

Data in scientific programming

Guillermo Aguilar & Pietro Berkes &
Zbigniew Jędrzejewski-Szmek & Victoria Shevchenko

Fork and clone repository
[https://git.aspp.school/
ASPP/2025-plovdiv-data.git](https://git.aspp.school/ASPP/2025-plovdiv-data.git)



Things one thinks about when thinking about data

Storage

- Size
- Access ease
- Access time

Processing

- Efficient processing (no for-loops!)
- Organizing data so that analyses are easy

Reproducibility and collaboration

- Versioning
- Lineage tracing (which script / other data was used to generate this?)
- Ease of sharing

Things one thinks about when thinking about data

Storage

- Size
- Access ease
- Access time

Processing

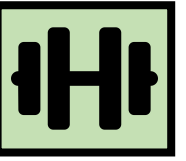
- Efficient processing (no for-loops!)
- Organizing data so that analyses are easy

Reproducibility and collaboration

- Versioning
- Lineage tracing (which script / other data was used to generate this?)
- Ease of sharing

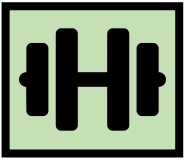
Hands-on

What data structure would you use to represent...

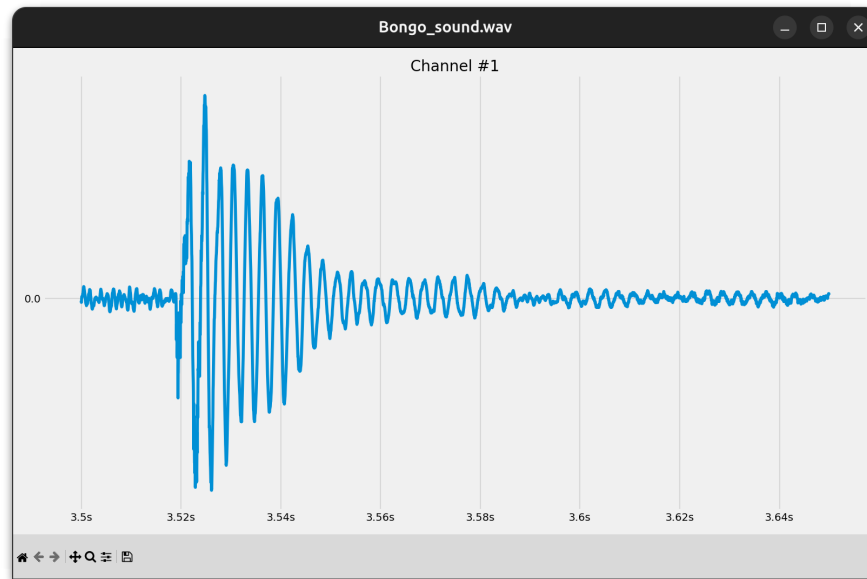


Hands-on

What data structure would you use to represent...

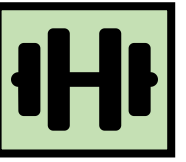


A sound wave?

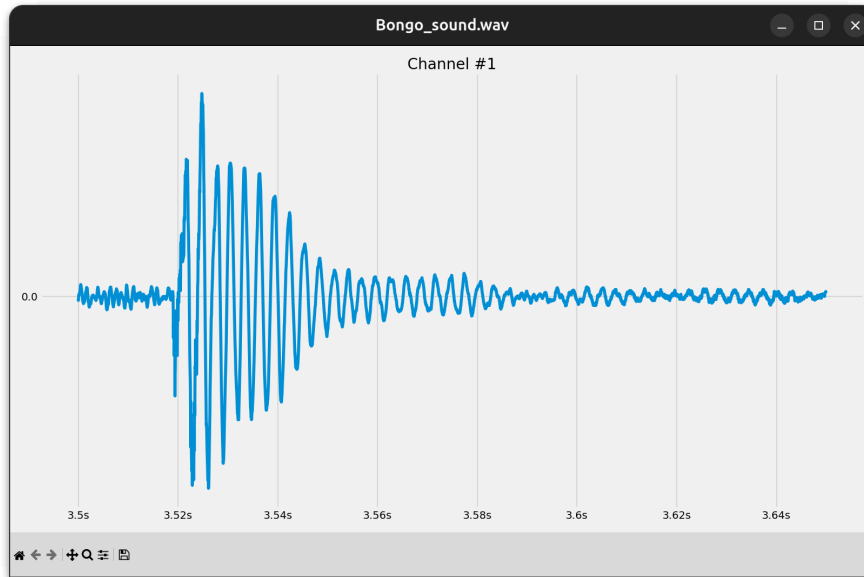


Hands-on

What data structure would you use to represent...



A sound wave?



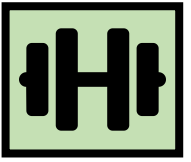
NumPy array

```
In [6]: sound_data
```

```
Out[6]: array([0.66709183, 0.55973494, 0.95416669, 0.60810949, 0.05188879,
0.58619063, 0.25555136, 0.72451477, 0.2646681 , 0.08694215,
0.75592186, 0.67261696, 0.62847452, 0.06232598, 0.20549438,
0.11718457, 0.25184725, 0.48625729, 0.8103058 , 0.18100915,
0.81113341, 0.62055231, 0.9046905 , 0.56664205, 0.73235338,
0.74382869, 0.64856368, 0.80644398, 0.46199345, 0.78516632,
0.91298397, 0.48290914, 0.20847714, 0.99162659, 0.26374781,
0.3602381 , 0.07173351, 0.8584085 , 0.32248766, 0.39167573,
0.67944923, 0.00930429, 0.21714217, 0.58810089, 0.17668711,
0.57444803, 0.25760187, 0.43785728, 0.39119371, 0.68268063,
0.95954499, 0.45934239, 0.03616905, 0.23896063, 0.61872801,
0.76332531, 0.96272817, 0.57169277, 0.50225193, 0.01361629,
0.15357459, 0.8057233 , 0.0642748 , 0.95013941, 0.38712684,
0.97231498, 0.20261775, 0.74184693, 0.26629893, 0.84672705,
0.67662718, 0.96055977, 0.64942314, 0.66487937, 0.86867536,
0.40815661, 0.1139344 , 0.95638066, 0.87436447, 0.18407227,
0.64457074, 0.19233097, 0.24012179, 0.90399279, 0.39093908,
0.26389161, 0.97537645, 0.14209784, 0.75261696, 0.10078122,
0.87468408, 0.77990102, 0.92983283, 0.45841805, 0.61470669,
0.87939755, 0.09266009, 0.41177209, 0.46973971, 0.43152144])
```

Hands-on

What data structure would you use to represent...

