[320] Complexity

Tyler Caraza-Harter

Review

The situation where git cannot auto-merge is called a ________________

What is the missing step?

1. nano file.txt
2. ????
3. git commit -m "I changed file.txt"
4. git push

What type does check_output return?

How can you use time.time() to measure an operation that is much faster than calling time.time()?

Today's Reading

Required: Think Python, Appendix B

http://www.greenteapress.com/thinkpython/html/thinkpython022.html (skip B.4)

Optional [math heavy]:

http://web.mit.edu/16.070/www/lecture/big_o.pdf

Complexity

Performance vs. Complexity

Things that affect performance (total time to run):

- speed of the computer (CPU, etc)
- speed of Python (quality+efficiency of interpretation)
- **algorithm**: strategy for solving the problem
- **input size:** how much data do we have?

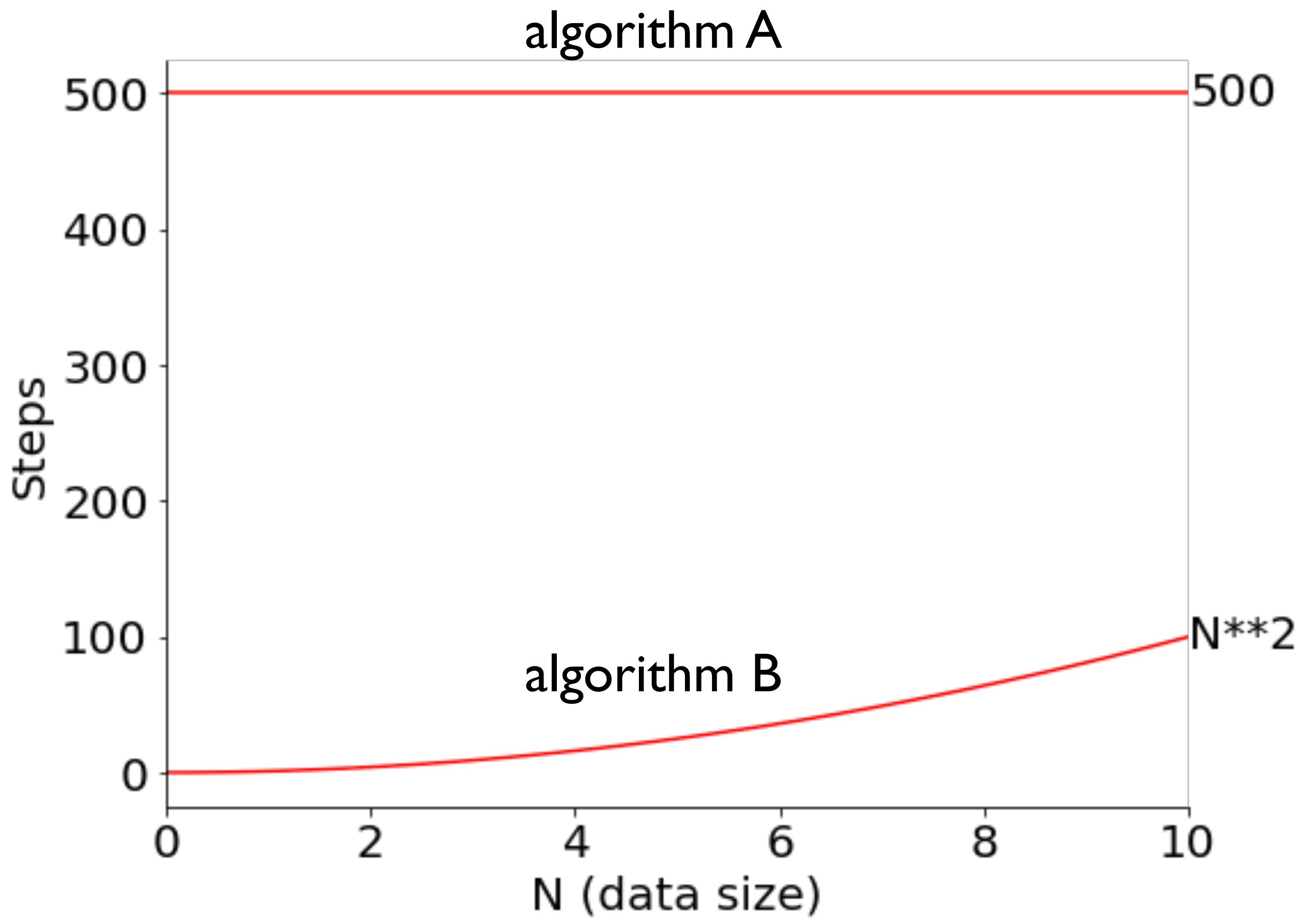
Performance vs. Complexity

Things that affect performance (total time to run):

- speed of the computer (CPU, etc)
- speed of Python (quality+efficiency of interpretation)
- **algorithm**: strategy for solving the problem
- **input size:** how much data do we have?

