AVL Trees 1

CptS 223 - Fall 2017 - Aaron Crandall

Today’s Agenda

- Announcements
- Getting into AVL trees

Special Office hours: Tuesday from 10-12

Today’s office hours: vomiting 6 year old instead

Announcements

- MA3 is up Due next Sunday @ 11:59pm
- PA2 will be out tomorrow, due Oct 1st at 11:59pm
- Remember HW1 is also there - it shouldn't take incredibly long if you chip away at it, but the algorithm design questions and the time calculation ones can be tricky
- Disney's Matt Estes is here tomorrow: Sept 19th @4-5pm in the Chinook Events Room
  - https://vcea.wsu.edu/event/walt-disney-co-wsu-the-magic-behind-disney/
  - Other event is a drop in for coffee, jobs & internship talk with Matt Estes: Sept 20th 9:30-12:00 in Murrow 218 Conference Room: https://goo.gl/sp9EKA
  - Brown bag lunch with Matt Estes, Sept 20th @ 12-1pm in CUB 208: https://goo.gl/YnBFu4
- Career Fair Prep #4 – Workshop: Networking/Social Media - Sept 20, 4:10-5:30pm, Sloan 169
  - https://goo.gl/d92pmy

Reddit brings down North Korea's entire internet after links to country's 28 websites are posted online

11:00, 21 SEP 2016 | **UPDATED** 11:02, 21 SEP 2016 | **BY** JEFF PARSONS

All 28 websites on North Korea's version of the internet were opened to outsiders, causing a surge of demand that flooded the state's systems

SHARE COMMENTS

Enter your e-mail for our daily newsletter Subscribe

★ Recommended In Technology

ANDROID

Google Pixel and Pixel XL: Specs, price, screen size, news and rumours about Google's new Android smartphones

VIDEO GAMES

Games with Gold: The latest Xbox One and Xbox 360 free deals for October

IOS 10

iPhone users claim iOS 10 has drastically reduced the battery life of their devices

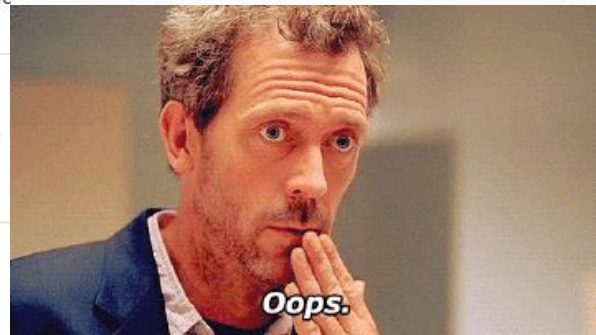
APPLE

Apple iCar: iPhone maker 'in talks' to buy legendary British F1 company McLaren

APPLE

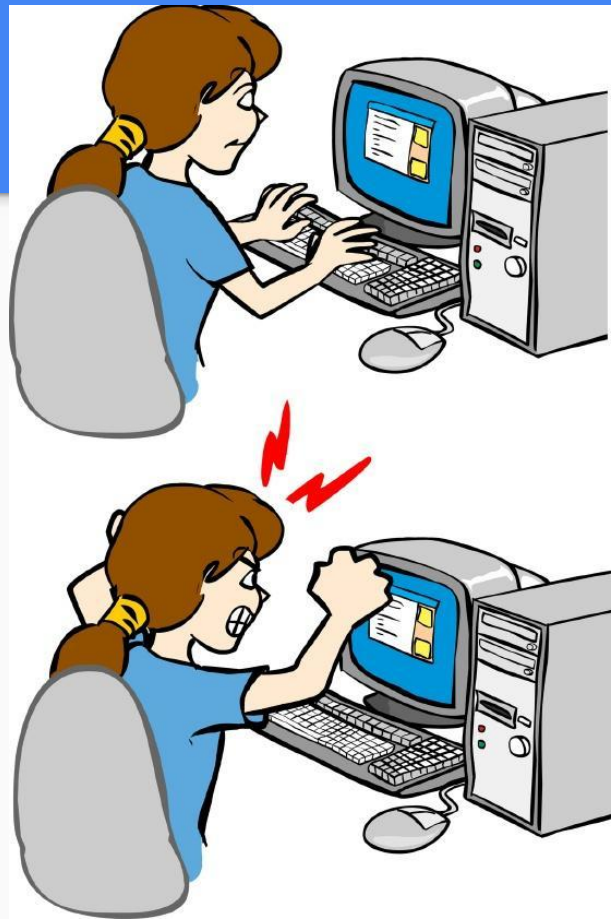
When is Apple's next 2016 event? New Macbooks and iPads

Internet go...



