What does it take to prove an algo is correct?

① Termination: The algorithm terminates T
  (a) give asymptotic analysis
  (b) decrementing fcn ← we'll get to this

② Partial Correctness: If the algorithm terminates, then it is correct.
  → usu. through loop / recursion invariant (we'll get to this)

$$\frac{T \Rightarrow C}{\therefore C}$$

(modus ponens!)

```
        FIND (x, A)
Θ(1)      i ← 1
          while i ≤ |A|
Θ(1)        if x = A[i]
Θ(1)  ×n      return TRUE
Θ(1)        end if
Θ(1)        i++
Θ(1)      end while
Θ(1)      return FALSE
```

Runtime

$n = |A|$

Runtime is

$$\underbrace{\Theta(1)}_{\text{before loop}} + \underbrace{n \cdot \Theta(1)}_{\text{loop}} + \underbrace{\Theta(1)}_{\text{after the loop}}$$

$$= \Theta(n) + 2\,\Theta(1)$$

$$= \boxed{\Theta(n)}$$

Recursive Runtimes

① Binary Search an array of size $n \rightarrow$ runtime $T(n)$

Runtime: 1: check middle $\Theta(1)$

2: recursively call on half $T(n/2)$

3: Use recursive ans as my own $\Theta(1)$

$$\boxed{T(n) = T(n/2) + \Theta(1)}$$

Let's be a bit specific for now.
Suppose steps 1 & 3 take d units of time.
↑ a constant

Suppose $T(1) = c$.

$$\boxed{T(n) = \begin{cases} c, & n = 1 \\ T(n/2) + d, & n > 1 \end{cases}}$$

both are Recursive Form

note: when using Θ's in recursive form, can ignore base case b/c always $\Theta(1)$!

Recursion Tree:
(Assume n is a power of 2)

$T(n)$ d
|
$T(n/2)$ d
|
$T(n/4)$ d
|
$T(n/8)$ d
⋮
$T(2)$ d
|
$T(1)$ c

how many levels?
$\log_2 n + 1$,
all of which except the last "cost" me d.
∴ total RT / cost is

$$\boxed{d \cdot \log_2 n + c}$$

$$\boxed{T(n) = d \log_2 n + c}$$

↓
closed form
"exact formula"

Asymptotic Form:
$T(n) \in \Theta(\log n)$

2 Recurrence Relation:

$$T(n) = \begin{cases} 1 & , n = 1 \\ T(n-1) + n & , n > 1 \end{cases}$$

Questions

a) What is the closed form? $T(n) = \sum_{i=1}^{n} i$ ← Summation form

$$\boxed{T(n) = \frac{n(n+1)}{2}}$$

b) Asymptotic Form? $\Theta(n^2)$

c) Name an algorithm with the recurrence $T(n) = T(n-1) + \Theta(n)$.

↳ selection sort

<u>examples</u>

$n = 3$

with recurrence relation:

$$\begin{aligned} T(3) &= T(2) + 3 && \text{via recursion} \\ &= (T(1) + 2) + 3 && \text{via base case} \\ &= (1 + 2) + 3 \\ &= 1 + 2 + 3 = 6 \\ &= \sum_{i=1}^{3} i \end{aligned}$$

with closed form: $T(3) = \frac{3(4)}{2} = 3 \cdot 2 = 6$

with recursion tree

$\cancel{T(3)}\ 3$
|
$\cancel{T(2)}\ 2$
|
$\cancel{T(1)}\ 1$

Sum values on my nodes

$$\sum_{i=1}^{n} i = \sum_{i=1}^{n} n - (i-1)$$

Solving Recurrence Relations

If we have no better option:

Guess + Check!

Guess:
- → gut feeling
- → top-down: unravel via the recursion tree
- → bottom-up: start w/ small #s + see if a pattern is found.

Check: → induction, either for closed form or asymptotic.