22 Jan 2026

Induction

1. Set things up (Base Case)
2. Base case, n_0 ("easy proof") usu., plug in #s & the eq'n or ineq. follows
1.A.3 Let $k \geq n_0$. Assume ...
4. Prove the stmt ~~holds~~ for $n = k+1$. (inductive step)
5. Conclude

More concretely,

Claim: $\forall n \geq n_0 \in \mathbb{N}$, stmt $S(n)$ = true ~~holds~~.

↑ a sentence that is either true or false

Proof: We proceed by induction.

For the base case, let $n = n_0$ ↑ Maybe your proof never explicitly uses the variable n_0

[[plug in n_0 into $S(n)$ and hopefully we find that yes, this is a true stmt]]

Let $k \geq n_0$. Assume $S(k)$ holds.

For the inductive step, we will show that $S(k+1)$ holds.

[[Prove, but start by choosing an arbitrary $k+1$ case. e.g., Let G be a graph with $k+1$ vertices.]]

□ (1)

Ch 0 of Jeff Erickson's Algorithms

Observations

- uses pseudocode, not code
  → more general, more readable
- emphasis on big-O for runtime (and Θ, if possible!)
  → trends in runtimes
- 4 components of analyzing an algorithm
  1. What? What is the problem we're solving?
  2. Why? Proof of correctness. Why does the algo do what it claims it should do.
  3. How? Give algo + explain how it works.
  4. How Fast? Runtime
  * Space complexity
    primitive complexity (moves away from real-RAM)

```
mult (x, y)
1:    ans ← x
2:    ans ← ans · y
3:    return ans
```

Before I start	P (preconditions) is true
after L-1	S_1 is true
after L2	S_2 is true
at. L3	S_3 is true (last one)

"My algo is correct"
"My algo did what it said it would do"

What? This alg will mult. two numbers (input as x, y)

Why does this work?

In line 1, ans is x. S_1 $(P \Rightarrow S_1)$

Then, in line 2, my ans is ans · y, $(S_1 \Rightarrow S_2)$

which means ans = x · y

Finally, we returned ans = x · y in Line 3. $(S_2 \Rightarrow S_3)$ □

How fast? $\Theta(1)$

- Line 1: 3 operations

  Locate x
  Locate ans
  Set ans to x

  $\Theta(1)$ step

- Line 2:

  Locate y
  Locate x
  Multiply
  Set ans

  $\Theta(1)$

↑ These are const time ops by our Real-RAM MOC choice

- Line 3: $\Theta(1)$

My total time is: $\Theta(1) + \Theta(1) + \Theta(1) = \Theta(1+1+1)$ (1+1+1 = 3)

$= \Theta(1)$

Let's Analyze a counting song!

"1 little dinosaur
2 little dinosaur
⋮
n little dinosaurs"

"n bottles of beer on the wall
n-1 bottles of beer on the wall
⋮
0 bottles of beer on the wall

(1) Write as an algorithm
(2) Analyze runtime

Sing song (int n)

```
1:  for i = n; i ≥ 0
2:      print/sing "i bottles of beer on the wall"
3:      i ← i - 1
```

What is the cost of Line 3? $\Theta(1)$, b/c 3 operations.
How many times does Line 3 run? n times
In total, what is the cost associated w/ all calls to Line 3? $\Theta(n-1) = \Theta(n)$

($n-1$: # times we see it; cost per)

This song is $\Theta(n)$!

Why Base case: brute force 0.
Assume works for some $k \geq 0 \in \mathbb{N}$.
Let $n = k+1$. Walk through algo, when we get to Line 3's return, use I.A.

$n + (n-1) + (n-2) + \dots + 0$
$\sum_{i=n}^{0} i = \frac{n(n-1)}{2} = \Theta(n^2)$