CS 61BL Lab 20

Ryan Purpura

Announcements

- BearMaps due tomorrow at 11:59 PM! Remember the extra-credit portion (Auto-complete with Tries) is worth 5 extra credit points.
- The final is next Thursday! Remember there is no dead week over summer. Start studying *now*!

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?
- First, let's check if $\log(n!) \in O(n \log n)$.

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?
- First, let's check if $\log(n!) \in O(n \log n)$.
  - $\log(n!) = \log 1 + \log 2 + \ldots + \log n$

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?
- First, let's check if $\log(n!) \in O(n \log n)$.
  - $\log(n!) = \log 1 + \log 2 + \ldots + \log n$
  - $n \log n = \log n + \log n + \ldots + \log n$

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?
- First, let's check if $\log(n!) \in O(n \log n)$.
  - $\log(n!) = \log 1 + \log 2 + \ldots + \log n$
  - $n \log n = \log n + \log n + \ldots + \log n$
  - So $\log(n!) \leq n \log n \implies \log(n!) \in O(n \log n)$

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?
- Next, let's check if $\log(n!) \in \Omega(n \log n)$.

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?
- Next, let's check if $\log(n!) \in \Omega(n \log n)$.
    - $\log(n!) = \log 1 + \log 2 + \ldots + \log n$

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?
- Next, let's check if $\log(n!) \in \Omega(n \log n)$.
  - $\log(n!) = \log 1 + \log 2 + \ldots + \log n$
  - $\log(n!) \geq \log\left(\frac{n}{2}\right) + \log\left(\frac{n}{2} + 1\right) + \ldots + \log(n)$

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?
- Next, let's check if $\log(n!) \in \Omega(n \log n)$.
  - $\log(n!) = \log 1 + \log 2 + \ldots + \log n$
  - $\log(n!) \geq \log\left(\frac{n}{2}\right) + \log\left(\frac{n}{2} + 1\right) + \ldots + \log(n)$
  - $\log(n!) \geq \log\left(\frac{n}{2}\right) + \log\left(\frac{n}{2}\right) + \ldots + \log\left(\frac{n}{2}\right)$

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?
- Next, let's check if $\log(n!) \in \Omega(n \log n)$.
  - $\log(n!) = \log 1 + \log 2 + \ldots + \log n$
  - $\log(n!) \geq \log\left(\frac{n}{2}\right) + \log\left(\frac{n}{2} + 1\right) + \ldots + \log(n)$
  - $\log(n!) \geq \log\left(\frac{n}{2}\right) + \log\left(\frac{n}{2}\right) + \ldots + \log\left(\frac{n}{2}\right)$
  - $\log(n!) \geq \frac{n}{2} \log\left(\frac{n}{2}\right)$

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?
- Next, let's check if $\log(n!) \in \Omega(n \log n)$.
  - $\log(n!) = \log 1 + \log 2 + \ldots + \log n$
  - $\log(n!) \geq \log\left(\frac{n}{2}\right) + \log\left(\frac{n}{2} + 1\right) + \ldots + \log(n)$
  - $\log(n!) \geq \log\left(\frac{n}{2}\right) + \log\left(\frac{n}{2}\right) + \ldots + \log\left(\frac{n}{2}\right)$
  - $\log(n!) \geq \frac{n}{2} \log\left(\frac{n}{2}\right)$
  - $\frac{n}{2} \log\left(\frac{n}{2}\right) \in \Theta(n \log n)$ so we know that $\log(n!) \in \Omega(n \log n)$

An Asymptotic Puzzle

- Is $\log(n!) \in \Theta(n \log n)$?
- We know $\log(n!) \in \Omega(n \log n)$ and $\log(n!) \in O(n \log n)$
- So yes, $\log(n!) \in \Theta(n \log n)$

Why do I care?

Why do I care?

- Given an array of N distinct items, how many ways can it be ordered?

Why do I care?

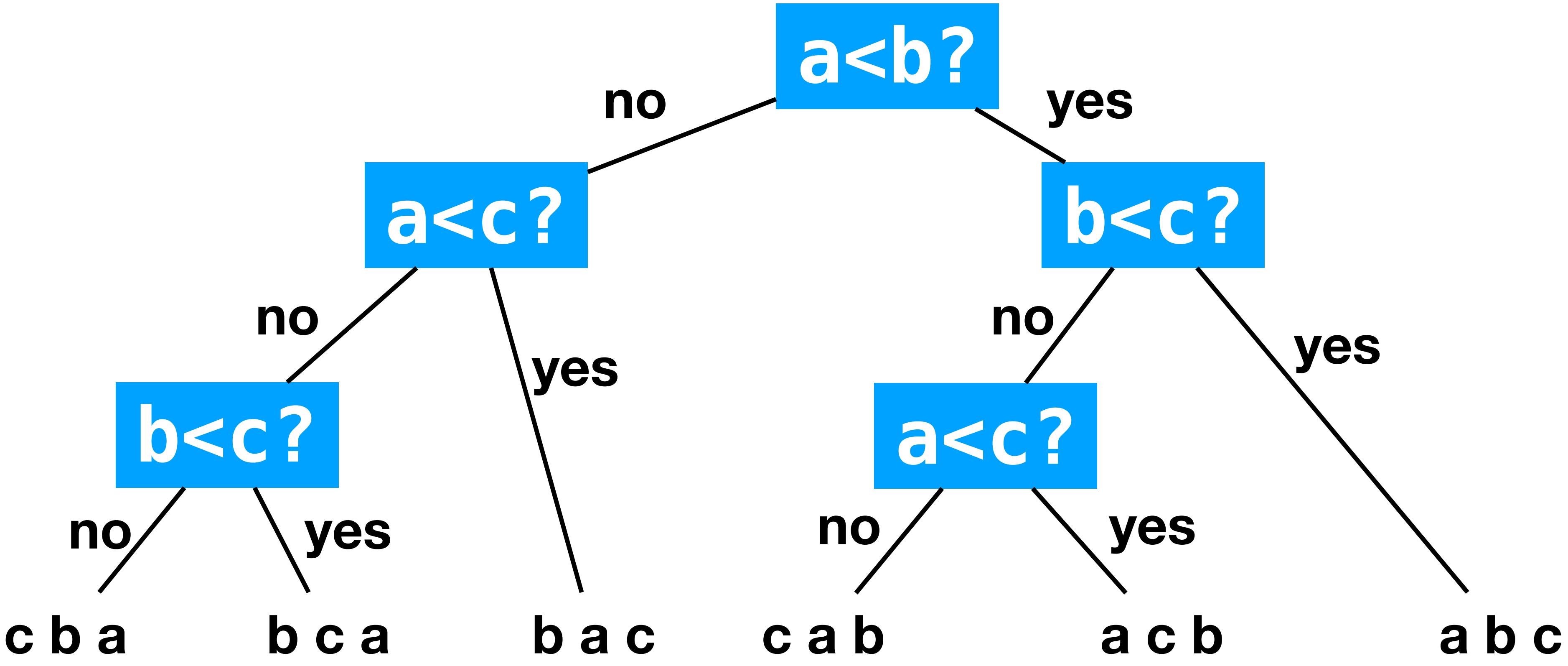
- Given an array of N distinct items, how many ways can it be ordered?
- Answer: $N!$

Why do I care?

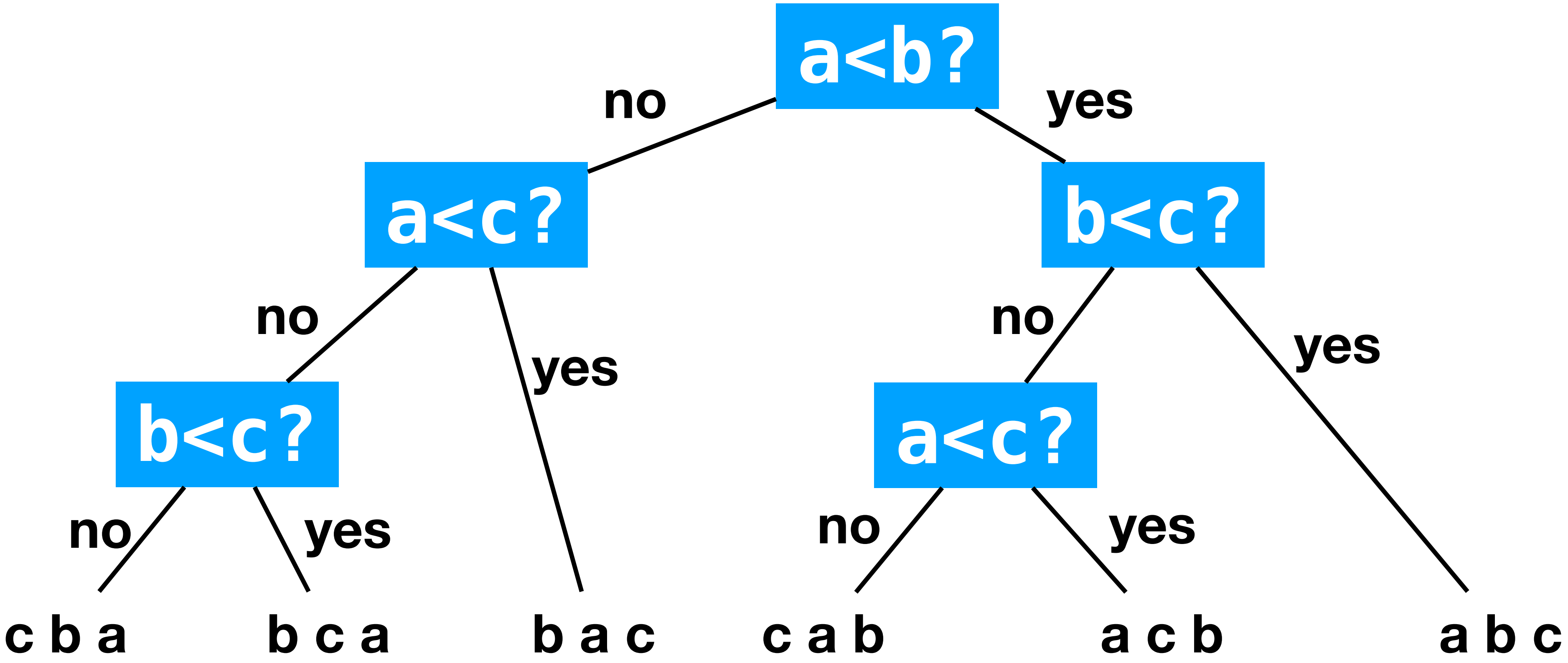
- Given an array of N distinct items, how many ways can it be ordered?
- Answer: $N!$
- When we sort a list, we perform *comparisons* (yes-or-no questions involving two items) in order to choose which permutation is the "sorted" one consistent with the comparisons.

