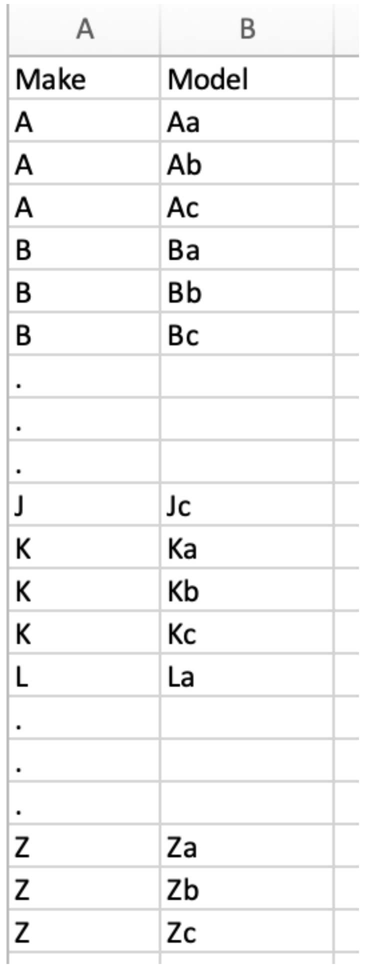
WASHINGTON STATE UNIVERSITY

CPTS 223: Advanced Data Structure in C/C++

Introduction: why advanced data structure?

First impression in daily life

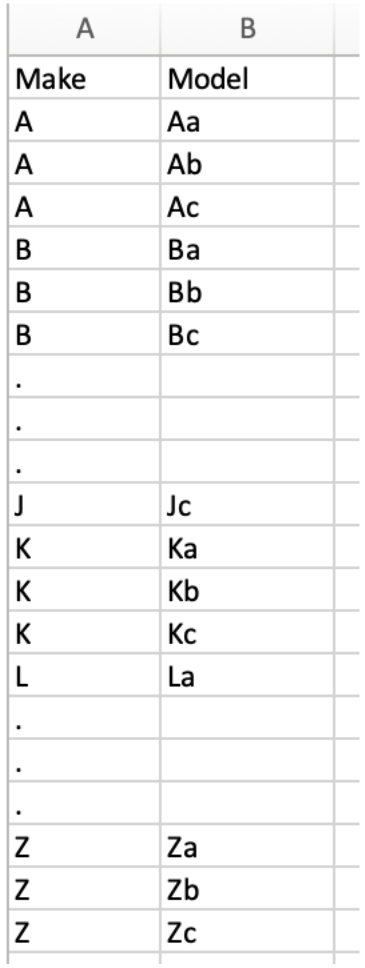
- Find a correct car wiper blade rom a size chart book



A	B
Make	Model
A	Aa
A	Ab
A	Ac
B	Ba
B	Bb
B	Bc
.	
.	
.	
J	Jc
K	Ka
K	Kb
K	Kc
L	La
.	
.	
.	
Z	Za
Z	Zb
Z	Zc

First impression in daily life

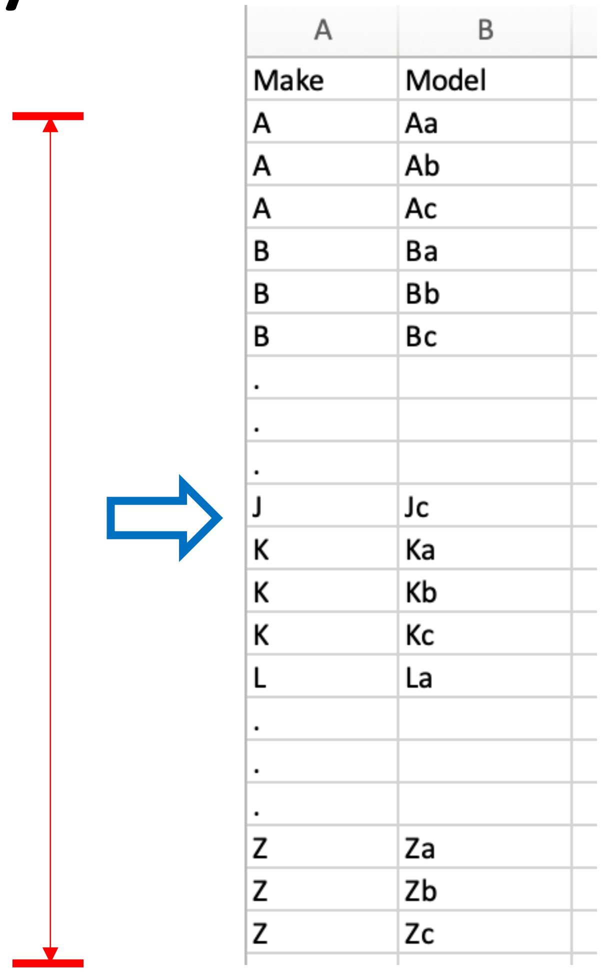
- Find a correct car wiper blade rom a size chart book
- Target: Model Kb
- Approaches to locating Make K?