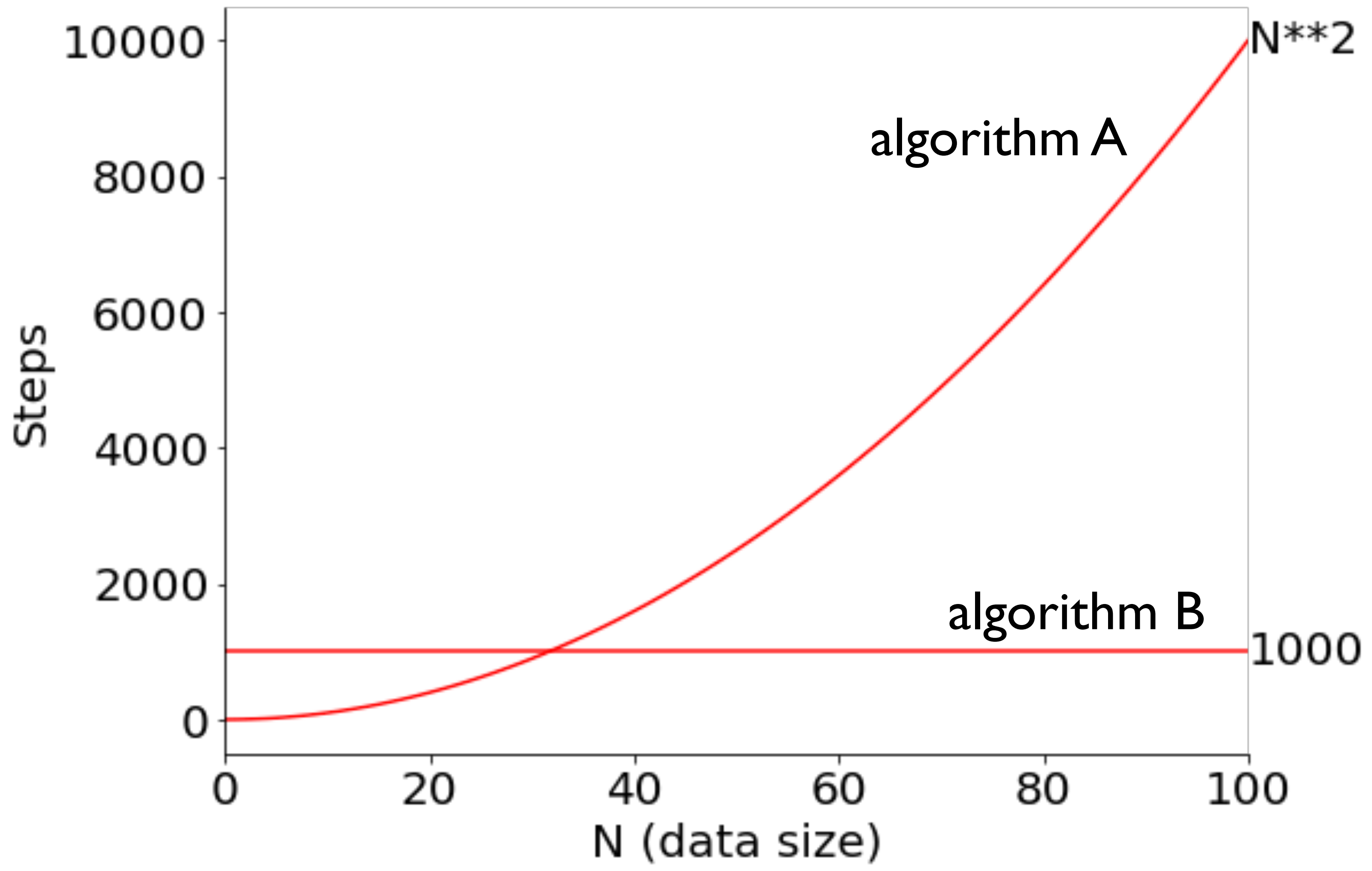
complexity analysis: how many steps must the algorithm perform, as a function of input size?

Which algorithm is better?