Why do I care?

- Given an array of N distinct items, how many ways can it be ordered?
- Answer: $N!$
- When we sort a list, we perform *comparisons* (yes-or-no questions involving two items) in order to choose which permutation is the "sorted" one consistent with the comparisons.

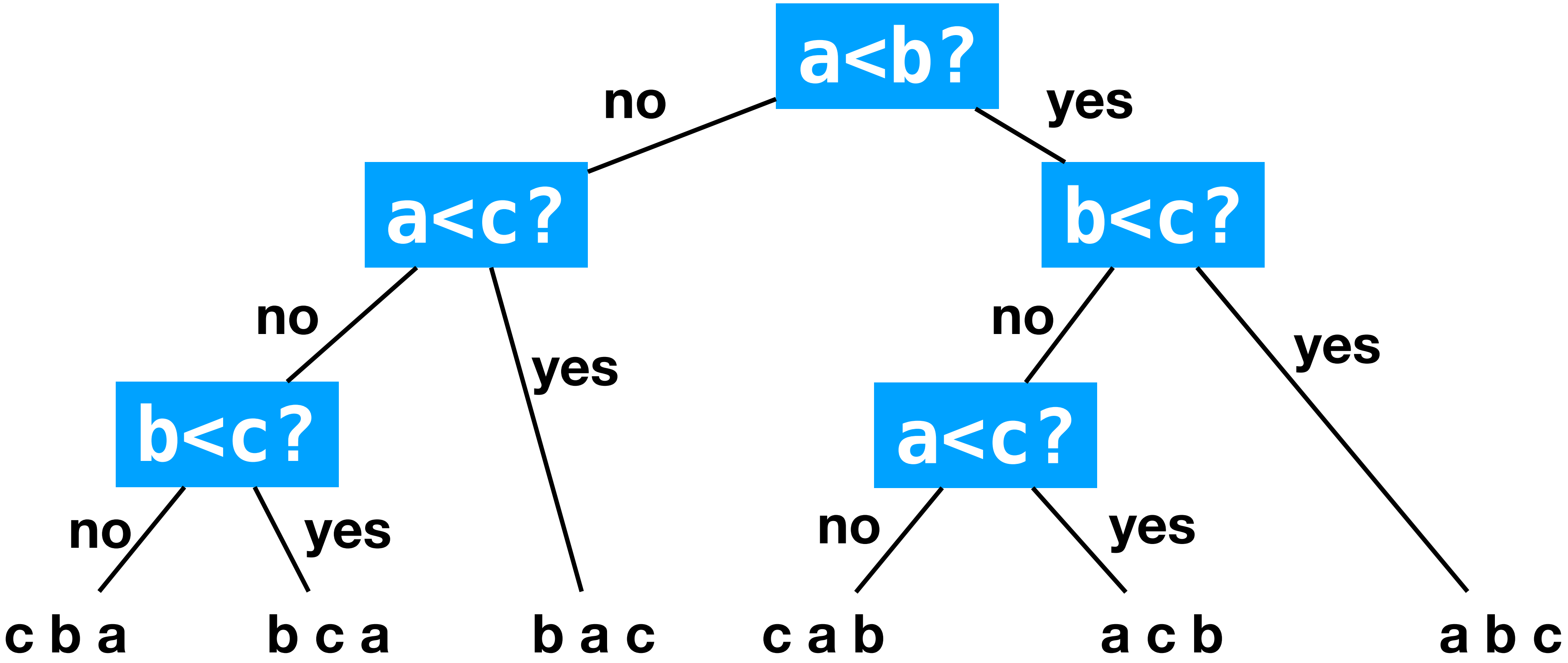


Why do I care?


a<b?
no
yes
a<c?
b<c?
no
yes
no
yes
b<c?
a<c?
no
yes
no
yes
c b a
b c a
b a c
c a b
a c b
a b c

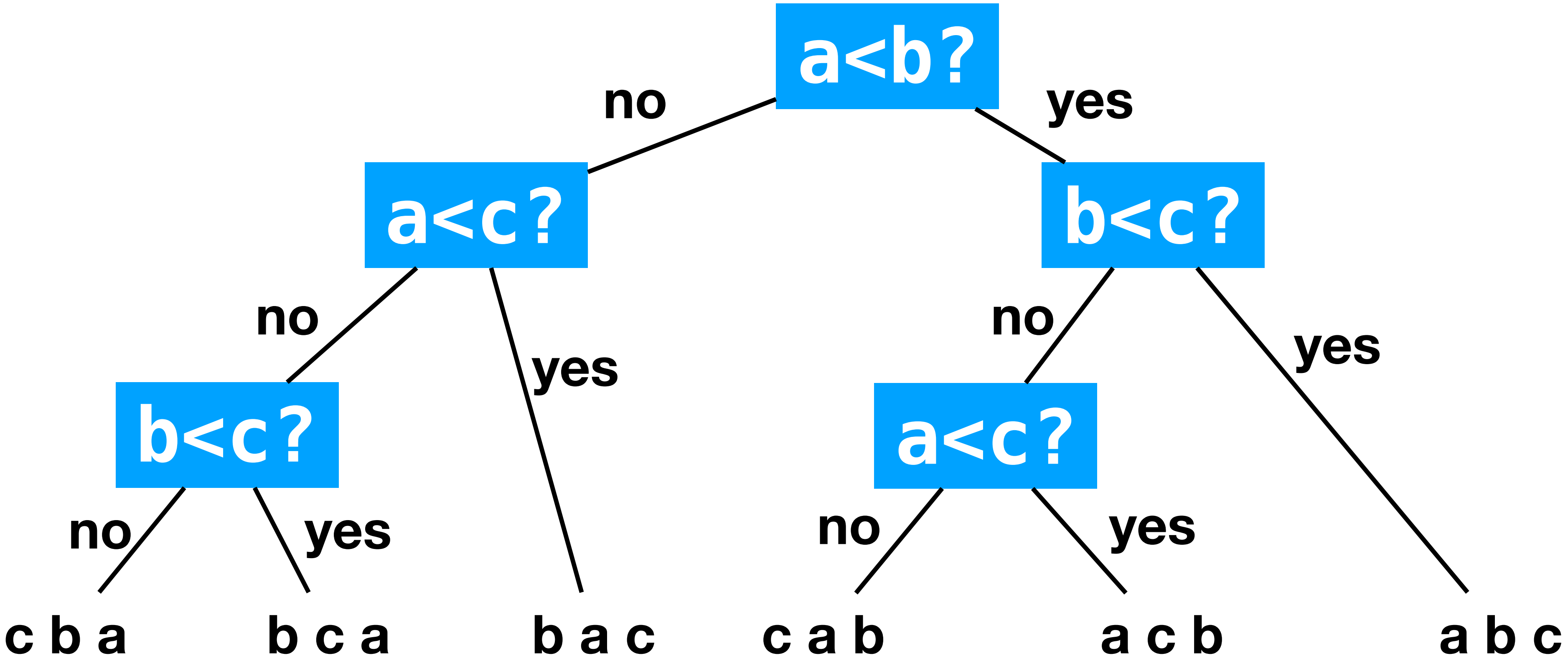
Why do I care?

- If we have n! leaves of our tree, how tall must our tree be?



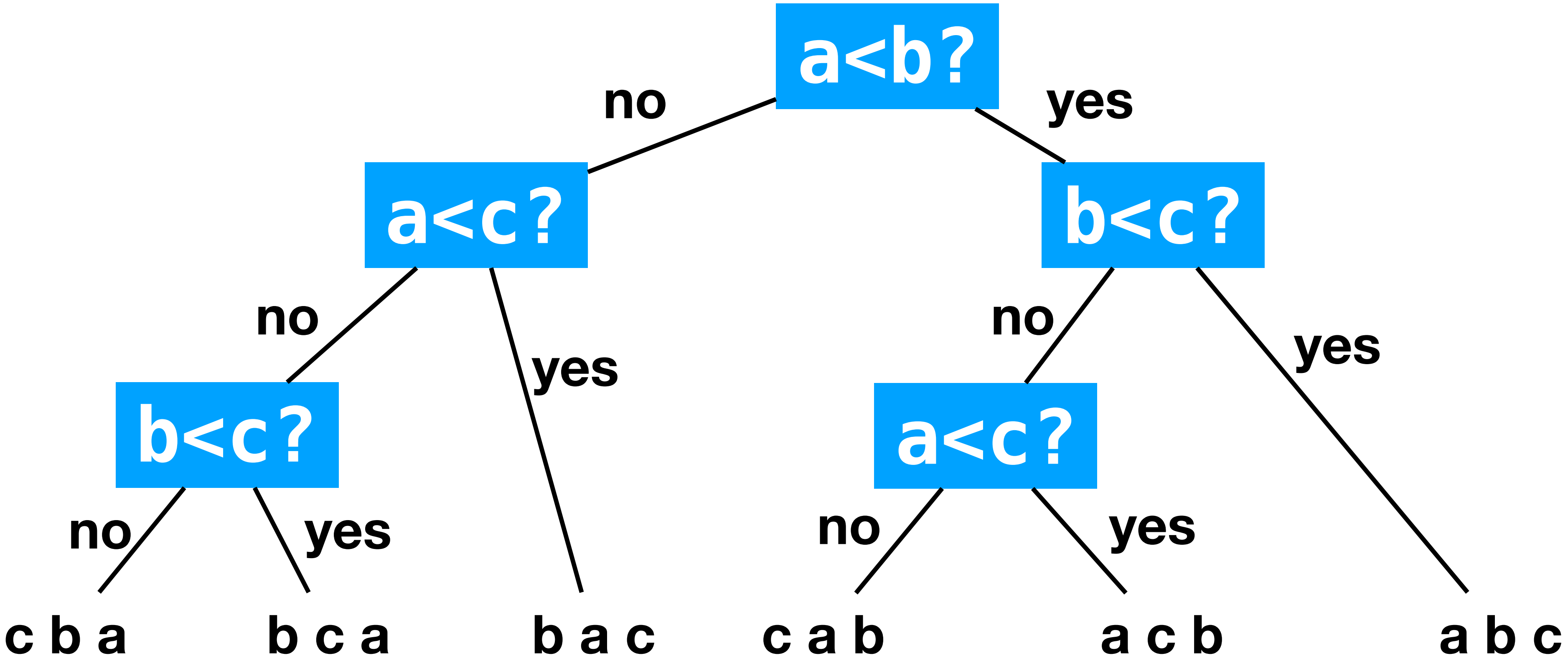
Why do I care?

