5 = \{3, 5, 6\}$, and $H \odot 6 = \{3, 5, 6\}$. So any two cosets are either identical or disjoint. This holds in general; the collection of cosets of a subgroup H of G is a partition of the group G. Thus, the relation $R = \{(a, b) \mid a \text{ and } b \text{ are in the same right coset of } H\}$ is an equivalence relation.

Lemma Let H be a subgroup of G, $x, y \in G$. Then either $Hx = Hy$ or $Hx \cap Hy = \emptyset$.

Pf. Suppose that $Hx \cap Hy \neq \emptyset$. Then there is some $b \in Hx \cap Hy$, so $b = h_1 \odot x = h_2 \odot y$ for some $h_1, h_2 \in H$. Thus, $x = h_1^{-1} \odot h_2 \odot y$, and $Hx = H(h_1^{-1} \odot h_2 \odot y)$. Hence any element of Hx is of the form $h \odot (h_1^{-1} \odot h_2 \odot y) = (h \odot h_1^{-1} \odot h_2) \odot y$ for some $h \in H$. Since H is a subgroup, $h \odot h_1^{-1} \odot h_2 \in H$, so $(h \odot h_1^{-1} \odot h_2) \odot y \in Hy$. Thus, $Hx \subseteq Hy$. Since, by the previous lemma, the two cosets have the same size, they must be equal. So if $Hx \cap Hy \neq \emptyset$, then $Hx = Hy$. $\square$

These two lemmas tell us that the group G must be a disjoint union of right cosets of any subgroup H, all of which have the same size. This gives us the following:

Thm. [Lagrange] If G is a finite group and H is any subgroup of G, then the order of H divides the order of G.

Corollary Let G be a finite group and $g \in G$. The order of the element g divides the order of the group G.

Corollary Suppose $|G| = n$ and $g \in G$. Then $g^n = e$, where e is the identity in G.

Pf. Let s be the order of the subgroup generated by g. By a previous theorem, $g^s = e$, where e is the identity. By Lagrange's Theorem, s divides n, so there is an integer c such that $n = sc$. Note that $g^n = g^{sc} = (g^s)^c = e^c = e$, so the corollary follows. $\square$

Recall Fermat's Little Theorem from section 2.5 of the textbook. It becomes much easier to prove using a little bit of group theory.

Thm. [Fermat's Little Theorem] If p is prime and a is an integer not divisible by p, then $a^{p-1} \equiv 1 \pmod{p}$.

Pf. Consider the multiplicative group $\mathbb{Z}_p^*$. This contains the positive integers less than p which are relatively prime to p. Since p is prime, all integers between 1 and $p-1$ are relatively prime to p. Thus, the order of $\mathbb{Z}_p^*$ is $p-1$. By the previous corollary, for any a in this group, $a^{p-1} \equiv 1 \pmod{p}$, so this holds for any integer a which when reduced modulo p is between 1 and $p-1$. This is any integer not divisible by p. $\square$

Using this same group theory, we can more easily show that encryption with RSA, followed by decryption with RSA yields the original message, if the message is less than n and relatively prime to n. (The other cases can be proven using group theory and the Chinese Remainder Theorem by considering all the cases separately.) Recall that $n = pq$ where p and q are large primes. Then, $|\mathbb{Z}_n^*| = \phi(n) = (p-1)(q-1)$ (one can get this using the Principle of Inclusion-Exclusion — see problems 22 and 23 on page 348). If the message M is less than n and relatively prime to n, $M \in \mathbb{Z}_n^*$. Recall that $ed \equiv 1 \pmod{(p-1)(q-1)}$, so there is an integer k such that $ed = 1 + k(p-1)(q-1)$. If $C \equiv M^e \pmod{n}$, then $C^d \equiv (M^e)^d \equiv M^{ed} \equiv M^{1+k(p-1)(q-1)} \equiv M \cdot M^{k(p-1)(q-1)} \equiv M \cdot (M^{\phi(n)})^k \equiv M \cdot 1^k \equiv M \pmod{n}$.

3.5 Rings and fields

The integers and real numbers have two operations addition and multiplication which can be considered together. For example, there is a distributive law which says that $x(y+z) = xy + xz$. Groups are generalizations of sets which have one "nice" operation; rings and fields are generalizations of sets which have two "nice" operations.