How do I debug my program?

- Primary tool on the terminal is GDB
- A good GUI option (widely available) is DDD
  - DDD uses GDB and gives you a GUI output
- Install on Debian (ubuntu)
  - sudo apt-get install gdb ddd
- Install on CentOS (RedHat)
  - sudo yum install gdb ddd
- Install on Arch Linux
  - sudo pacman -S ddd



A quick git command note:

- Git command I love:

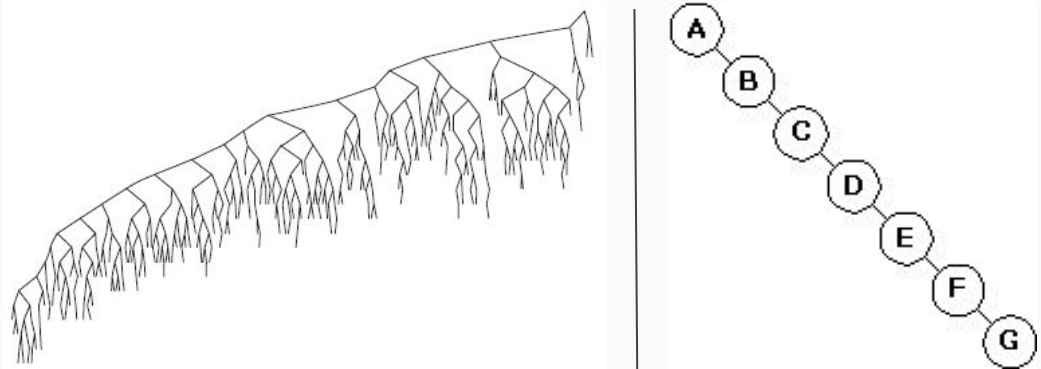
  git config --global credential.helper cache

  - Makes it so you only have to authenticate every 15 minutes while working.
  - No need to set up SSH keys, you type your password once
  - As long as you keep pushing/pulling every 15 minutes or less, no more password

AVL Trees - Adelson-Velskii and Landis

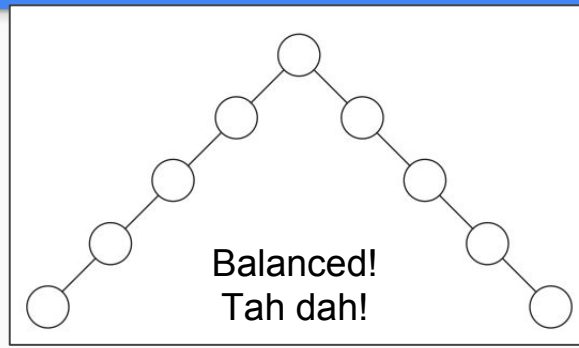
- AVL trees are self-balancing trees
  - Seek to give read-search focused applications guaranteed O(log N) access times
  - This means the most balanced tree reasonably possible
- They are binary trees (but not Binary Search Trees)
  - They add a Balance Condition to the algorithm for insertion and deletion

The goal is to avoid this ⇒



The Balance Condition

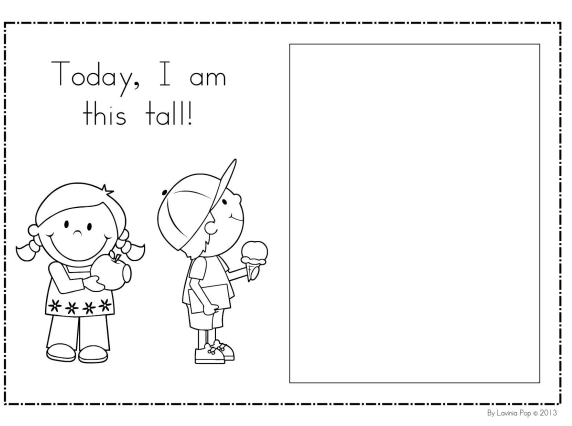
- Generates log(n) height for the tree
- Balance at the root can still have poor performance:
- Every node (sub tree) is checked for balance
  - But can’t be perfect!
  - If you refused any imbalance how would node #2 go in?
- An empty tree (nullptr) has a height of -1
- Every tree node must satisfy the requirement that:
  - abs(Height of left subtree - Height of right subtree) <= 1



Bookkeeping needed!

