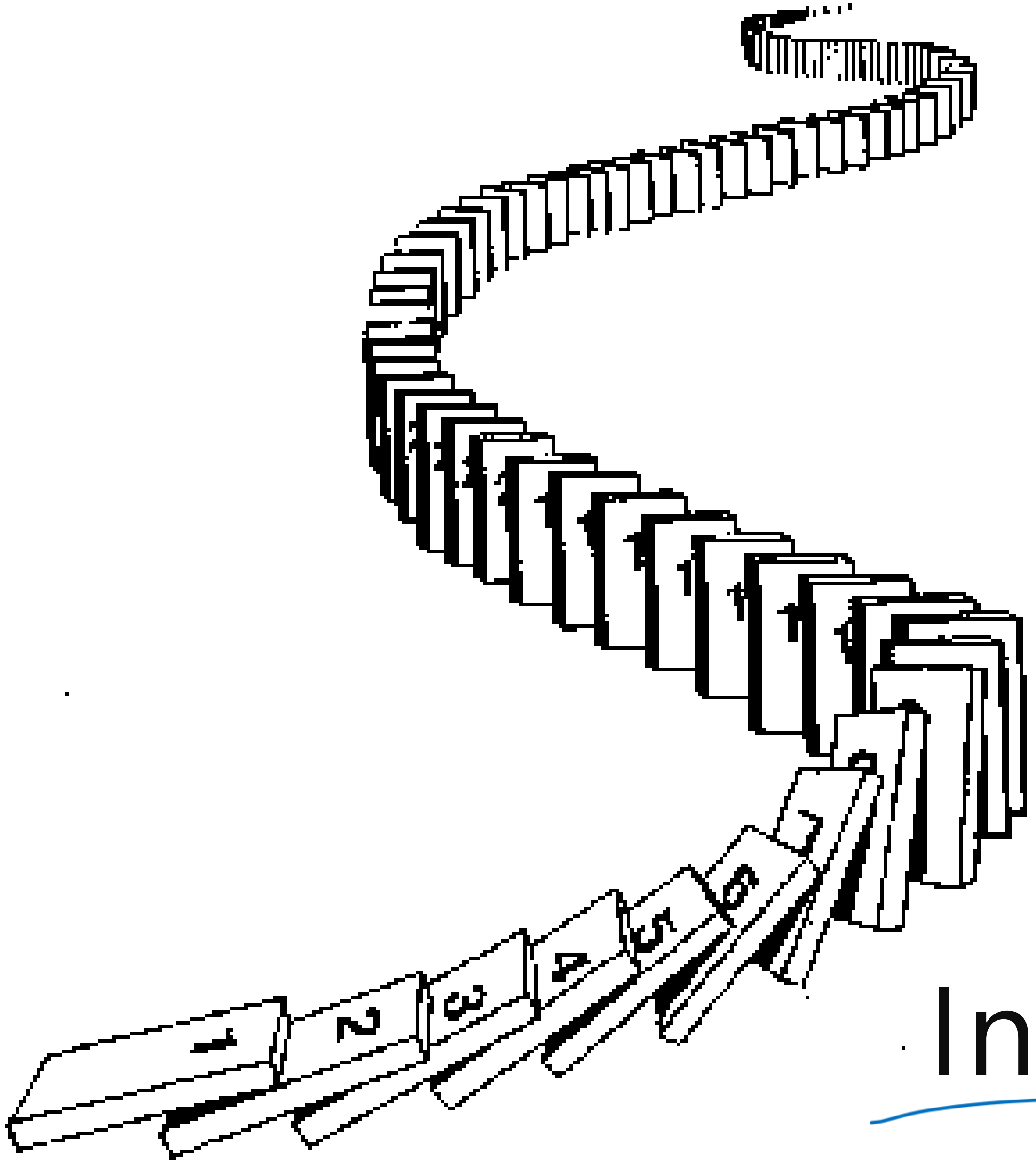

Induction

CSE 311 Autumn 2024
Lecture 15

How do we know recursion works?

```
//Assume i is a nonnegative integer
//returns 2^i.
public int CalculatesTwoToTheI(int i){
    if(i == 0)
        return 1;
    else
        return 2*CaclulatesTwoToTheI(i-1);
}
```

Why does `CalculatesTwoToTheI(4)` calculate `2^4`?