Def. A *ring* is a set R closed under two binary operations $+$ and $\bullet$ such that

- R1. R is an abelian group with respect to the operation $+$.
- R2. The operation $\bullet$ is associative.
- R3. [Distributive Laws] For all $x, y, z \in R$, the following hold:

$$\begin{aligned} x \bullet (y+z) &= x \bullet y + x \bullet z \\ (y+z) \bullet x &= y \bullet x + z \bullet x \end{aligned}$$

The first operation $+$ is called addition and the second operation $\bullet$ is called multiplication.

Note that any ring must contain at least one element since it contains an identity with respect to addition. One can define a ring with just one element 0. Then $0 + 0 = 0$

and $0 \bullet 0 = 0$. This is called the *trivial ring*. A *nontrivial ring* is a ring with more than one element. The identity element with respect to addition is called *zero*, and all other elements are called *nonzero elements*. If the ring R has an identity element i with respect to multiplication, then for all $x \in G$, $i \bullet x = x \bullet i = x$, and R is said to be a *ring with identity*. This identity is denoted by 1. The ring R is *commutative* if for all $x, y \in R$, $x \bullet y = y \bullet x$. Note that the integers and the reals are both commutative rings with identity.

Def. A *field* is a nontrivial commutative ring with identity in which every nonzero element has a multiplicative inverse.

Example: $\mathbb{R}$ is a field and $\mathbb{Q}$ is a field, but $\mathbb{Z}$ is not because 2, for instance, does not have a multiplicative inverse.

Example: $\mathbb{Z}_n$ is a field when n is prime. It is a ring when n is composite, but it is not a field. To see this, consider p which divides n, where $1 < p < n$. Suppose $n = pr$. This element p is in the ring $\mathbb{Z}_n$, but it does not have a multiplicative inverse because it is not relatively prime to n. The element r is also in the ring $\mathbb{Z}_n$. Note that $p \bullet r \equiv 0 \pmod{n}$, so p and r are called *zero divisors*. Any element which is relatively prime to n, though, has an inverse.

4 Probability

The textbook has too little on probability, so this is a supplement to what it has.

Note that in section 4.5, there is no need for the restriction to finite sample spaces, even though we are restricting ourselves to discrete probability spaces. For example, consider the experiment in which a fair coin is flipped until the first time "heads" is the result. Then, the sample space is

$$S = \{H, TH, TTH, TTTH, TTTTH, ..., T^nH, ...\}.$$

This is not finite, but we can assign probabilities to each event. The probability of T^nH is $(1/2)^{n+1}$ for $n \geq 0$. If the coin is biased so that the probability of "heads" is p and the probability of "tails" is $q = 1 - p$, then the probability of T^nH is $q^n p$. In this case we have independent Bernoulli trials which are repeated until the first success.

If the sample space is $\{x_i \mid i \geq 1\}$, the probabilities must satisfy:

- $0 \leq p(x_i) \leq 1 \quad \forall i \geq 1$
- $\sum_{i=1}^{\infty} p(x_i) \;=\; 1.$
- For any events $A_1, A_2, ...$ that are pairwise mutually exclusive (i.e. $A_i \cap A_j = \emptyset$ for $i \neq j$), $p(\bigcup\limits_i A_i) = \sum\limits_i p(A_i)$.

Note that the last is not included in the textbook, but really should be. These probability axioms can be written more generally, so they also apply to any finite sample space S.

- $0 \leq p(x_i) \leq 1 \quad \forall x_i \in S$
- $\sum_{x_i \in S} p(x_i) \;=\; 1.$
- For any events $A_1, A_2, ...$ that are pairwise mutually exclusive (i.e. $A_i \cap A_j = \emptyset$ for $i \neq j$), $p(\bigcup\limits_i A_i) = \sum\limits_i p(A_i)$.

When the probabilities satisfy these axioms, the mapping which is defined by p is a *probability distribution*. We are interested in *discrete probability distributions*; those defined over finite or countably infinite sample spaces. For a finite sample space S, if $p(x_i) = 1/|S|$ for all $x_i \in S$, then we have the *uniform probability distribution*; each outcome is equally likely. The probability distributions defined by a fair coin and by fair dice are uniform. The probabilities defined above for the number of coin tosses required before the result is a "head" is called the *geometric distribution*. As mentioned on page 274, the probability distribution defined by the probability of k successes in n independent Bernoulli trials is called the *binomial distribution*.

The more general proofs of theorems 1 and 2 from section 4.4, given on pages 270-271 in section 4.5 clearly hold with infinite sample spaces, if one simply changes n to ∞. The definitions of conditional probability, independence, and random variables are exactly the same whether one is talking about a finite or a countably infinite sample space.