- Every node needs to calculate its height to test for balancing
  - Should this be done for the whole tree after every insert or delete?
  - What's the cost of that? O(?)
  - Instead… just update all nodes in the path from the root to the new node
  - This means each node keeps a field to store it's height

```
struct AvlNode {
        Comparable element;
        AvlNode *left;
        AvlNode *right;
        int height;
};
```



Cute function to calculate a node’s height

```
/**
  * Return the height of node t or -1 if nullptr.
  */
int height( AvlNode *t ) const
{
        return t == nullptr ? -1 : t->height;
}
```

Remember: an empty tree has a height of -1!
This function is a simple way to create a base case when recursively calculating node heights.

Side note about calculating height

- I was taught (back when the Flintstones were on Saturday morning cartoons) to track the relative heights of sub trees, not absolute heights
- This means keeping a BalanceFactor value:
  - BalanceFactor(N) := −Height(LeftSubtree(N)) + Height(RightSubtree(N))
- This BalanceFactor(N) is always kept:
  - BalanceFactor(N) ∈ {−1,0,+1}
  - When you add to a sub-tree, if it changes the tree height, you walk back up updating balance factors. If they get |BF(N)| > 1, then you rotate at that node

How tall will the tree be?

- log(n), right?
- Effectively, but actually about: 1.44 * log(n + 2) - 0.328 (book wrong?)
  - No, I won't be trying to derive that on stage today
- It comes from this, though. Min nodes in an AVL tree of height h:
  - S(h) = S(h-1) + S(h-2) + 1
  - This is closely related to a Fibonacci number series where:
    - h = 0 -> S(h) = 1 and h = 1 -> S(h) = 2
  - Note the S(h-1) + S(h-2) is very similar to a recursive Fib calculator return of:
    - return fib(x-1)+fib(x-2);

But here’s the calcs for the height interval

The height h of an AVL tree with n nodes lies in the interval:

$$\log_2(n+1) \le h < c \log_2(n+2)+b$$

with the golden ratio $\varphi := (1+\sqrt{5}) / 2 \approx 1.618$

$c := 1 / \log_2 \varphi \approx 1.44$

$b := c/2 \log_2 5 - 2 \approx -0.328.$

This is because an AVL tree of height h contains at least $F_{(h+2)} - 1$ nodes where $\{F_h\}$ is the Fibonacci sequence with the seed values $F_1 = 1$, $F_2 = 1$.

Cost of AVL tree operations:

Functions:	Ave Complexity:
contains(Comparable & x)	O(log N)
findMin( )	O(log N)
findMax( )	O(log N)
insert(Comparable & x)	O(log N) - but with extra rotate overhead
remove(Comparable & x)	O(log N) - but with extra rotate overhead
makeEmpty()	O(N)
copy()	O(N)

What is that extra overhead?

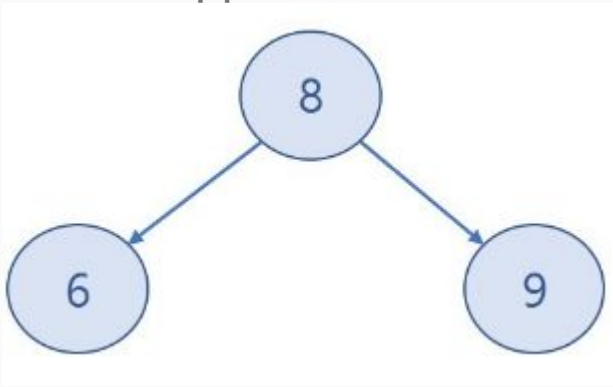
1) Updating height information
2) Rotating as needed

There’s 2 kinds of rotations:
1) Single rotations
2) Double rotations

Rotation cases

For a node (α) that needs rebalancing, these are the cases that happened:

1) An insertion into the left subtree of the left child of α
2) An insertion into the right subtree of the left child of α
3) An insertion into the left subtree of the right child of α
4) An insertion into the right subtree of the right child of α



