6/15/2020 - Recursion #1

Monday, June 15, 2020 4:09 PM

A recursive function programming is a function that (sometimes) calls itself.

Note: a recursive function can't ALWAYS call itself…if it did, the first function call would never end…

The idea is as follows:

Break the problem down into at least one piece that is a problem of the exact same nature. So, instead of solving it, we just call our own function **WITH DIFFERENT PARAMETERS** to solve that smaller piece. Then we use that solution to solve our specific query.

Other times, the problem is so easy, we just directly solve it. This is typically called the base case (or base cases).

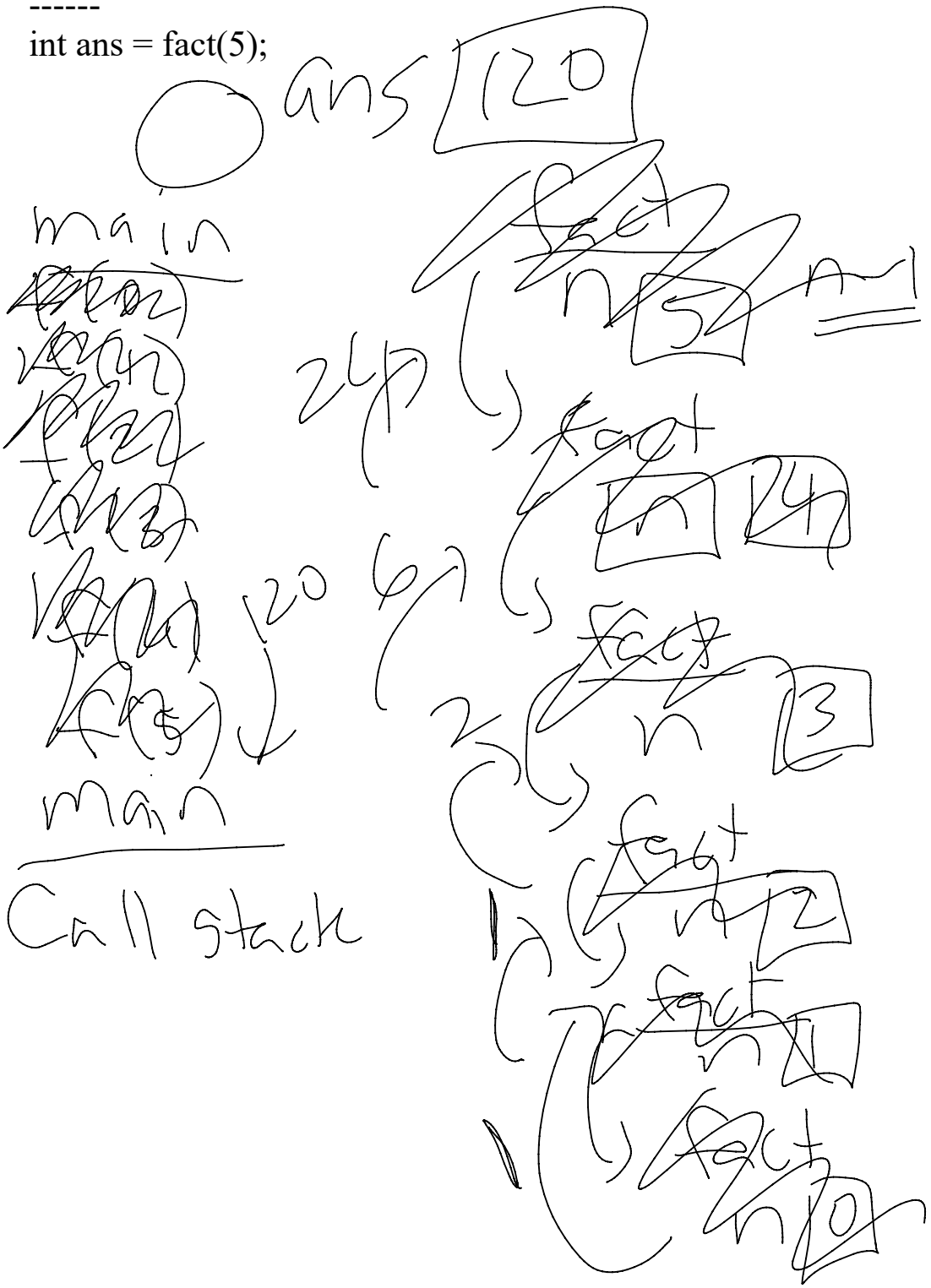
Outline

1. factorial
2. fibonacci
3. power
4. tipChart
5. sumDigits
6. decToBin
7. BinarySearch
8. modular exponentiation (modPow)
9. Towers of Hanoi
10. Linked List Stuff Revisited, With Recursion

n! = 1 x 2 x 3 … x n, also define 0! = 1

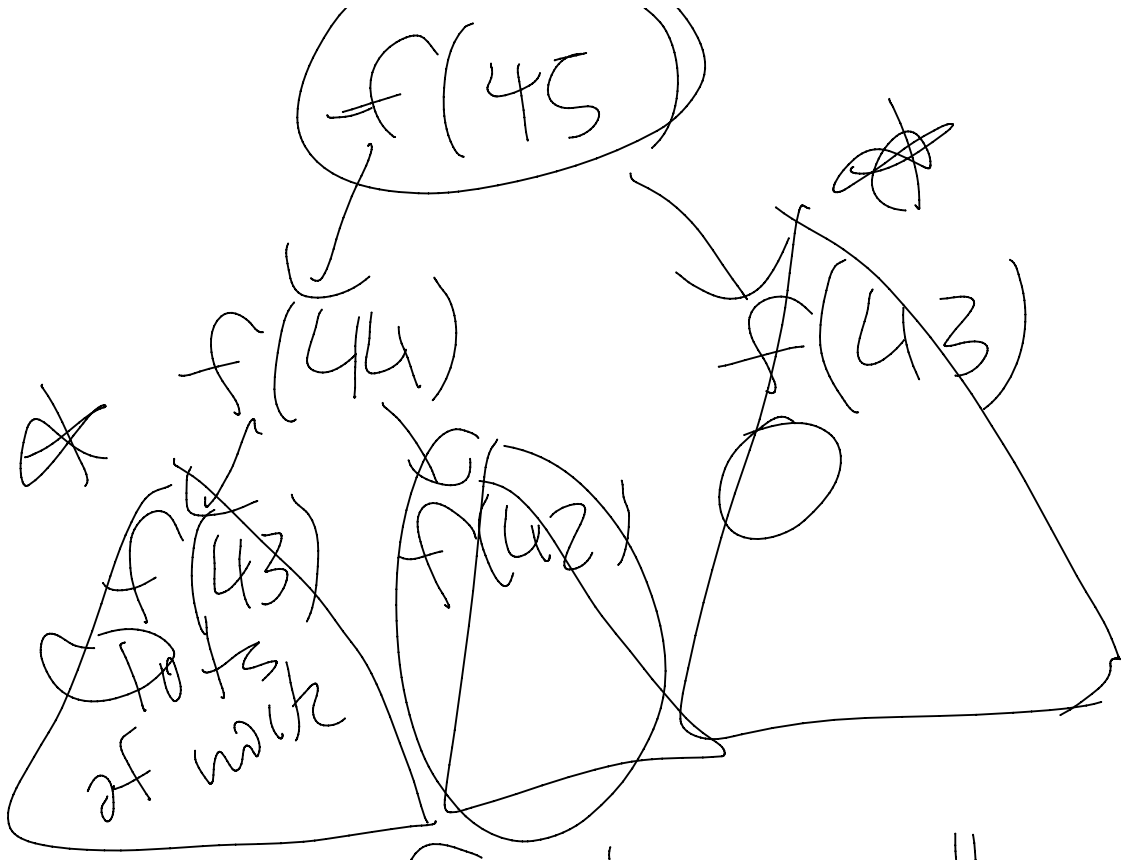
n! = (1 x 2 x 3 … x (n-1)) x n
= (n-1)! x n, this is true so long as n > 0…
and when n = 0, the answer is 1.

main
---
int ans = fact(5);



Fibonacci…

fib(0) = 0, fib(1) = 1
fib(n) = fib(n-1)+fib(n-2) for all ints n > 1.