A	B
Make	Model
A	Aa
A	Ab
A	Ac
B	Ba
B	Bb
B	Bc
.	
.	
.	
J	Jc
K	Ka
K	Kb
K	Kc
L	La
.	
.	
.	
Z	Za
Z	Zb
Z	Zc

First impression in daily life

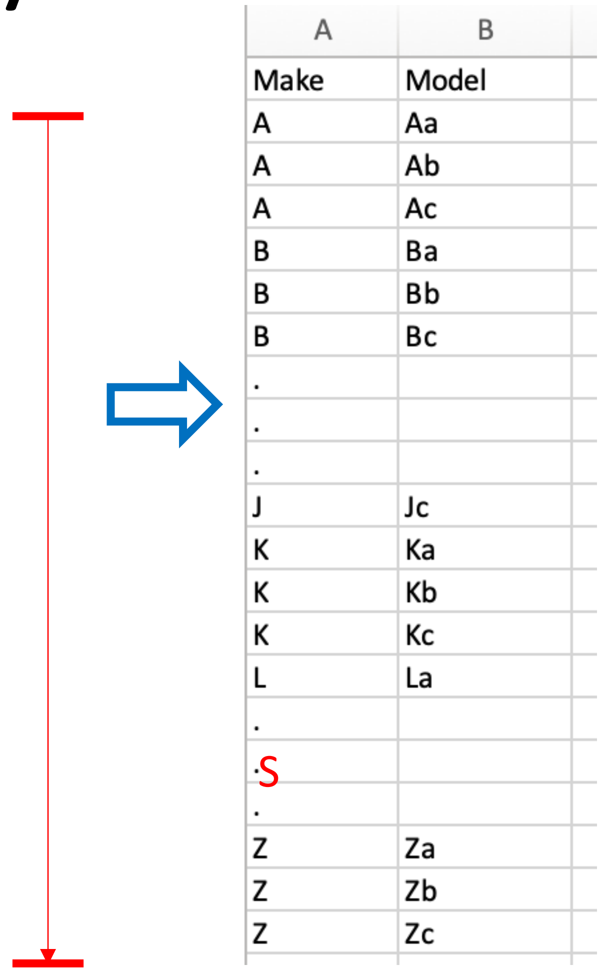
- Find a correct car wiper blade rom a size chart book
- Target: Model Kb
- Approaches to locating Make K?
    - Random pick a row?



A	B
Make	Model
A	Aa
A	Ab
A	Ac
B	Ba
B	Bb
B	Bc
.	
.	
.	
J	Jc
K	Ka
K	Kb
K	Kc
L	La
.	
.	
.	
Z	Za
Z	Zb
Z	Zc

First impression in daily life

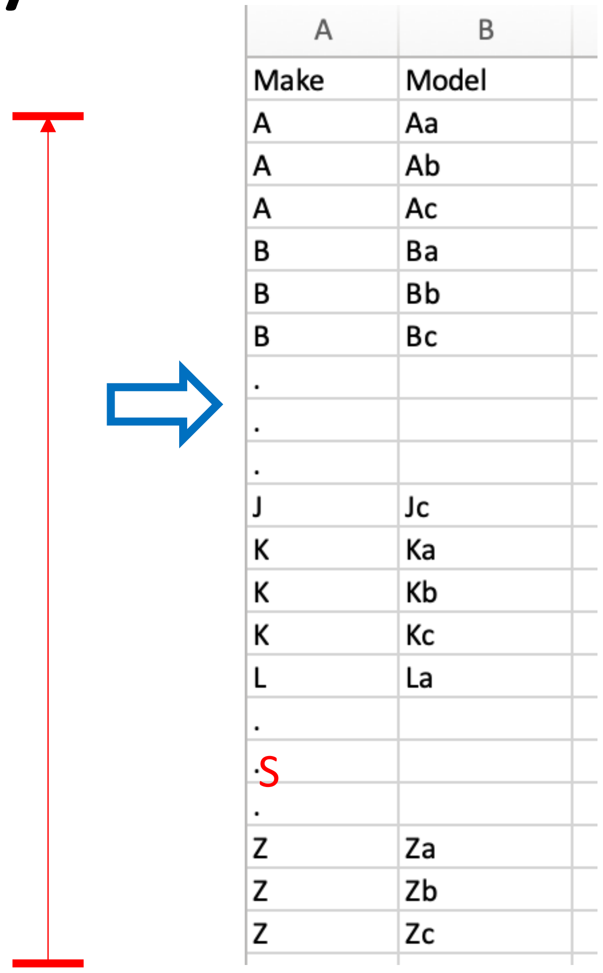
- Find a correct car wiper blade rom a size chart book
- Target: Model Kb
- Approaches to locating Make K?
  - Random pick a row?
  - From 1st to last row?



A	B
Make	Model
A	Aa
A	Ab
A	Ac
B	Ba
B	Bb
B	Bc
.	
.	
.	
J	Jc
K	Ka
K	Kb
K	Kc
L	La
.	
.S	
.	
Z	Za
Z	Zb
Z	Zc