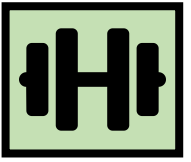


A map between color names and
RGB values?



Hands-on

What data structure would you use to represent...



A map between color names and
RGB values?

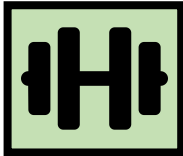
Dictionary



```
colors_hex = {  
    "tawny orange": "#CD5700",  
    "very peri": "#6667AB",  
    "iced coffee": "#C5A582",  
    "pink flambé": "#DC4C8B",  
    # ...  
}
```


Hands-on

What data structure would you use to represent...

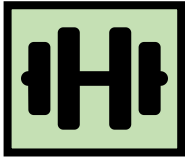


Phone book entries?

San Francisco	415 729-9283	Timothy	415
Uta	415	Uta	415
Harrington's Moving & Storage	415	Harrington's Moving & Storage	415
4415 Paradise Dr Tibrn	415	4415 Paradise Dr Tibrn	415
HARRIS Adam 106 Baltimore Ave C M	415	HARRIS Adam 106 Baltimore Ave C M	415
Alan & Christine	415	Alan & Christine	415
Andrew & Mary 8 Via Capistrano Tibrn	415	Andrew & Mary 8 Via Capistrano Tibrn	415
Anne 102 Ryan Av M Vly	415	Anne 102 Ryan Av M Vly	415
Anne 102 Ryan Av M Vly	415	Anne 102 Ryan Av M Vly	415
Anne 102 Ryan Av M Vly	415	Anne 102 Ryan Av M Vly	415
Arlene L	415	Arlene L	415
B	415	B	415
Harris Bail bonds 775 E Blithedale AV M Vly	415	Harris Bail bonds 775 E Blithedale AV M Vly	415
HARRIS Barbara	415	HARRIS Barbara	415
Barbra	415	Barbra	415
Barry	415	Barry	415
Bernard & Bette	415	Bernard & Bette	415
Bernice	415	Bernice	415
Bourke	415	Bourke	415
Brent & Nanette 50 La Cuesta Lagntas	415	Brent & Nanette 50 La Cuesta Lagntas	415
C	415	C	415

Hands-on

What data structure would you use to represent...



Phone book entries?

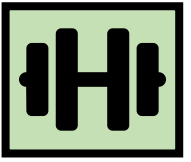
Francisco	415 729-9283	Timothy	415
415 729-9283	415 729-9283	Uta	415
415 729-9283	415 729-9283	Harrington's Moving & Storage	415
415 729-9283	415 729-9283	4415 Paradise Dr Tibrn	415
415 729-9283	415 729-9283	HARRIS Adam 106 Baltimore Ave C M	415
415 729-9283	415 729-9283	Alan & Christine	415
415 729-9283	415 729-9283	Andrew & Mary 8 Via Capistrano Tibrn	415
415 729-9283	415 729-9283	Anne 102 Ryan Av M Vly	415
415 729-9283	415 729-9283	Anne 102 Ryan Av M Vly	415
415 729-9283	415 729-9283	Anne 102 Ryan Av M Vly	415
415 729-9283	415 729-9283	Arlene L	415
415 729-9283	415 729-9283	B	415
415 729-9283	415 729-9283	Harris Bail bonds 775 E Blithedale AV M Vly	415
415 729-9283	415 729-9283	HARRIS Barbara	415
415 729-9283	415 729-9283	Barbra	415
415 729-9283	415 729-9283	Barry	415
415 729-9283	415 729-9283	Bernard & Bette	415
415 729-9283	415 729-9283	Bernice	415
415 729-9283	415 729-9283	Bourke	415
415 729-9283	415 729-9283	Brent & Nanette 50 La Cuesta Lagntas	415

Pandas DataFrame

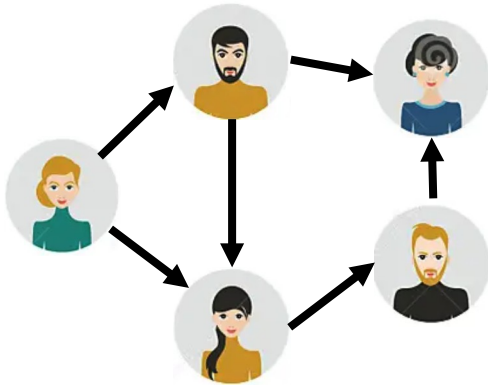
first_name	last_name	phone_nr	address	ZIP	city
John	Doe	555-1234	123 Maple St	12345	Springfield
Jane	Smith	555-5678	456 Oak St	67890	Rivertown
Alice	Johnson	555-8765	789 Pine St	54321	Lakeside
Bob	Brown	555-4321	321 Birch St	09876	Hilltop
Emma	Davis	555-7890	654 Elm St	11223	Greendale

Hands-on

What data structure would you use to represent...

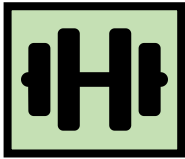


Friendship relations?