- If we have n! leaves of our tree, how tall must our tree be?
    - $\log(n!) \in \Theta(n \log n)$



Why do I care?

- If we have n! leaves of our tree, how tall must our tree be?
  - $\log(n!) \in \Theta(n \log n)$
- Conclusion: we can never do better than $n \log n$ in the worst case for comparison-based sorting algorithms 😭



Sorting with Limited Variety of Items

- What if the the list we are sorting only has a limited number of varieties?

- For example, we have a list of Strings where the only values are "water", "coffee", and "tea" and want to sort in alphabetical order.

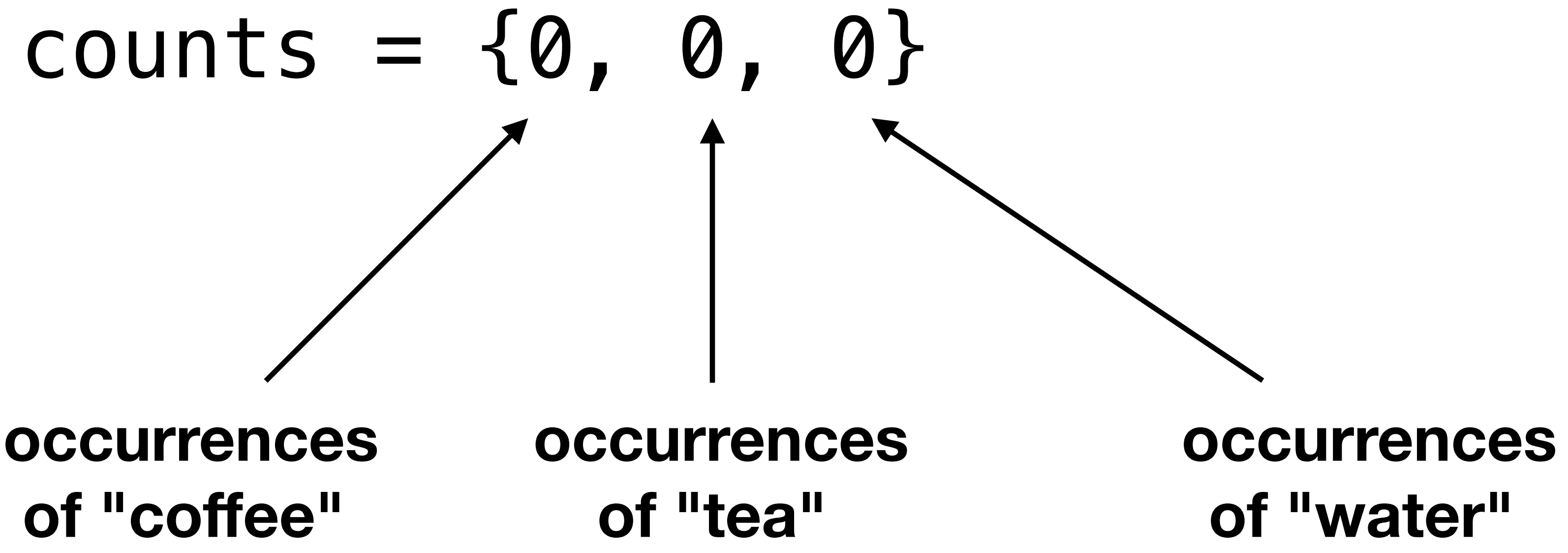
-
  ```
  myDrinks = {"water", "coffee", "water", "water",
      "coffee", "water", "tea", "water", "tea"}
  ```

- Can we sort this in better than $n \log n$ time?

Counting Sort

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`
- Idea: count occurrences of each. Then we know where each group starts!

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`



```
counts = {0, 0, 0}
```

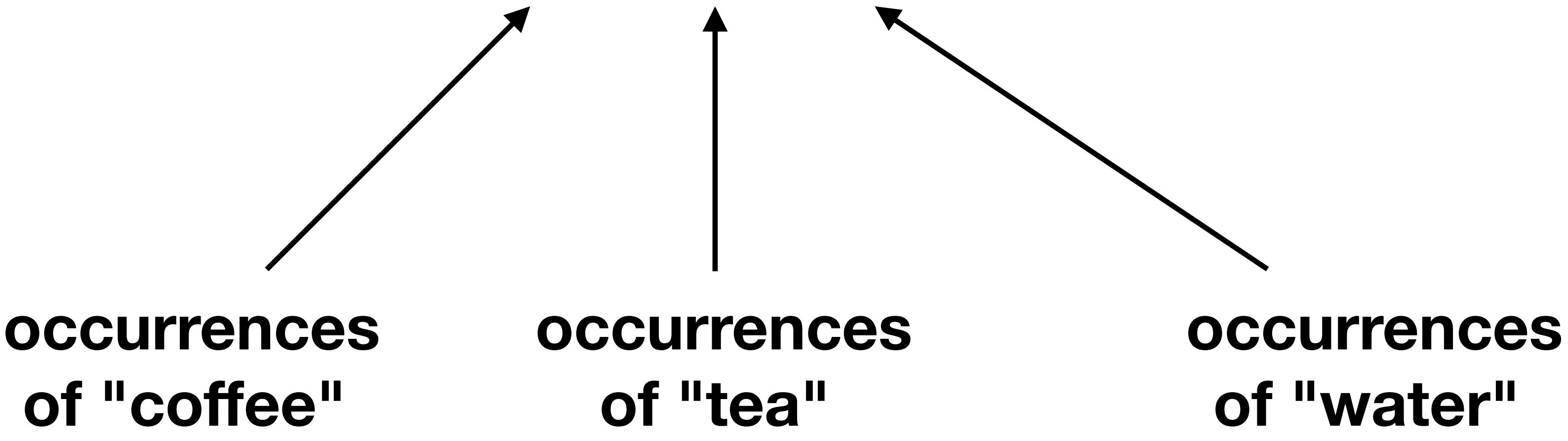
occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {0, 0, 1}
```

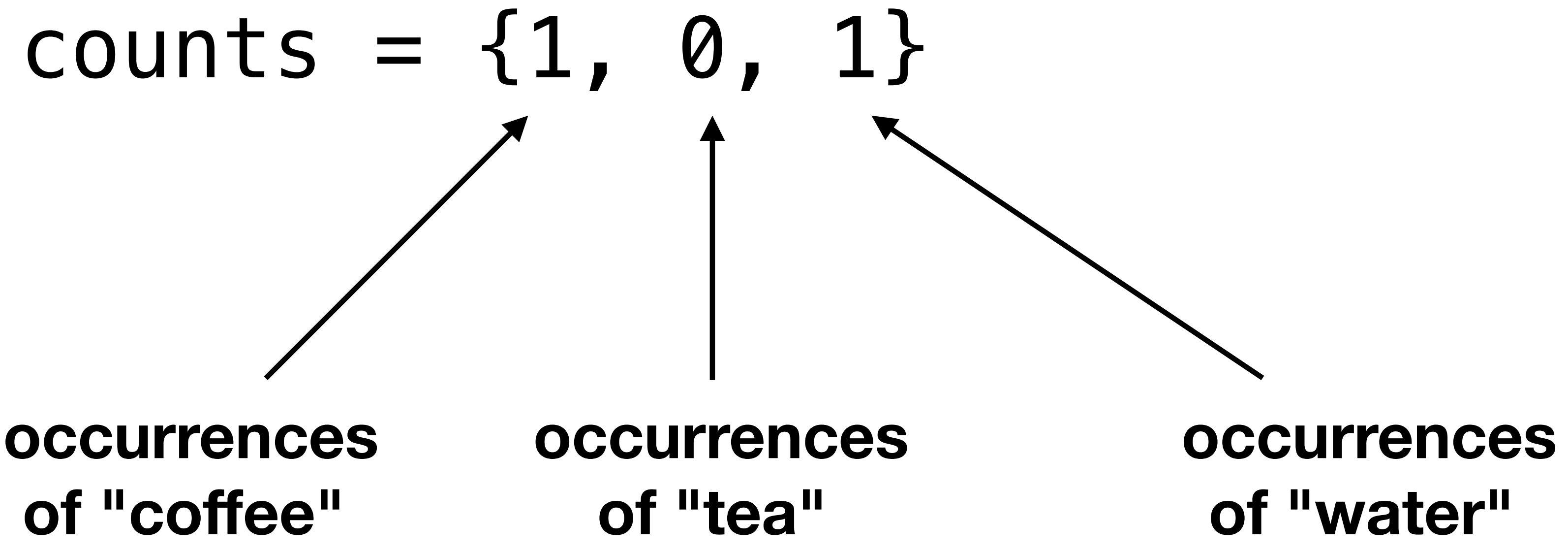


occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`



```
counts = {1, 0, 1}
```

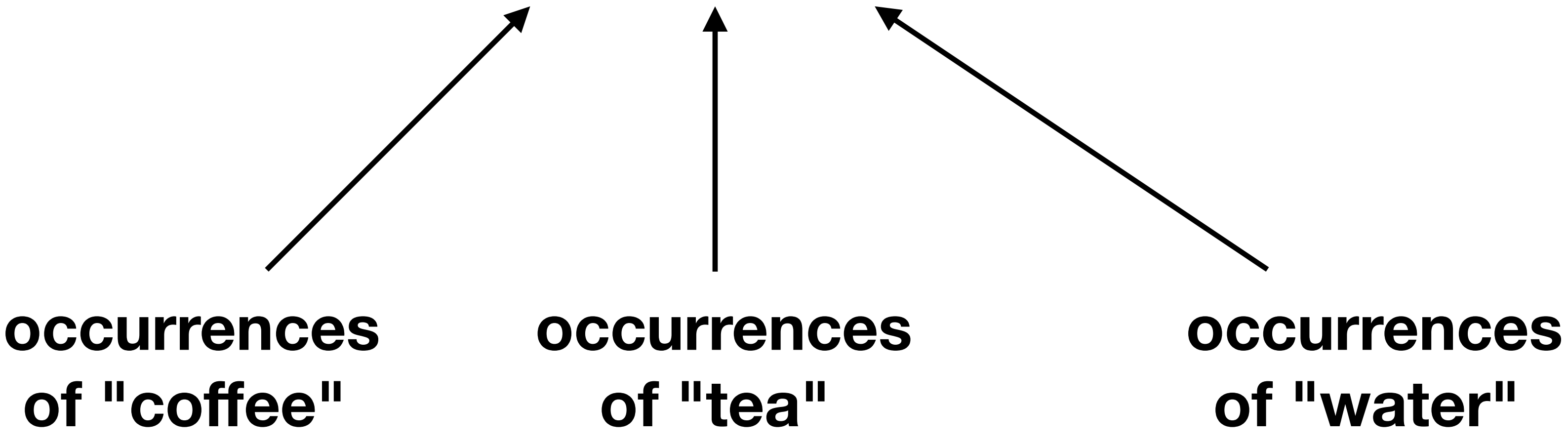
occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {1, 0, 2}
```