The events $A_1, A_2, ..., A_n$ are *pairwise independent* if $p(A_i \cap A_j) = p(A_i) \cdot p(A_j)$ for $i \neq j$, and they are mutually independent if, for every subset of $2 \leq k \leq n$ events $A_{i_1}, A_{i_2}, ...A_{i_k}$ where $1 \leq i_1 < i_2 < ... < i_k \leq n$, we have $p(A_{i_1} \cap A_{i_2} \cap ... \cap A_{i_k}) = p(A_{i_1})p(A_{i_2}) \cdots p(A_{i_k})$.

A well-known theorem concerning conditional probabilities is

Thm. [Baye's Theorem] For two events A and B such that $p(A) > 0$ and $p(B) > 0$,

$$p(A|B) = \frac{p(A)p(B|A)}{p(B)}$$

Pf. By the definition of conditional probability, $p(A \cap B) = p(B)p(A|B)$ and $p(A \cap B) = p(A)p(B|A)$. Thus, $p(B)p(A|B) = p(A)p(B|A)$, and the result follows by dividing both sides by $p(B)$. $\square$

Def. The *expected value* of a random variable $X(s)$ on the sample space $\{x_i \mid i \geq 1\}$ is

$$E(X) = \sum_{i=1}^{\infty} p(x_i)X(x_i)$$

The expected value is also called the *mean.* Consider the sample space S defined above, with the probability of "heads" being p and the probability of "tails" being $q = 1 - p$. The expected value of the number of times the coin must be flipped to get the result "heads" once is

$$\sum_{i=1}^{\infty} iq^{i-1}p = \sum_{i=1}^{\infty} iq^{i-1} - \sum_{i=1}^{\infty} iq^i = \sum_{j=0}^{\infty}(j+1)q^j - \sum_{i=1}^{\infty} iq^i$$

by changing the index of summation so $j = i - 1$. Pulling out the term with $i = 0$ and combining, we get

$$\sum_{i=1}^{\infty} iq^{i-1}p = 1 + \sum_{j=1}^{\infty}(j+1-j)q^j = 1 + \sum_{j=1}^{\infty} q^j = 1 + (\sum_{j=0}^{\infty} q^j) - 1 = \lim_{j\to\infty} \frac{q^{j+1} - 1}{q - 1}$$

since this is a geometric progression as in example 6 on page 190 of the textbook. Since $q < 1$, $\lim_{j\to\infty} q^{j+1}$ is clearly zero, so

$$\sum_{i=1}^{\infty} iq^{i-1}p = \frac{-1}{q-1} = \frac{1}{p}$$

We summarize this as a theorem:

Thm. The expected number of times a coin with probability p for "heads" must be flipped before the result is "heads" is $\frac{1}{p}$.

Thus, with a fair coin, the expected number of tosses before the result is "heads" is 2, and with a fair die, the expected number of throws before a 4 is 6.

This result is particularly useful in the analysis of randomized algorithms. Such algorithms often involve making random choices for some object until one is found satisfying a certain property P. If $1/c$ of the objects satisfy that property P, we expect to have c trials before success. Often this expected running time is much faster in practice than one can get with any known deterministic algorithm for finding an object with the given property P.

The expectation of random variables has an important property, known as the *linearity of expectation*. A linear funtion of n random variables, $X_1, X_2, ..., X_n$ has the form $f(X_1, X_2, ..., X_n) = a_0 + a_1X_1 + a_2X_2 + ... + a_nX_n$ where $a_i \in \mathbb{R}$ for $0 \leq i \leq n$. The expected value of a linear function of random variables has the same value as that function applied to the expectations of the random variables. Thus, $E[f(X_1, X_2, ..., X_n)] = f(E[X_1], E[X_2], ..., E[X_n])$. This does not depend on whether or not the random variables are independent, so in a randomized algorithm which finds two objects with some property, the expected time for this is the sum of the expected times for finding each one. The linearity of expectation can be proven directly from the definition of expected values.