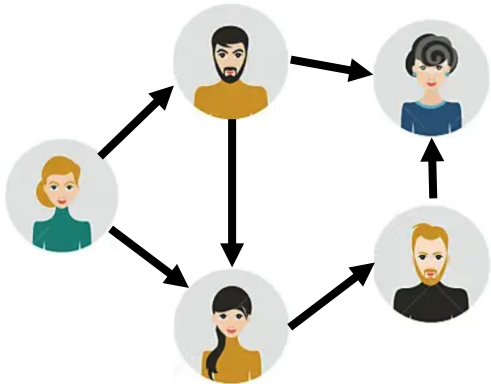


Hands-on

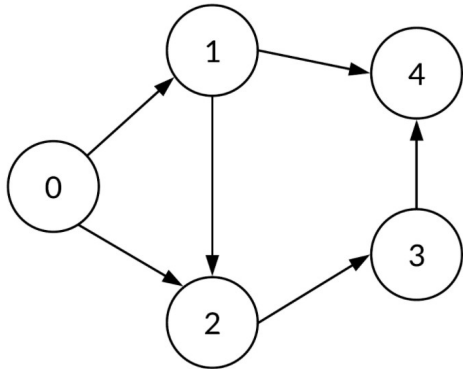
What data structure would you use to represent...



Friendship relations?



Graph



Implemented as

	0	1	2	3	4
0	0	1	1	0	0
1	0	0	1	0	1
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	0	0	0

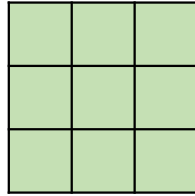
Adjacency matrix
(array)

```
A_dict = {
    '0': [1, 2],
    '1': [2],
    '2': [3],
    '3': [4],
    '4': []
}
```

Dictionary

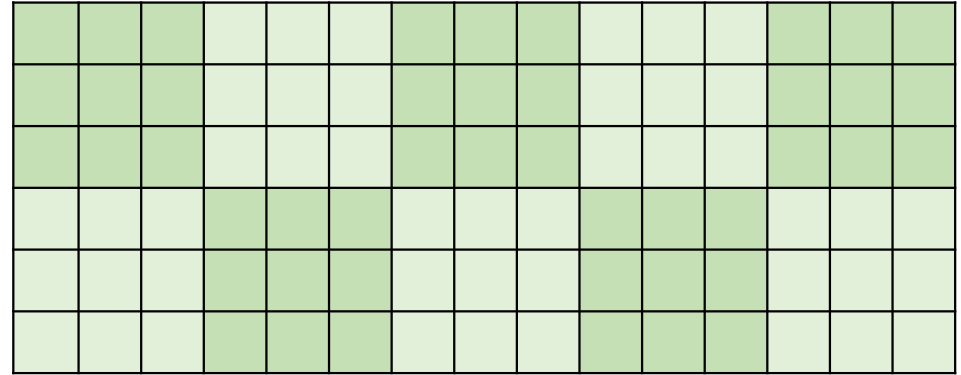
You develop your code on a small data set, how is it going to scale to the complete data set?

Development data



N data points,
Processing time T

Real data

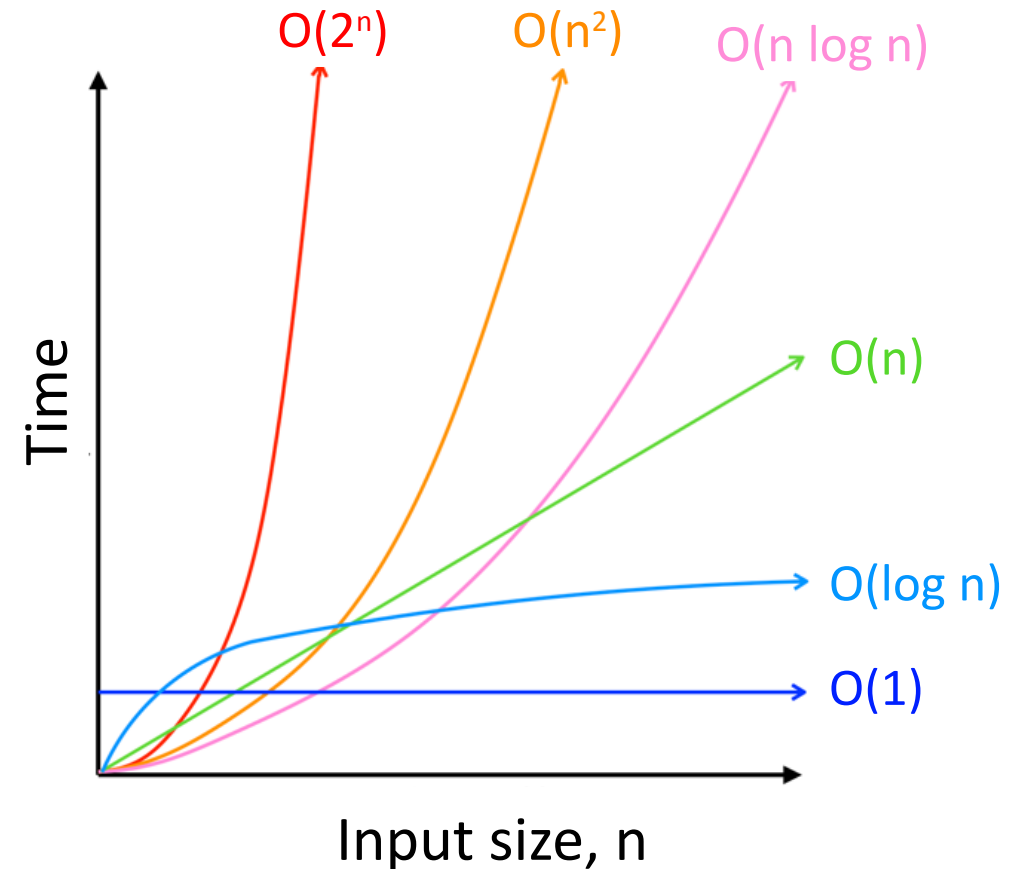


10x N data points
Processing time -> ?

