CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson

Fall 2017

Swarthmore College

Outline Dec 4:

- Recursion
- Wed: go over Quiz 5

Notes

•Lab 11 is optional but STRONGLY recommended
•Lab 11 attendance is NOT optional

Begin: recursive functions

What is recursion?

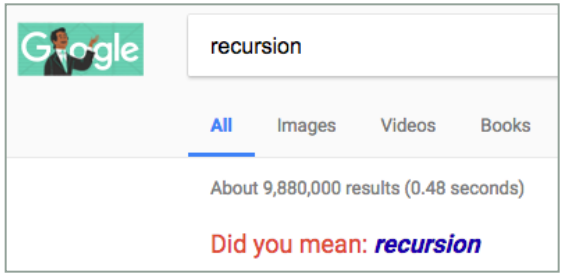
- A recursive function calls **itself**.

- Usually two key components to a recursive function:
  1) A **base case** or way to stop the recursion
  2) A **recursive call** using a modified argument

What is recursion?

- A recursive function calls **itself**.

- Usually two key components to a recursive function:
  1) A **base case** or way to stop the recursion
  2) A **recursive call** using a modified argument



"To understand recursion, you must first understand recursion"

Image: "Matryoshka Nesting Dolls: Meaning of Russian Wooden Stacking Doll" - Legomenon

Factorial function

n	recursion	n!
0	base case	1
1		
2		
3		
4		
5		
6		

$n! = n*(n-1)*(n-2)\ldots3*2*1$

or

$n! = n*(n-1)!$

with

base case: $0! = 1$

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2		
3		
4		
5		
6		

$n! = n*(n-1)*(n-2)\ldots3*2*1$

or

$n! = n*(n-1)!$

with

base case: $0! = 1$

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2	2*1	2
3		
4		
5		
6		

$n! = n*(n-1)*(n-2)\ldots3*2*1$

or

$n! = n*(n-1)!$

with

base case: $0! = 1$

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2	2*1	2
3	3*2	6
4		
5		
6		

$n! = n*(n-1)*(n-2)\ldots3*2*1$

or

$n! = n*(n-1)!$

with

base case: $0! = 1$

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2	2*1	2
3	3*2	6
4	4*6	24
5		
6		

n! = n*(n-1)*(n-2)…3*2*1

or

n! = n*(n-1)!

with

base case: 0! = 1

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2	2*1	2
3	3*2	6
4	4*6	24
5	5*24	120
6		

$n! = n*(n-1)*(n-2)\ldots3*2*1$

or

$n! = n*(n-1)!$

with

base case: $0! = 1$

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2	2*1	2
3	3*2	6
4	4*6	24
5	5*24	120
6	6*120	720

$n! = n*(n-1)*(n-2)\ldots3*2*1$

or

$n! = n*(n-1)!$

with

base case: $0! = 1$

Reverse String (recursion)

Reversing a string

```
                                reverse("summer")

                        reverse("ummer")                + "s"

                    (reverse("mmer")            + "u") + "s"

                ((reverse("mer")        + "m") + "u") + "s"

            (((reverse("er")        + "m") + "m") + "u") + "s"

        ((((reverse("r")     + "e") + "m") + "m") + "u") + "s"

    (((((reverse("") + "r") + "e") + "m") + "m") + "u") + "s"
```

Summer

s

us

mus

⋮

remmus

pretend we can reverse

factorial

$$n! = 1 \cdot 2 \cdot 3 \cdot 4 \cdots (n-1) \cdot n$$

$$fac(3) = 1 \cdot 2 \cdot 3 = 6$$

$$fac(4) = 1 \cdot 2 \cdot 3 \cdot 4 = 24 \rightarrow fac(3) \cdot 4$$

$$fac(5) = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120 \rightarrow fac(4) \cdot 5$$

remmu

s ← s[0]

call function on s[1:]