CS/ENGRD 2110
Object-Oriented Programming and Data Structures

Spring 2012

Thorsten Joachims

Lecture 22: 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)
- They are very closely related
- Induction can be used to establish the *correctness* and *complexity* of programs

Defining Functions

- It is often useful to describe a function in different ways
  - Let S : int → int be the function where S(n) is the sum of the integers from 0 to n. For example,
    - S(0) = 0
    - S(3) = 0+1+2+3 = 6
  - Definition: iterative form
    - S(n) = 0+1+ … + n
  - Another characterization: closed form
    - S(n) = n(n+1)/2

Sum of Squares

- A more complex example
  - Let SQ : int → int be the function that gives the sum of the squares of integers from 0 to n:
    - SQ(0) = 0
    - SQ(3) = $0^2 + 1^2 + 2^2 + 3^2 = 14$
  - Definition (iterative form):
    - SQ(n) = $0^2 + 1^2 + \ldots + n^2$
  - Is there an equivalent closed-form expression?

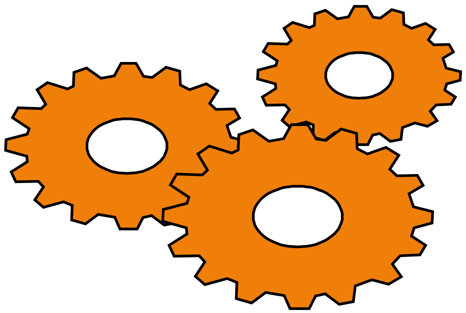
Closed-Form Expression for SQ(n)

- Idea:
  - Sum of integers between 0 through n was n(n+1)/2 which is a quadratic in n (that is, $O(n^2)$)
  - Inspired guess: perhaps sum of squares of integers between 0 through n is a cubic in n

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

Finding Coefficients

- Solve:
  - $SQ(n) = 0^2+1^2+\ldots+n^2 = an^3+bn^2+cn+d$
- Use n = 0, 1, 2, 3
  - SQ(0) = 0 = $a\cdot 0 + b\cdot 0 + c\cdot 0 + d$
  - SQ(1) = 1 = $a\cdot 1 + b\cdot 1 + c\cdot 1 + d$
  - SQ(2) = 5 = $a\cdot 8 + b\cdot 4 + c\cdot 2 + d$
  - SQ(3) = 14 = $a\cdot 27 + b\cdot 9 + c\cdot 3 + d$



  - Solve these 4 equations to get
    - $a = 1/3$ $b = 1/2$ $c = 1/6$ $d = 0$

Is the Formula Correct?

- This suggests

$$\begin{aligned} SQ(n) &= 0^2 + 1^2 + \dots + n^2 \\ &= n^3/3 + n^2/2 + n/6 \\ &= n(n+1)(2n+1)/6 \end{aligned}$$

- Question: Is this closed-form solution true for all n?
  - Remember, we only used n = 0,1,2,3 to determine these coefficients
  - Need to show 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…

→We can never prove validity of the closed-form solution for all values of n this way, since there is an infinite number of values of n

A Recursive Definition

- To solve this problem, let's express SQ(n) in a different way:
  - $SQ(n) = \boxed{0^2 + 1^2 + \ldots + (n-1)^2} + n^2$
  - The part in the box is just SQ(n-1)
- This leads to the following *recursive* definition
  - $SQ(0) = 0$ ← Base Case
  - $SQ(n) = SQ(n-1) + n^2, \; n > 0$ ← Recursive Case
- Thus,
  - $SQ(4) = SQ(3) + 4^2 = SQ(2) + 3^2 + 4^2 = SQ(1) + 2^2 + 3^2 + 4^2 = SQ(0) + 1^2 + 2^2 + 3^2 + 4^2 = 0 + 1^2 + 2^2 + 3^2 + 4^2$

Are These Two Functions Equal?