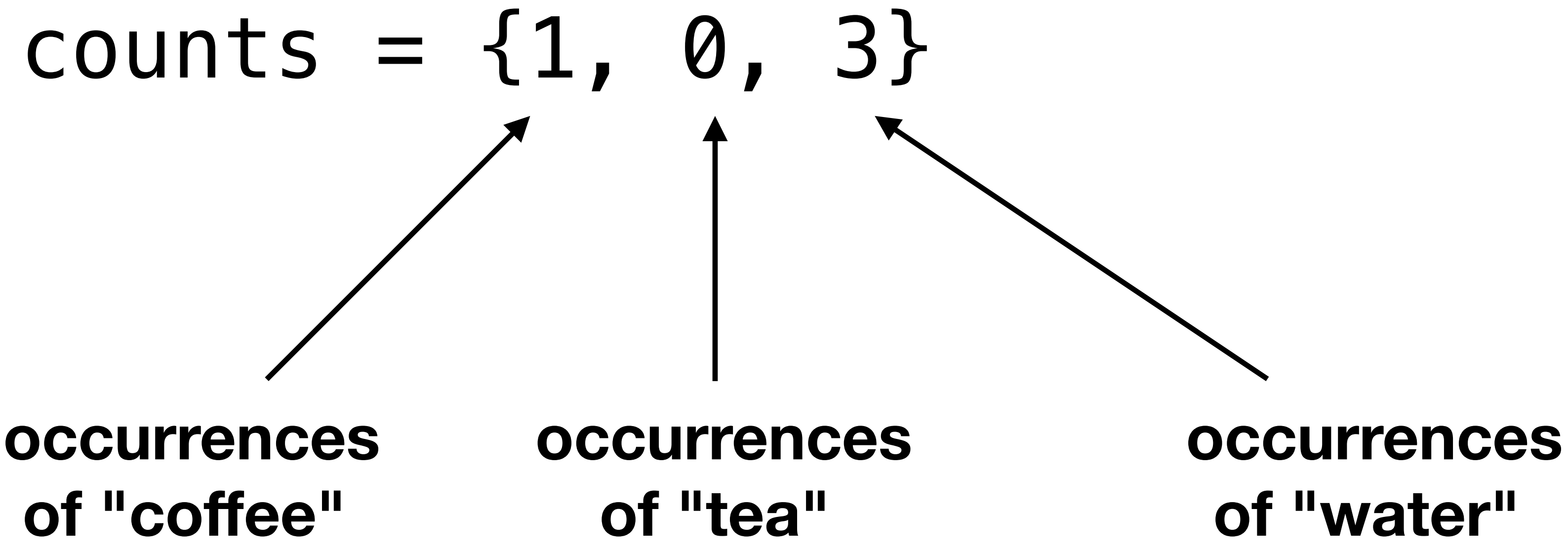


occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`



```
counts = {1, 0, 3}
```

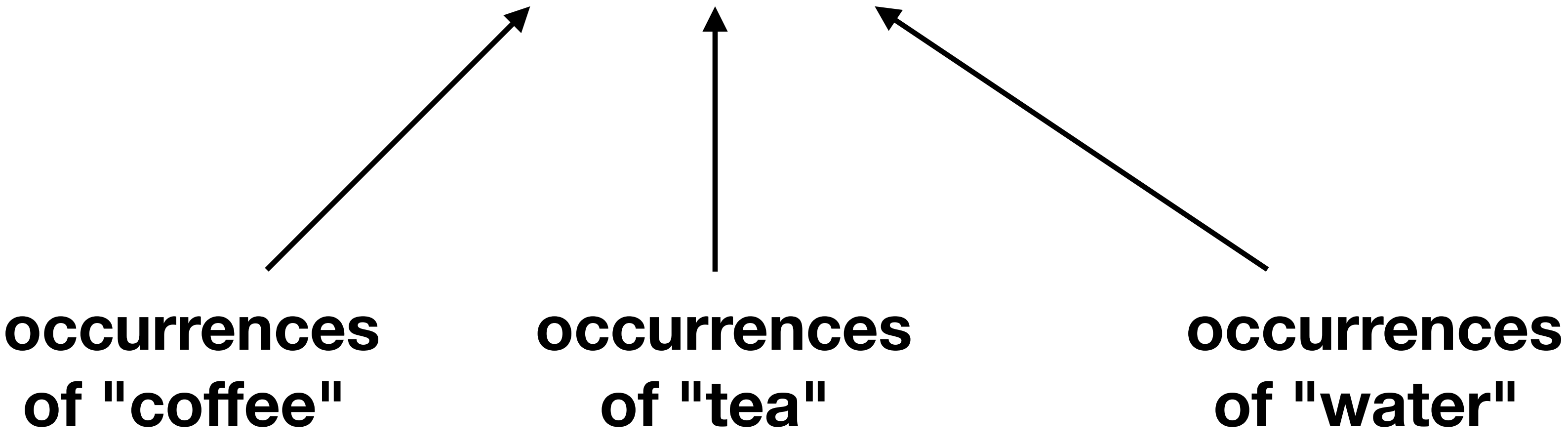
occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {2, 0, 3}
```

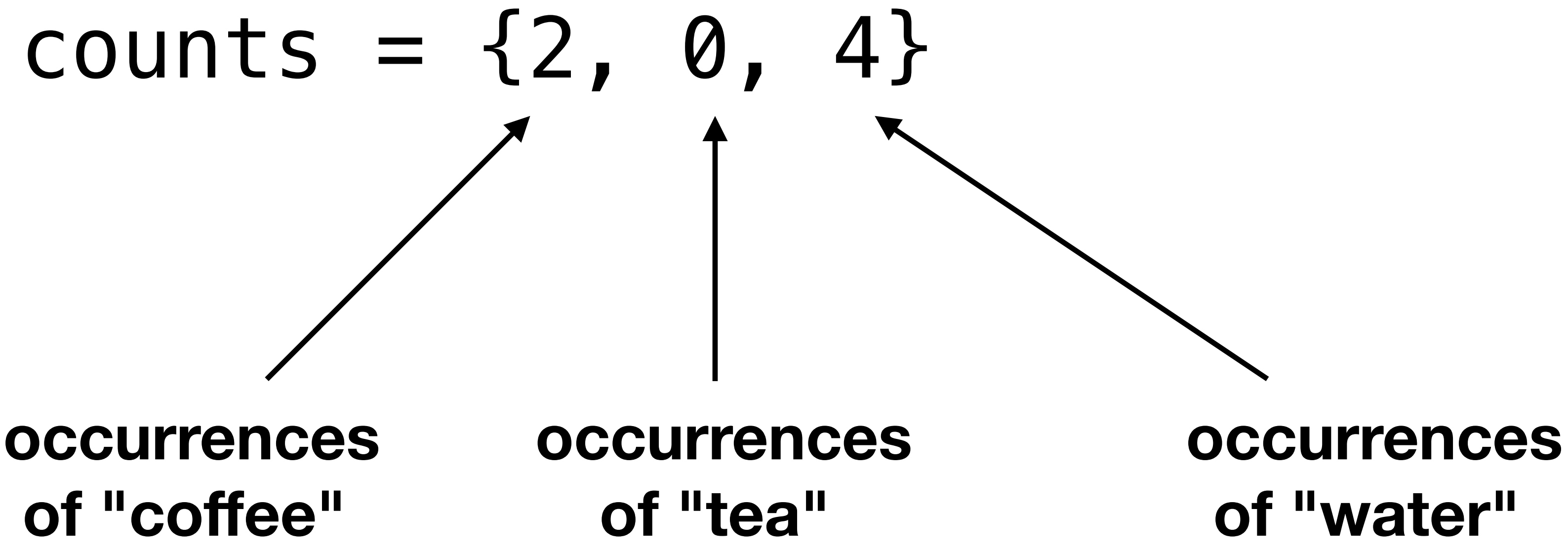


occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

-
  ```
  myDrinks = {"water", "coffee", "water", "water",
      "coffee", "water", "tea", "coffee", "tea"}
  ```



```
counts = {2, 0, 4}
```

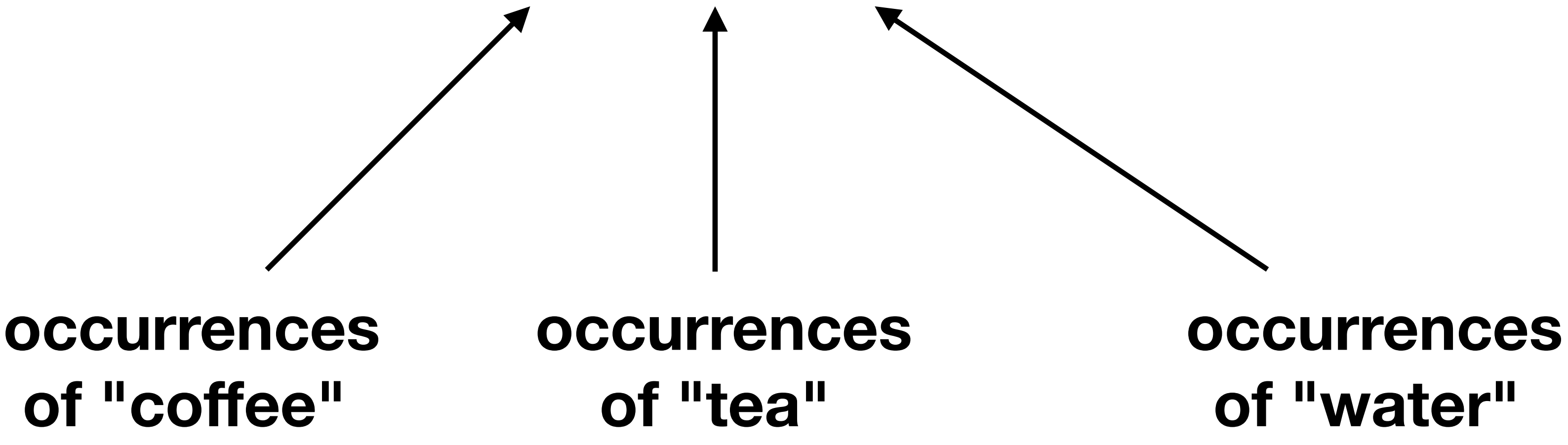
occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {2, 1, 4}
```