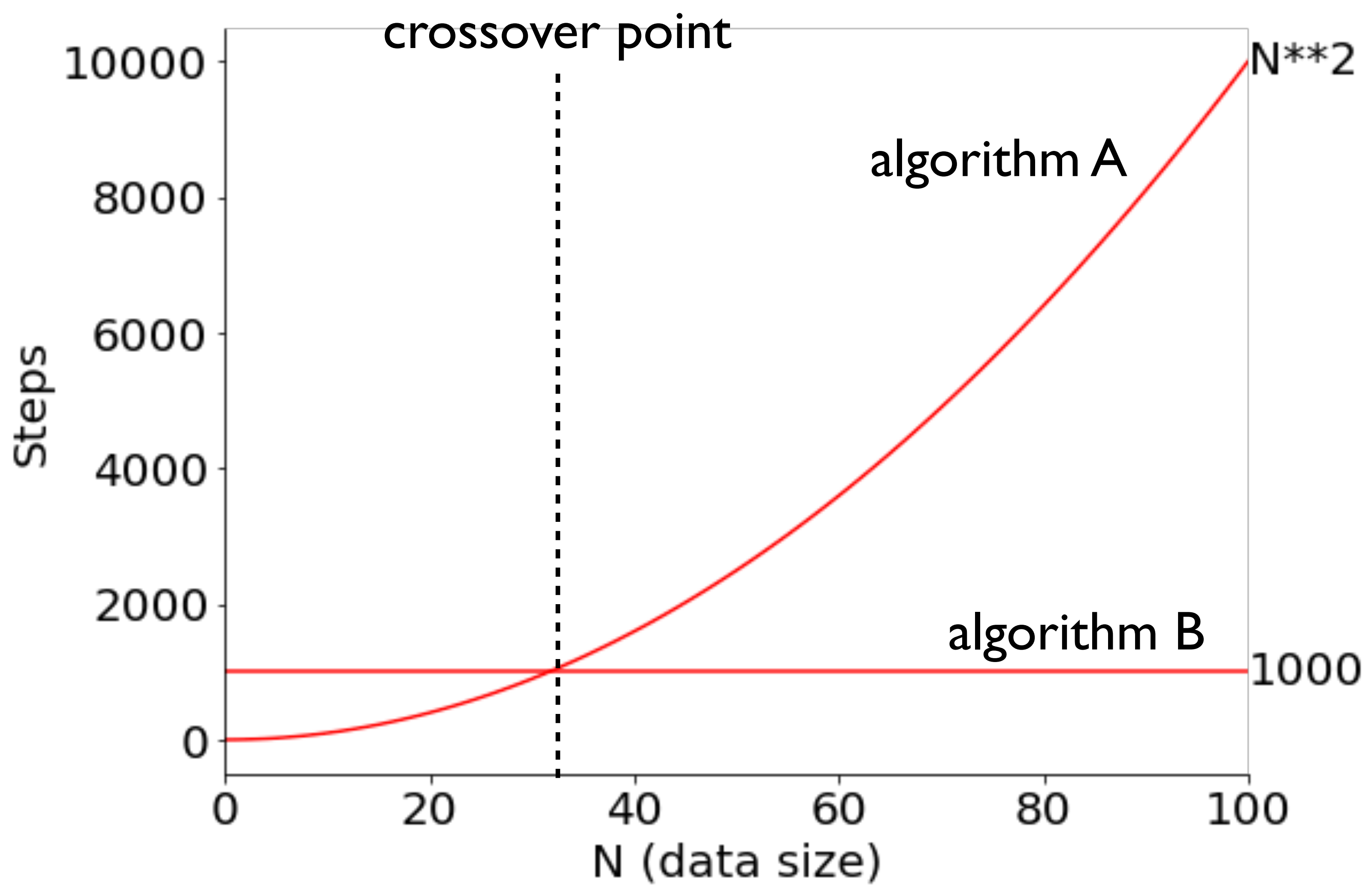
Do you prefer A or B?

Which algorithm is better?



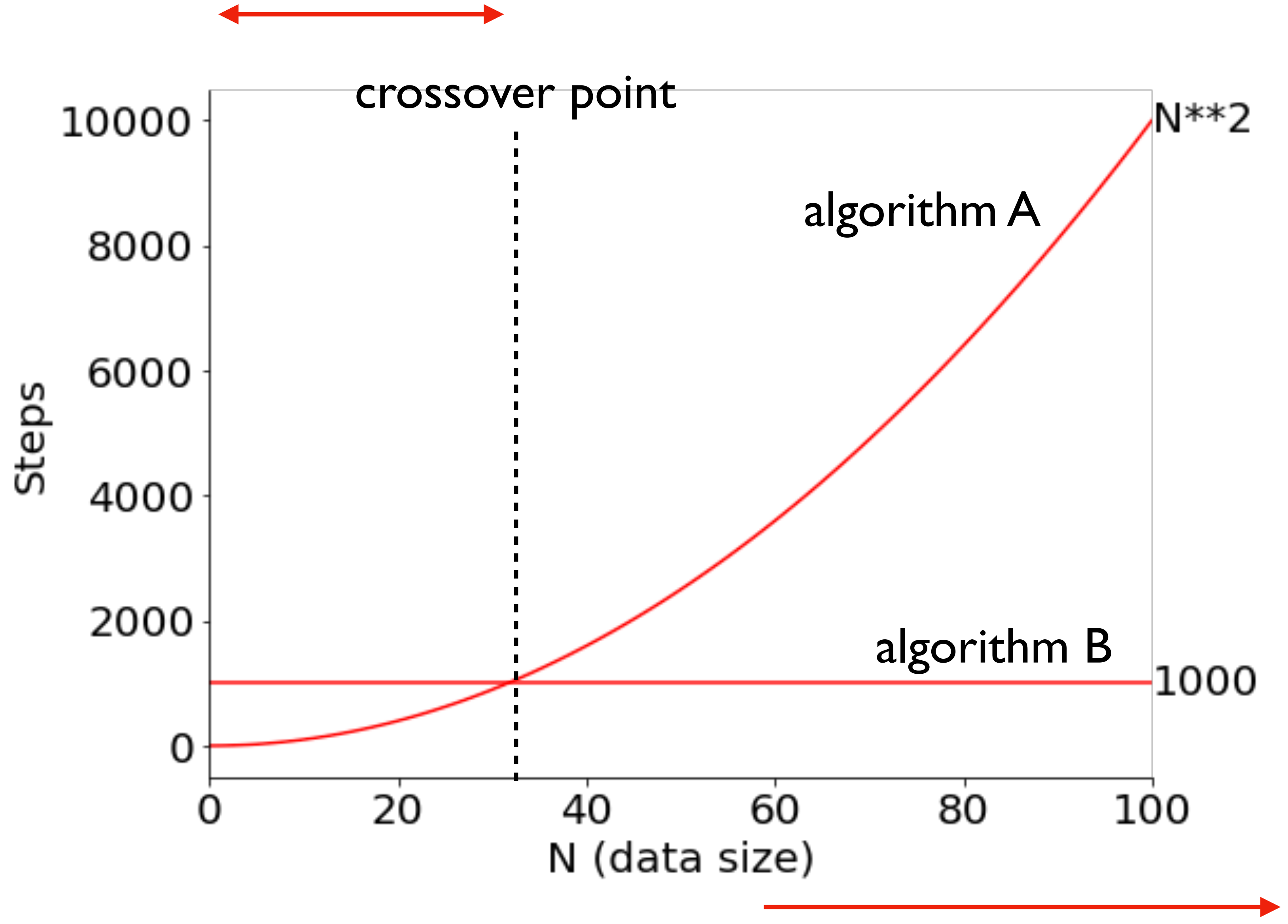
Do you prefer A or B?

