CS157 Lecture #10

ROUNDING `c = myconv(a,b)` ← $c \approx conv$

c may be off by 10^{-12} → fix by rounding!

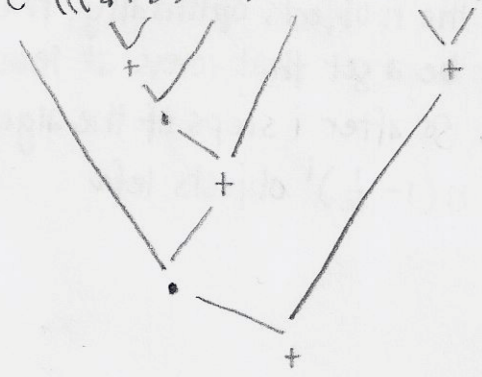
```
c = round(c)
    n = length(c)
    for i = n:-1:2
        c(i-1) = c(i-1) + ⌊c(i)/10⌋
        c(i) = c(i) mod 10
```

Q: How many times do we carry the leftmost digit?

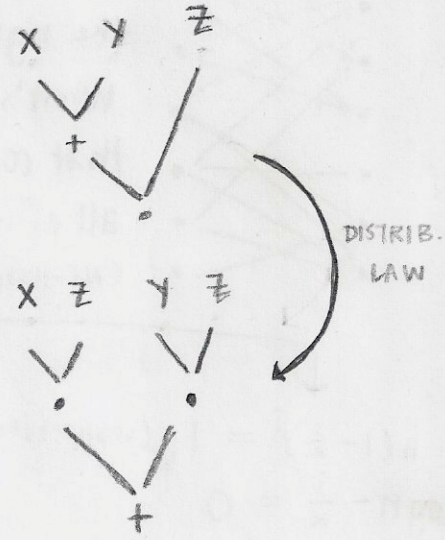
2 big #'s multiplying together, but both bounded by 1000... (1 more digit each) so the product of this bound has 2 extra digits, which means our number can have only 1 extra digit, so we'll only have to carry the leftmost digit at most once

DISTRIBUTIVE LAW: $a \cdot (b+c) = a \cdot b + b \cdot c$

$e \cdot (((a+b) \cdot c) + d) + (f+g)$



GOAL: use the distributive law to get a tree with only one addition (at the root)



* How do we know that this terminates?

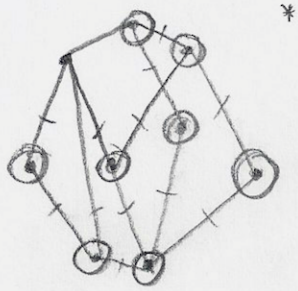
Claim 1: If it works for a,b,c, it works for all z's (or any number ≠ 1)

Claim 2: The number of leaves always increases

Claim 3: The evaluation of the expression must remain constant, so this evaluation must give an upper bound for the number of leaves

But we said the number of leaves always increases, so this upper bound gives a contradiction ⇒ the algorithm must terminate in finite time

VERTEX COVER: Graph $G=(V,E)$. Want to choose a set of vertices $V' \subset V$ such that every edge in E is connected to a vertex in V'



*How do we do this optimally?

↓

NP-HARD

(but there's a greedy algorithm that gets us close)

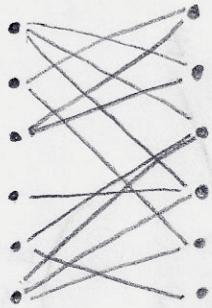
Idea #1: Swap edges / vertices : choose the set of edges that covers all vertices

ALGO ① Find edges such that every edge is adjacent to the ones that you've picked (pick edges not adjacent to edges we've already picked until we can't)
② Take both endpoints of the chosen edges as our vertex cover

* Does it approximate the optimal solution (within a factor of 2)?

↳ Consider a chosen edge. The optimal solution has to cover this edge, and the vertices that cover this edge are the endpoints, so the optimal solution must pick at least one of these endpoints. We choose exactly 2, so we must be within a factor of 2 of the optimal solution.

SET COVER: Each thing on the left is a "set", each thing on the right is an "object". What's the fewest # of sets that collectively contain all of the objects? (NP-Hard, even to approximate)



Idea #1: Choose the set with the most elements that hasn't been chosen yet

* If you are told that there are k sets that cover the n objects optimally, then there must be a set that covers at least $\frac{n}{k}$ objects. So after i steps of the algo, there are $n(1-\frac{1}{k})^i$ objects left

$n(1-\frac{1}{k})^i = 1$ (want 1 item left)

$\log n - \frac{i}{k} = 0$

$i = k \log n$

$\Rightarrow \log n$ approximation