Warm-up as you walk in

Complete the recursive function for factorial!

Note: A recursive function calls itself

```
def factorial(int n):
    if ____________:
        return ____________
    return _____________________
```

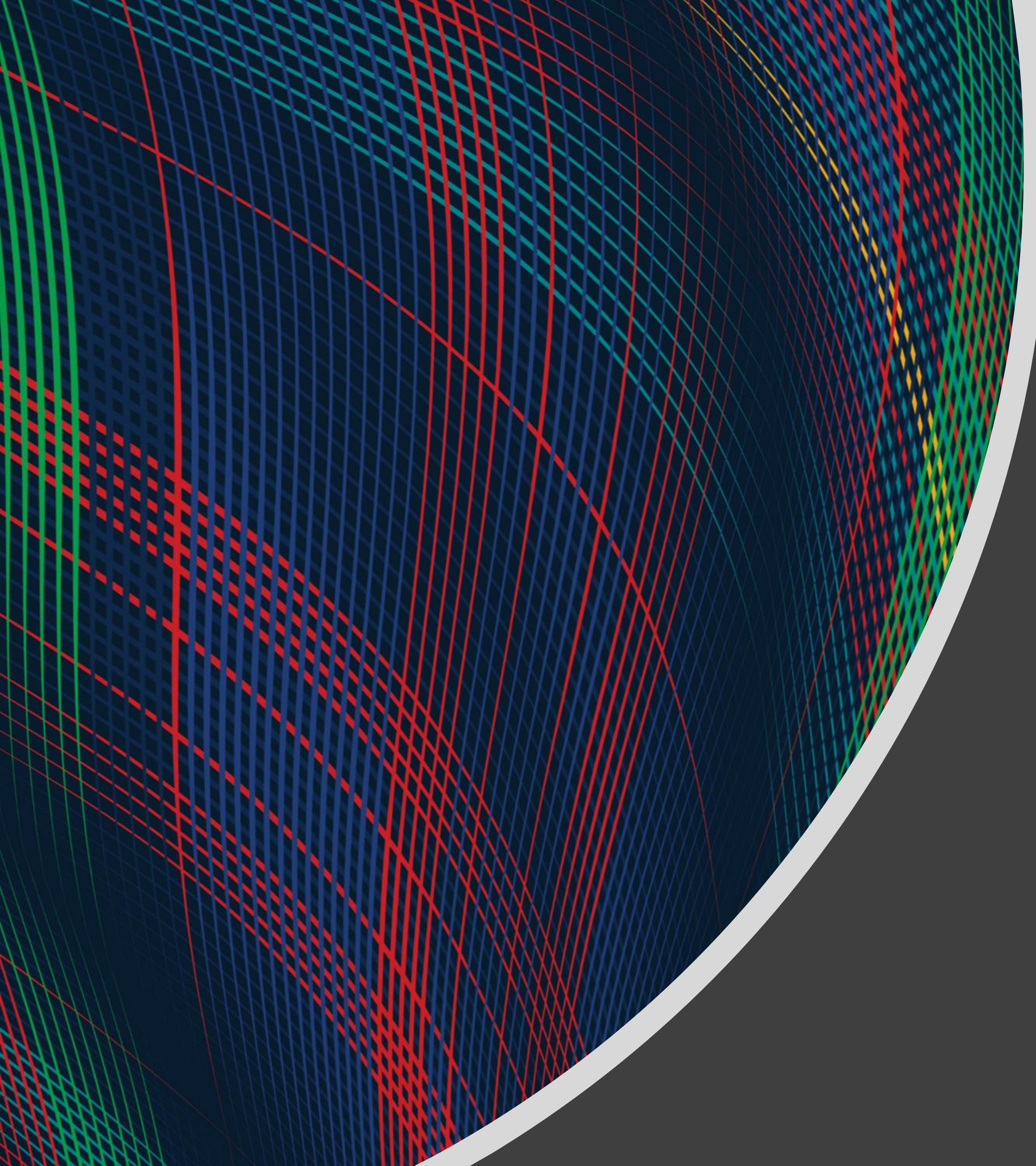
0! = 1

1! = 1

2! = 2

3! = 6

4! = 24



10-607 Computational Foundations for Machine Learning

Recursion

Instructor: Pat Virtue

Plan

Computational Complexity

- Dealing with multiple inputs
- Exercises

Recursion

- Guess a number
- Binary search
- Computational complexity
- Recursion
- Practical considerations
- Proof by induction

Computation Complexity

Previous lecture slides

Plan

Computational Complexity

- Dealing with multiple inputs
- Exercises

Recursion

- Guess a number
- Binary search
- Computational complexity
- Recursion
- Practical considerations
- Proof by induction

Poll 1

I’m thinking of a number between 1 and 64. After each guess, I’ll tell you if you’re correct or if my number is higher or lower.

What is the maximum number of guesses you’ll need to play this game?

A: 6

B: 7

C: 32

D: 64

Guess a Number

I'm thinking of a number between 1 and N. After each guess, I'll tell you if you're correct or if my number is higher or lower.

What is the maximum number of guesses you'll need to play this game?