Convince the people around you!

How do we know recursion works?

Something like this:

Well, as long as `CalculatesTwoToTheI(3) = 8`, we get 16...

Which happens as long as `CalculatesTwoToTheI(2) = 4`

Which happens as long as `CalculatesTwoToTheI(1) = 2`

Which happens as long as `CalculatesTwoToTheI(0) = 1`

And it is! Because that’s what the base case says.

How do we know recursion works?

There's really only two cases.

The Base Case is Correct

`CalculatesTwoToTheI(0) = 1 (which it should!)`

And that means `CalculatesTwoToTheI(1) = 2`, (like it should)

And that means `CalculatesTwoToTheI(2) = 4`, (like it should)

And that means `CalculatesTwoToTheI(3) = 8`, (like it should)

And that means `CalculatesTwoToTheI(4) = 16`, (like it should)

IF the recursive call we make is correct
THEN our value is correct.

How do we know recursion works?

The code has two big cases,

So our proof had two big cases

"The base case of the code produces the correct output"

"IF the calls we rely on produce the correct output THEN the current call produces the right output"

A bit more formally…

"The base case of the code produces the correct output"

"IF the calls we rely on produce the correct output THEN the current call produces the right output"

Let $P(i)$ be "`CalculatesTwoToTheI(i)` returns 2^i."

How do we know $P(4)$?

$P(0)$ is true.

And $P(0) \to P(1)$, so $P(1)$.

And $P(1) \to P(2)$, so $P(2)$.

And $P(2) \to P(3)$, so $P(3)$.

And $P(3) \to P(4)$, so $P(4)$.

A bit more formally…

This works alright for $P(4)$.

What about $P(1000)$? $P(1000000000)$?

At this point, we'd need to show that implication $P(k) \rightarrow P(k+1)$ for A BUNCH of values of k.

But the code is the same each time.

And so was the argument!

We should instead show $\forall k[P(k) \rightarrow P(k+1)]$.

Induction

Your new favorite proof technique!

How do we show $\forall n, P(n)$?

Show $P(0)$

Show $\forall k(P(k) \rightarrow P(k+1))$

Induction

```
//Assume i is a nonnegative integer
public int CalculatesTwoToTheI(int i){
    if(i == 0)
        return 1;
    else
        return 2*CaclulatesTwoToTheI(i-1);
}
```

P(0)

Let $P(n)$ be "`CalculatesTwoToTheI(n)` returns 2^n."

Note that if the input n is 0, then the if-statement evaluates to true, and $1 = 2^\wedge 0$ is returned, so $P(0)$ is true.

Suppose $P(k)$ holds for an arbitrary $k \geq 0$.

Consider the code run on $k+1$. Since $k \geq 0$, $k+1 \geq 1$ and we are in the else branch. By inductive hypothesis, `CalculatesTwoToTheI`(k) returns 2^k, so the code run on $k+1$ returns $2 \cdot 2^k = 2^{k+1}$.

So $P(k+1)$ holds.

Therefore $P(n)$ holds for all $n \geq 0$ by the principle of induction.

Making Induction Proofs Pretty

1

Let $P(n)$ be the predicate "`CalculatesTwoToTheI(n)` returns 2^n." We prove $P(n)$ holds holds for all natural numbers n by induction on n.

Base Case $(\mathrm{n} = 0)$ Note that if the input n is 0, then the if-statement evaluates to true, and $1 = 2\string^0$ is returned, so $P(0)$ is true. 2

Inductive Hypothesis: Suppose $P(k)$ holds for an arbitrary $k \geq 0$. 3

Inductive Step: Since $k \geq 0, k + 1 \geq 1$, so the code goes to the recursive case. We will return $2 \cdot$ `CalculatesTwoToTheI(k)`. By Inductive Hypothesis,

4

`CalculatesTwoToTheI(k)` $= 2^k$. Thus we return $2 \cdot 2^k = 2^{k+1}$.

