CSE373: Data Structures and Algorithms

Lecture 2b: Proof by Induction and Powers of Two

Nicki Dell

Spring 2014

Mathematical induction

Suppose *P(n)* is some statement (mentioning integer *n*)

Example: $n \geq n/2 + 1$

We can use induction to prove P(n) for all integers $n \geq n_0$.

We need to

1. Prove the "base case" i.e. $P(n_0)$. For us n_0 is usually 1.
2. Assume the statement holds for P(k).
3. Prove the "inductive case" i.e. if P(k) is true, then P(k+1) is true.

Why we will care:

To show an algorithm is correct or has a certain running time *no matter how big a data structure or input value is*

(Our "*n*" will be the data structure or input size.)

Example

P(n) = “the sum of the first *n* powers of 2 (starting at 0) is 2^n-1”

Theorem: *P(n)* holds for all $n \geq 1$
Proof: By induction on *n*

- Base case: *n*=1. Sum of first 1 power of 2 is 2^0 , which equals 1.
  And for *n*=1, 2^n-1 equals 1.
- Inductive case:
  - Assume the sum of the first *k* powers of 2 is 2^k-1
  - Show the sum of the first (*k*+1) powers of 2 is 2^{k+1}-1

  Using assumption, sum of the first (*k*+1) powers of 2 is
  $(2^k-1) + 2^{(k+1)-1} = (2^k-1) + 2^k = 2^{k+1}-1$

Example

n	1	2	3	4
sum of first n powers of 2	$2^0 = 1$	$1 + 2^1 = 3$	$3 + 2^2 = 7$	$7 + 2^3 = 15$
P(n)	$2^1 - 1 = 1$			

Example

n	1	2	3	4
sum of first n powers of 2	$2^0 = 1$	$1 + 2^1 = 3$	$3 + 2^2 = 7$	$7 + 2^3 = 15$
P(n)	$2^1 - 1 = 1$	$2^2 - 1 = 3$		

Example

n	1	2	3	4
sum of first n powers of 2	$2^0 = 1$	$1 + 2^1 = 3$	$3 + 2^2 = 7$	$7 + 2^3 = 15$
P(n)	$2^1 - 1 = 1$	$2^2 - 1 = 3$	$2^3 - 1 = 7$	

Example

n	1	2	3	4	
sum of first n powers of 2	$2^0 = 1$	$1 + 2^1 = 3$	$3 + 2^2 = 7$	$7 + 2^3 = 15$	…
P(n)	$2^1 - 1 = 1$	$2^2 - 1 = 3$	$2^3 - 1 = 7$	$2^4 - 1 = 15$	

Powers of 2

- A bit is 0 or 1 (just two different "letters" or "symbols")
- A sequence of *n* bits can represent 2^n distinct things
  - For example, the numbers 0 through 2^n-1
- 2^{10} is 1024 ("about a thousand", kilo in CSE speak)
- 2^{20} is "about a million", mega in CSE speak
- 2^{30} is "about a billion", giga in CSE speak

Java: an **`int`** is 32 bits and signed, so "max int" is "about 2 billion"

a **`long`** is 64 bits and signed, so "max long" is 2^{63}-1

Therefore…

Could give a unique id to…

- Every person in the U.S. with 29 bits
- Every person in the world with 33 bits
- Every person to have ever lived with 38 bits (estimate)
- Every atom in the universe with 250-300 bits

So if a password is 128 bits long and randomly generated, do you think you could guess it?