Discussion 04:
Growth and Nonlocal

TA: **Jerry Chen**
Email: **jerry.c@berkeley.edu**
TA Website: **jerryjrchen.com/cs61a**

Agenda

1. Attendance
2. Announcements
3. List Mutation
4. Check Your Understanding
5. Orders of Growth
6. Nonlocal (fast)

Attendance

Sign in at bit.do/jerrydisc

OR

Come to me for check-in

Announcements

Maps due Today!

- I hope you've started…

Lab feedback: bit.do/jerrylabfb

Discussion feedback: bit.do/jerrydiscfb

Whoops

Slicing is confusing (and probably still is). I'll try to not make it any worse this time

Default start and end index **depend on step size**

Start(inclusive):End(exclusive)

```
lst[::-1] ==
lst[len(lst)-1:-(len(lst)+1):-1]

lst[:] == lst[0:len(lst):1]
```

List Mutation

Static lists are great, but…

- Having to copy information can be wasteful!
- Would like to modify our existing lists

For more, see http://cs61a.org/disc/disc04.pdf

List Mutation

Operation	Description
`lst.append(x)`	Add x to the end of lst
`lst[1] = x`	Assign x to index 1
`lst = lst + [x]`	Append x to a **copy of** lst
`lst.remove(x)`	Remove first occurrence of x
`lst.pop(2)`	Remove **and** return element at index 2

```
>>> lst1 = [1, 2, 3]
>>> lst2 = [1, 2, 3]
>>> lst1 == lst2 #compares each value
True
>>> lst1 is lst2 #compares references
False
>>> lst2 = lst1
>>> lst2 is lst1
True
>>> lst1.append(4)
>>> lst1
[1, 2, 3, 4]
>>> lst2
[1, 2, 3, 4]
```

```
>>> lst2[1] = 42
>>> lst2
[1, 42, 3, 4]
>>> lst1 = lst1 + [5]
>>> lst1 == lst2
False
>>> lst1
[1, 42, 3, 4, 5]
>>> lst2
[1, 42, 3, 4]
>>> lst2 is lst1
False
```

Check Your Understanding

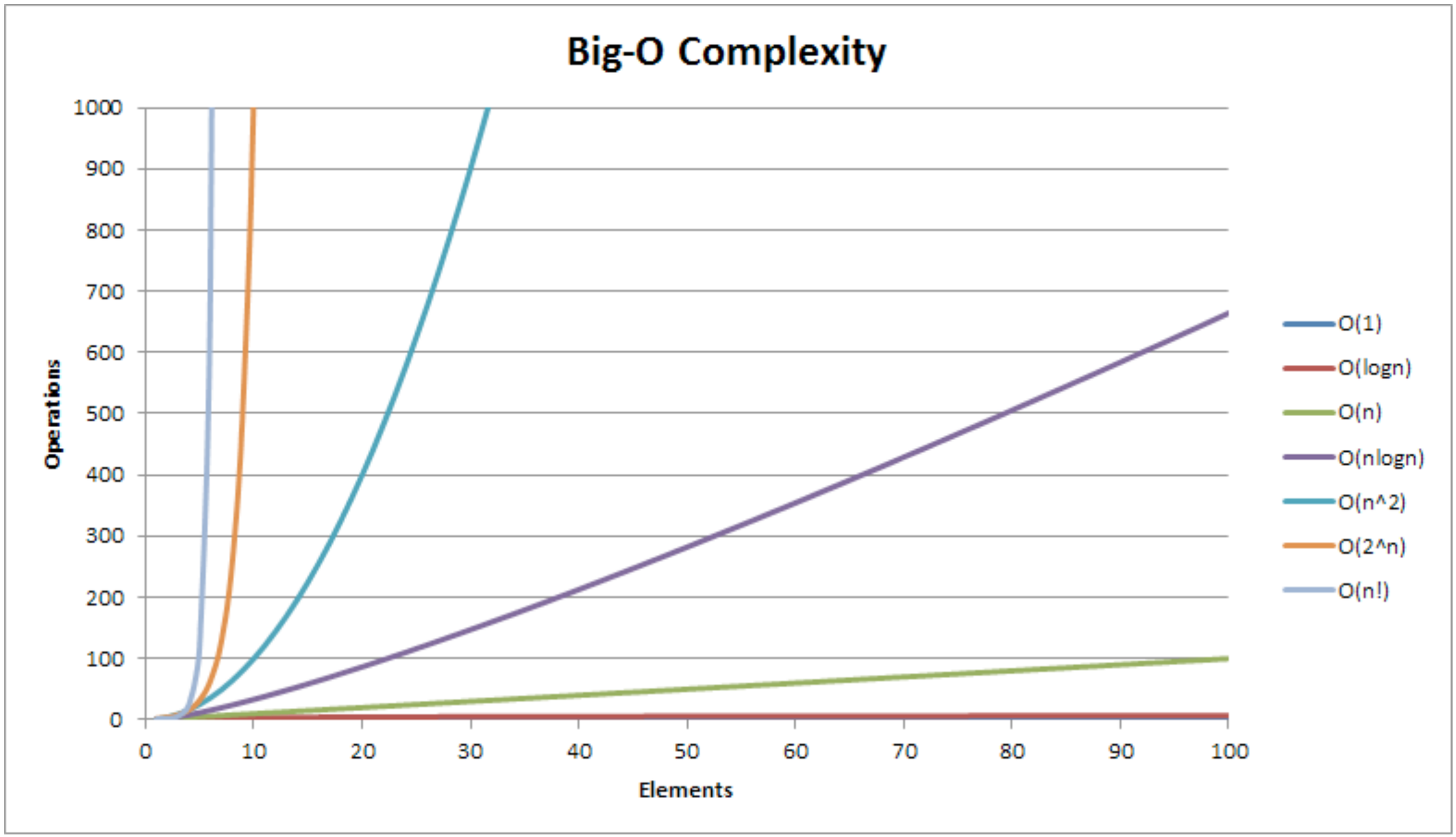
1.

```
def pairs_to_dict(pairs):
    """
    Convert a list of pairs into a dictionary.
     >>> p = [['c', 6], ['s', 1], ['c', 'a']]
    >>> pairs_to_dict(p)
    {'c': 'a', 's': 1}
    """
```