First impression in daily life

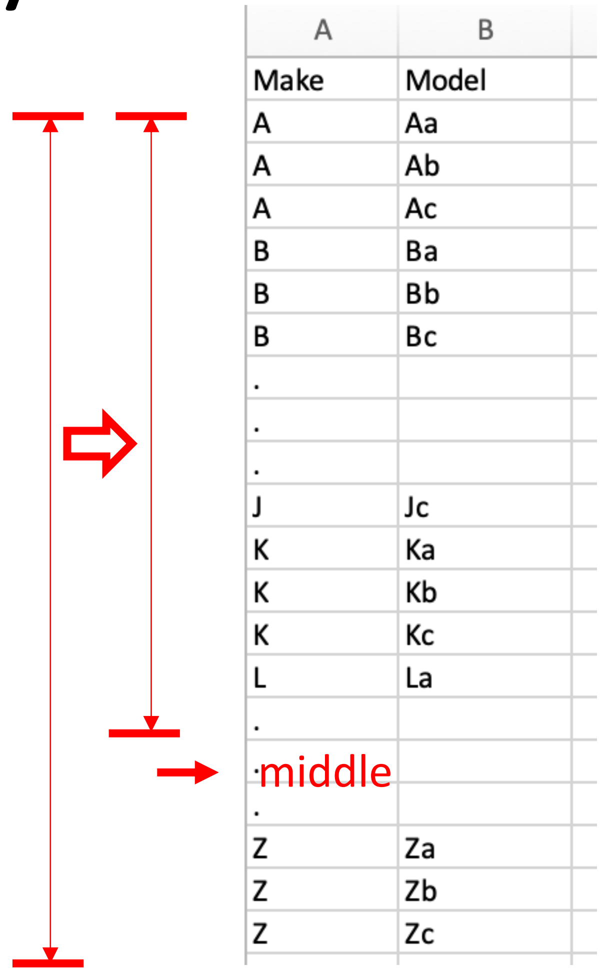
- Find a correct car wiper blade rom a size chart book
- Target: Model Kb
- Approaches to locating Make K?
    - Random pick a row?
    - From 1st to last row?
    - From last to 1st row?



A	B
Make	Model
A	Aa
A	Ab
A	Ac
B	Ba
B	Bb
B	Bc
.	
.	
.	
J	Jc
K	Ka
K	Kb
K	Kc
L	La
.	
.S	
.	
Z	Za
Z	Zb
Z	Zc

First impression in daily life

- Find a correct car wiper blade rom a size chart book
- Target: Model Kb
- Approaches to locating Make K?
  - Random pick a row?
  - From 1st to last row?
  - From last to 1st row?
  - Query Makes from middle?
    - If queried Make < K → query next one in latter section
    - If queried Make > K → query next one in former section

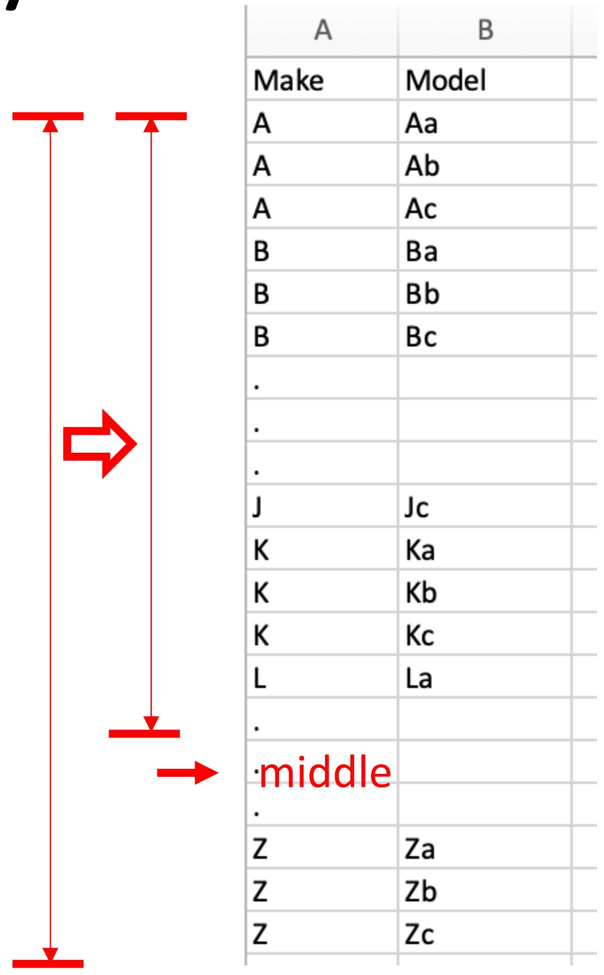


A	B
Make	Model
A	Aa
A	Ab
A	Ac
B	Ba
B	Bb
B	Bc
.	
.	
.	
J	Jc
K	Ka
K	Kb
K	Kc
L	La
.	
.	
.	
Z	Za
Z	Zb
Z	Zc

First impression in daily life

- Find a correct car wiper blade rom a size chart book
- Target: Model Kb
- Approaches to locating Make K?
  - Random pick a row?
  - From 1st to last row?
  - From last to 1st row?
  - Query Makes from middle?
    - If queried Make < K → query next one in latter section
    - If queried Make > K → query next one in former section
- Which one is the best? Why?

Difference?



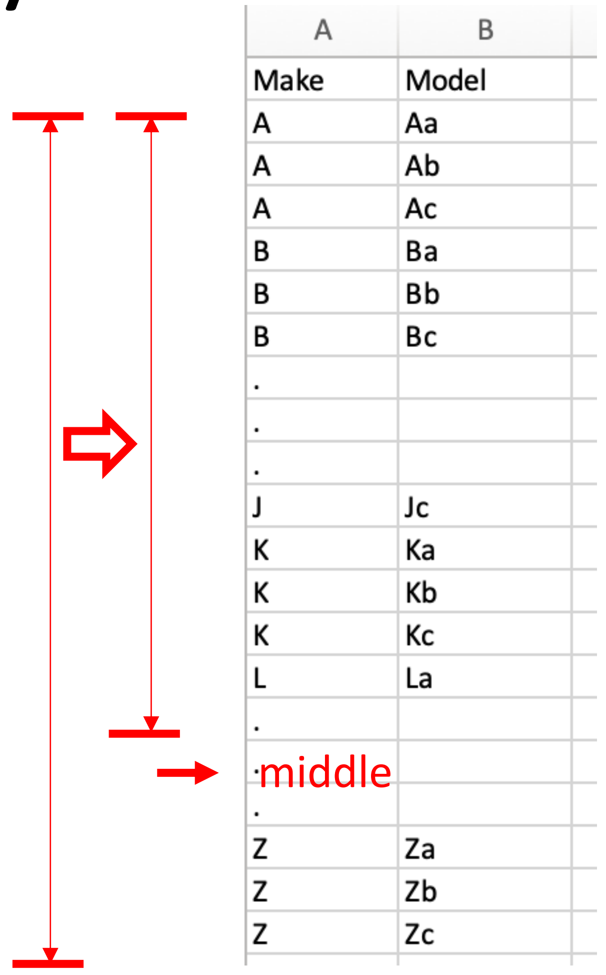
A	B
Make	Model
A	Aa
A	Ab
A	Ac
B	Ba
B	Bb
B	Bc
.	
.	
.	
J	Jc
K	Ka
K	Kb
K	Kc
L	La
.	
.	
.	
Z	Za
Z	Zb
Z	Zc