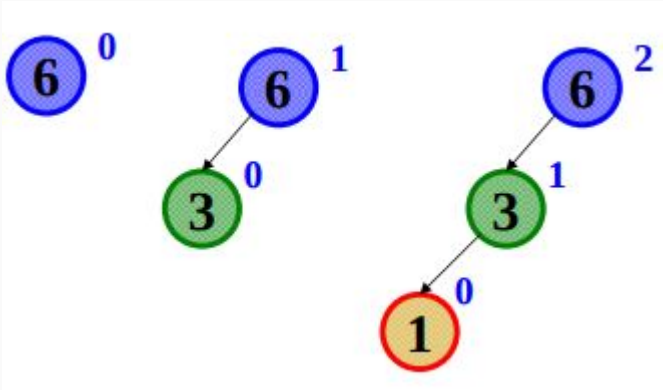
1 & 4 are mirrors, as are 2 & 3.

* Left-Left and Right-Right from α are single rotations
* Left-Right and Right-Left from α are double rotations

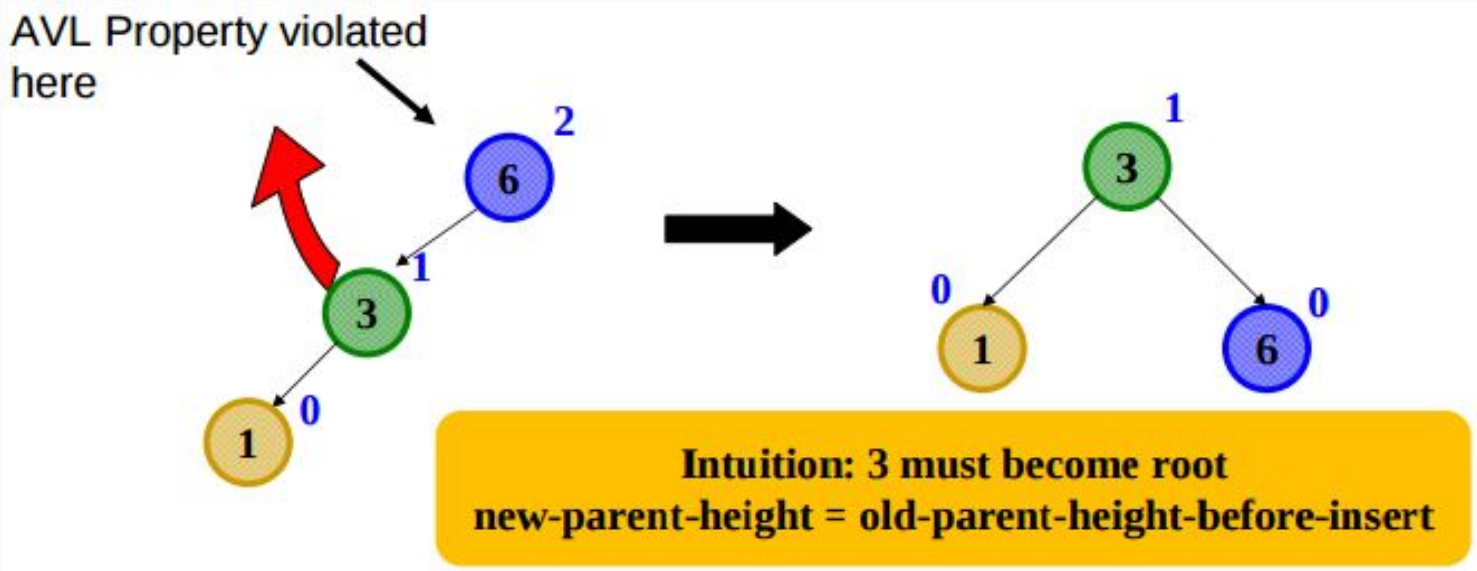
Example inserts

Insert(6)
Insert(3)
Insert(1)

What next?