N	10	100	1000	10K	100K	1M	10M	100M
$\log_2 N$	3.3	6.6	10.0	13.3	16.6	19.9	23.3	26.6
$\lfloor \log_2 N \rfloor + 1$	4	7	11	14	17	20	24	27

Classic CS problem: Searching

Imagine storing **sorted** data in an array

How long does it take us to find a number we are looking for?

0	1	2	3	4	5	6	7	8	9	10
2	7	8	13	25	29	33	51	89	90	95

Slide credit: http://www.peerinstruction4cs.org/2013/07/13/cs2-in-c-peer-instruction-materials/

Imagine storing **sorted** data in an array

How long does it take us to find a number we are looking for?

0	1	2	3	4	5	6	7	8	9	10
2	7	8	13	25	29	33	51	89	90	95

If you start at the front and proceed forward, each item you examine rules out 1 item

Slide credit: http://www.peerinstruction4cs.org/2013/07/13/cs2-in-c-peer-instruction-materials/

Imagine storing sorted data in an array

0	1	2	3	4	5	6	7	8	9	10
2	7	8	13	25	29	33	51	89	90	95

If instead we **jump right to the middle**, one of three things can happen:

A. The middle one happens to be the number we were looking for, yay!
B. We realize we went too far
C. We realize we didn't go far enough

Ruling out HALF the options in one step is so much faster than only ruling out one!

Slide credit: http://www.peerinstruction4cs.org/2013/07/13/cs2-in-c-peer-instruction-materials/

Imagine storing <u>sorted</u> data in an array

0	1	2	3	4	5	6	7	8	9	10
2	7	8	13	25	29	33	51	89	90	95

If instead we **jump right to the middle**, one of three things can happen:

A. The middle one happens to be the number we were looking for, yay!
B. We realize we went too far
C. We realize we didn't go far enough

Ruling out HALF the options in one step is <u>so much</u> faster than only ruling out one!

Slide credit: http://www.peerinstruction4cs.org/2013/07/13/cs2-in-c-peer-instruction-materials/

Binary search

0	1	2	3	4	5	6	7	8	9	10
2	7	8	13	25	29	33	51	89	90	95

Let’s say the answer was C, “we didn’t go far enough”

We ruled out the entire first half, and now only have the second half to search

We could start at the front of the second half and proceed forward…

Slide credit: http://www.peerinstruction4cs.org/2013/07/13/cs2-in-c-peer-instruction-materials/

Binary search

0	1	2	3	4	5	6	7	8	9	10
2	7	8	13	25	29	33	51	89	90	95

Let’s say the answer was C, “we didn’t go far enough”

We ruled out the entire first half, and now only have the second half to search

We could start at the front of the second half and proceed forward…but why do that when we know we have a better way?

Jump right to the middle of the region to search

Slide credit: http://www.peerinstruction4cs.org/2013/07/13/cs2-in-c-peer-instruction-materials/

Binary search

0	1	2	3	4	5	6	7	8	9	10
2	7	8	13	25	29	33	51	89	90	95

Let’s say the answer was C, “we didn’t go far enough”

We ruled out th… …and now only have the secon…

RECURSION !!

We could star… …of the second half and proceed forward…but why do that when we know we have a better way?

Jump right to the middle of the region to search

Slide credit: http://www.peerinstruction4cs.org/2013/07/13/cs2-in-c-peer-instruction-materials/

Binary search

```
bool binarySearch(list data, int key) {
      return binarySearch2(data, key, 0, len(data)-1);
}

bool binarySearch2(list data, int key, int first, int last) {
      // TODO base case
      int mid = (first + last) // 2;
      if (data[mid] == key)
            return true;
      else if (data[mid] > key)
            return binarySearch2(data, key, first, mid-1);
      else
            return binarySearch2(data, key, mid+1, last)
}
```

Poll 2

What would be a good base case for our Binary Search function?

Select all that apply

A. Only three items remain: save yourself an unnecessary function call that would trivially divide them into halves of size 1, and just check all three.
B. Only two items remain: can't divide into two halves with a middle, so just check the two.
C. Only one item remains: just check it.
D. No items remain: obviously we didn't find it.

Slide credit: http://www.peerinstruction4cs.org/2013/07/13/cs2-in-c-peer-instruction-materials/

Poll 3

What is the time complexity of binary search?

A: $O(1)$

B: $O(N)$

C: $O(N/2)$

D: $O(\log N)$

E: None of the above

Plotting log functions

Notebook

Plan

Computational Complexity

- Dealing with multiple inputs
- Exercises

Recursion

- Guess a number
- Binary search
- Computational complexity O(log n)
- Recursion
- Practical considerations
- Proof by induction

Factorial

Complete the recursive function for factorial!

Note: A recursive function calls itself

```
def factorial(int n):
    if ____________:
        return ____________
    return _____________________
```

0! = 1

1! = 1

2! = 2

3! = 6

4! = 24

Recursion

https://www.cs.cmu.edu/~112/notes/notes-recursion-part1.html

Template

```
def recursiveFunction():
    if (this is the base case):
        do something non-recursive
    else:
        do something recursive
```