So $P(k+1)$ holds.

Therefore $P(n)$ holds for all $n \geq 0$ by the principle of induction. 5

Making Induction Proofs Pretty

All of our induction proofs will come in 5 easy(?) steps!

1. Define $P(n)$. State that your proof is by induction on n.
2. Show $P(0)$ i.e. show the base case
3. Suppose $P(k)$ for an arbitrary k.
4. Show $P(k+1)$ (i.e. get $P(k) \rightarrow P(k+1)$)
5. Conclude by saying $P(n)$ is true for all n by induction.

Some Other Notes

Always state where you use the inductive hypothesis when you're using it in the inductive step.

It's usually the key step, and the reader really needs to focus on it.

Be careful about what values you're assuming the Inductive Hypothesis for – the smallest possible value of k should assume the base case but nothing more.

The Principle of Induction (formally)

Principle of Induction	If you know: $P(0); \forall k(P(k) \rightarrow P(k+1))$
	You can conclude: $\forall n(P(n))$

Informally: if you knock over one domino, and every domino knocks over the next one, then all your dominoes fell over.

More induction!

More Induction

Induction doesn't **only** work for code!

Show that $\sum_{i=0}^{n} 2^i = 1 + 2 + 4 + \cdots + 2^n = 2^{n+1} - 1$.

More Induction

Induction doesn't **only** work for code!

Show that $\sum_{i=0}^{n} 2^i = 1 + 2 + 4 + \cdots + 2^n = 2^{n+1} - 1.$

Let $P(n) =$ "$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1.$"

We show $P(n)$ holds for all natural numbers n by induction on n.

Base Case ($n=0$) $\sum_{i=0}^{0} 2^i = 2^0 = 1 = 2-1 = 2^{0+1}-1$

Inductive Hypothesis:

Inductive Step: $\sum_{i=0}^{k+1} 2^i = \left(\sum_{i=0}^{k} 2^i\right) + 2^{k+1}$

$= (2^{k+1} - 1) + 2^{k+1} = 2^{k+2} - 1$

$P(n)$ holds for all $n \geq 0$ by the principle of induction.

More Induction

Induction doesn't **only** work for code!

Show that $\sum_{i=0}^{n} 2^i = 1 + 2 + 4 + \cdots + 2^n = 2^{n+1} - 1$.

Let $P(n) =$"$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$."

We show $P(n)$ holds for all natural numbers n by induction on n.

Base Case $(n = 0)$ $\sum_{i=0}^{0} 2^i = 1 = 2 - 1 = 2^{0+1} - 1$.

Inductive Hypothesis: Suppose $P(k)$ holds for an arbitrary $k \geq 0$.

Inductive Step: We show $P(k+1)$. Consider the summation $\sum_{i=0}^{k+1} 2^i = 2^{k+1} + \sum_{i=0}^{k} 2^i = 2^{k+1} + 2^{k+1} - 1$, where the last step is by IH.

Simplifying, we get: $\sum_{i=0}^{k+1} 2^i = 2^{k+1} + 2^{k+1} - 1 = 2 \cdot 2^{k+1} - 1 = 2^{(k+1)+1} - 1$.

$P(n)$ holds for all $n \geq 0$ by the principle of induction.

Algebra Block Formatting

In an English proof it's very common to have algebra broken out (instead of separate sentences); this is often easier to read.

Inductive Step: We show $P(k+1)$. Consider the summation

$$\begin{aligned}\sum_{i=0}^{k+1} 2^i &= 2^{\mathrm{k+1}} + \sum_{i=0}^{k} 2^i \text{ breaking off final term}\\ &= 2^{k+1} + 2^{k+1} - 1 \text{ by IH}\\ &= 2 \cdot 2^{k+1} - 1 \text{ combining two copies of } 2^{k+1}\\ &= 2^{(k+1)+1} - 1.\end{aligned}$$

$P(n)$ holds for all $n \geq 0$ by the principle of induction.