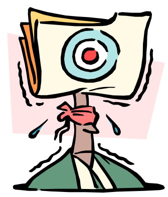
Induction



Overview

- Recursion
  - a programming strategy that solves a problem by reducing it to simpler or smaller instance(s) of the same problem
- Induction
  - a mathematical strategy for proving statements about natural numbers 0,1,2,... (or more generally, about inductively defined objects)
- Induction and recursion are very closely related

Defining Functions

- It is often useful to write a given function in different ways
  - Let S : int $\rightarrow$ int be the function where S(n) is the sum of the integers from 0 to n. E.g.,

    $S(0) = 0 \qquad S(3) = 0+1+2+3 = 6$
  - Definition: iterative form
    - $S(n) = 0+1+ \ldots + n$
  - Another characterization: closed form
    - $S(n) = n(n+1)/2$

Sum of Squares

- Here is a more complex example.
  - Let SQ : int $\rightarrow$ int be the function that gives the sum of the squares of integers from 0 to n. E.g.,

    $SQ(0) = 0 \quad SQ(3) = 0^2 + 1^2 + 2^2 + 3^2 = 14$
- Definition: $SQ(n) = 0^2 + 1^2 + \ldots + n^2$
- Is there an equivalent closed-form expression?

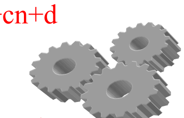
Closed-form expression for SQ(n)

- Sum of integers between 0 through n was n(n+1)/2 which is a quadratic in n.
- Inspired guess: perhaps sum of squares of integers between 0 through n is a cubic in n.

- So conjecture: $SQ(n) = an^3+bn^2+cn+d$ where a,b,c,d are unknown coefficients.
- How can we find the values of the four unknowns?
  - Use any 4 values of n to generate 4 linear equations, and solve

Finding coefficients

$SQ(n) = 0^2+1^2+\ldots+n^2 = an^3+bn^2+cn+d$



- Use n=0,1,2,3
- $SQ(0) = 0 = a\cdot0 + b\cdot0 + c\cdot0 + d$
- $SQ(1) = 1 = a\cdot1 + b\cdot1 + c\cdot1 + d$
- $SQ(2) = 5 = a\cdot8 + b\cdot4 + c\cdot2 + d$
- $SQ(3) = 14 = a\cdot27 + b\cdot9 + c\cdot3 + d$
- Solve these 4 equations to get

  $a = 1/3, b = ½, c = 1/6, d = 0$

- This suggests

$$SQ(n) \equiv 0^2 + 1^2 + \ldots + n^2$$
$$= n^3/3 + n^2/2 + n/6$$
$$= n(n+1)(2n+1)/6$$

- Question: How do we know this closed-form solution is true for all values of n?
  - Remember, we only used n = 0,1,2,3 to determine these co-efficients. We do not know that the closed-form expression is valid for other values of n.

- One approach:
  - Try a few other values of n to see if they work.
  - Try n = 5: SQ(n) = 0+1+4+9+16+25 = 55
  - Closed-form expression: 5·6·11/6 = 55
  - Works!
  - Try some more values…
- Problem: we can never prove validity of closed-form solution for all values of n this way since there are an infinite number of values of n.

To solve this problem, let us express SQ(n) in another way.

$$SQ(n) = \underbrace{0^2 + 1^2 + \ldots + (n-1)^2}_{SQ(n-1)} + n^2$$

This leads to the following recursive definition of SQ:

$$SQ(0) = 0$$
$$SQ(n) = SQ(n-1) + n^2, \quad n > 0$$

To get a feel for this definition, let us look at

$$SQ(4) = SQ(3) + 4^2 = SQ(2) + 3^2 + 4^2 = SQ(1) + 2^2 + 3^2 + 4^2$$
$$= SQ(0) + 1^2 + 2^2 + 3^2 + 4^2 = 0 + 1^2 + 2^2 + 3^2 + 4^2$$

Notation for recursive functions

Base case

$$SQ(0) = 0$$
$$SQ(n) = SQ(n-1) + n^2, \quad n > 0$$

Recursive case

Can we show that these two functions are equal?