Which algorithm is better?



Which algorithm is better?

you might still reasonably care about this portion!



complexity analysis only cares about "big" inputs

What is the asymptotic behavior of the function?

Performance vs. Complexity

Things that affect performance (total time to run):

- speed of the computer (CPU, etc)
- speed of Python (quality+efficiency of interpretation)
- **algorithm:** strategy for solving the problem
- **input size:** how much data do we have?

complexity analysis: how many steps must the algorithm perform, as a function of input size?

Performance vs. Complexity

Things that affect performance (total time to run):

- speed of the computer (CPU, etc)
- speed of Python (quality+efficiency of interpretation)
- **algorithm**: strategy for solving the problem
- **input size:** how much data do we have?

what is this?

complexity analysis: how many **steps** must the algorithm perform, as a function of input size?

What is a "step"?

What is a step?

A **step** is any unit of work that has a time to execute that **does not** depend on the input size

input size is length of this list

```
           input_nums = [2, 3, ...]

STEP       odd_count = 0
STEP       odd_sum = 0
STEP       for num in input_nums:
STEP           if num % 2 == 1:
STEP               odd_count += 1
STEP               odd_sum += num
STEP       odd_avg = odd_sum
STEP       odd_avg /= odd_count
```

✓

What is a step?

A **step** is any unit of work that has a time to execute that **does not** depend on the input size

```
      input_nums = [2, 3, ...]

STEP  odd_count = 0
      odd_sum = 0
STEP  for num in input_nums:
STEP      if num % 2 == 1:
STEP          odd_count += 1
              odd_sum += num
STEP  odd_avg = odd_sum
      odd_avg /= odd_count
```

✓

What is a step?

A **step** is any unit of work that has a time to execute that **does not** depend on the input size

```
        input_nums = [2, 3, ...]

STEP    odd_count = 0
        odd_sum = 0
STEP    for num in input_nums:
STEP        if num % 2 == 1:
STEP            odd_count += 1
                odd_sum += num
STEP    odd_avg = odd_sum / odd_count
```

✓

One line can do a lot, so no reason to have lines and steps be equivalent

What is a step?

A **step** is any unit of work that has a time to execute that **does not** depend on the input size