- SQ_r (r = recursive):

  $SQ_r(0) = 0$

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

- SQ_c (c = closed-form):

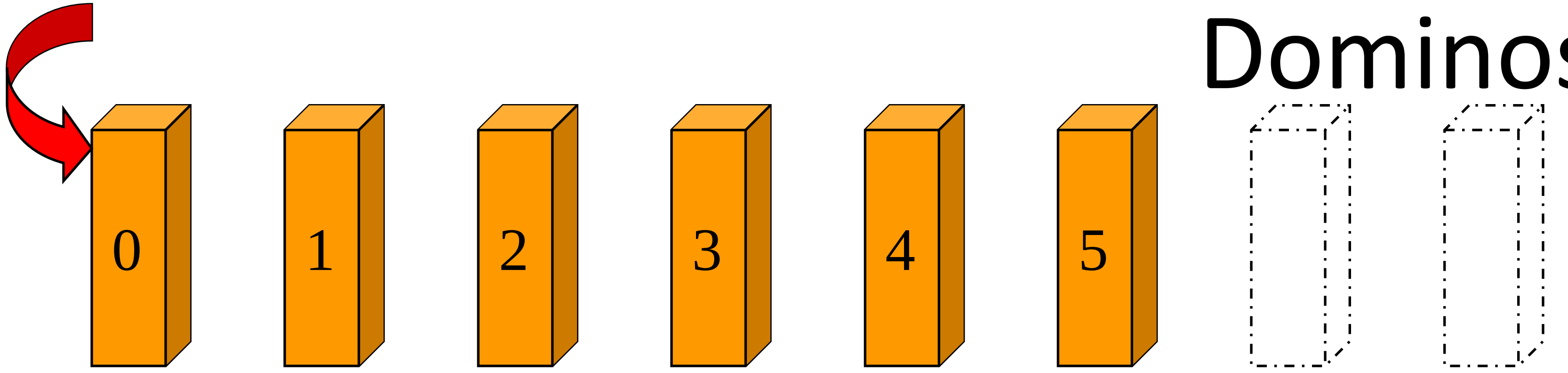
  $SQ_c(n) = n(n+1)(2n+1)/6$

Induction over Integers

- To prove that some property P(n) holds for all integers $n \geq 0$,
  - Basis: Show that P(0) is true
  - Induction Step: Assuming that P(k) is true for an unspecified integer k, show that P(k+1) is true

- Conclusion: Because we could have picked any k, we conclude that P(n) holds for all integers $n \geq 0$

Dominos



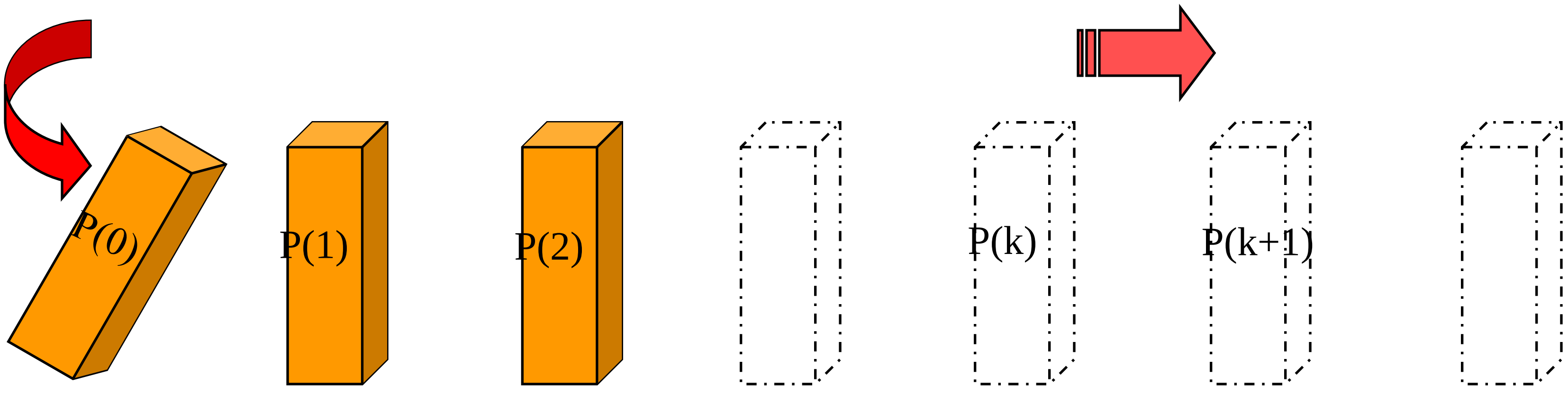
- Assume equally spaced dominos, and assume that spacing between dominos is less than domino length
- How would you argue that all dominos would fall?
- Dumb argument:
  - Domino 0 falls because we push it over
  - Domino 0 hits domino 1, therefore domino 1 falls
  - Domino 1 hits domino 2, therefore 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 or Basis)
  – Assume that domino k falls over (Induction 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 version is called *weak induction*
  – There is also *strong induction* (later)
- Not only is this argument more compact, it works for an arbitrary number of dominoes!