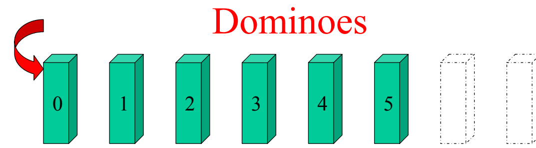
$$SQ_r(0) = 0$$
$$SQ_r(n) = SQ_r(n-1) + n^2, \quad n > 0$$

(r=recursive)

$$SQ_c(n) = n(n+1)(2n+1)/6$$

(c=closed-form)

Dominoes



- Assume equally spaced dominoes, and assume that spacing between dominoes is less than domino length.
- How would you argue that all dominoes would fall?
- Dumb argument:
  - Domino 0 falls because we push it over.
  - Domino 0 hits domino 1, therefore domino 1 falls.
  - Domino 1 hits domino 2, therefore domino 2 falls.
  - Domino 2 hits domino 3, therefore domino 3 falls.
  - ...
- Is there a more compact argument we can make?

Better argument

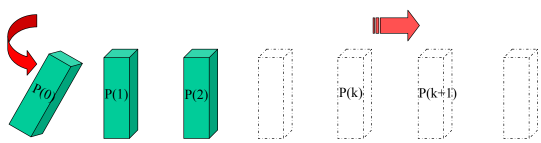
- Argument:
  - Domino 0 falls because we push it over (base case).
  - Assume that domino k falls over (inductive hypothesis).
  - Because domino k's length is larger than inter-domino spacing, it will knock over domino k+1 (inductive step).
  - Because we could have picked any domino to be the k^{th} one, we conclude that all dominoes will fall over (conclusion).
- This is an inductive argument.
- This is called weak induction. There is also strong induction (later).
- Not only is it more compact, but it works for an infinite number of dominoes!

Weak induction over integers

- We want to prove that some property P(n) holds for all integers $n \geq 0$.
- Inductive argument:
  - Base case P(0): Show that property P is true for 0.
  - Inductive step: P(k) implies P(k+1): Assume that P(k) is true for an unspecified integer k (this is the inductive hypothesis). Under this assumption, show that P(k+1) is true.
  - Because we could have picked any k, we can conclude that P(n) holds for all integers $n \geq 0$.

$SQ_r(n) = SQ_c(n)$ for all n?

Define P(n) as $SQ_r(n) = SQ_c(n)$



Prove P(0).

Assume P(k) for unspecified k, and prove P(k+1) under this assumption.

$SQ_r(0) = 0$

$SQ_r(n) = SQ_r(n-1) + n^2, \; n > 0$

$SQ_c(n) = n(n+1)(2n+1)/6$

Let P(n) be the proposition that $SQ_r(n) = SQ_c(n)$.

Proof by induction:

P(0): $SQ_r(0) = 0 = SQ_c(0)$

P(k) => P(k+1): Assume $SQ_r(k) = SQ_c(k)$, prove that $SQ_r(k+1) = SQ_c(k+1)$

$SQ_r(k+1) = SQ_r(k) + (k+1)^2$	(definition of SQ_r)
$= SQ_c(k) + (k+1)^2$	(inductive hypothesis)
$= k(k+1)(2k+1)/6 + (k+1)^2$	(definition of SQ_c)
$= (k+1)(k+2)(2k+3)/6$	(algebra)
$= SQ_c(k+1)$	(definition of SQ_c)

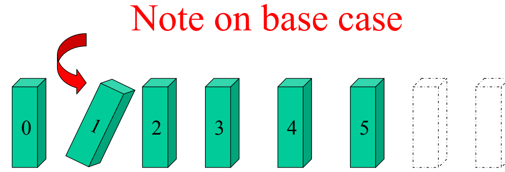
Therefore $SQ_r(n) = SQ_c(n)$ for all n.

Another example

Prove that 0+1+...+n = n(n+1)/2

- Basis n=0:
  - 0 = 0
- Inductive step:
  - Assume 1+2+...+k = k(k+1)/2 for an unspecified k. This is the inductive hypothesis.
  - Under this assumption, show that 1+2+...+(k+1) = (k+1)(k+2)/2.
  - $0 + 1 + \ldots + k + (k+1) = (0 + 1 + \ldots + k) + (k+1)$
    $= k(k+1)/2 + (k+1)$
    $= (k+1)(k+2)/2$
  - Therefore, if result is true for k, it is true for k+1.
- Conclusion: the result holds for all n.

Note on base case



- Sometimes we are interested in showing some proposition is true for integers $\geq b$
- Intuition: we knock over domino b, and dominoes in front get knocked over. Not interested in 0,1,…,(b–1)
- In general, base case in induction does not have to be 0.
- If base case is some integer b, induction proves the proposition for n = b, b+1, b+2, …
- Does not say anything about n = 0,1,…,b–1

Weak induction: nonzero base case

- Sometimes we want to prove that some property P holds for all integers $n \geq b$
- Inductive argument:
  - P(b): show that property P is true for b
  - P(k) => P(k+1): show that if property P is true for k, then it is true for k+1
- We can conclude that P(n) holds for all $n \geq b$
- We don't care about $n < b$ (and in fact, P(n) may not be true for $n < b$!)