2.

```
def remove_all(el, lst):
    """
    >>> x = [3, 1, 2, 1, 5, 1, 1, 7]
    >>> remove_all(1, x)
    >>> x
    [3, 2, 5, 7]
    """
```

Orders of Growth

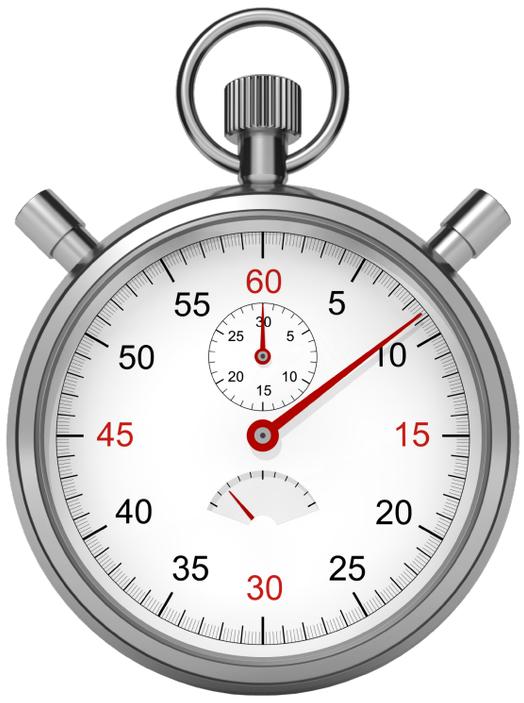


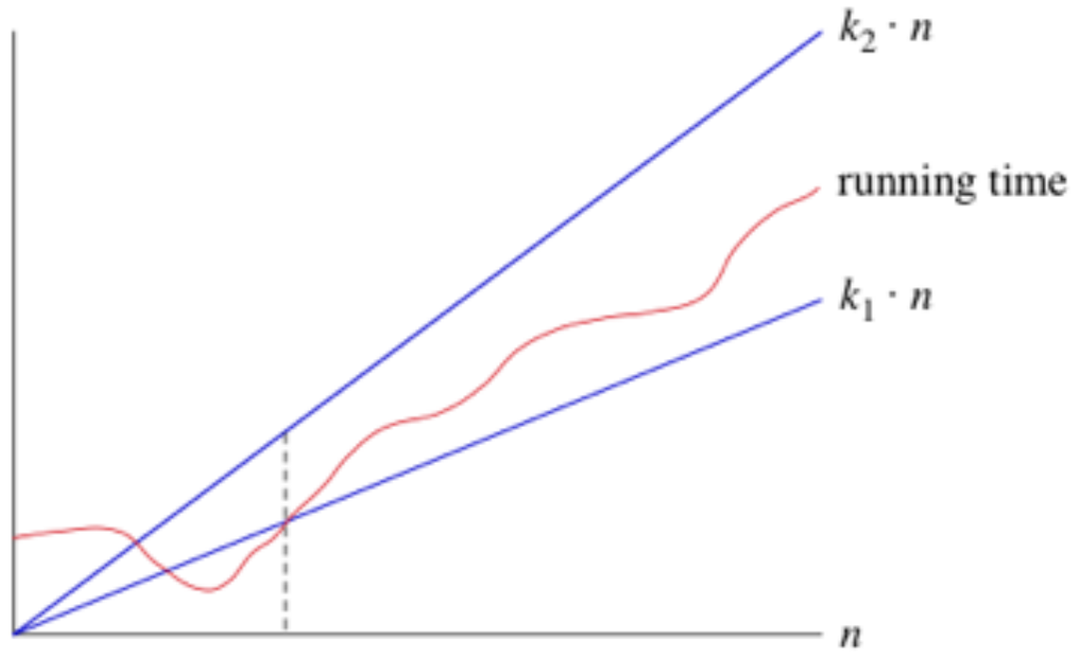
http://bigocheatsheet.com/img/big-o-complexity.png

Orders of Growth

How do we describe how fast a program is?

- Orders of growth — "as input size changes, how does run time?"





Orders of Growth

Time (μsec) for problem size N	Max N Possible in 1 second	1 hour	1 month	1 century
$\lg N$	10^{300000}	$10^{1000000000}$	$10^{8 \cdot 10^{11}}$	$10^{9 \cdot 10^{14}}$
N	10^6	$3.6 \cdot 10^9$	$2.7 \cdot 10^{12}$	$3.2 \cdot 10^{15}$
$N \lg N$	63000	$1.3 \cdot 10^8$	$7.4 \cdot 10^{10}$	$6.9 \cdot 10^{13}$
N^2	1000	60000	$1.6 \cdot 10^6$	$5.6 \cdot 10^7$
N^3	100	1500	14000	150000
2^N	20	32	41	51

Orders of Growth

Simplify	Answer
$\theta(3n)$	$\theta(n)$ — ignore const factors
$\theta(n^3 + 1000n^2)$	$\theta(n^3)$ — larger term dominates
$\theta(\log n + n)$	$\theta(n)$ — larger term dominates
$\theta(n \log n + n)$	$\theta(n \log n)$ — larger term dominates

Caveat — **these are NOT mathematically precise ways of describing growth relationships!

Orders of Growth

Question**	Answer
$\Theta(\log_2 n) > \Theta(\log_{10} n)$	**No** — Use change of base formula.