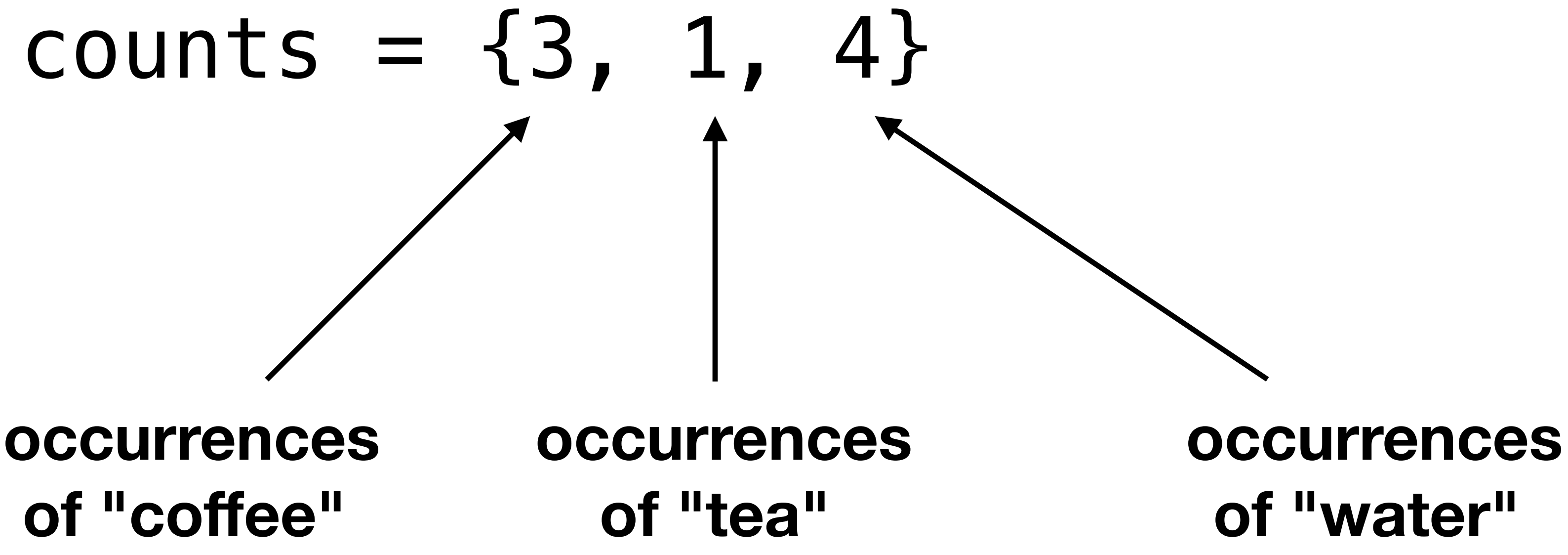


occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`



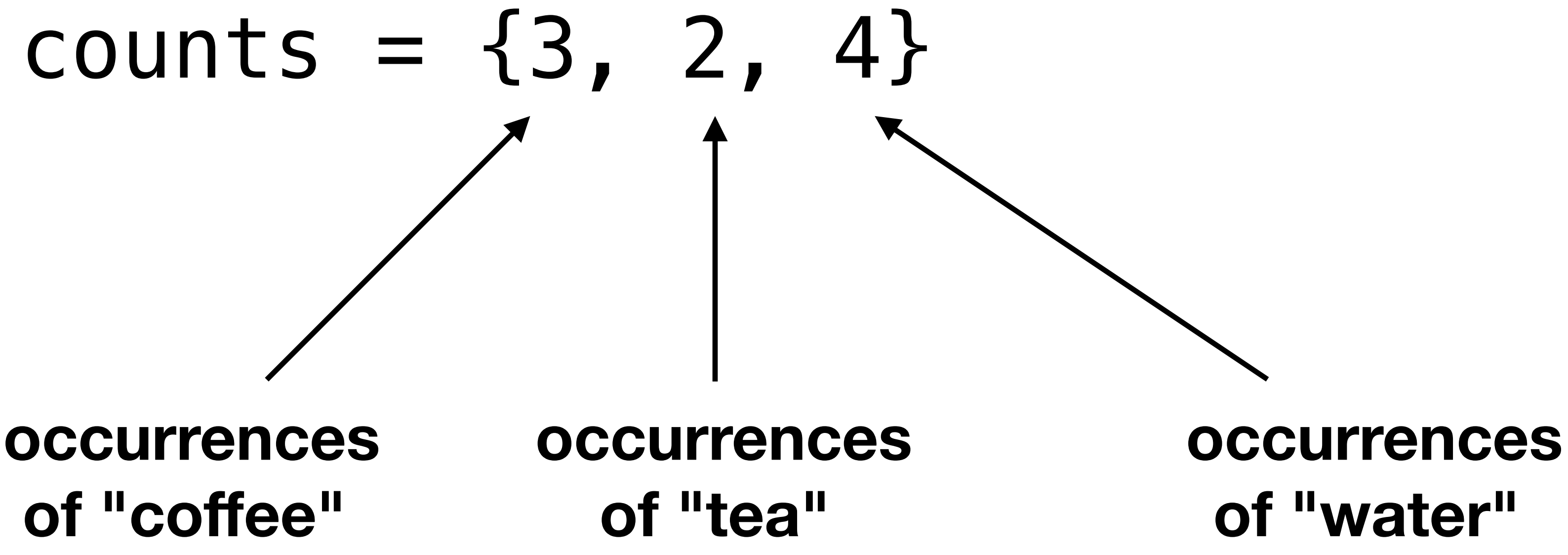
```
counts = {3, 1, 4}
```

occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`



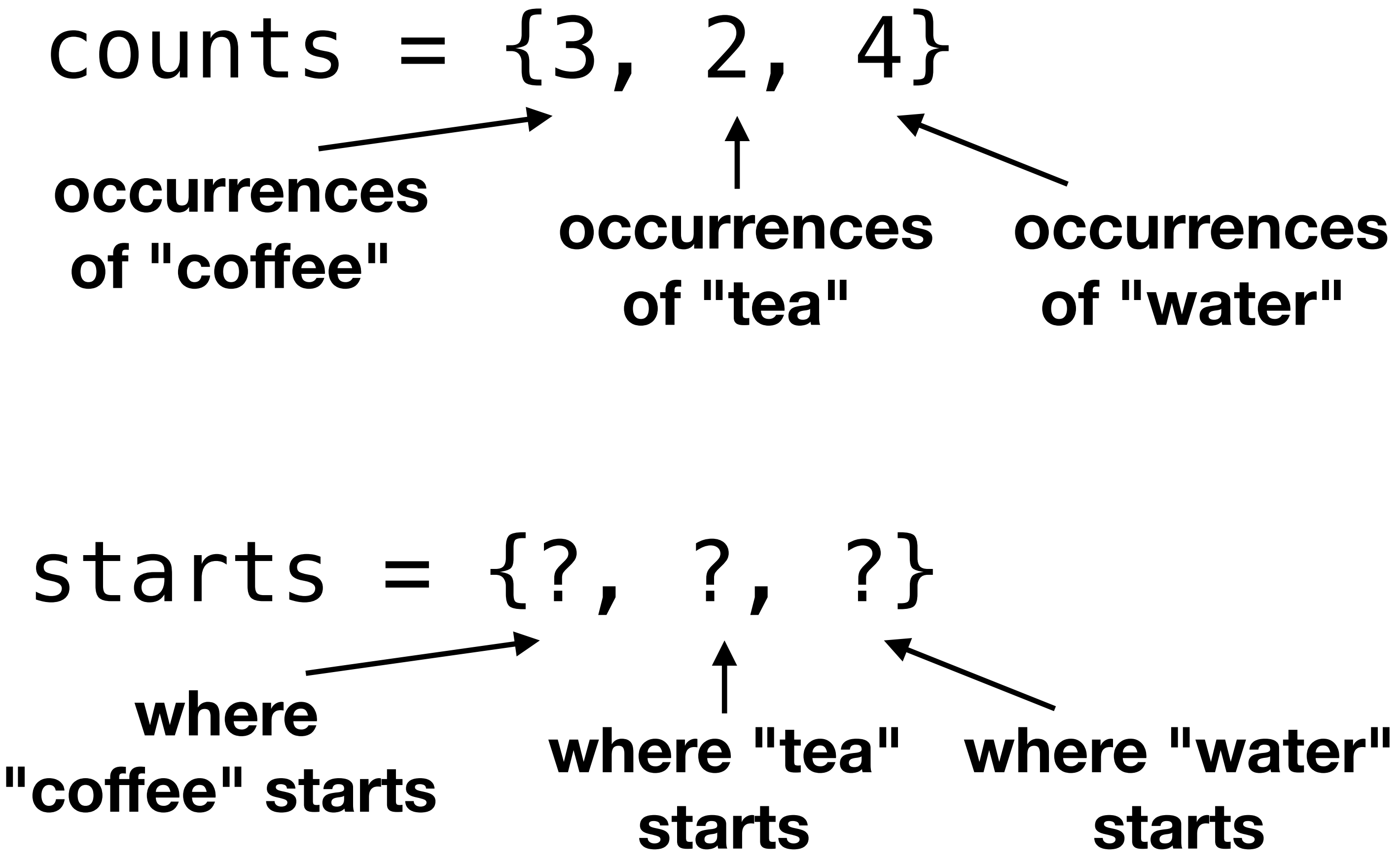
```
counts = {3, 2, 4}
```

occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`



```
counts = {3, 2, 4}
```

occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

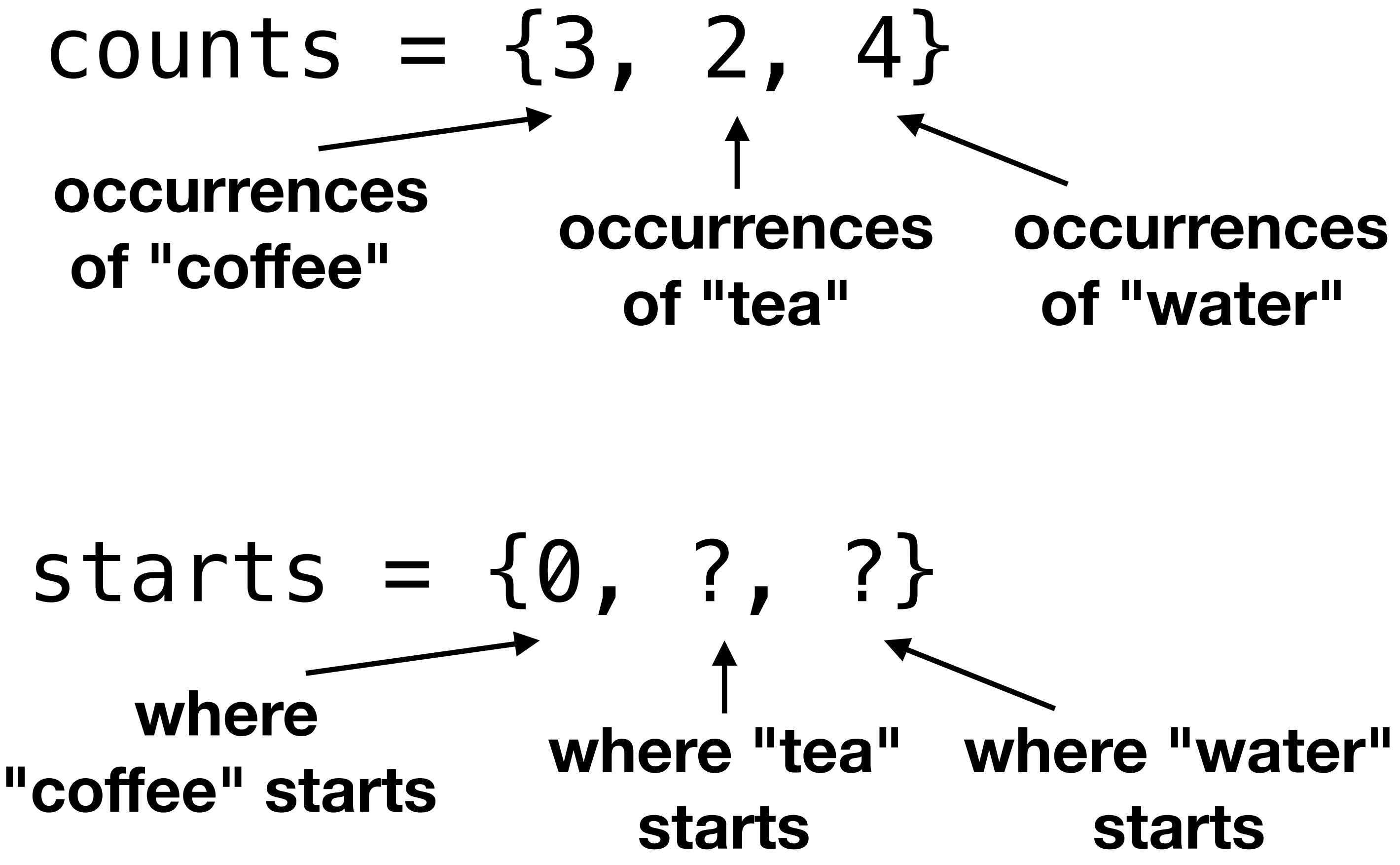
```
starts = {?, ?, ?}
```

where "coffee" starts

where "tea" starts

where "water" starts

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`



```
counts = {3, 2, 4}
```

occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

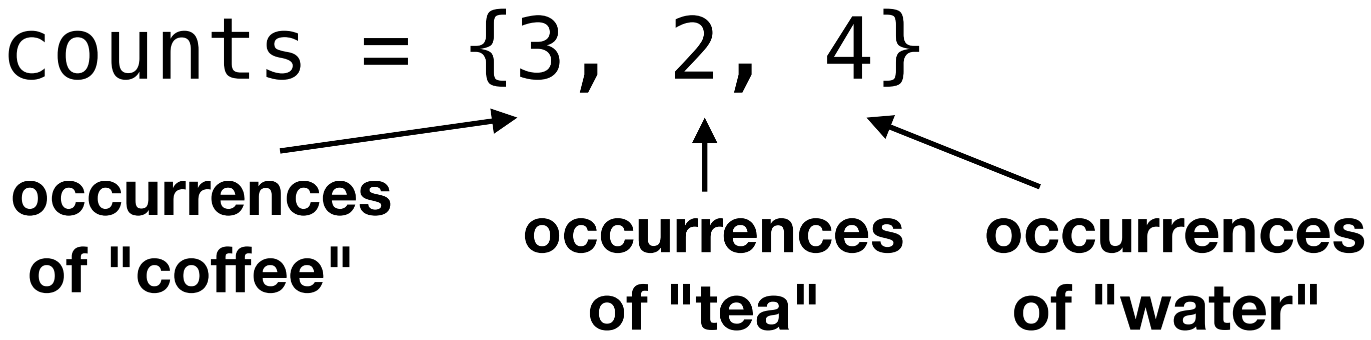
```
starts = {0, ?, ?}
```

where "coffee" starts

where "tea" starts

where "water" starts

-
  ```
  myDrinks = {"water", "coffee", "water", "water",
      "coffee", "water", "tea", "coffee", "tea"}
  ```

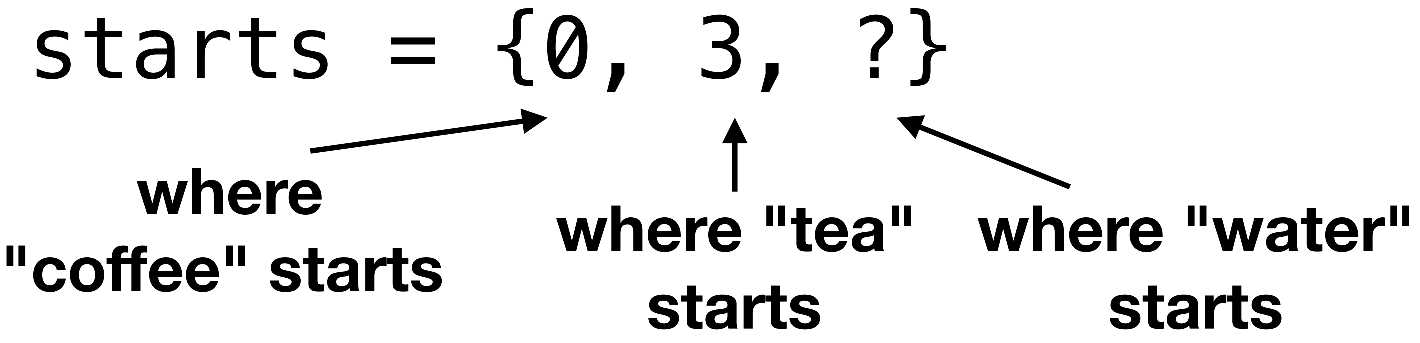


```
counts = {3, 2, 4}
```

occurrences of "coffee"

occurrences of "tea"

occurrences of "water"



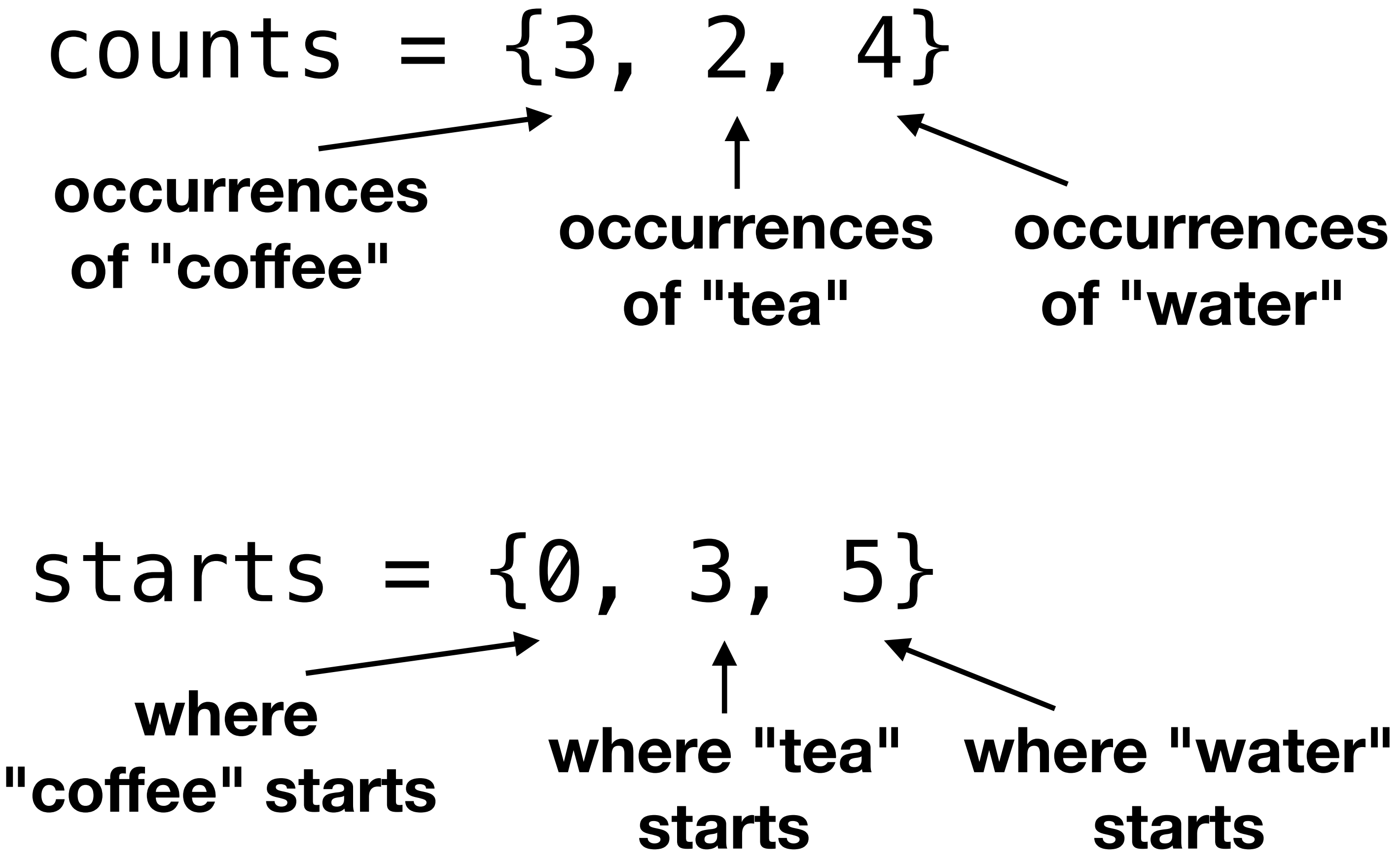
```
starts = {0, 3, ?}
```

where "coffee" starts

where "tea" starts

where "water" starts

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`



```
counts = {3, 2, 4}
```