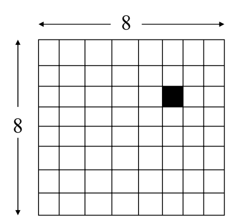
Weak induction: nonzero base case

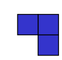
- Example: You can make any amount of postage above 8¢ with some combination of 3¢ and 5¢ stamps.
- Basis: true for 8¢: 8 = 3 + 5
- Induction step: suppose true for k.
  - If used a 5¢ stamp to make k, replace it by two 3¢ stamps. Get k+1.
  - If did not use a 5¢ stamp to make k, must have used at least three 3¢ stamps. Replace three 3¢ stamps by two 5¢ stamps. Get k+1.

More on induction

- In some problems, it may be tricky to determine how to set up the induction:
  - What are the dominoes?
- This is particularly true in geometric problems that can be attacked using induction.

A Tiling Problem



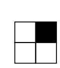


- A chessboard has one square cut out of it. Can the remaining board be tiled using tiles of the shape shown in the picture (rotation allowed)?
- Not obvious that we can use induction!

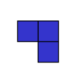
Idea

- Consider boards of size $2^n \times 2^n$ for n = 1,2,…
- Basis: show that tiling is possible for 2 x 2 board.
- Inductive step: assuming $2^k \times 2^k$ board can be tiled, show that $2^{k+1} \times 2^{k+1}$ board can be tiled.
- Conclude that any $2^n \times 2^n$ board can be tiled, n = 1,2,…
- Chessboard (8 x 8) is a special case of this argument. We have proved the 8 x 8 special case by solving a more general problem!

Basis

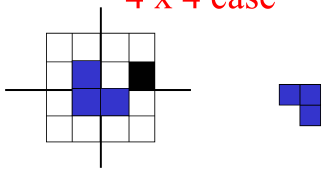


2 x 2 board



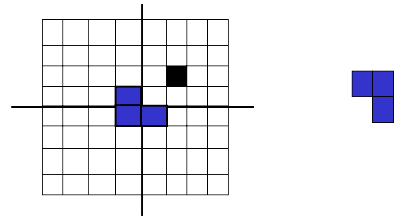
- The 2 x 2 board can be tiled regardless of which one of the four pieces has been omitted

4 x 4 case



- Divide the 4 x 4 board into four 2 x 2 sub-boards.
- One of the four sub-boards has the missing piece.
- By the induction hypothesis, that sub-board can be tiled since it is a 2 x 2 board with a missing piece.
- Tile the center squares of the three remaining sub-boards as shown.
- This leaves 3 2 x 2 boards with a missing piece, which can be tiled by the induction hypothesis.

2^{n+1} x 2^{n+1} case



- Divide board into four sub-boards and tile the center squares of the three complete sub-boards.
- The remaining portions of the sub-boards can be tiled by the assumption about 2^n x 2^n boards.

When induction fails

- Sometimes an inductive proof strategy for some proposition may fail.
- This does not necessarily mean that the proposition is wrong.
  - It may just mean that the inductive strategy you are trying fails.
- A different induction hypothesis (or a different proof strategy altogether) may succeed.

Tiling example (cont.)

- Let us try a different inductive strategy which will fail.
- Proposition: any *n x n* board with one missing square can be tiled.
- Problem: a *3 x 3* board with one missing square has 8 remaining squares, but our tile has 3 squares. Tiling is impossible.
- Therefore, any attempt to give an inductive proof is proposition must fail.
- This does not say anything about the 8x8 case.

Strong induction

- We want to prove that some property P holds for all n.
- Weak induction:
  - P(0): show that property P is true for 0
  - P(k) => P(k+1): show that if property P is true for k, it is true for k+1
  - Conclude that P(n) holds for all n.
- Strong induction:
  - P(0): show that property P is true for 0
  - P(0) and P(1) and … and P(k) => P(k+1): show that if P is true for numbers less than or equal to k, it is true for k+1
  - Conclude that P(n) holds for all n.
- Both proof techniques are equally powerful.

Conclusion

- Induction is a powerful proof technique
- Recursion is a powerful programming technique
- Induction and recursion are closely related. We can use induction to prove correctness and complexity results about recursive programs. Examples next time!