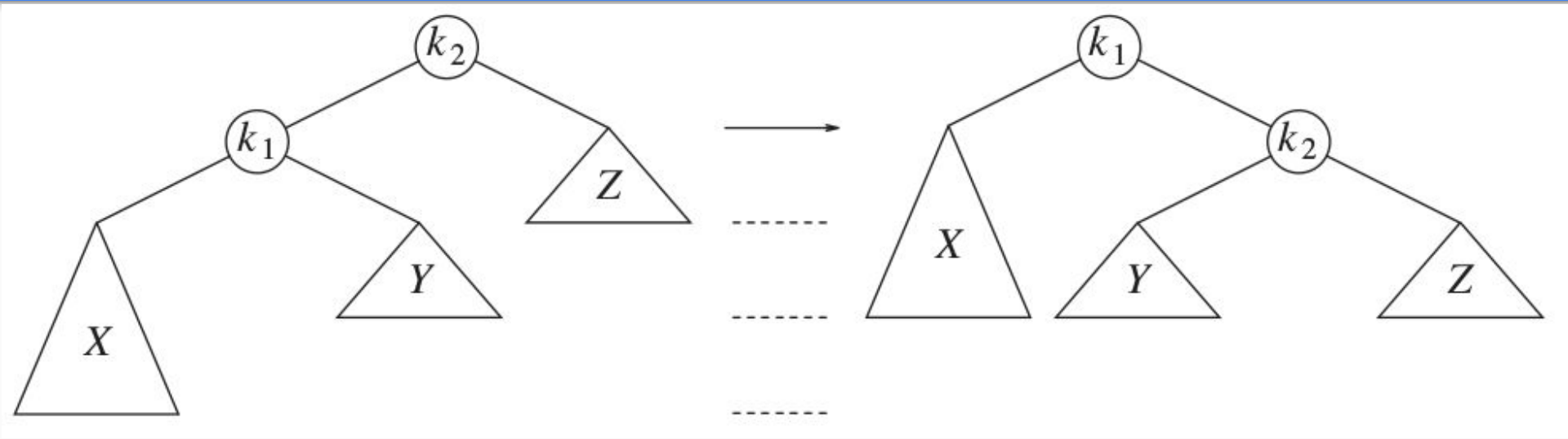


Time to rotate!



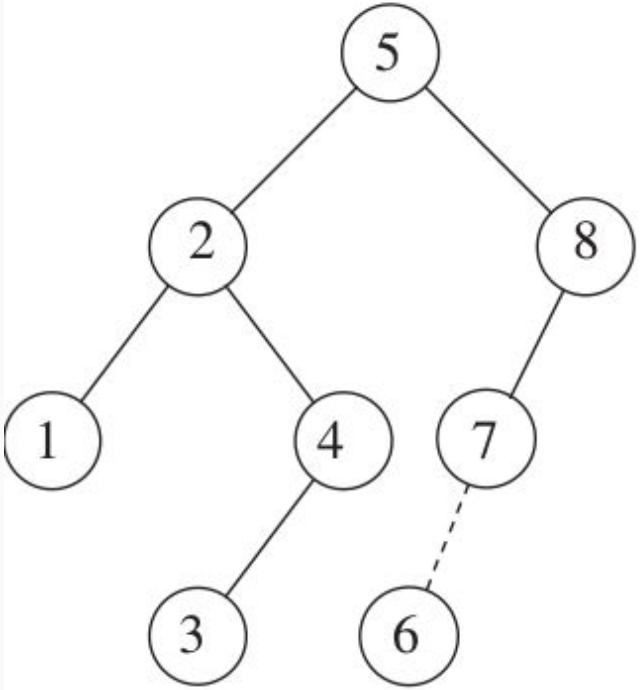
* The old root gets height + 1 of it's new children's max height

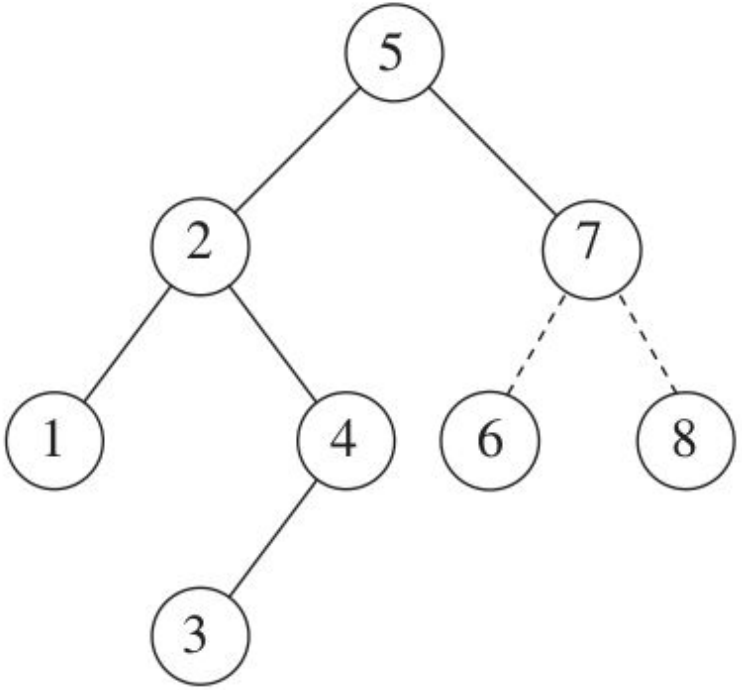
In general:



