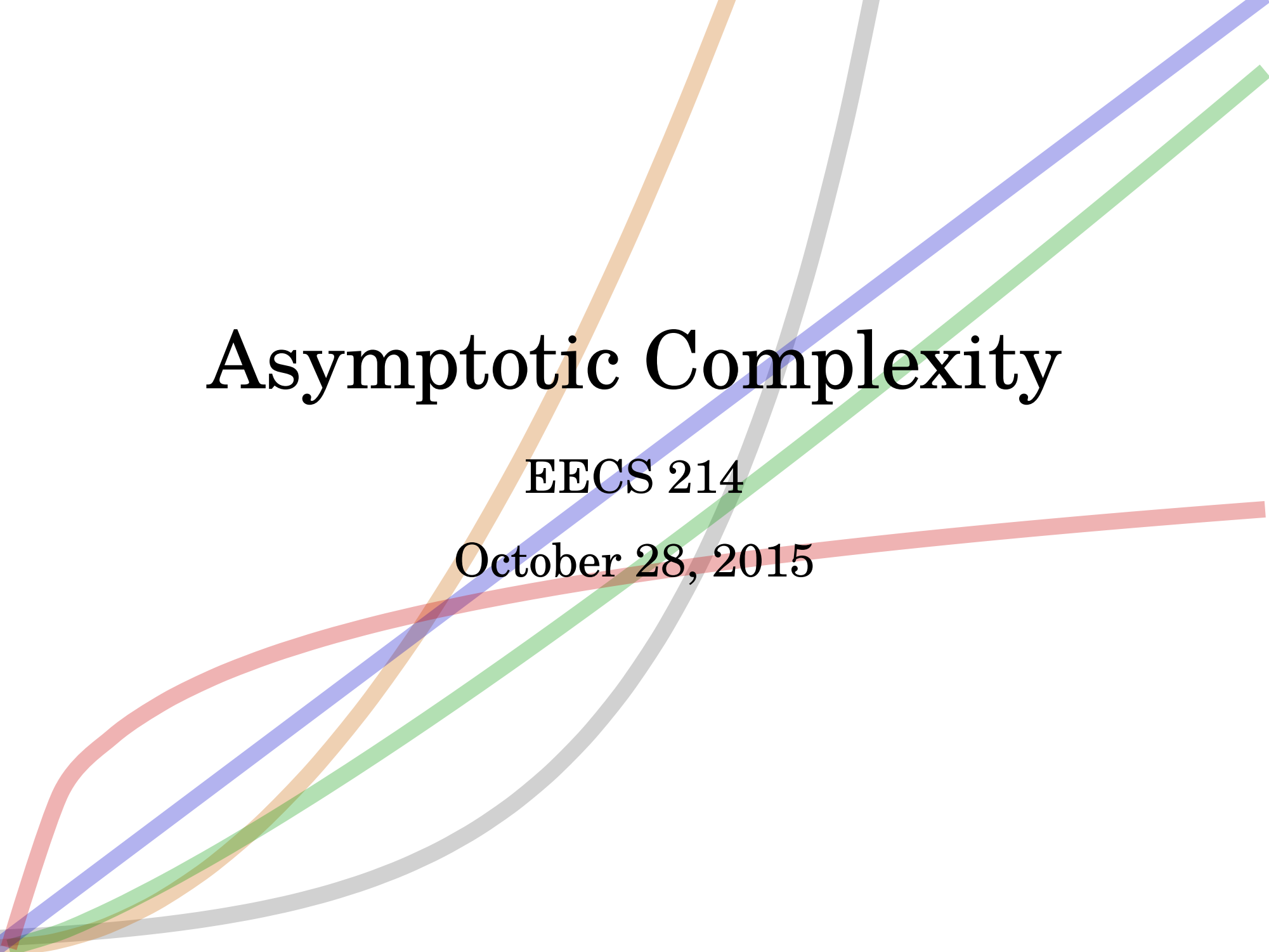

Asymptotic Complexity

EECS 214

October 28, 2015

Finding things in an array

```
int find_in_unsorted_array(int needle, int* haystack, int size)
{
    for (int i = 0; i < size; ++i) {
        if (needle == haystack[i]) return i;
    }

    return -1;
}
```

Finding things in an array

```
int find_in_unsorted_array(int needle, int* haystack, int size)
{
    for (int i = 0; i < size; ++i) {
        if (needle == haystack[i]) return i;
    }

    return -1;
}
```

In the worst case, how long does this take?

Finding things in an array

```
int find_in_unsorted_array(int needle, int* haystack, int size)
{
    for (int i = 0; i < size; ++i) {
        if (needle == haystack[i]) return i;
    }

    return -1;
}
```

In the worst case, how long does this take? `size` iterations

Finding things in a sorted array

```
bool find_in_sorted_array(int needle, int* haystack, int size)
{
    int start = 0, limit = size;

    while (start < limit) {
        int mid = (start + limit) / 2;
        if (needle < haystack[mid]) limit = mid;
        else if (needle > haystack[mid]) start = mid + 1
        else return mid;
    }

    return false;
}
```

Finding things in a sorted array

```
bool find_in_sorted_array(int needle, int* haystack, int size)
{
    int start = 0, limit = size;

    while (start < limit) {
        int mid = (start + limit) / 2;
        if (needle < haystack[mid]) limit = mid;
        else if (needle > haystack[mid]) start = mid + 1
        else return mid;
    }

    return false;
}
```

In the worst case, how long does this take?