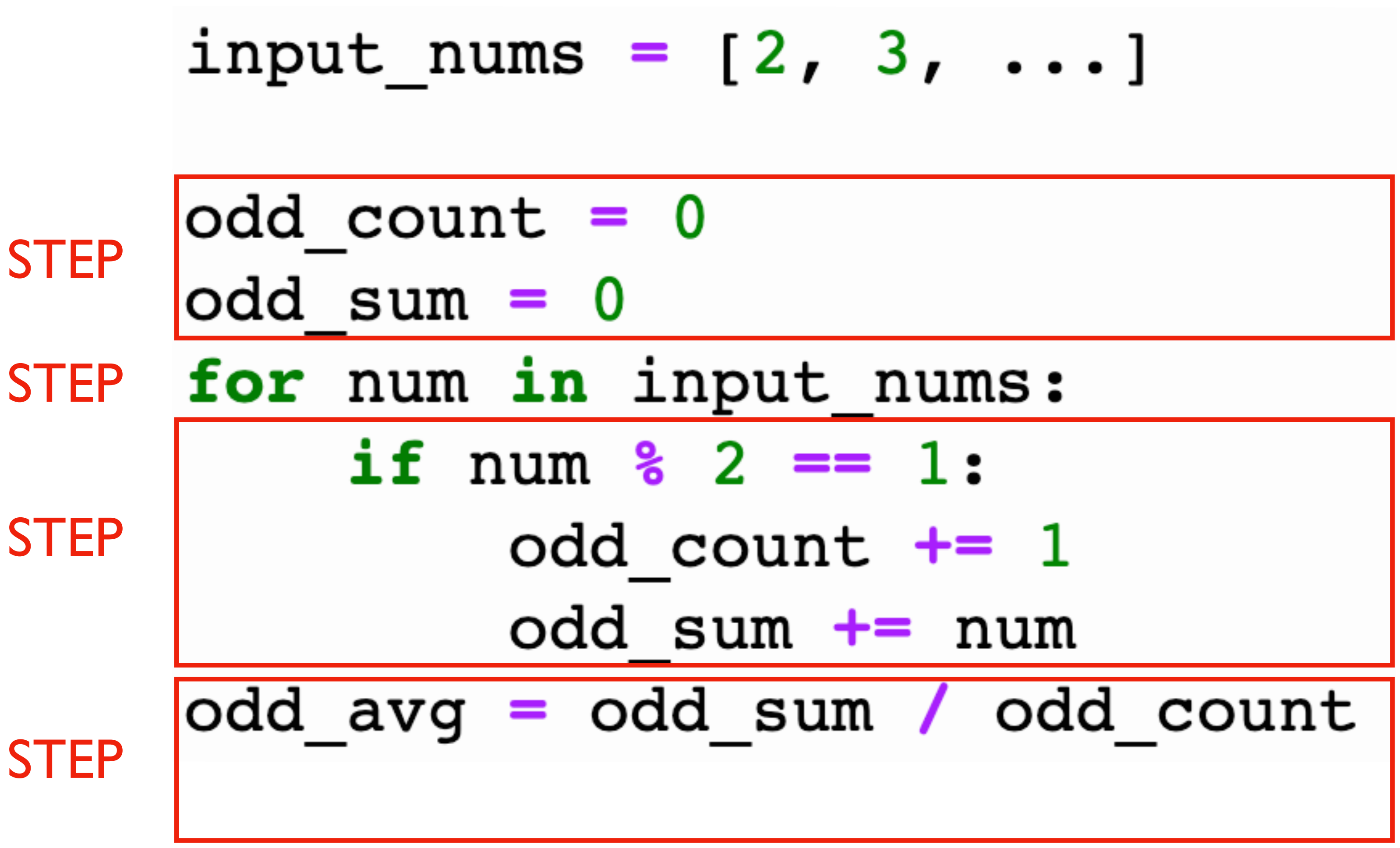
```
        input_nums = [2, 3, ...]

STEP    odd_count = 0
        odd_sum = 0
STEP    for num in input_nums:
STEP        if num % 2 == 1:
                odd_count += 1
                odd_sum += num
STEP    odd_avg = odd_sum / odd_count
```

???

What is a step?

A **step** is any unit of work that has a time to execute that **does not** depend on the input size