First impression in daily life

- Find a correct car wiper blade rom a size chart book
- Target: Model Kb
- Approaches to locating Make K?

Difference?

  - Random pick a row?
  - From 1st to last row?
  - From last to 1st row?
  - Query Makes from middle? binary search
    - If queried Make < K → query next one in latter section
    - If queried Make > K → query next one in former section
- Which one is the best? Why?



A	B
Make	Model
A	Aa
A	Ab
A	Ac
B	Ba
B	Bb
B	Bc
.	
.	
.	
J	Jc
K	Ka
K	Kb
K	Kc
L	La
.	
.	
.	
Z	Za
Z	Zb
Z	Zc

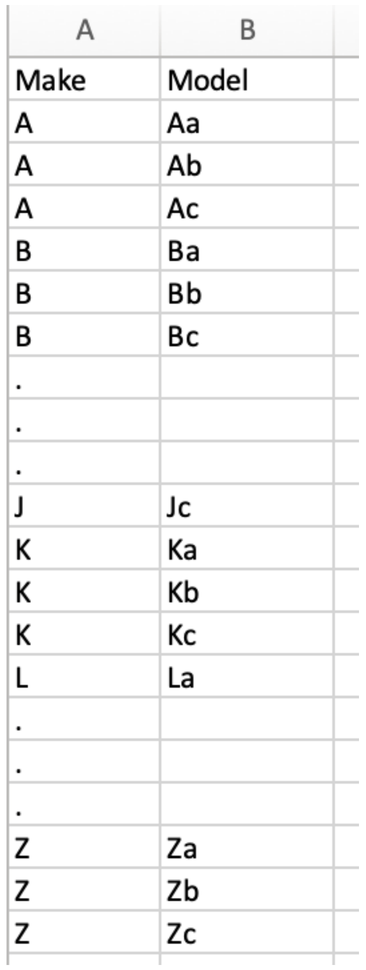
WSU

First impression in daily life

- Conclusions:
  - All searching algorithms works
  - Efficiency varies

First impression in daily life

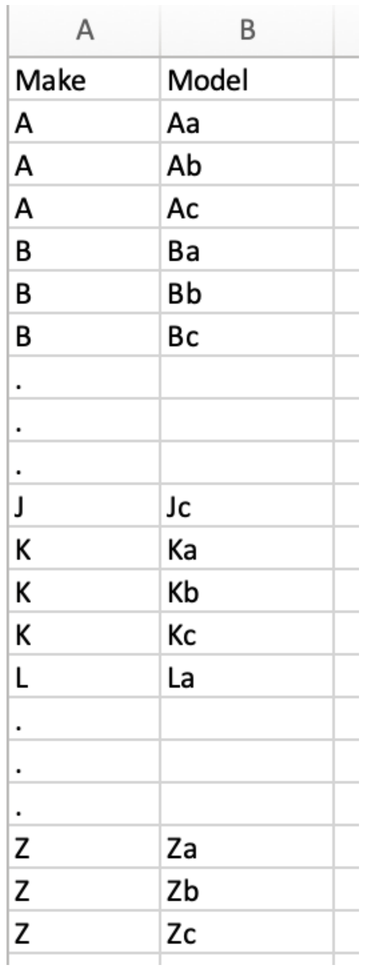
- Conclusions:
  - All searching algorithms works
  - Efficiency varies
  - Is there any **assumption** about how data is maintained?
    - Already sorted: the size chart book ranks elements from A → Z



A	B
Make	Model
A	Aa
A	Ab
A	Ac
B	Ba
B	Bb
B	Bc
.	
.	
.	
J	Jc
K	Ka
K	Kb
K	Kc
L	La
.	
.	
.	
Z	Za
Z	Zb
Z	Zc

First impression in daily life

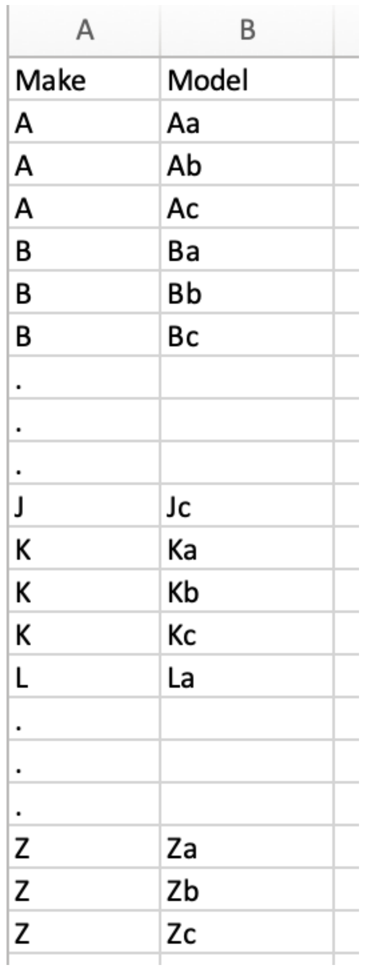
- Conclusions:
  - All searching algorithms works
  - Efficiency varies
  - Is there any **assumption** about how data is maintained?
    - Already sorted: the size chart book ranks elements from A → Z
- Questions:
  - How to compare efficiency of algorithms?
    - How about executing/running time?



