FRIDAY WEEK 2 NOTES

1. Binary Search

Input:
L - a sorted list of numbers
x - the number we're looking for
a - place in L to start looking
b - place in L to stop looking

Output:
true or false - is x in the list between a and b or not

Some code for binary search:

```
def Find(L,x,a,b):
    let c = ⌊(a+b)/2⌋
    if a == b:
        return (L(a) == x)
    else:
        if L[c] < x :
            return Find(L,x,c,b)
        else:
            return Find(L,x,a,c)
```

Let's analyze how long this takes to run:
Let $T(n)$ = worst-case running time on list of length n
Claim. $T(n) = max\left(T(\lfloor \frac{n}{2} \rfloor), T(\lceil \frac{n}{2} \rceil)\right) + C$

Claim. T is increasing, so $T(n) = T(\lceil \frac{n}{2} \rceil) + C$
If $n = 2^k$,

$$T(2^k) = T(2^{k-1}) + C$$
$$= T(2^{k-2} + 2C$$
$$\ldots$$
$$= T(1) + kC$$

We learn $T(n) \leq Clog_2 n$ (assuming T is increasing), i.e. $T(n) = O(logn)$.

Date: October 11, 2019.

2. Memoization

If you have something like:

```
def F(a,b,c):
    do some things
    return answer
```

and you would like this to run faster the next time you call it, you can do:

```
memoize = {}
def F(a,b,c):
    if (a,b,c) not in memoize:
        do some things
        memoize[(a,b,c)] = answer
    return memoize[(a,b,c)]
```

note that (a,b,c) have to be things you can put in a dictionary

Example. Up-Right lattice points from $(0,0)$ to (a,b)

Let $n(a,b)$ be the number of U-R paths from $(0,0$ to (a,b)

It's clear that

$$n(a,b) = \begin{cases} 1 \text{ if } a = 0 \text{ or } b = 0 \\ n(a-1,b) + n(a,b-1) \end{cases}$$

You have some options:
- could memoize
- could compute these for (a,b) in increasing lex order
- answer really though is $\binom{a+b}{b}$, could do math

3. Graph Theory

- A graph $G = (V,E)$ has a set V of vertices and $E \subseteq (V \times V)$ of edges.
- We say G is simple if $(v,v) \notin E$ for every $v \in V$ (i.e. no loops of vertex to itself).
- We say G is directed if E is a set of ordered pairs (a,b).
- We say G is undirected if E is a set of unordered pairs $\{a,b\}$.
- A walk from $a \in V$ to $b \in V$ is a sequence of edges

$$(a_1,a_1),(a_1,a_2),...,(a_{k-1},a_k),(a_k,b)$$

in E.

- A connected graph has a walk from a to b for all $a, b \in V$.
- A path from a to b is a walk with no repeated vertex.
- A cycle is a walk from $a \in V$ to a of length ≥ 3 with no repeated vertices except for endpoints.
- A tree is a connected graph with no cycles.

Claim. If $T = (V, E)$ is a tree, then $|V| = |E| + 1$.