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

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



- Prove P(0)
- Assume P(k) for unspecified k, and then prove P(k+1) under this assumption

Proof (by Induction)

- Recall:

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

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

- Let P(n) be the proposition that $SQ_r(n) = SQ_c(n)$
- Basis "P(0) true": holds because $SQ_r(0) = 0$ and $SQ_c(0) = 0$
- Induction Hypothesis "P(k) assumed": Assume $SQ_r(k) = SQ_c(k)$
- Inductive Step: "P(k) → P(k+1) true"

  $SQ_r(k+1) = SQ_r(k) + (k+1)^2$ by definition of $SQ_r(k+1)$
  $= SQ_c(k) + (k+1)^2$ by the Induction Hyp.
  $= k(k+1)(2k+1)/6 + (k+1)^2$ by definition of $SQ_c(k)$
  $= (k+1)(k+2)(2k+3)/6$ algebra
  $= SQ_c(k+1)$ by definition of $SQ_c(k+1)$

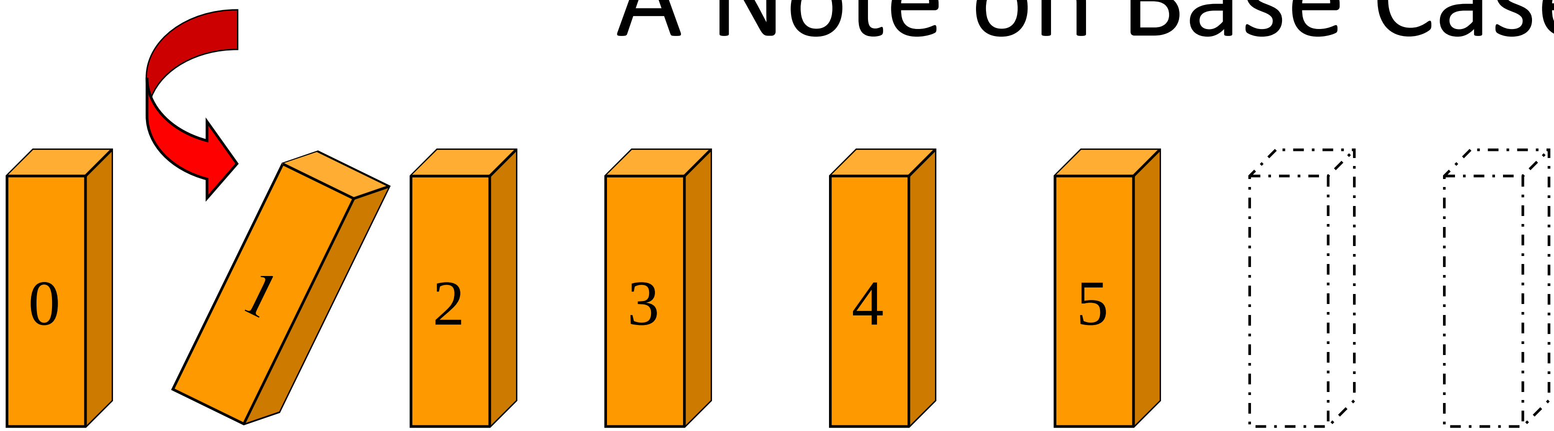
- Conclusion "forall $k \geq 0$: P(k) true":