A	B
Make	Model
A	Aa
A	Ab
A	Ac
B	Ba
B	Bb
B	Bc
.	
.	
.	
J	Jc
K	Ka
K	Kb
K	Kc
L	La
.	
.	
.	
Z	Za
Z	Zb
Z	Zc

First impression in daily life

- Conclusions:
  - All searching algorithms works
  - Efficiency varies
  - Is there any **assumption** about how data is maintained?
    - Already sorted: the size chart book ranks elements from A → Z
- Questions:
  - How to compare efficiency of algorithms?
    - How about executing/running time?
    - Is it reliable?



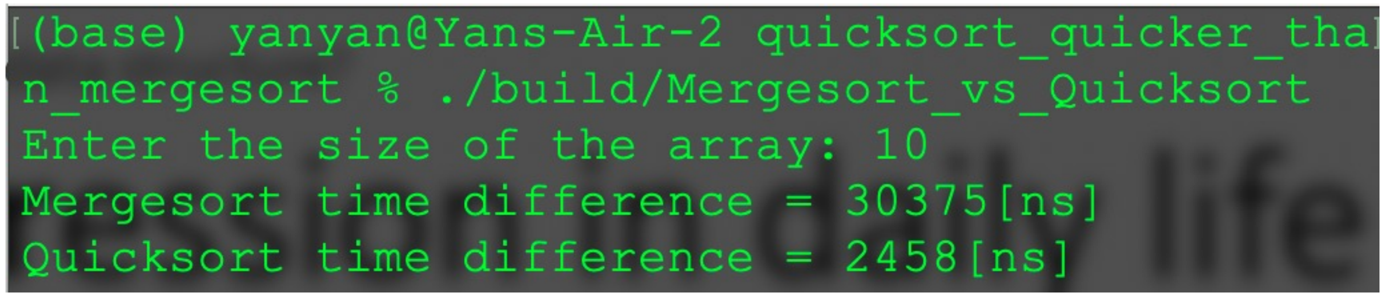
A	B
Make	Model
A	Aa
A	Ab
A	Ac
B	Ba
B	Bb
B	Bc
.	
.	
.	
J	Jc
K	Ka
K	Kb
K	Kc
L	La
.	
.	
.	
Z	Za
Z	Zb
Z	Zc

First impression: case study

- Case study: comparing sorting algorithms:
    - Mergesort V.S. Quicksort (in terms of running time)

First impression: case study

- Case study: comparing sorting algorithms:
  - Mergesort V.S. Quicksort (in terms of running time)

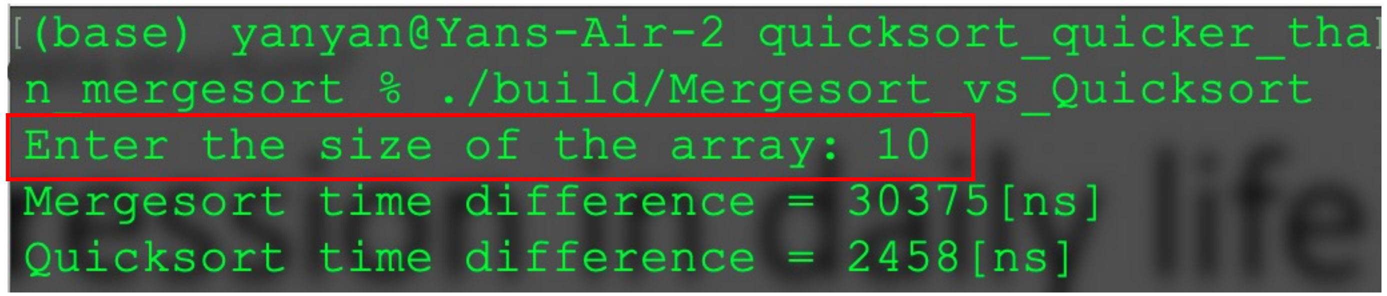


```
[(base) yanyan@Yans-Air-2 quicksort_quicker_tha]
n_mergesort % ./build/Mergesort_vs_Quicksort
Enter the size of the array: 10
Mergesort time difference = 30375[ns]
Quicksort time difference = 2458[ns]
```