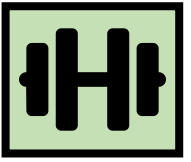
We're interested in orders of magnitude

How performance scales: big-O

Big-O class	What we call it	Time increase, when data increases 10x
$O(1)$	constant	1x time
$O(n)$	linear	10x time
$O(n^2)$	quadratic	100x time
$O(n \times \log n)$	linearithmic	~10-20x time
$O(\log n)$	logarithmic	~1-2x time

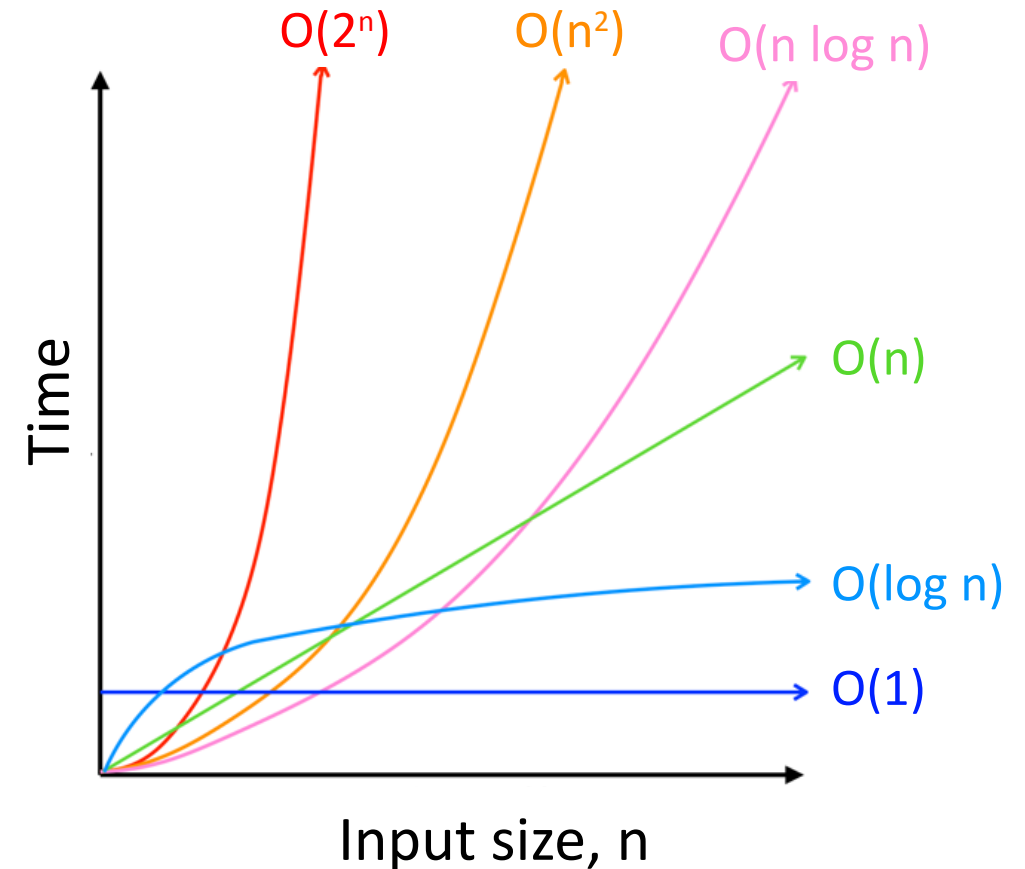


Hands-on: Operations on lists

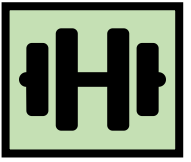


Big-O class	What we call it	Time increase, when data increases 10x
$O(1)$	constant	1x time
$O(n)$	linear	10x time
$O(n^2)$	quadratic	100x time
$O(n \times \log n)$	linearithmic	~10-20x time
$O(\log n)$	logarithmic	~1-2x time

Big-O class	Operation on lists that scales this way
$O(1)$	
$O(n)$	
$O(n^2)$	
$O(n \times \log n)$	
$O(\log n)$	

