3.1 Structure of a Proof by Induction

Induction can be used to a prove that a given proposition, $P(n)$, holds for all integers $n \geq n_0$, where n_0 is some fixed integer. The proof consists of two steps:

1. Base Step: Prove directly that the proposition $P(n_0)$ is true.
2. Induction Step: Prove $\forall n \geq n_0$: $P(n) \rightarrow P(n+1)$. In other words, for an arbitrary n (where $n \geq n_0$) we assume that $P(n)$ is true and show as a consequence that $P(n+1)$ is true. The left side of the implication is called the induction hypothesis, since it is what is assumed in the induction step.

Note: The induction step is also equivalent to: *Prove* $\forall n > n_0$: $P(n-1) \rightarrow P(n)$.

A proof by induction is akin to climbing a ladder (having an infinite number of steps). One is able to climb all the steps of a ladder if both of the following are true:

1. He is able to climb to the first step; this is the base step.
2. From an arbitrary step n, he is able to climb one step higher; this is the induction step.

Note that climbing to the second step is implied by the preceding steps 1 and 2 with n=1. Applying step 2 again with n=2, enables climbing to the third step, and so on. This shows that the proof method is sound and that the induction hypothesis is not something coming out of thin air; rather, it is being gradually established for each successive value of n.

The preceding form of induction is known as *weak induction*. For *strong induction*., we use a slightly different induction step with a *stronger* induction hypothesis.

Induction Step for Strong Induction: Prove $\forall n \geq n_0$: $(\forall k \bullet n: P(n)) \rightarrow P(n+1)$. That is, we assume that $P(k)$ is true for all k in the range $n_0 \leq k \leq n$, and then prove as a consequence that $P(n+1)$ is true. An equivalent form of this is to assume that $P(k)$ is true for all k in the range $n_0 \leq k < n$, and then prove as a consequence that $P(n)$ is true.

3.1.1 Examples of Induction Proofs

We start with a classical example of an induction proof.

Example 3.1 Show that $1+2+ \ldots +n = n(n+1)/2$ for all $n \geq 1$.

Solution:

Base Step: We are to show $P(n)$ for n=1. In this case, LHS = 1 and RHS = 1(1+1)/2 = 1. Thus, the proposition is true for n=1.

Induction Step: We are to show that, for $n \geq 1$, $P(n) \rightarrow P(n+1)$. Thus, we assume (*induction hypothesis*) the following:

$$1+2+ \ldots +n = n(n+1)/2 \tag{3.1}$$

We proceed to show $P(n+1)$. We are to show that

$$1+2+ \ldots + n+(n+1) = (n+1)((n+1)+1)/2 \tag{3.2}$$

LHS of (3.2) = $1+2+ \dots +n+(n+1) = n(n+1)/2 + (n+1)$, where the sum of the first n terms is replaced by RHS of (3.1). The latter expression = $(n+1)(n/2+1) = (n+1)(n/2+2/2) = (n+1)(n+2)/2$ = RHS of (3.2).

Example 3.2 Show that $1+a+a^2+ \dots +a^n = (a^{n+1}-1)/(a-1)$ for all $n \geq 0$. Assume $a \neq 1$.

Note: The terms in this sum form a *geometric progression*, where every term is obtained from the previous term by multiplying by some fixed factor a.

Solution:

Base Step: We show $P(0)$. LHS = 1; RHS = $(a - 1)/(a-1) = 1$. Thus, the proposition is true for n=0.

Induction Step: Assume $P(n)$ for $n \geq 0$ and show $P(n+1)$. Thus, assume (induction hypothesis) the following:

$$1+a+a^2+ \dots +a^n = (a^{n+1} -1)/(a-1) \qquad (3.3)$$

We proceed to show $P(n+1)$. We are to show that

$$1+a+a^2 + \dots +a^{n+1} = (a^{n+2} -1)/(a-1) \qquad (3.4)$$

LHS of (3.4) = $1+a+a^2+ \dots +a^n+a^{n+1} = [(a^{n+1} -1)/(a-1)] + a^{n+1}$, where the sum of the terms up to a^n is replaced by RHS of (3.3). The latter expression gives: $1/(a-1)\,[a^{n+1} -1 + (a-1)\, a^{n+1}] = (a^{n+2} -1)/(a-1)$ = RHS of (3.4).

Note: A special case of a geometric progression is when summing powers of 2: $1+2+ 2^2 + \dots + 2^n = 2^{n+1} -1$.

Example 3.3 Find a formula for $1/2+ 1/4 + \dots + 1/2^n$ and prove your claim.