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining
1	500,000

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining
1	500,000
2	250,000

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining
1	500,000
2	250,000
3	125,000

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining
1	500,000
2	250,000
3	125,000
4	62,500

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining
1	500,000
2	250,000
3	125,000
4	62,500
5	31,250

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining
1	500,000
2	250,000
3	125,000
4	62,500
5	31,250
6	15,675

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining
1	500,000
2	250,000
3	125,000
4	62,500
5	31,250
6	15,675
7	7,837

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining
1	500,000
2	250,000
3	125,000
4	62,500
5	31,250
6	15,675
7	7,837
8	3,918

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining
1	500,000
2	250,000
3	125,000
4	62,500
5	31,250
6	15,675
7	7,837
8	3,918
9	1,959

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining
1	500,000
2	250,000
3	125,000
4	62,500
5	31,250
6	15,675
7	7,837
8	3,918
9	1,959
10	980

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining	# of divs.	size remaining
1	500,000	11	490
2	250,000		
3	125,000		
4	62,500		
5	31,250		
6	15,675		
7	7,837		
8	3,918		
9	1,959		
10	980		

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining	# of divs.	size remaining
1	500,000	11	490
2	250,000	12	245
3	125,000		
4	62,500		
5	31,250		
6	15,675		
7	7,837		
8	3,918		
9	1,959		
10	980		

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining	# of divs.	size remaining
1	500,000	11	490
2	250,000	12	245
3	125,000	13	123
4	62,500		
5	31,250		
6	15,675		
7	7,837		
8	3,918		
9	1,959		
10	980		

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining	# of divs.	size remaining
1	500,000	11	490
2	250,000	12	245
3	125,000	13	123
4	62,500	14	62
5	31,250		
6	15,675		
7	7,837		
8	3,918		
9	1,959		
10	980		

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining	# of divs.	size remaining
1	500,000	11	490
2	250,000	12	245
3	125,000	13	123
4	62,500	14	62
5	31,250	15	32
6	15,675		
7	7,837		
8	3,918		
9	1,959		
10	980		

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining	# of divs.	size remaining
1	500,000	11	490
2	250,000	12	245
3	125,000	13	123
4	62,500	14	62
5	31,250	15	32
6	15,675	16	16
7	7,837		
8	3,918		
9	1,959		
10	980		

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining	# of divs.	size remaining
1	500,000	11	490
2	250,000	12	245
3	125,000	13	123
4	62,500	14	62
5	31,250	15	32
6	15,675	16	16
7	7,837	17	8
8	3,918		
9	1,959		
10	980		

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining	# of divs.	size remaining
1	500,000	11	490
2	250,000	12	245
3	125,000	13	123
4	62,500	14	62
5	31,250	15	32
6	15,675	16	16
7	7,837	17	8
8	3,918	18	4
9	1,959		
10	980		

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining	# of divs.	size remaining
1	500,000	11	490
2	250,000	12	245
3	125,000	13	123
4	62,500	14	62
5	31,250	15	32
6	15,675	16	16
7	7,837	17	8
8	3,918	18	4
9	1,959	19	2
10	980		

How many times can we divide the array in half?

Suppose `size` is 1,000,000. Then:

# of divs.	size remaining	# of divs.	size remaining
1	500,000	11	490
2	250,000	12	245
3	125,000	13	123
4	62,500	14	62
5	31,250	15	32
6	15,675	16	16
7	7,837	17	8
8	3,918	18	4
9	1,959	19	2
10	980	20	1

How many times can we divide an array of size n in half?

$$\log_2 n$$

Time to search array

Linear search	Binary search
$c_1 n + c_2$	$d_1 \log n + d_2$

Definition of big O

Let $f : \mathbb{R} \to \mathbb{R}$. Then we define the set of functions $\mathcal{O}(f)$ as follows:

A function $g \in \mathcal{O}(f)$ *iff* there are some m and c
such that for all $n > m$,

$$g(n) \leq cf(n) .$$

Definition of big O

Let $f : \mathbb{R} \to \mathbb{R}$. Then we define the set of functions $\mathcal{O}(f)$ as follows:

A function $g \in \mathcal{O}(f)$ *iff* there are some m and c
such that for all $n > m$,

$$g(n) \leq cf(n) .$$

Intuitively:

- m means that we care about large inputs and can ignore small inputs up to some size.
- c is a constant factor, since we don't care about differences of a constant factor (since that just corresponds to making a computer faster)

Big O definition example

Let's show that $d_1 \log n \in \mathcal{O}(c_1 n)$.

Big O definition example

Let's show that $d_1 \log n \in \mathcal{O}(c_1 n)$.

Choose $c = \frac{d_1}{c_1}$ and $m = 1$. (These are guesses.)

Big O definition example

Let's show that $d_1 \log n \in \mathcal{O}(c_1 n)$.

Choose $c = \frac{d_1}{c_1}$ and $m = 1$. (These are guesses.)

To show:

$$\begin{aligned} d_1 \log n &\leq \frac{d_1}{c_1}(c_1 n) && \text{for } n > 1 \\ \log n &\leq n && \text{by algebra} \end{aligned}$$

This is true for $n > 0$, so it is true for $n > 1$.