occurrences of "coffee"

occurrences of "tea"

occurrences of "water"

```
starts = {0, 3, 5}
```

where "coffee" starts

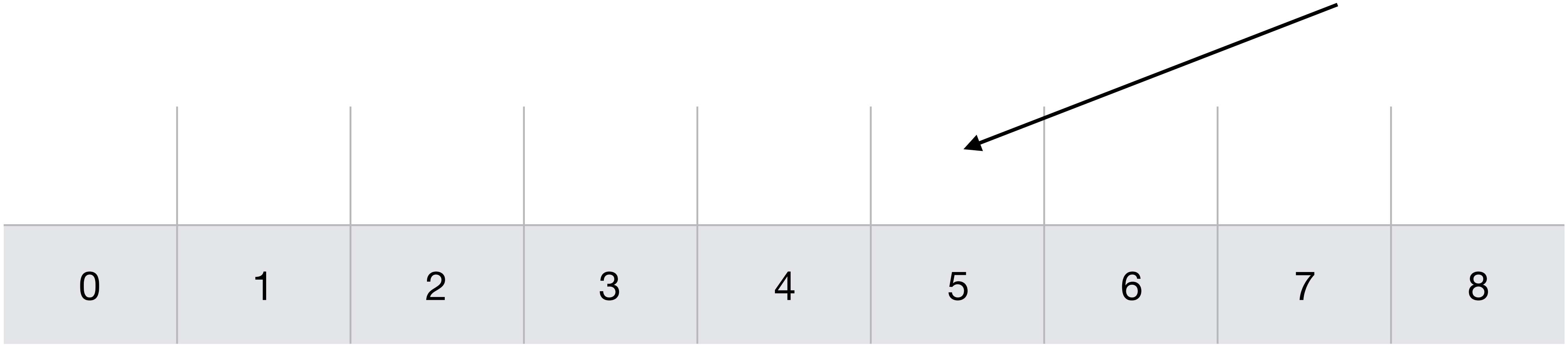
where "tea" starts

where "water" starts

- myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}

counts = {3, 2, 4}
☕ 🍵 💧

starts = {0, 3, 5}
☕ 🍵 💧



- myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}

counts = {3, 2, 4}

☕ 🍵 💧

starts = {0, 3, 5}

☕ 🍵 💧

					water			
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}            starts = {0, 3, 6}
          ☕  🍵  💧                       ☕  🍵  💧
```

					water			
0	1	2	3	4	5	6	7	8

-
  ```
  myDrinks = {"water", "coffee", "water", "water",
          "coffee", "water", "tea", "coffee", "tea"}
  ```

```
counts = {3, 2, 4}            starts = {0, 3, 6}
          ☕ 🍵 💧                      ☕ 🍵 💧
```

coffee					water			
0	1	2	3	4	5	6	7	8

-
  ```
  myDrinks = {"water", "coffee", "water", "water",
      "coffee", "water", "tea", "coffee", "tea"}
  ```

```
counts = {3, 2, 4}
          ☕ 🍵 💧
```

```
starts = {1, 3, 6}
          ☕ 🍵 💧
```

coffee					water			
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}
          ☕  🍵  💧
```

```
starts = {1, 3, 6}
          ☕  🍵  💧
```

coffee					water	water		
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}            starts = {1, 3, 7}
          ☕  🍵  💧                     ☕  🍵  💧
```

coffee					water	water		
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}          starts = {1, 3, 7}
          ☕  🍵  💧                    ☕  🍵  💧
```

coffee					water	water	water	
0	1	2	3	4	5	6	7	8

- myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}

counts = {3, 2, 4}
☕ 🍵 💧

starts = {1, 3, 8}
☕ 🍵 💧

coffee					water	water	water	
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}              starts = {1, 3, 8}
          ☕ 🍵 💧                          ☕ 🍵 💧
```

coffee	coffee				water	water	water	
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

`counts = {3, 2, 4}`

☕ 🍵 💧

`starts = {2, 3, 8}`

☕ 🍵 💧

coffee	coffee				water	water	water	
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}           starts = {2, 3, 8}
          ☕  🍵  💧                    ☕  🍵  💧
```

coffee	coffee				water	water	water	water
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}          starts = {2, 3, 9}
          ☕  🍵  💧                   ☕  🍵  💧
```

coffee	coffee				water	water	water	water
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}          starts = {2, 3, 9}
          ☕  🍵  💧                    ☕  🍵  💧
```

coffee	coffee		tea		water	water	water	water
0	1	2	3	4	5	6	7	8

- myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}

```
counts = {3, 2, 4}          starts = {2, 4, 9}
          ☕  🍵  💧                    ☕  🍵  💧
```

coffee	coffee		tea		water	water	water	water
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}          starts = {2, 4, 9}
          ☕  🍵  💧                   ☕  🍵  💧
```

coffee	coffee	coffee	tea		water	water	water	water
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}
          ☕  🍵  💧
```

```
starts = {3, 4, 9}
          ☕  🍵  💧
```

coffee	coffee	coffee	tea		water	water	water	water
0	1	2	3	4	5	6	7	8

-
  ```
  myDrinks = {"water", "coffee", "water", "water",
      "coffee", "water", "tea", "coffee", "tea"}
  ```

```
counts = {3, 2, 4}            starts = {3, 4, 9}
          ☕ 🍵 💧                      ☕ 🍵 💧
```

coffee	coffee	coffee	tea	tea	water	water	water	water
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}          starts = {3, 5, 9}
          ☕  🍵  💧                     ☕  🍵  💧
```

coffee	coffee	coffee	tea	tea	water	water	water	water
0	1	2	3	4	5	6	7	8

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}          starts = {3, 5, 9}
          ☕ 🍵 💧                     ☕ 🍵 💧
```

coffee	coffee	coffee	tea	tea	water	water	water	water
0	1	2	3	4	5	6	7	8

stable?

- myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}

```
counts = {3, 2, 4}          starts = {3, 5, 9}
          ☕  🍵  💧                   ☕  🍵  💧
```

coffee	coffee	coffee	tea	tea	water	water	water	water
0	1	2	3	4	5	6	7	8

stable? Yes!

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

↗

```
counts = {3, 2, 4}          starts = {3, 5, 9}
          ☕ 🍵 💧                     ☕ 🍵 💧
```

coffee	coffee	coffee	tea	tea	water	water	water	water
0	1	2	3	4	5	6	7	8

Runtime? Let # distinct elements = K.

- `myDrinks = {"water", "coffee", "water", "water", "coffee", "water", "tea", "coffee", "tea"}`

```
counts = {3, 2, 4}          starts = {3, 5, 9}
          ☕  🍵  💧                   ☕  🍵  💧
```

coffee	coffee	coffee	tea	tea	water	water	water	water
0	1	2	3	4	5	6	7	8

Runtime? Let # distinct elements = K.

$\Theta(N + K)$

Extending Counting Sort

- Although counting sort only works if there are a small set of values, there are many cases that items to be sorted are composed of elements that can only take on a small set of values.
- For example, numbers in base 10 like *1283642* are made up of digits, which can only be one of ten values (0 through 9).
- Can we use counting sort on each digit to quickly sort a list of numbers?

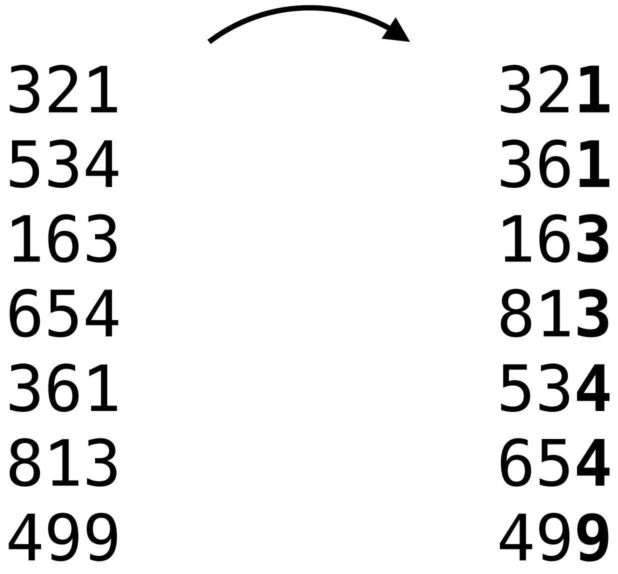
Least Significant Digit Radix Sort

- Let's sort from least significant digit to most significant digit.
- We'll use counting sort, but only using one digit at a time as our search key.

```
321
534
163
654
361
813
499
```

Least Significant Digit Radix Sort

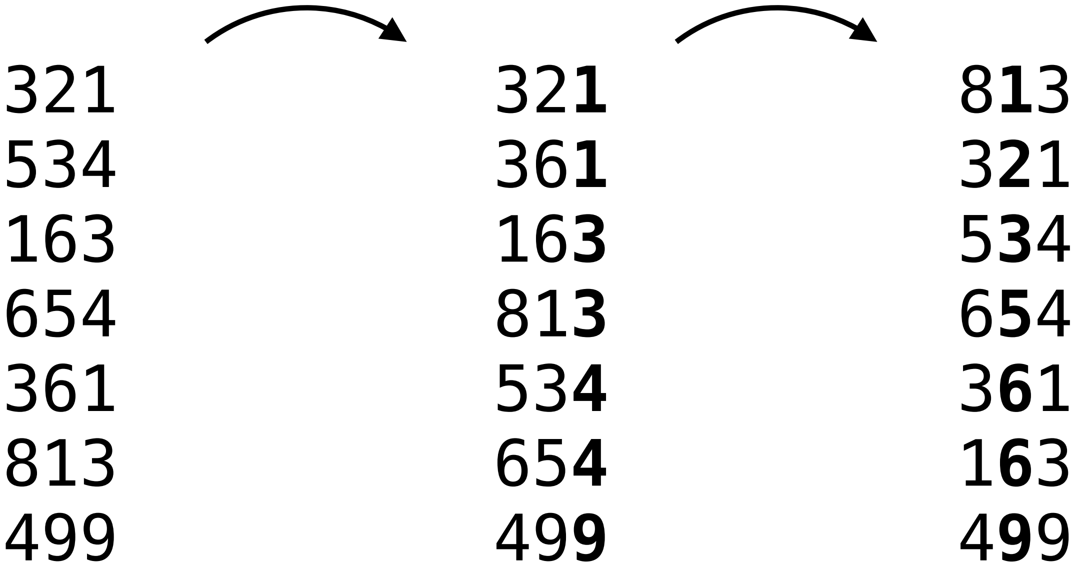
- Let's sort from least significant digit to most significant digit.
- We'll use counting sort, but only using one digit at a time as our search key.