Thm. Let f be a linear function, S be a sample space, and $X_1, X_2, ..., X_n$ be random variables defined on S. Then,

$$E[f(X_1, X_2, ..., X_n)] = f(E[X_1], E[X_2], ..., E[X_n])$$

Pf. Let $f(X_1, X_2, ..., X_n) = a_0 + a_1X_1 + a_2X_2 + ... + a_nX_n$ where $a_i \in \mathbb{R}$ for $0 \leq i \leq n$. Then,

$$\begin{aligned}
E[f(X_1, X_2, ..., X_n)] &= \sum_{s\in S} p(s)f(X_1(s), X_2(s), ..., X_n(s)) \\
&= \sum_{s\in S} p(s)(a_0 + a_1X_1(s) + a_2X_2(s) + ... + a_nX_n(s)) \\
&= \sum_{s\in S}(p(s)a_0 + \sum_{i=1}^{n} p(s)a_iX_i(s)) \\
&= a_0 + \sum_{i=1}^{n}(\sum_{s\in S} p(s)a_iX_i(s)) \\
&= a_0 + \sum_{i=1}^{n}(a_i \sum_{s\in S} p(s)X_i(s)) \\
&= f(E[X_1], E[X_2], ..., E[X_n]). \qquad \square
\end{aligned}$$

The expectation of products of random variables does not work quite as nicely as sums of random variables.

Thm. If two random variables X and Y are independent, then $E[XY] = E[X] \cdot E[Y]$.

Def. The *variance* of a random variable X is

$$\mathrm{Var}[X] = E[(X - E[X])^2]$$

Thm. The variance of a random variable X can also be written as

$$\mathrm{Var}[X] = E[X^2] - E^2[X]$$

Pf. Expanding $E[(X-E[X])^2]$, we get $\mathrm{Var}[X] = E[X^2-2XE[X]+E^2[X]]$. By the linearity of expectations, this is $E[X^2]-2E[XE[X]]+E[E^2[X]]$. Since $E[X]$ is a real number, rather than a random variable, this is $E[X^2] - 2E^2[X] + E^2[X]$. Thus $\mathrm{Var}[X] = E[X^2] - E^2[X]$. $\square$

The variance is meant to tell how far from the mean one expects the value to be. If X and Y are independent random variables, then $\mathrm{Var}[X+Y] = \mathrm{Var}[X]+\mathrm{Var}[Y]$. More generally,

Thm. If $X_1, X_2, ..., X_n$ are pairwise independent random variables, then

$$\mathrm{Var}[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} \mathrm{Var}[X_i]$$

Def. The *standard deviation* of a random variable X is

$$\sigma[X] = \sqrt{\mathrm{Var}[X]}$$

Thm. For the geometric distribution with probality p, we have

1. $E[X] = \frac{1}{p}$ and
2. $\mathrm{Var}[X] = \frac{1-p}{p}$

Note that the first part of this theorem is just the first theorem on page 15.

Thm. For the binomial distribution with probality p and with X denoting the number of successes in n trials, we have

1. $E[X] = np$ and
2. $\mathrm{Var}[X] = np(1-p)$

Pf. The expected value of the binomial distribution is shown on page 278 to be np. Let X_i be the random variable which says how many successes there were in the ith trial. This value must be either 0 or 1, so $E[X_i^2] = E[X_i] = p$. To compute the variance of the binomial distribution, we have $\mathrm{Var}[X] = \mathrm{Var}[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} \mathrm{Var}[X_i]$ since the X_i are pairwise independent. But $\mathrm{Var}[X_i] = E[X_i^2] - E^2[X_i] = p - p^2 = p(1-p)$. Thus for the binomial distribution, $\mathrm{Var}[X] = \sum_{i=1}^{n} p(1-p) = np(1-p)$. $\square$

5 Huffman encoding

Representing text files in the most straightforward manner, using the ASCII representation of each character, wastes a considerable amount of space. Part of this is wasted simply because this ASCII representation uses 8 bits for every character. In fact most text files would use no more than 64 different characters, so 6 bits per character would be sufficient. However, even more space is wasted because there is so much redundancy in all natural languages.

Notice that in English text (or in Danish text) some letters occur much more frequently than others. For example, in the previous paragraph, the letter "e" occurred 48 times, while the letters "j" and "z" did not occur at all. Representing the characters with variable length bit strings could save considerable space if the most frequently occurring characters had the shortest representations and the least frequently occurring characters the longest. Even more space can be saved by considering pairs of letters. In English, the pair th occurs most frequently, but the pairs tb and qp essentially do not occur. In fact, by using this idea and other methods, it is possible to represent English text using just over 1.5 bits per character on average.