WSU

First impression: case study

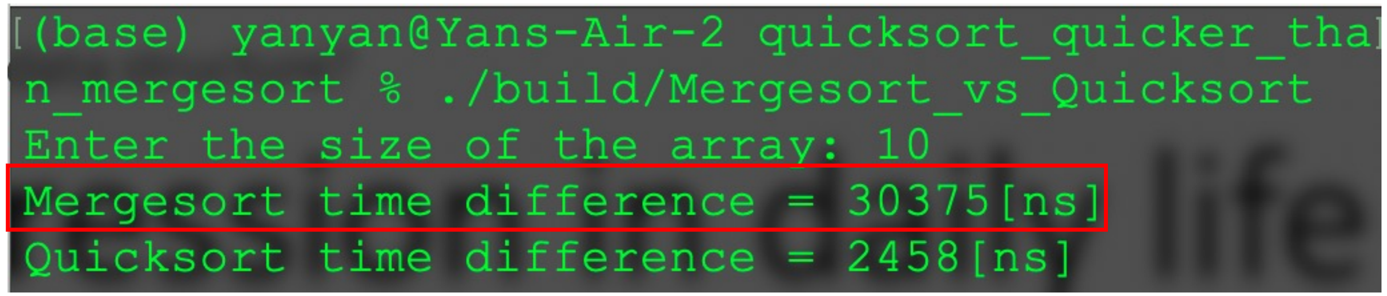
- Case study: comparing sorting algorithms:
  - Mergesort V.S. Quicksort (in terms of running time)



```
[(base) yanyan@Yans-Air-2 quicksort_quicker_tha
n mergesort % ./build/Mergesort_vs_Quicksort
Enter the size of the array: 10
Mergesort time difference = 30375[ns]
Quicksort time difference = 2458[ns]
```

First impression: case study

- Case study: comparing sorting algorithms:
  - Mergesort V.S. Quicksort (in terms of running time)

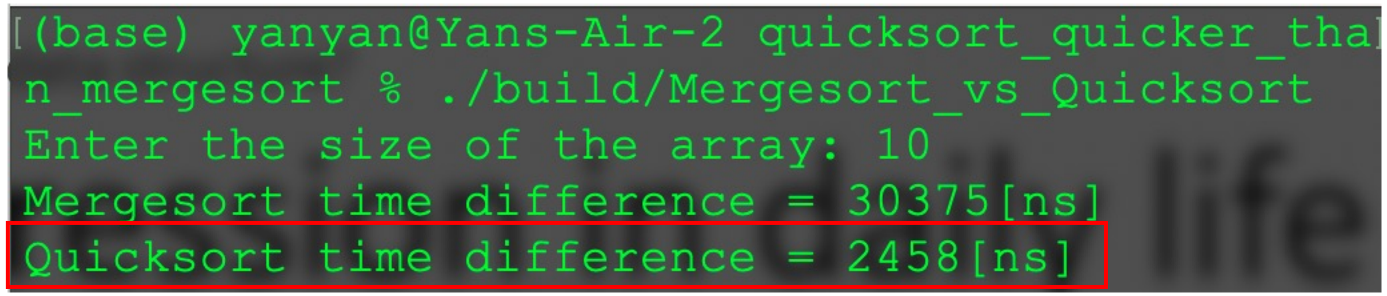


```
[(base) yanyan@Yans-Air-2 quicksort_quicker_tha
n_mergesort % ./build/Mergesort_vs_Quicksort
Enter the size of the array: 10
Mergesort time difference = 30375[ns]
Quicksort time difference = 2458[ns]
```

WSU

First impression: case study

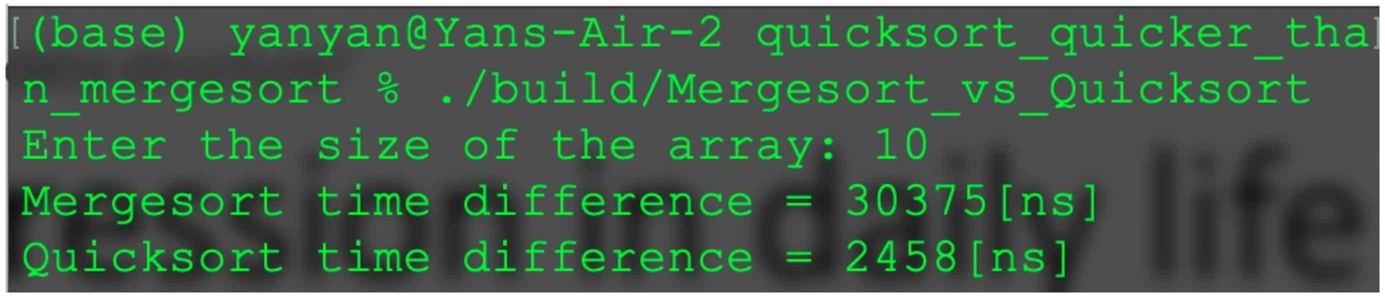
- Case study: comparing sorting algorithms:
  - Mergesort V.S. Quicksort (in terms of running time)



```
(base) yanyan@Yans-Air-2 quicksort_quicker_tha
n_mergesort % ./build/Mergesort_vs_Quicksort
Enter the size of the array: 10
Mergesort time difference = 30375[ns]
Quicksort time difference = 2458[ns]
```

First impression: case study

- Case study: comparing sorting algorithms:
  - Mergesort V.S. Quicksort (in terms of running time)

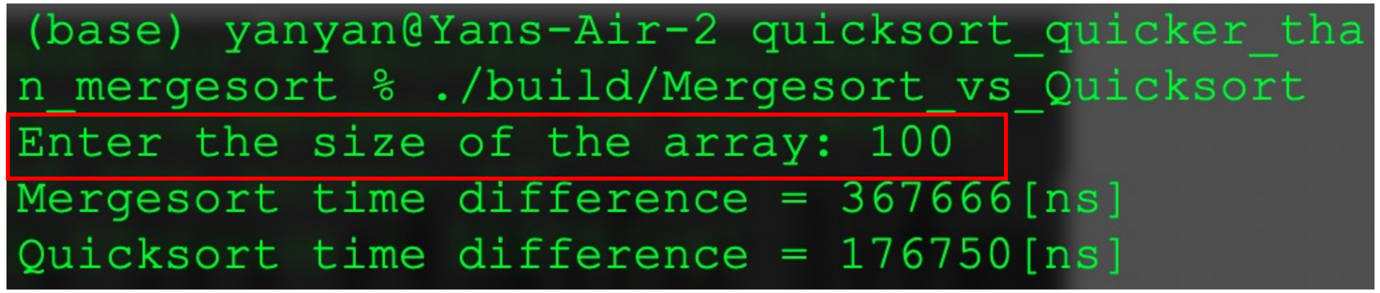


```
[(base) yanyan@Yans-Air-2 quicksort_quicker_tha
n_mergesort % ./build/Mergesort_vs_Quicksort
Enter the size of the array: 10
Mergesort time difference = 30375[ns]
Quicksort time difference = 2458[ns]
```