$\Theta(n \log(n^8)) > \Theta(n^2 \log(n^3))$	**No** — use log rules to get $\Theta(n \log n)$ vs $\Theta(n^2 \log n)$
$\Theta(n \log n) < \Theta((\log n)^{\log n})$	**Yes** — RHS is $n^{\log \log n}$. Or take log of both sides.

Orders of Growth

Why do we care?

In the news

Google's DeepMind defeats legendary Go player Lee Se-dol in historic victory

The Verge - 1 day ago

DeepMind founder Demis Hassabis expressed "huge respect for Lee Se-dol and his ...

Match 1 - Google DeepMind Challenge Match: Lee Sedol vs AlphaGo

YouTube - 1 day ago

Google's Deepmind AI beats Go world champion in first match

Engadget - 23 hours ago

More news for deepmind

Nonlocal

Laughing tourist with map at Rio de Janeiro (a "non-local")

Nonlocal

Why do we need nonlocal?

What will be the result of the output below?

```
1 def mdfy(x):
2     def inner():
3         x = 10
4         x = x + 2
5     inner()
6     return x
```

A	10
B	20
C	12
D	22
E	Error

```
>>> x = mdfy(20)
>>> x
```

_____

Nonlocal

Why do we need nonlocal?

What will be the result of the output below?

```
1 def mdfy(x):
2     def inner():
3         x = 10
4         x = x + 2
5     inner()
6     return x
```

```
>>> x = mdfy(20)
>>> x
```

A	10
B	20
C	12
D	22
E	Error

Nonlocal

What's happening in inner()?