  $SQ_r(n) = SQ_c(n)$ for all $n \geq 0$

Another Example

- Prove: P(n) ⇔ [0+1+...+n = n(n+1)/2]
- Basis "P(0) true": Obviously holds for n = 0
- Induction Hypothesis "P(k) assumed":
  Assume 0+1+…+k = k(k+1)/2
- Inductive Step "P(k) → P(k+1) true":
  0+1+…+(k+1) = [0+1+…+k] + (k+1)   by def
  = k(k+1)/2 + (k+1)   by I.H.
  = (k+1)(k+2)/2   algebra
- Conclusion "forall k ≥ 0: P(k) true":
  0+1+…+n = n(n+1)/2 for all n ≥ 0

A Note on Base Cases



- In general, the base case in induction does not have to be 0
  - Sometimes we are interested in showing some proposition is true for integers ≥ b
  - Intuition: we knock over domino b, and dominoes in front get knocked over; not interested in 0,1,…,(b-1)
- 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

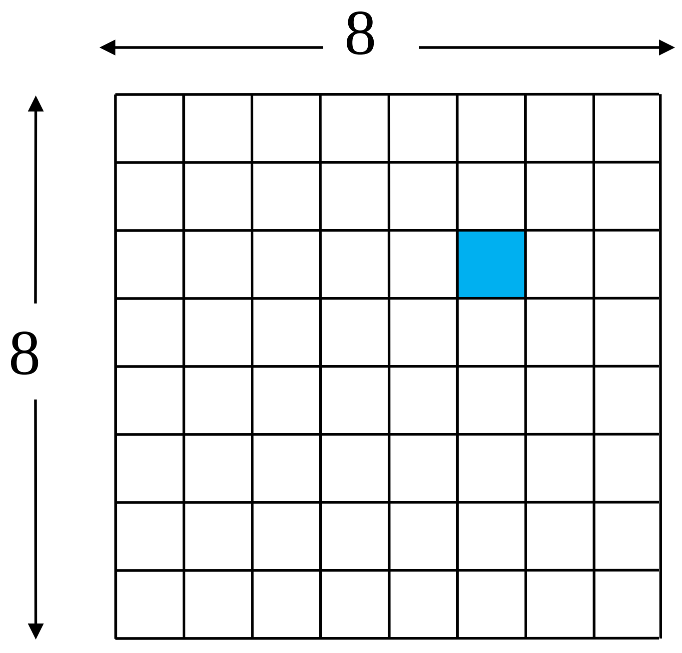
Weak Induction: Nonzero Base Case

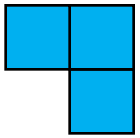
- Claim: You can make any amount of postage above 8¢ with some combination of 3¢ and 5¢ stamps
- Basis: True for 8¢: 8 = 3 + 5
- Induction Hypothesis: Suppose true for some $k \geq 8$
- Inductive Step:
  - If used a 5¢ stamp to make k, replace it by two 3¢ stamps. Get k+1.
  - If did not use a 5¢ stamp to make k, must have used at least three 3¢ stamps. Replace three 3¢ stamps by two 5¢ stamps. Get k+1.
- Conclusion: Any amount of postage above 8¢ can be made with some combination of 3¢ and 5¢ stamps

What are the “Dominos”?

- In some problems, it can be tricky to determine how to set up the induction
- This is particularly true for geometric problems that can be attacked using induction

A Tiling Problem



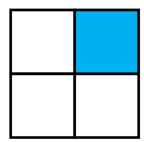


- A chessboard has one square cut out of it somewhere
- Can the remaining board be tiled using tiles of the shape shown in the picture (rotation allowed)?
- Not obvious that we can use induction!