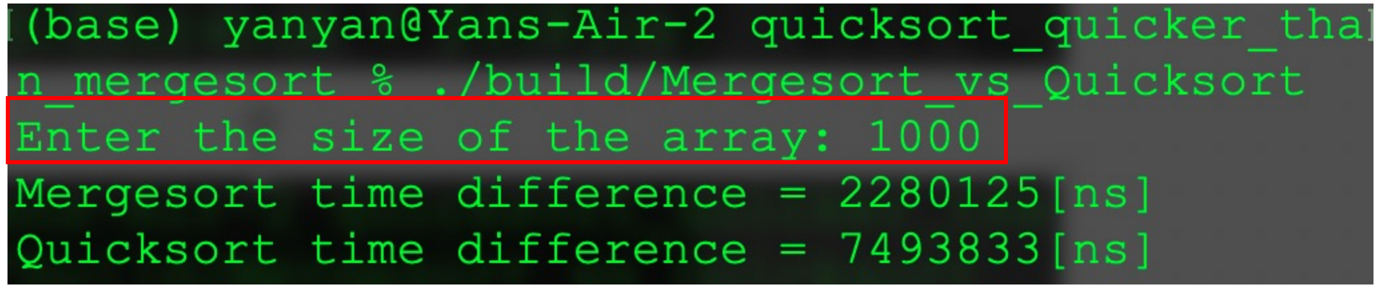
  - Conclusion: Quicksort is better than Mergesort?

First impression: case study

- Case study: comparing sorting algorithms:
  - Mergesort V.S. Quicksort (in terms of running time)



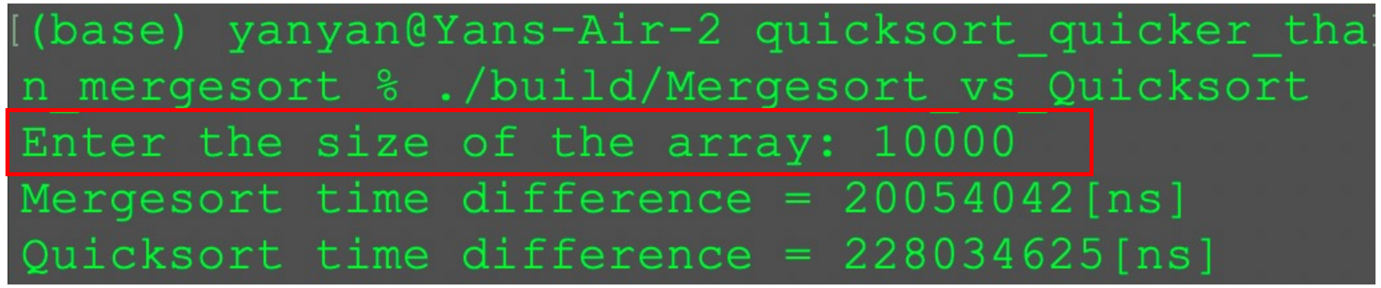
```
(base) yanyan@Yans-Air-2 quicksort_quicker_tha
n_mergesort % ./build/Mergesort_vs_Quicksort
Enter the size of the array: 100
Mergesort time difference = 367666[ns]
Quicksort time difference = 176750[ns]
```



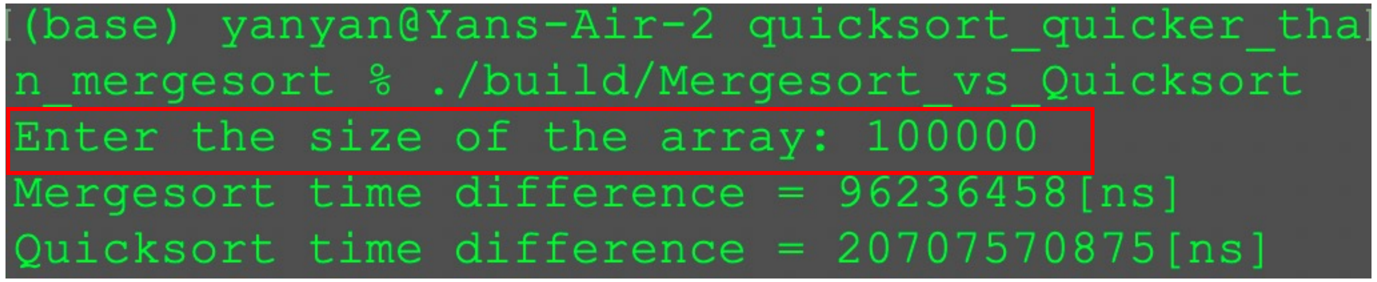
```
(base) yanyan@Yans-Air-2 quicksort_quicker_tha
n_mergesort % ./build/Mergesort_vs_Quicksort
Enter the size of the array: 1000
Mergesort time difference = 2280125[ns]
Quicksort time difference = 7493833[ns]
```