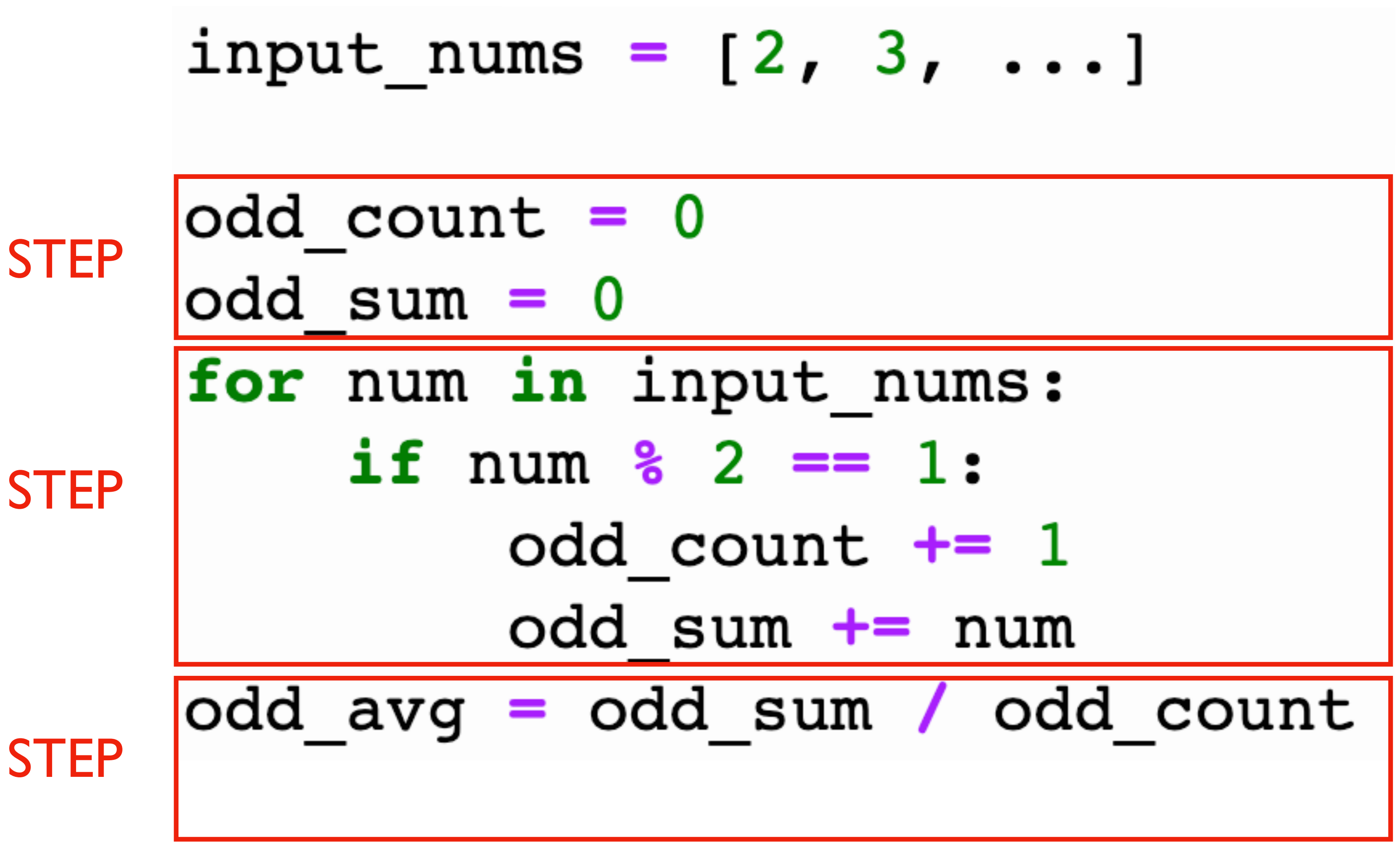


```python
input_nums = [2, 3, ...]

odd_count = 0
odd_sum = 0
for num in input_nums:
    if num % 2 == 1:
        odd_count += 1
        odd_sum += num
odd_avg = odd_sum / odd_count
```

What is a step?

A **step** is any unit of work that has a time to execute that **does not** depend on the input size



```
input_nums = [2, 3, ...]

odd_count = 0
odd_sum = 0
for num in input_nums:
    if num % 2 == 1:
        odd_count += 1
        odd_sum += num
odd_avg = odd_sum / odd_count
```

???

is this a valid way to identify steps?

What is a step?

A **step** is any unit of work that has a time to execute that **does not** depend on the input size

```
input_nums = [2, 3, ...]

odd_count = 0
odd_sum = 0
for num in input_nums:
    if num % 2 == 1:
        odd_count += 1
        odd_sum += num
odd_avg = odd_sum / odd_count
```

STEP

STEP

not a "step", because exec time depends on input size

STEP

✗

Counting Executed Steps

A **step** is any unit of work that has a time to execute that **does not** depend on the input size

```
          input_nums = [2, 3, ...]

STEP      odd_count = 0
          odd_sum = 0
STEP      for num in input_nums:
              if num % 2 == 1:
STEP              odd_count += 1
                  odd_sum += num
STEP      odd_avg = odd_sum / odd_count
```

How many total steps will **execute** if `len(input_nums) == 10`?

Counting Executed Steps

A **step** is any unit of work that has a time to execute that **does not** depend on the input size

```
input_nums = [2, 3, ...]

odd_count = 0
odd_sum = 0
for num in input_nums:
    if num % 2 == 1:
        odd_count += 1
        odd_sum += num
odd_avg = odd_sum / odd_count
```

1 STEP

+ 11 STEP

+ 10 STEP

+ 1 STEP

= 23 steps

For **N** elements, there will be **2*N+3** steps

Counting Executed Steps

A **step** is any unit of work that has a time to execute that **does not** depend on the input size

```
      input_nums = [2, 3, ...]

STEP  odd_count = 0
STEP  odd_sum = 0
STEP  for num in input_nums:
STEP      if num % 2 == 1:
STEP          odd_count += 1
STEP          odd_sum += num
STEP  odd_avg = odd_sum
STEP  odd_avg /= odd_count
```

How many total steps will **execute** if `len(input_nums) == 10`?