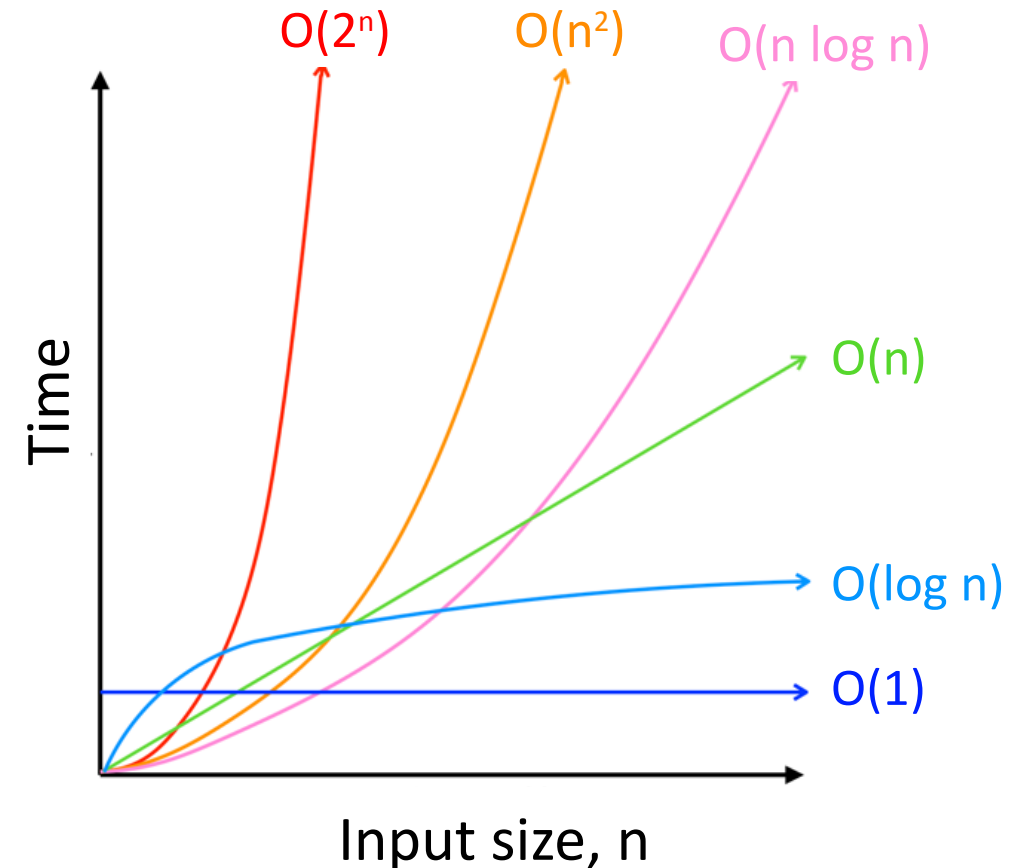


Hands-on: Operations on lists



Big-O class	What we call it	Time increase, when data increases 10x
$O(1)$	constant	1x time
$O(n)$	linear	10x time
$O(n^2)$	quadratic	100x time
$O(n \times \log n)$	linearithmic	~10-20x time
$O(\log n)$	logarithmic	~1-2x time

Big-O class	Operation on lists that scales this way
$O(1)$	Getting an element by its index
$O(n)$	Summing elements in list
$O(n^2)$	Computing distance between all pairs of elements in the list
$O(n \times \log n)$	Sorting the list
$O(\log n)$	Searching an element in a sorted list



Example: Find common words

Given two lists of words, extract all the words that are in common

```
words1 = ['apple', 'orange', 'banana', 'melon', 'peach']  
words2 = ['orange', 'kiwi', 'avocado', 'apple', 'banana']
```

Expected result: ['apple', 'orange', 'banana']

Implementation with two for-loops

```
words1 = ['apple', 'orange', 'banana', 'melon', 'peach']
words2 = ['orange', 'kiwi', 'avocado', 'apple', 'banana']

common = []
for x in words1:
    if x in words2:
        common += [x]
```

What is the big-O complexity of this implementation?

Implementation with two for-loops

```
words1 = ['apple', 'orange', 'banana', 'melon', 'peach']
words2 = ['orange', 'kiwi', 'avocado', 'apple', 'banana']

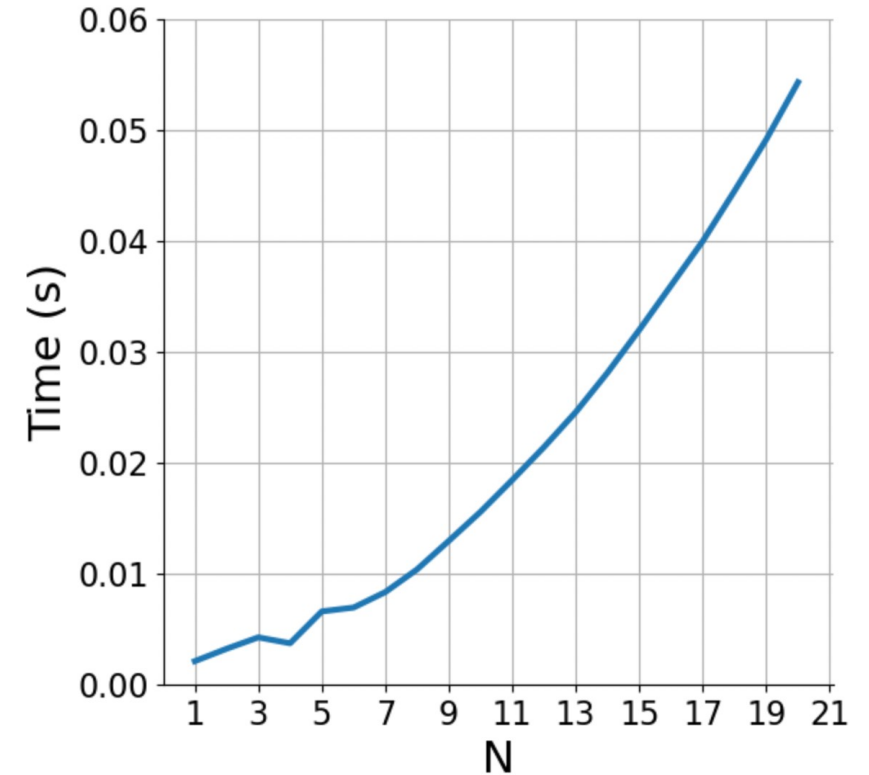
common = []
for x in words1:
    for y in words2:
        if x == y:
            common += [x]
```

What is the big-O complexity of this implementation?

Implementation with two for-loops

```
words1 = ['apple', 'orange', 'banana', 'melon', 'peach']
words2 = ['orange', 'kiwi', 'avocado', 'apple', 'banana']

common = []
for w in words1:          #  $O(N)$ 
    if w in words2:        #  $O(N)$ 
        common.append(w)  #  $O(1)$ 
```



What is the big-O complexity of this implementation?