WSU

First impression: case study



```
[(base) yanyan@Yans-Air-2 quicksort_quicker_tha
n_mergesort % ./build/Mergesort_vs_Quicksort
Enter the size of the array: 10000
Mergesort time difference = 20054042[ns]
Quicksort time difference = 228034625[ns]
```



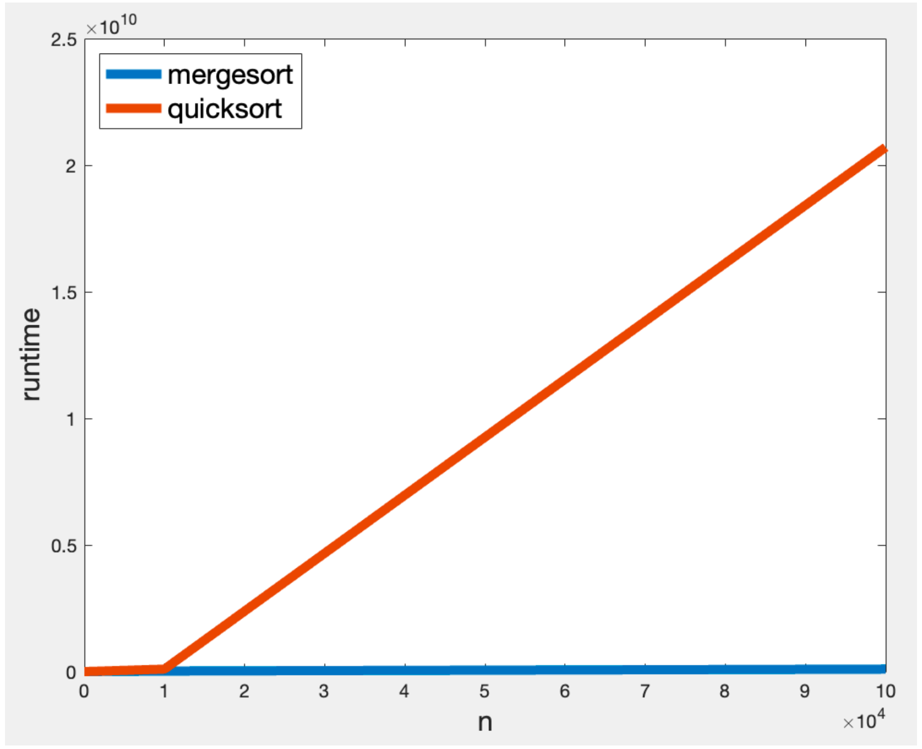
```
[(base) yanyan@Yans-Air-2 quicksort_quicker_tha
n_mergesort % ./build/Mergesort_vs_Quicksort
Enter the size of the array: 100000
Mergesort time difference = 96236458[ns]
Quicksort time difference = 20707570875[ns]
```

First impression: case study

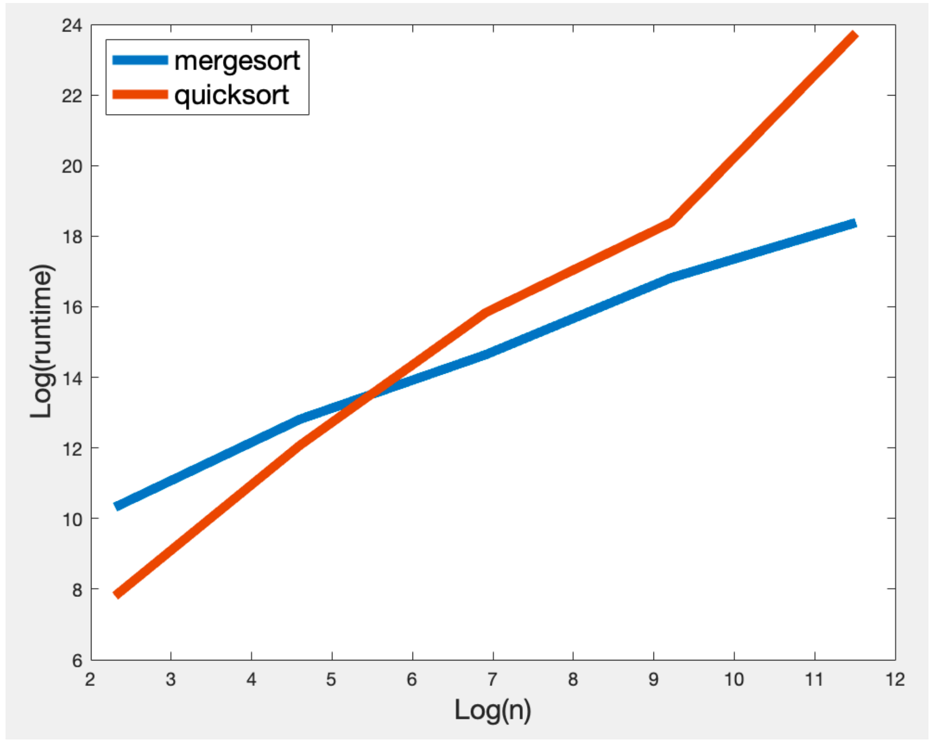
Running time (in ns)

n (input size)	**Mergesort**	**Quicksort**
10	30375	2458
100	367666	176750
1000	2280125	7493833
10000	20054042	96236458
100000	96236458	20707570875

First impression: case study



First impression: case study



- Is it possible to have a comparison benchmark?