# of function calls is $O(Fib(n))$.

$\sim O(1.61^n)$

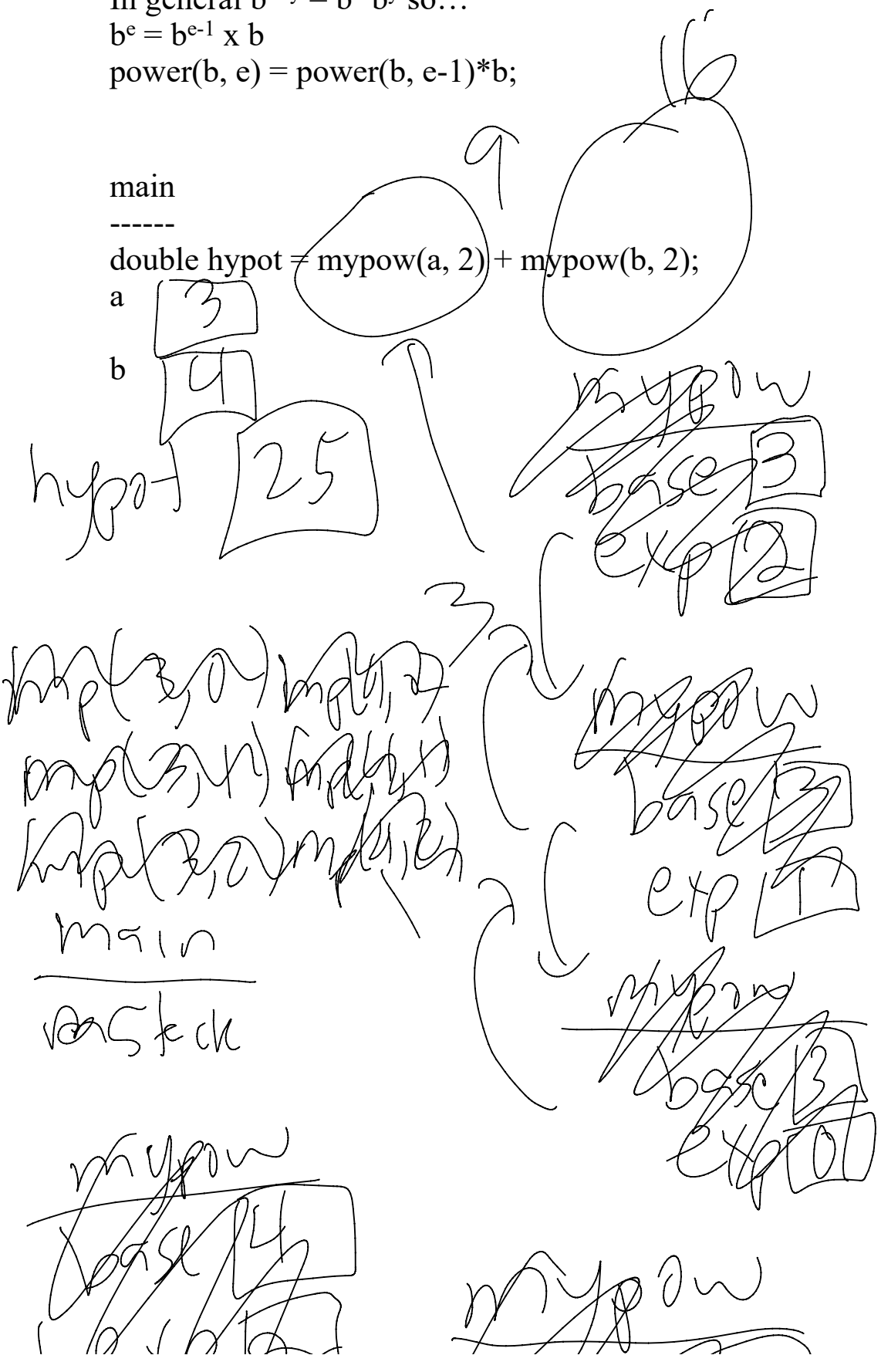
Golden ratio $\phi = \frac{1+\sqrt{5}}{2}$