$N * N \sim O(N^2)$

Implementation with sorted lists

```
words1 = ['apple', 'orange', 'banana', 'melon', 'peach']
words2 = ['orange', 'kiwi', 'avocado', 'apple', 'banana']

words1 = sorted(words1) # ['apple', 'banana', 'melon', 'orange', 'peach']
words2 = sorted(words2) # ['apple', 'avocado', 'banana', 'kiwi', 'orange']

common = []
idx2 = 0
for w in words1:
    while idx2 < len(words2) and words2[idx2] < w:
        idx2 += 1

    if idx2 >= len(words2):
        break

    if words2[idx2] == w:
        common.append(w)
```

What is the big-O complexity of this implementation?

Implementation with sorted lists

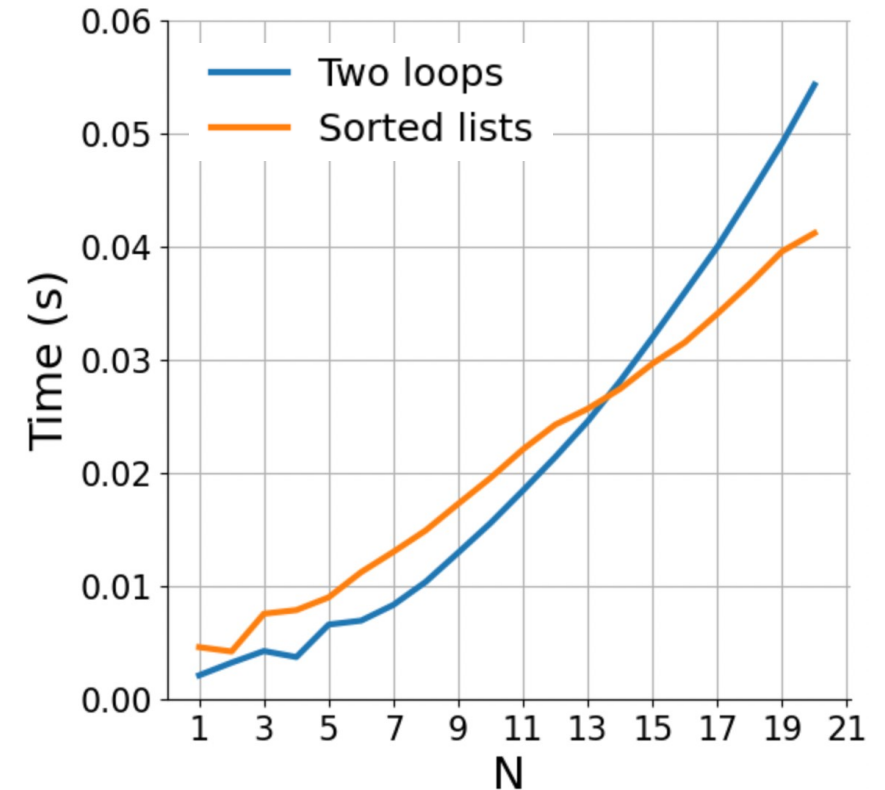
```
words1 = ['apple', 'orange', 'banana', 'melon', 'peach']
words2 = ['orange', 'kiwi', 'avocado', 'apple', 'banana']

words1 = sorted(words1)      #  $O(N * \log(N))$ 
words2 = sorted(words2)      #  $O(N * \log(N))$ 

common = []
idx2 = 0
for w in words1:              #  $O(N)$ 
    while idx2 < len(words2) and words2[idx2] < w: #  $O(N)$  in total
        idx2 += 1

    if idx2 >= len(words2): #  $O(1)$ 
        break

    if words2[idx2] == w:    #  $O(1)$ 
        common.append(w)
```



What is the big-O complexity of this implementation?

$$2 \times (N \times \log N) + 2 \times N \sim \mathbf{O(N \log N)}$$

Implementation with sets

```
words1 = ['apple', 'orange', 'banana', 'melon', 'peach']
words2 = ['orange', 'kiwi', 'avocado', 'apple', 'banana']

words2 = set(words2)

common = []
for w in words1:
    if w in words2:
        common.append(w)
```

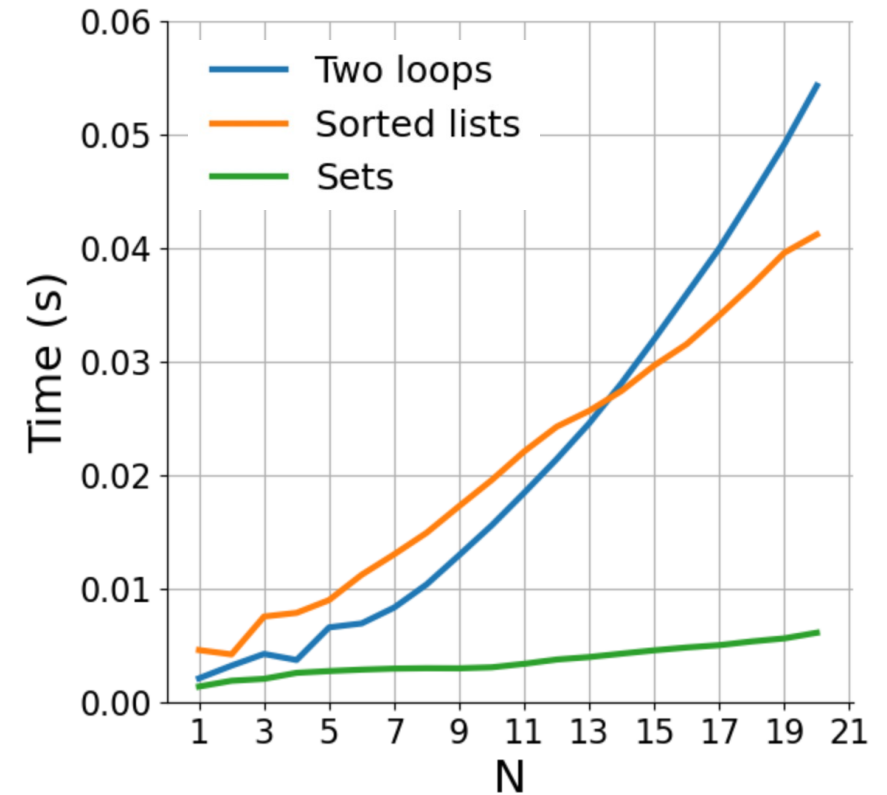
What is the big-O complexity of this implementation?