k_2 needs rebalancing, so k_1 takes it's place. What's the rule about the values in subtree Y and how do they relate to k_2?

A specific example case





A great tool to help visualize this: VisuAlgo

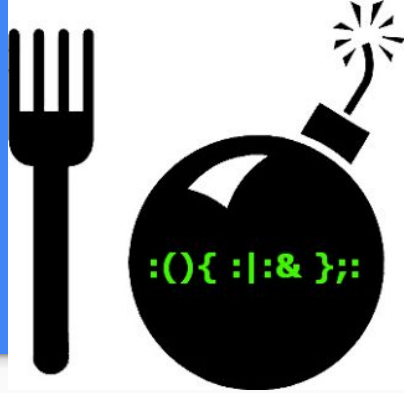
This web site (we've mentioned it before) shows many algorithms. It has some great AVL tree tools:

https://visualgo.net/bst

And for Wednesday?

- AVL trees and double rotations
- Visualizing an AVL tree with DDD
  - I’m going to spend some time getting the resolution nice and the fonts bigger!

Forkbombs: :(){ :|: & };:
Fun for the whole family!



A denial-of-service attack wherein a process continually replicates itself to deplete available system resources, slowing down or crashing the system due to resource starvation.

Around 1978, an early variant of a fork bomb called wabbit was reported to run on a System/360. It may have descended from a similar attack called RABBITS reported from 1969 on a Burroughs 5500 at the University of Washington.

Ubuntu forums are gold sometimes!

`:()` means you are defining a function called `:`

`{:|: &}` means run the function `:` and send its output to the `:` function again and run that in the background.

The `;` is a command separator, like `&&`.

`:` runs the function the first time.

Essentially you are creating a function that calls itself twice every call and doesn't have any way to terminate itself. It will keep doubling up until you run out of system resources.

Running in Virtualbox was quite sensible really otherwise you would have had to restart your pc.

Three Forkbombs

Poor little Linux box

DOS Batch Forkbomb:

```
%0|%0
```

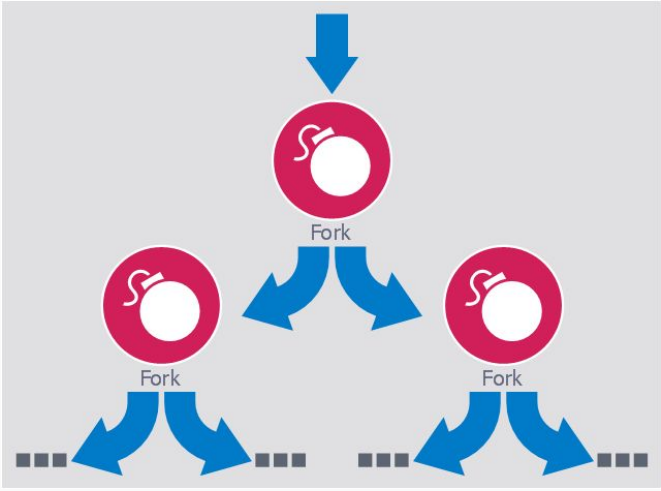
(save as .bat & run!)

BASH shell:

```
:(){ :|:& };:
```

C/C++:

```
#include <unistd.h>
int main(void)
{
  while(1)
    fork();
  return 0;
}
```



* Note: DO NOT RUN ON EECS SERVERS! *
I highly suggest a virtual machine to crash

Thing of the Day

CES had a single passenger drone demonstrated. Any takers?

https://www.theguardian.com/technology/2016/jan/07/first-passenger-drone-makes-world-debut