Counting Executed Steps

A **step** is any unit of work that has a time to execute that **does not** depend on the input size

```
                       input_nums = [2, 3, ...]

       1      STEP     odd_count = 0
     + 1      STEP     odd_sum = 0
     + 11     STEP     for num in input_nums:
     + 10     STEP         if num % 2 == 1:
+ 0 to 10     STEP             odd_count += 1
+ 0 to 10     STEP             odd_sum += num
     + 1      STEP     odd_avg = odd_sum
     + 1      STEP     odd_avg /= odd_count
```

For **N** elements, there will be between
2*N+5 and **4*N+5** steps

Counting Executed Steps

A **step** is any unit of work that has a time to execute that **does not** depend on the input size

```
input_nums = [2, 3, ...]

odd_count = 0
odd_sum = 0
for num in input_nums:
    if num % 2 == 1:
        odd_count += 1
        odd_sum += num
odd_avg = odd_sum / odd_count
```

Important: we might not identify steps the same, but our execution counts can at most differ by a constant factor!

can we broadly (but rigorously) categorize based on this?

Big O Notation ("O" is for "order of growth")

If $f(N) < C * g(N)$ *for large x values and some fixed <u>constant</u> C*

Then $f(N) \in O(g(N))$

Big O Notation ("O" is for "order of growth")

If $f(N) < C * g(N)$ *for large x values and some fixed* *<u>constant</u>* *C*

Then $f(N) \in O(g(N))$

---

which ones are true?

$2N \in O(N)$

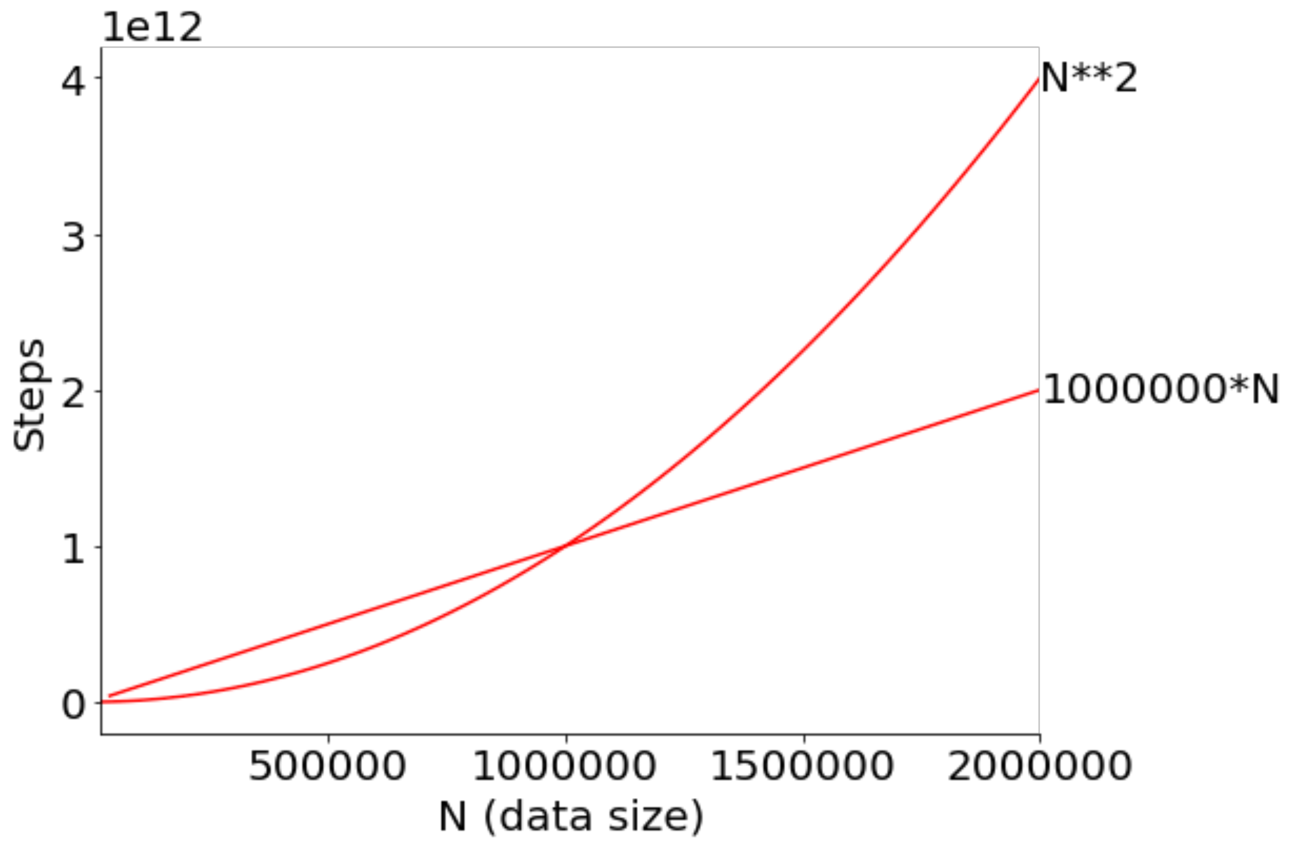
$100N \in O(N^2)$

$N^2 \in O(1000000N)$

Big O Notation ("O" is for "order of growth")

If $f(N) < C * g(N)$ *for large x values and some fixed constant C*

Then $f(N) \in O(g(N))$



$2N \in O(N)$

$100N \in O(N^2)$

~~$N^2 \in O(1000000N)$~~

Big O Notation ("O" is for "order of growth")

If $f(N) < C * g(N)$ *for large x values and some fixed constant C*

Then $f(N) \in O(g(N))$

---

which ones are true?