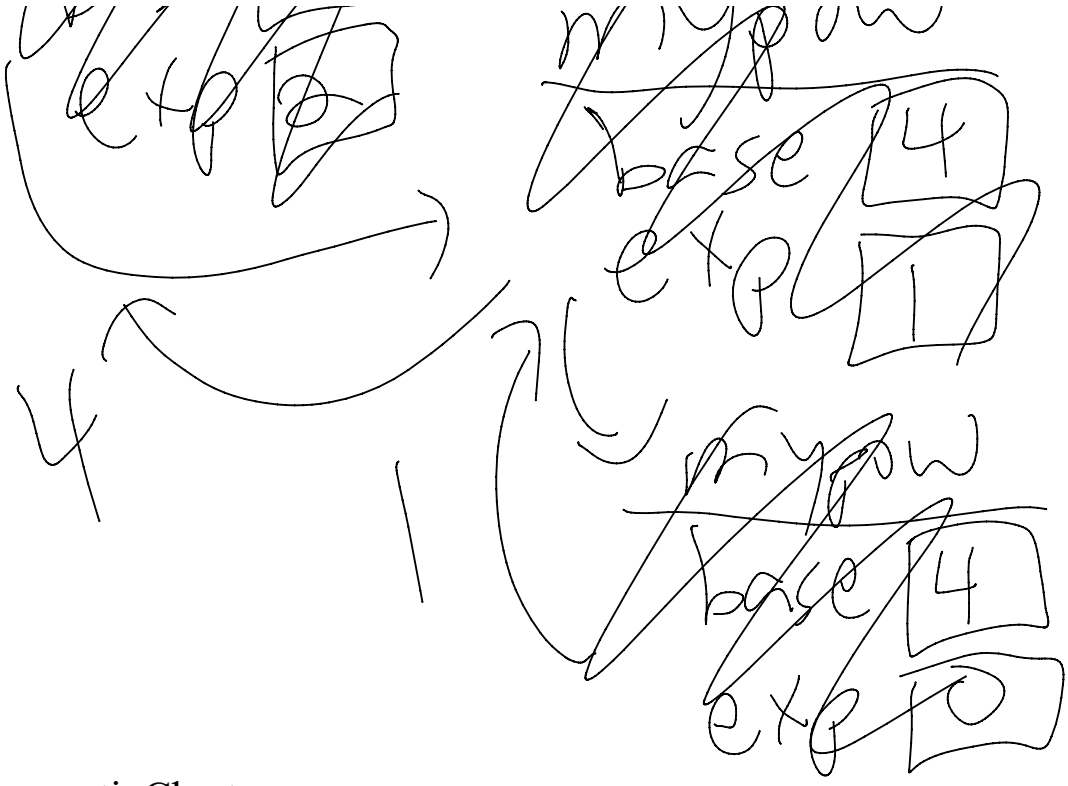
power

$b^{-e} = 1/b^e$, rule for negative exponents.