Implementation with sets

```
words1 = ['apple', 'orange', 'banana', 'melon', 'peach']
words2 = ['orange', 'kiwi', 'avocado', 'apple', 'banana']

words2 = set(words2)      # O(N)

common = []
for w in words1:          # O(N)
    if w in words2:        # O(1)
        common.append(w)  # O(1)
```



What is the big-O complexity of this implementation?

$N + N \sim \mathbf{O(N)}$

Basic reference sheet about Python data structures

Lists: collection of ordered, arbitrary data

Getting an element by index	$O(1)$
Appending	$O(1)$
Inserting an element at index	$O(n)$
Sorting	$O(n \log n)$
Finding an element by value (e.g., “if element in my_list: ...”, list.index, etc.)	$O(n)$
Copy a list	$O(n)$

Dictionaries (“hashmaps”)

Inserting	$O(1)$
Finding a value by key (e.g., “if element in my_dict: ...”)	$O(1)$
Create dictionary from lists	$O(n)$

Sets: it's dictionaries without values

Inserting	$O(1)$
Finding a value by key (e.g., “if element in my_set: ...”)	$O(1)$
Create set from list	$O(n)$

Hands-on



Exercise

`exercises/match_tarots`

- Open the notebook `match_tarots`, and follow the instructions!
- Submit a PR for Issue #1 on `git.aspp.school/ASPP/2025-plovdiv-data`

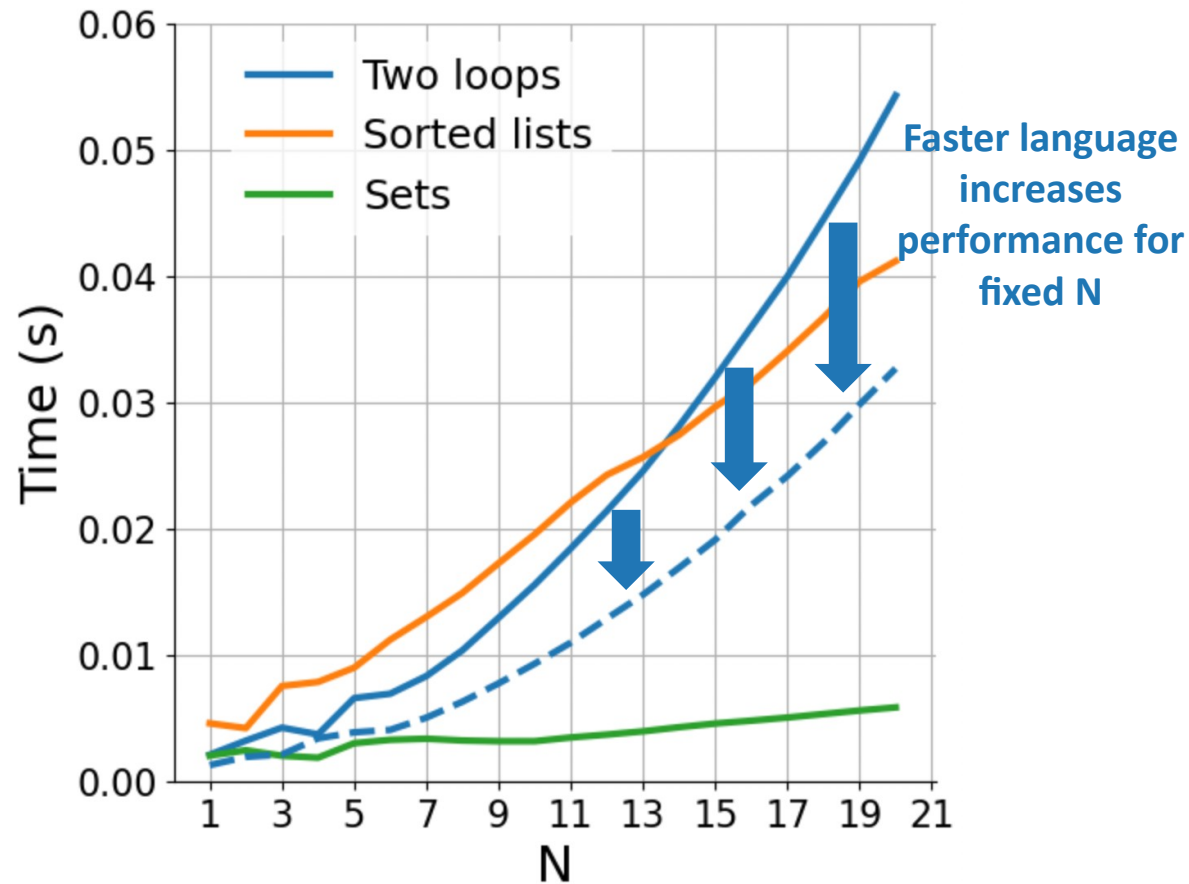
How does rewriting in C change the performance?