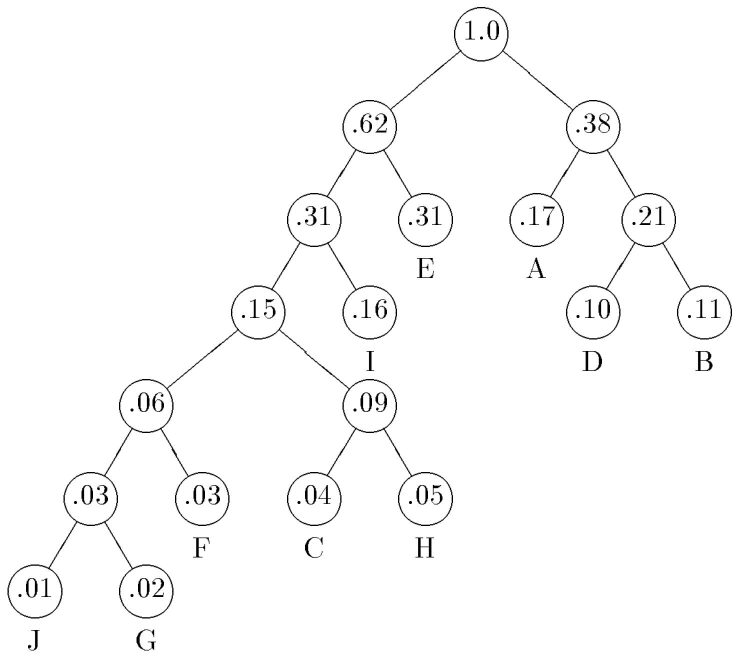
One technique for using variable-length representations for different characters, depending on the frequency of their occurrence, is called Huffman encoding. Huffman codes are an example of the prefix codes described in section 8.2 of the textbook by Rosen. These prefix codes are more often called *prefix-free*. The idea is that no code representing a character is a prefix of another code. (A prefix of a string is a substring consisting of the first k consecutive characters, for some k.) For example, the bit strings "0110" and "011010" could not both be codes in a prefix-free encoding, because the first is a prefix of the second. The advantage to a prefix-free encoding is that decoding is so easy. There is never any doubt, when decoding a long string of zeros and ones, as to when one has reached the end of the code for one character or where the code for the next character begins. For example, if an encoding for the 5-letter alphabet $\{a, b, c, d, e\}$ is given by the function f, and $f(a) = 01$, $f(b) = 0010$, $f(c) = 0011$, $f(d) = 10$, $f(e) = 11$, then the string 010011111000110111 can be decoded as $acedcae$, making just one pass through the string.

To do Huffman encoding, one builds a binary tree. Assume that there are k possible characters, and that the probability of character c_i is p_i (meaning more precisely that one expects a fraction p_i of the characters in the text to be c_i). Then, any of the k characters must occur with some positive probability, so $0 < p_i \leq 1$ for all i, and $\sum_{i=1}^{k} p_i = 1$. There will be one leaf in the binary tree for each character, and when that leaf node is created, a pointer to it is created so that during the encoding process, the node for any character can be accessed in constant time. There will also be a pointer to the root of the tree, plus parent, left and right child pointers for every node. Each node in the tree will contain a key which is a probability. The keys in the leaves will be the probabilities of the associated characters, and the leaves will also contain the characters associated with them (or pointers to these characters).

Initially, all of the nodes for the leaves are viewed as individual trees, with root probability equal to the probability of the associated character. At any point in time the roots of all the existing trees are sorted according to their probabilities. One step consists of choosing the two roots with the smallest probabilities, p_i and p_j. These two trees are made the left and right subtrees of a new root which is given probability $p_i + p_j$. This decreases the number of existing trees by one. This step is repeated until there is only one tree. Then, the pointer to the root of that tree is created (and will be used for decoding).

Notice that the binary tree created by this process will have relatively short paths from the root to the leaves associated with frequently occurring characters, and longer paths from the root to the leaves associated with less frequently occurring characters. The path from the root to a leaf will define the encoding of the character at that leaf as follows. An edge on this path going to a left child will be a zero, and an edge going to a right child will be a one. Consider the following example:

A	B	C	D	E	F	G	H	I	J
.17	.11	.04	.10	.31	.03	.02	.05	.16	.01



The encoding of the different characters is as follows:

A	B	C	D	E	F	G	H	I	J
10	111	00010	110	01	00001	000001	00011	001	000000

The fact that the codes are paths in the tree ensures that no code is a prefix of another. For one code to be a prefix of another, one leaf would have to be on the path from the root to another leaf. But that is impossible for two distinct leaves; the first one would not have been a leaf if it was on the path to the other.

In practice, the probabilities can either

1. be calculated from the known probabilities of the characters in text from the given language, in which case the Huffman tree can be a fixed tree shared by many users for many files, or
2. be calculated from the number of occurrences of the different characters in the file being compressed, in which case the Huffman tree would be included at the beginning of the compressed file.