- We created a local variable x and assigned 10.
- Then, we incremented that local variable by 2.
- The one in "mdfy" is unchanged!

```
1 def mdfy(x):
2     def inner():
3         x = 10
4         x = x + 2
5     inner()
6     return x
```

Nonlocal

Let's try again.
What will happen here?

```
 8 def mdfy2(x):
 9     def inner():
10         x = x + 10
11     inner()
12     return x
```

A	10
B	20
C	30
D	40
E	Error

```
>>> x = mdfy2(20)
>>> x
```

_____

Nonlocal

Let's try again.
What will happen here?

```
 8 def mdfy2(x):
 9     def inner():
10         x = x + 10
11     inner()
12     return x
```

A	10
B	20
C	30
D	40
E	Error

```
>>> x = mdfy2(20)
>>> x
```

_____

Nonlocal

Uh oh. This is even worse!

```
 8 def mdfy2(x):
 9     def inner():
10         x = x + 10
11     inner()
12     return x
```

- **Can** lookup x from parent frame
- **Cannot** also bind to an x in the current frame
- Confusingly, this will give an “unbound local error” claiming we referenced x before assignment (Read 2.4.4 in your textbook)

Nonlocal

As you may have guessed, nonlocal is required.
Here's the proper syntax:

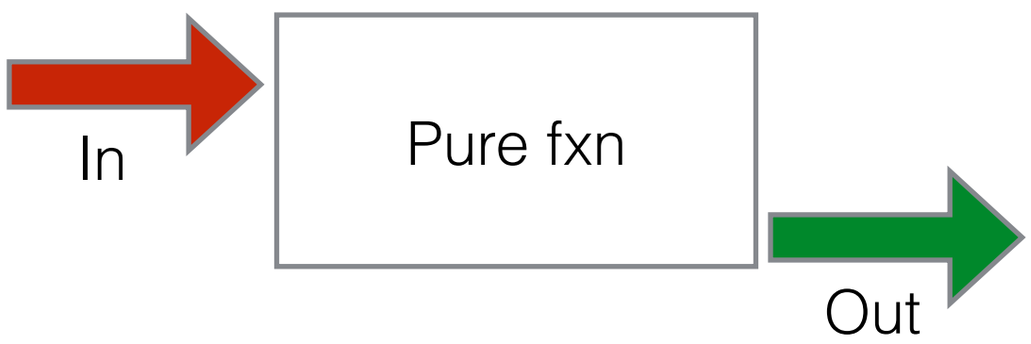
```
14 def mdfy3(x):
15     def inner():
16         nonlocal x
17         x = x + 1
18     inner()
19     return x
```

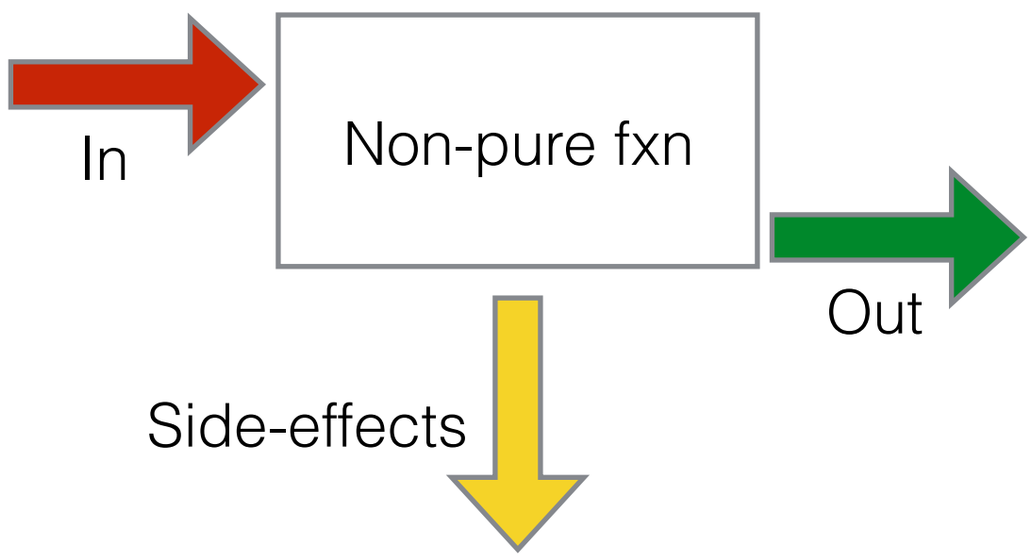
```
>>> x = mdfy3(20)
>>> x
21
```

Nonlocal

Exercise Caution:

- Nonlocal functions are non-pure
- As a reminder:





Side effects could be printing, nonlocal assignment, writing to a file, etc.