Solution: The sum of the first two terms is 3/4; the sum of the first three terms = 3/4+1/8 = 7/8. Thus, we guess that the sum of the first k terms is $(2^k -1)/2^k$, and because there are n terms (noting that the denominator goes from 2^1 to 2^n), we guess that the expression evaluates to $(2^n -1)/2^n$. Next, we use induction to prove this guess. We only show the induction step.

Induction Step: Assume $P(n)$ for $n \geq 1$ and show $P(n+1)$. Thus assume

$$1/2+1/4+ \dots +1/2^n = (2^n -1)/2^n \qquad (3.5)$$

We proceed to show $P(n+1)$. We are to show that

$$1/2+1/4 + \dots +1/2^{n+1} = (2^{n+1} -1)/2^{n+1} \qquad (3.6)$$

LHS of (3.6) = $1/2+1/4 + \dots + 1/2^{n+1} = [(2^n -1)/2^n]+ 1/2^{n+1} = (1/2^{n+1})\,(2(2^n -1)+1) = (2^{n+1}-1)/2^{n+1}$ = RHS of (3.6).

Note: A direct way to establish $P(n)$ in Example 3.3 is to note that the given expression is a geometric progression and utilize the formula of Example 3.2 with a =1/2. Alternatively, multiply (and divide) the given expression by 2^n to get, $(2^{n-1} + \dots +1)/2^n = (2^n -1)/2^n$.

is shown in Figure 3.1(b) — making the induction hypothesis $P(n)$ inapplicable! We are stuck, and properly so, since the claim is false.

3.1.3 Using Induction for Counting

Because induction is about recursive definitions, it becomes handy in solving counting problems. The idea is to parameterize a definition. For example, if we let f_n denote the number of binary strings of length n satisfying some condition C then, by definition, f_{n-1} will be the number of binary strings of length n–1 satisfying the same condition C.

Example 3.7 Let f_n denote the number of ways to cover the squares of a 2×n grid using plain dominos. Then it is easy to see, as illustrated by Figure 3.2, that f_1=1, f_2=2, and f_3=3. Derive a recurrence equation for f_n.

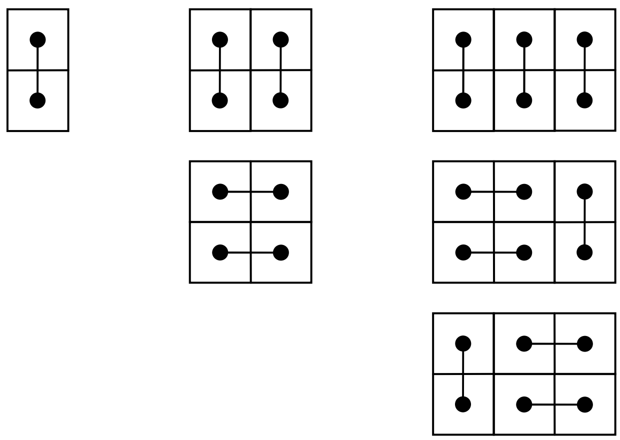

Figure 3.2 Ways of covering the squares of a 2×n grid with plain dominos for n=1,2 and 3.

Solution: The top-right square of the board can be covered by a domino that is either laid horizontally or vertically.

- If covered by a vertically-laid domino, this leaves a 2×(n–1) grid that can be covered in f_{n-1} ways.
- If covered by a horizontally-laid domino, the domino below it must also lie horizontally. This leaves a 2×(n–2) grid that can be covered in f_{n-2} ways.

Because these are all the cases, we have proven that $f_n = f_{n-1} + f_{n-2}$.

In Section 4.1, we discuss methods for solving a system of recurrence equations such as the one given previously. Interestingly, we can use induction for this; a solution can be guessed and then induction can be used to verify that the guess is correct.

When analyzing the running time of a recursive algorithm, recurrence equations can be used to quantify the number of operations executed by an algorithm. Then, induction can be used to solve the resulting equations.

Example 3.8 Let f_n be specified by the recurrence, $f_n = f_{n-1} + f_{n-1}$ for $n \geq 3$; f_1 =1, f_2 =1. Use induction to show that $f_n \geq \alpha^{n-2}$ for all integers $n \geq 3$, where $\alpha=(1+\sqrt{5}\,/2)$. Based on this, quantify f_n using the proper big-O notation.

3.5 The Coin Change Problem

The *coin change* problem calls for finding the number of ways of making a change for a given amount of *n* cents, using a given set of denominations $\{d_1, d_2, ..., d_m\}$. The problem is formulated as follows:

Given a positive integer *n*, and a set of positive integers $\{d_1, d_2, ..., d_m\}$, in how many ways can we express *n* as a linear combination of $\{d_1, d_2, ..., d_m\}$ with nonnegative integer coefficients?