In general $b^{x+y} = b^x\ b^y$ so…
$b^e = b^{e-1} \times b$
power(b, e) = power(b, e-1)*b;

main
---
double hypot = mypow(a, 2) + mypow(b, 2);
a 3
b 4
hypot 25





tipChart
---
1 0.15 -- do this
2 0.30
3 0.45
4 0.60
5 0.75

this is a smaller tip chart.

tipChart of row 1 to row 5 is kind of like
1. Print row 1
2. Print a tipChart of rows 2 through 5

// recursively print out a tip chart form minVal to maxVal
tipChart(int minVal, int maxVal, double percent)

Alternative breakdown

1	0.15
2	0.30
3	0.45
4	0.60
5	0.75

recursive

recursive

2	0.30
3	0.45
4	0.60
5	0.75

recursive

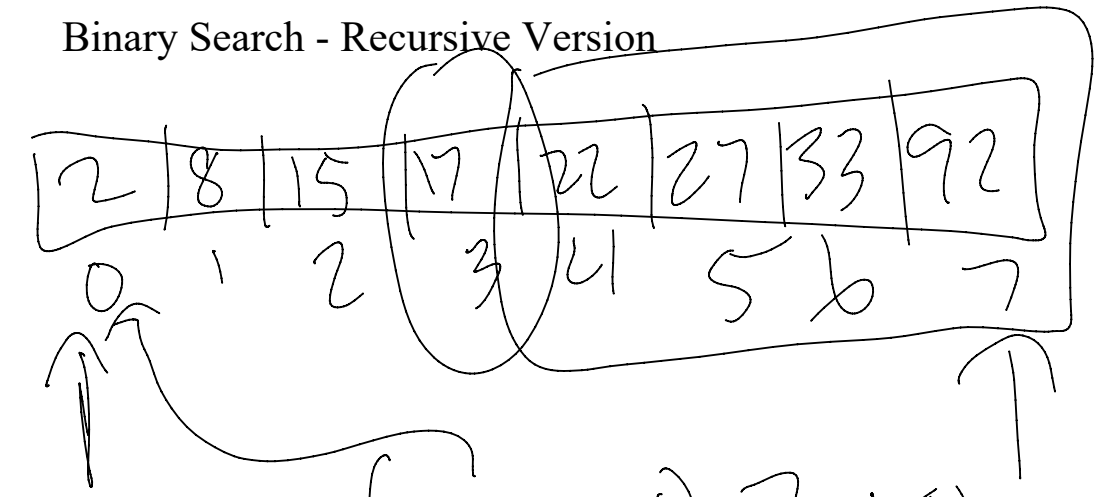
sum of digits of an integer

sumdigits(6427372) = sumdigits(642737) + 2

This is the easiest way to break this down recursively.

sumdigits(n) = sumdigits(n/10) + (n%10), this is our recursive formula

Binary Search - Recursive Version



binSearch(array, 0, 7, 45)

mid 3

Since 45 > 17 we will search for it in indexes [4..7]

In indexes [4..7]

Towers of Hanoi

3 poles
Move 1 disk at a time
Can never place a larger disk on a smaller disk

Towers(int startTower, int endTower, int nDisks)

Key Realization - we are FORCED to solve the puzzle for nDisks-1 to free up the bottom disk to move. We want to move this disks to a "temporary" pole.

Towers(startTower, tempTower, nDisks-1);
Move the bottom disk from startTower to endTower
Towers(tempTower, endTower, nDisk-1);

Number of moves (we did this empirically) is probably

$T(n) = 2^n - 1$.

We will prove this next time.

What is left to cover for this lecture:

1) fast modular exponentiation
2) Linked List Methods

Since we got a late start, I'll record both of these tonight and post them in the same place I post the regular class lectures.

Arup

Answers to questions at end of class
---
n + m might not be less than a constant times n, and it might

not be less than a constant times m, but it is definitely less than a constant times n+m, it is ALSO less than a constant times max(n,m). There are two correct answers, at least…

O(n+m) or O(max(n,m)).

It's perfectly valid for a run time to be something like $O(nm^2)$ …or say, O(nlg m)