$N^2 \in O(N^2+N+1)$

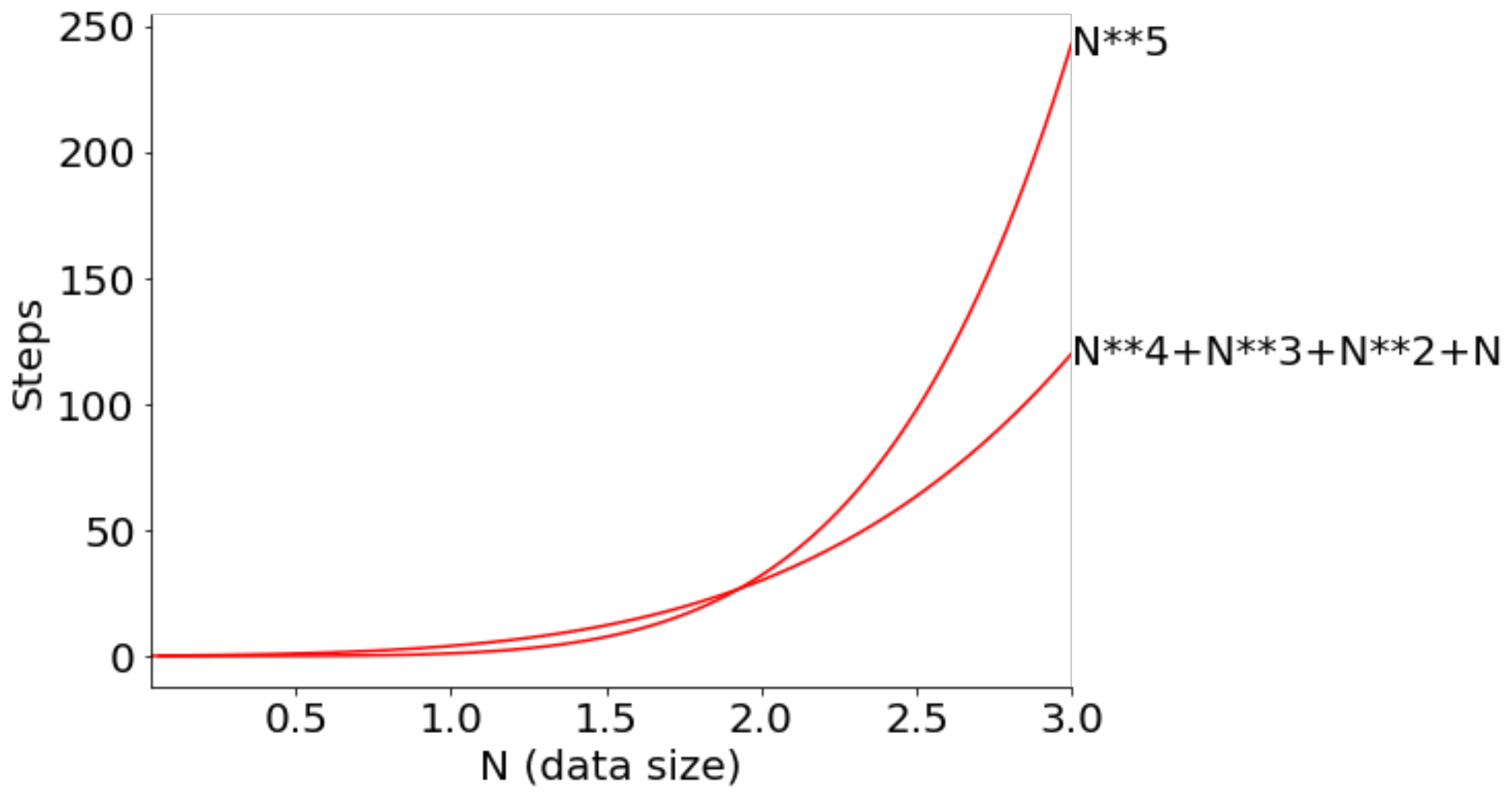
$N^2+N+1 \in O(N^2)$

$N^5 \in O(N^4 + N^3 + N^2 + N)$

Big O Notation ("O" is for "order of growth")

If $f(N) < C * g(N)$ *for large x values and some fixed constant C*

Then $f(N) \in O(g(N))$



$N^2 \in O(N^2+N+1)$

$N^2+N+1 \in O(N^2)$

~~$N^5 \in O(N^4 + N^3 + N^2 + N)$~~

TODO: find example: beginning vs. end. Worst case.

TODO: the "in" operator is not a step (or sum, sort, etc)

Summary of Terms

commit:	a snapshot of files at a point in time
HEAD:	a convenient label for the current commit
tag:	a label attached to a commit
branch:	a label attached to a commit that re-attaches to new commits
merge:	to combine changes on another branch into the current branch
conflict:	differences that cannot automatically be merged