(rewriting in C, parallelization; same algorithm)



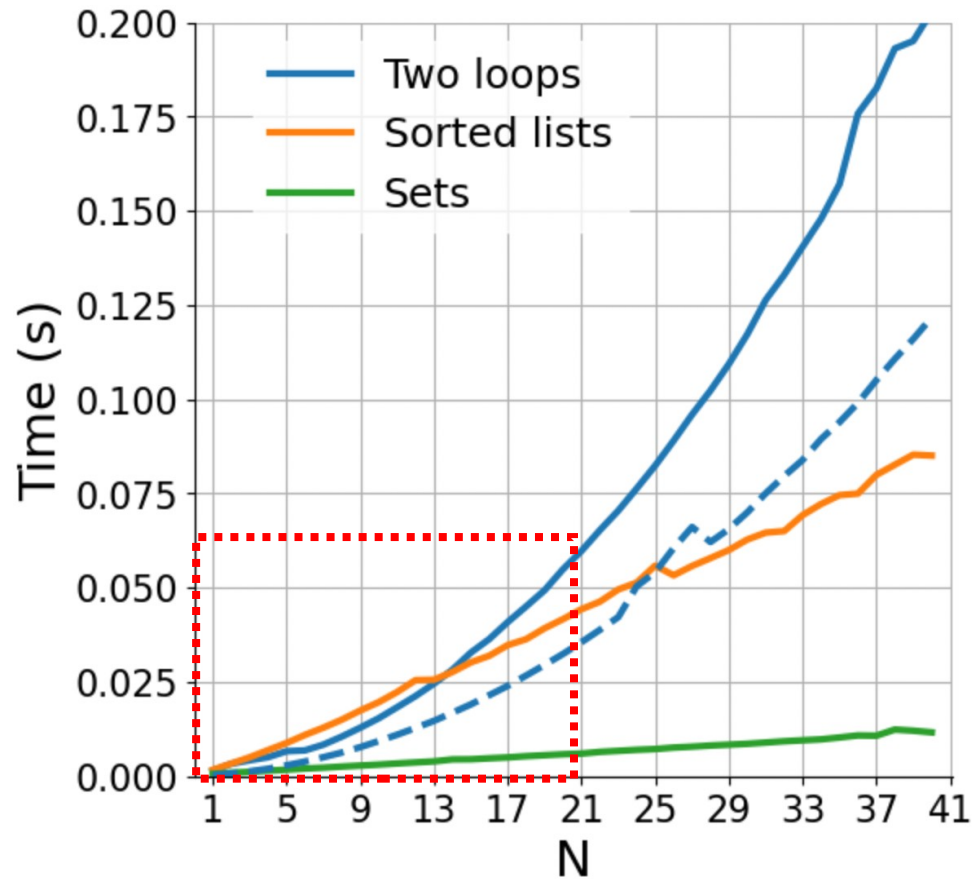
How does rewriting in C change the performance?

(rewriting in C, parallelization; same algorithm)



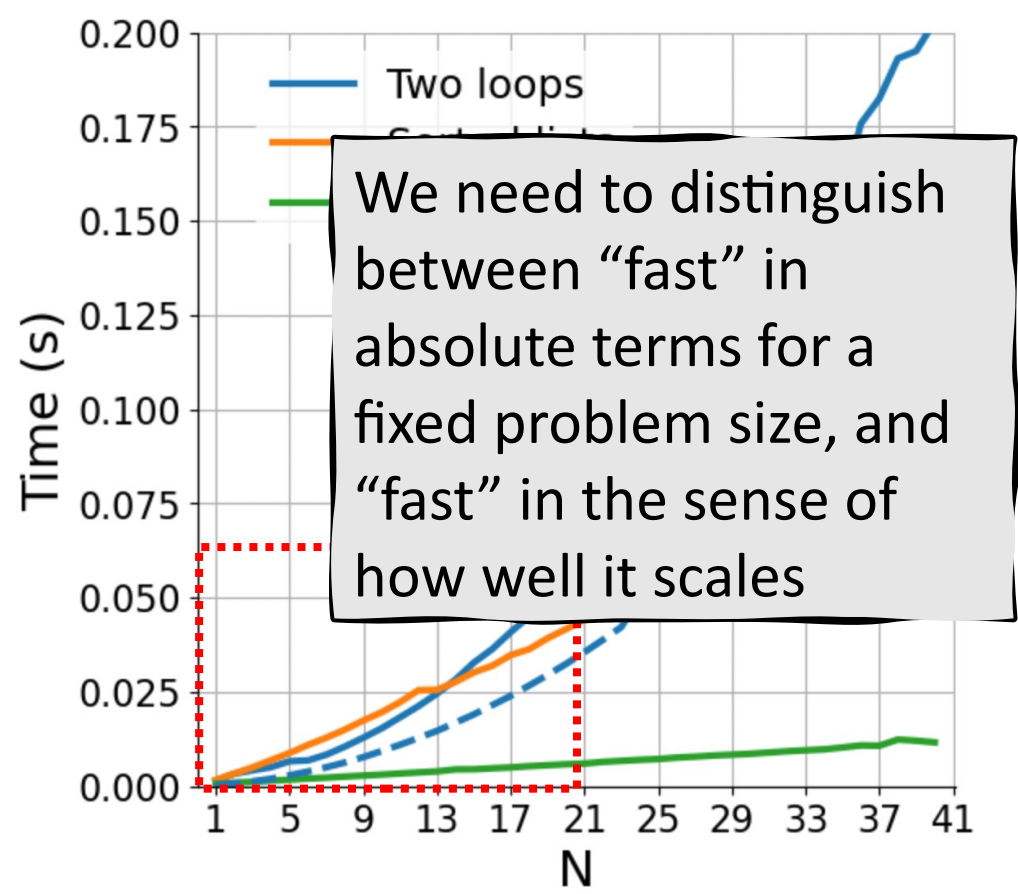
How does rewriting in C change the performance?

(rewriting in C, parallelization; same algorithm)



How does rewriting in C change the performance?

(rewriting in C, parallelization; same algorithm)



COMING UP NEXT:
NumPy and the array data structure

Some data structures

- list: ordered, heterogeneous storage, stack/queue, fast access by index, slow search
- set: unordered
- dictionary
- arrays: e.g. numpy, HDF5
- tables: e.g. pandas, dask, spark, SQL
- graph: social network structure
- tree: to rapidly search a dataset
- heap
- stack
- queue
- priority queue