In other words, if we are to make change for an amount of *n* cents using an infinite supply of each of d_1–d_m valued coins, in how many ways can we make the change (order of coins does not matter, {1,2,1}={1,1,2}={2,1,1})? For example, if *n*=4 and *d*={1,2,3}, we have a total of 4 ways, namely: {1,1,1,1}, {1,1,2}, {2,2}, {1,3}.

Here, we consider a special case of the coin-change problem, where we are given two denominations, and the problem is to determine whether there is a solution for all values of $n \geq n_0$.

Coin Change Problem. Show that any integer amount ≥ 60 cents can be changed using 6-cent and 11-cent coins. Equivalently, any integer $n \geq 60$ can be expressed as $n = 6a + 11b$, where *a* and *b* are nonnegative integers.

Proof by Induction: Let *P*(*n*) denote the proposition that an amount of *n* cents can be changed using 6-cent and 11-cent coins. In other words, *P*(*n*): $n = 6a + 11b$ where *a*, *b* are nonnegative integers.

Base Step: For *n* = 60, 60 = 6 (10) + 11 (0). Thus, *P*(60) is true.

Induction Step: We assume *P*(*n*) (for $n \geq 60$) and consider how to extend *P*(*n*) to *P*(*n*+1). If *P*(*n*) uses at least one 11-cent coin, then replace one 11-cent coin with two 6-cent coins. On the other hand, if *P*(*n*) does not use any 11-cent coins, then because $n \geq 60$, *P*(*n*) must use at least nine 6-cent coins. In this case, replace nine 6-cent coins with five 11-cent coins.

Listing 3.6 shows the corresponding recursive algorithm.

```
Input: an integer n; assume n ≥ 60
Output: a pair of integers (we can use a 2-element integer array for this)

integer_pair CoinChange(int n)
{  if (n==60)  // base case
      return (10,0);
   else
   { (a,b) = CoinChange(n-1);
      if (b > 0) return (a+2,b-1);
      else return (a-9,b+5);
   }
}
```

Listing 3.6 A recursive algorithm for the coin-change problem.

Exercise 3.9 Convert the recursive algorithm for the coin-change problem given in Listing 3.6 into an iterative algorithm, then go one step further and write it as a CSharp program method.

Exercise 3.10 Derive the order of running time for the coin-change algorithm given in Listing 3.6. Hint: Write a recurrence equation for the number of elementary operations performed by the algorithm.

3.5.1 Using Strong Induction for the Coin-Change Problem

Let us return to the problem of changing an amount of n cents ($n \geq 60$) using 6-cent and 11-cent coins, but this time we try to use strong induction.

A Faulty Inductive Proof

Base Step: For $n = 60$, $60 = 6\ (10) + 11\ (0)$.

Induction Step (*using strong induction*): Assume any amount $k \leq n$ is expressible in terms of 6 and 11. Then, since $n+1 = (n-5)+6$, we can add a 6-cent coin to the change corresponding to $P(n-5)$. This establishes $P(n+1)$.

To see why the preceding proof is faulty, consider using it to show $P(61)$. In this case, $P(61)$: $61=(60-5)+6$. This rests on the assumption that "(60–5)" is expressible in terms of 6 and 11, but the value "(60–5)" falls below the base-step value. How do we fix such a proof? *Answer:* Provide enough base cases. For $n-5$ not to fall below the base-step value, we have to provide *additional* base cases and have the induction step apply to n having values beyond those specified as base cases.

A Valid Inductive Proof

Base Step: $60 = 6\ (10) + 11\ (0)$; $61 = 6\ (1) + 11\ (5)$; $62 = 6\ (3) + 11\ (4)$;
$63 = 6\ (5) + 11\ (3)$; $64 = 6\ (7) + 11\ (2)$; $65 = 6\ (9) + 11\ (1)$;

Induction Step: We assume that k (where $60 \leq k \leq n$) is expressible in terms of 6 and 11 then the amount $n+1$ (where $n+1 > 65$) is expressible in terms of 6 and 11, since $n+1 = (n-5) + 6$.

Important Observation

In a strong induction proof where the induction step expresses $P(n+1)$ in terms of $P(n-k)$, the base step must be established for $k+1$ values: n_0, n_0+1, …, n_0+k. (Note: $k = 0$ corresponds to *weak induction*.) For divide-and-conquer algorithms (e.g. Binary search, Mergesort), we normally express $P(n)$ in terms of $P(\lfloor n/2 \rfloor)$ (and/or $P(\lceil n/2 \rceil)$). In such cases, $P(1)$ is never bypassed; therefore, it suffices to provide $P(1)$ as a base step.

Exercise 3.11 Write recursive and iterative program methods for the coin-change algorithm described by the preceding induction proof. Also, draw the tree of recursive calls for (the input) n=100.