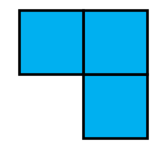
Proof Outline

- Consider boards of size 2^n x 2^n for n = 1,2,…
- Basis: Show that tiling is possible for 2 x 2 board
- Induction Hypothesis:
  Assume the 2^k x 2^k board can be tiled
- Inductive Step:
  Using I.H. show that the 2^{k+1} x 2^{k+1} board can be tiled
- Conclusion: Any 2^n x 2^n board can be tiled, n = 1,2,…
  - Our chessboard (8 x 8) is a special case of this argument
  - We will have proven the 8 x 8 special case by solving a more general problem!

Basis

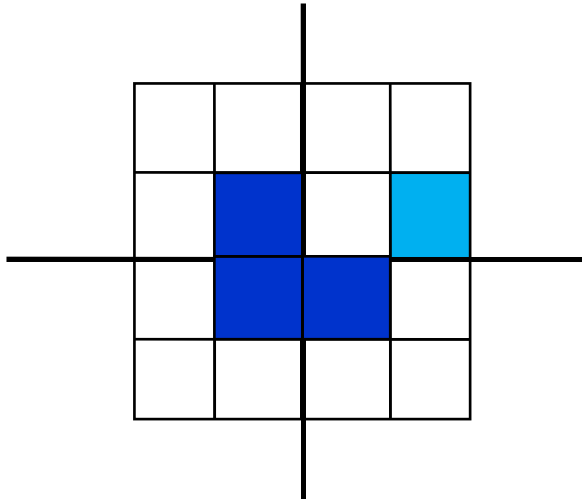


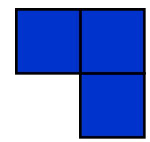
2 x 2 board



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

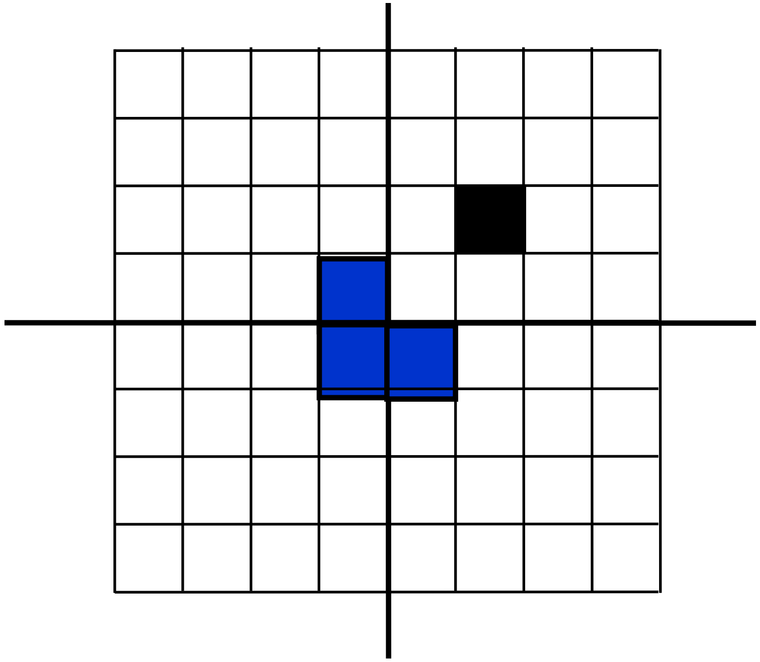
4 x 4 Case

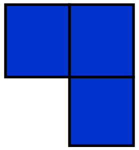




- Divide the 4 x 4 board into four 2 x 2 sub-boards
- One of the four sub-boards has the missing piece
  - By the I.H., that sub-board can be tiled since it is a 2 x 2 board with a missing piece somewhere
- Tile the center squares of the three remaining sub-boards as shown
  - This leaves three 2 x 2 boards, each with a missing piece
  - We know these can be tiled by the Induction Hypothesis

2^{k+1} x 2^{k+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 I.H. (which assumes we can tile 2^k x 2^k 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 particular inductive strategy you are using is the wrong choice

- A different induction hypothesis (or a different proof strategy altogether) may succeed

Tiling Example (Poor Strategy)

- Let's try a different induction strategy
- 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
- Thus, any attempt to give an inductive proof of this proposition *must fail*

- Note that this failed proof does not tell us anything about the 8 x 8 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