321	32**1**
534	36**1**
163	16**3**
654	81**3**
361	53**4**
813	65**4**
499	49**9**

Least Significant Digit Radix Sort

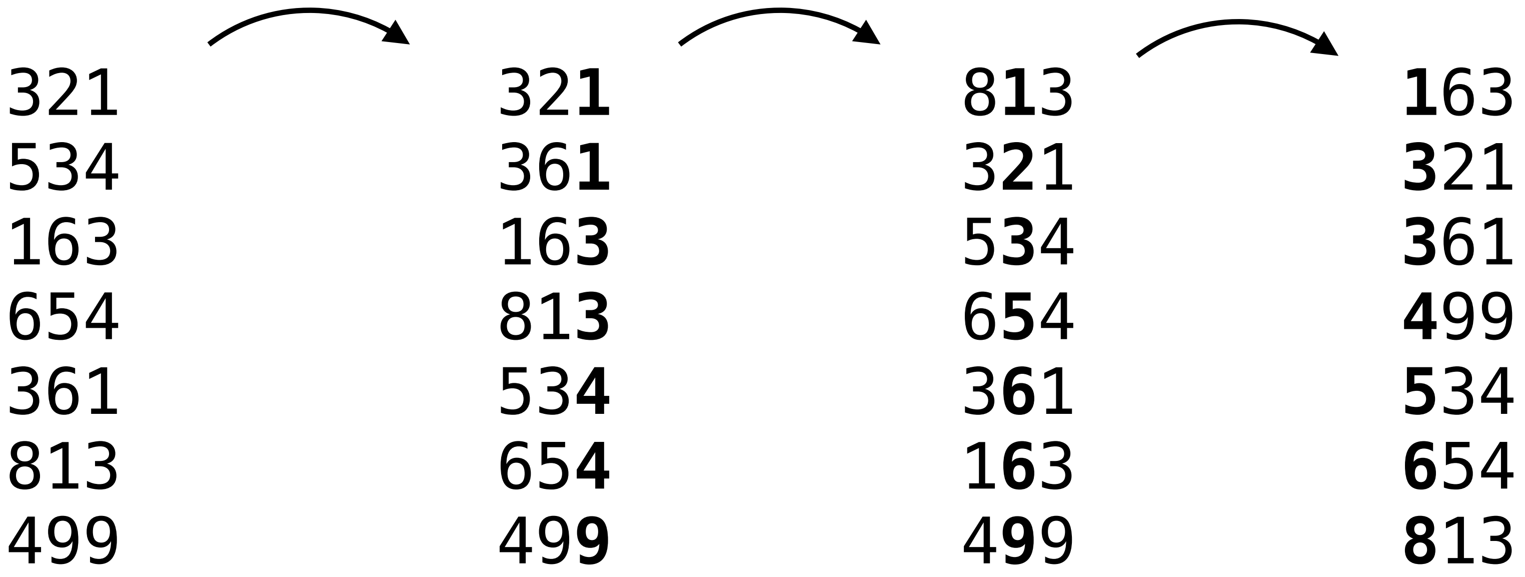
- Let's sort from least significant digit to most significant digit.
- We'll use counting sort, but only using one digit at a time as our search key.



321	32**1**	8**1**3
534	36**1**	3**2**1
163	16**3**	5**3**4
654	81**3**	6**5**4
361	53**4**	3**6**1
813	65**4**	1**6**3
499	49**9**	4**9**9

Least Significant Digit Radix Sort

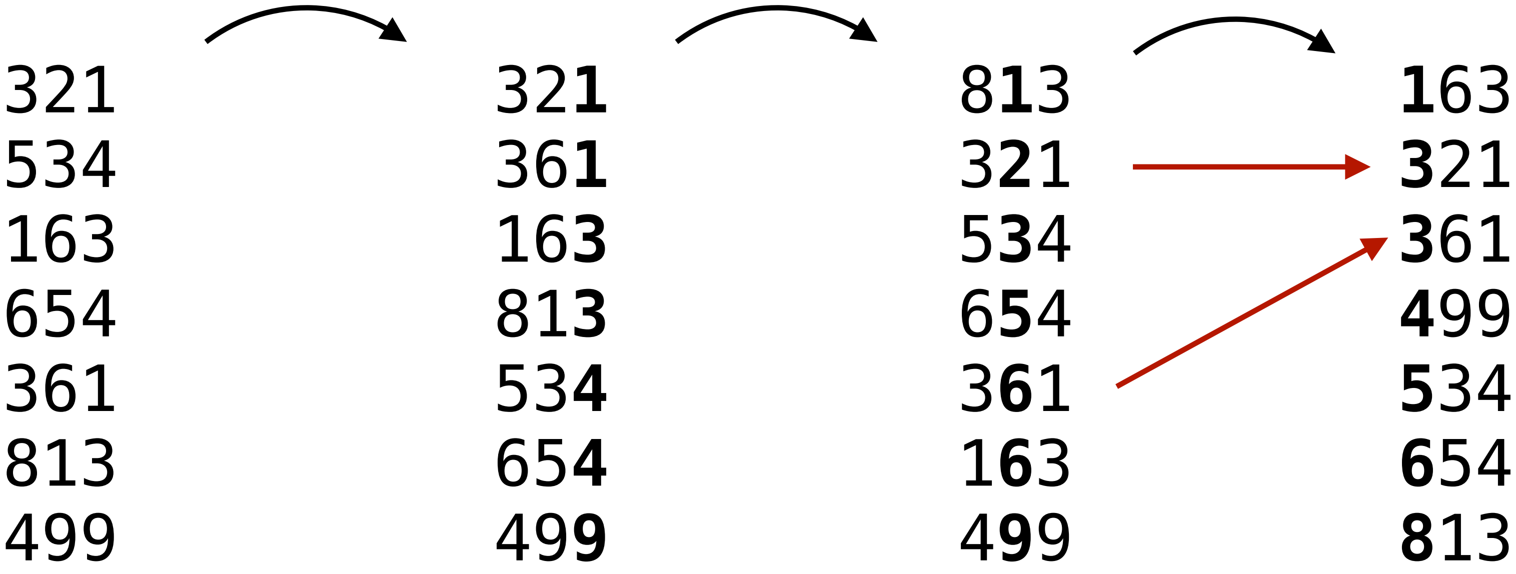
- Let's sort from least significant digit to most significant digit.
- We'll use counting sort, but only using one digit at a time as our search key.



321	32**1**	8**1**3	**1**63
534	36**1**	3**2**1	**3**21
163	16**3**	5**3**4	**3**61
654	81**3**	6**5**4	**4**99
361	53**4**	3**6**1	**5**34
813	65**4**	1**6**3	**6**54
499	49**9**	4**9**9	**8**13

Least Significant Digit Radix Sort

- Let's sort from least significant digit to most significant digit.
- We'll use counting sort, but only using one digit at a time as our search key.

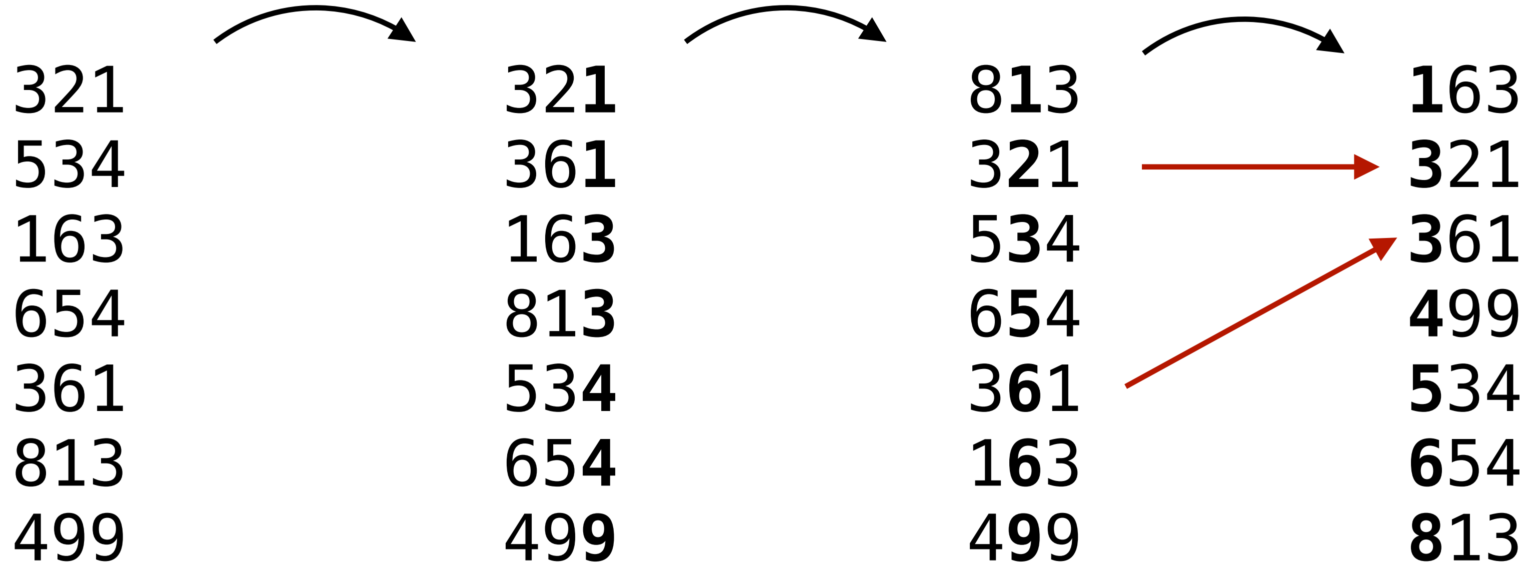


321	32**1**	8**1**3	**1**63
534	36**1**	3**2**1	**3**21
163	16**3**	5**3**4	**3**61
654	81**3**	6**5**4	**4**99
361	53**4**	3**6**1	**5**34
813	65**4**	1**6**3	**6**54
499	49**9**	4**9**9	**8**13

Notice how it is essential that counting sort is stable!

Least Significant Digit Radix Sort

- Let's sort from least significant digit to most significant digit.
- We'll use counting sort, but only using one digit at a time as our search key.

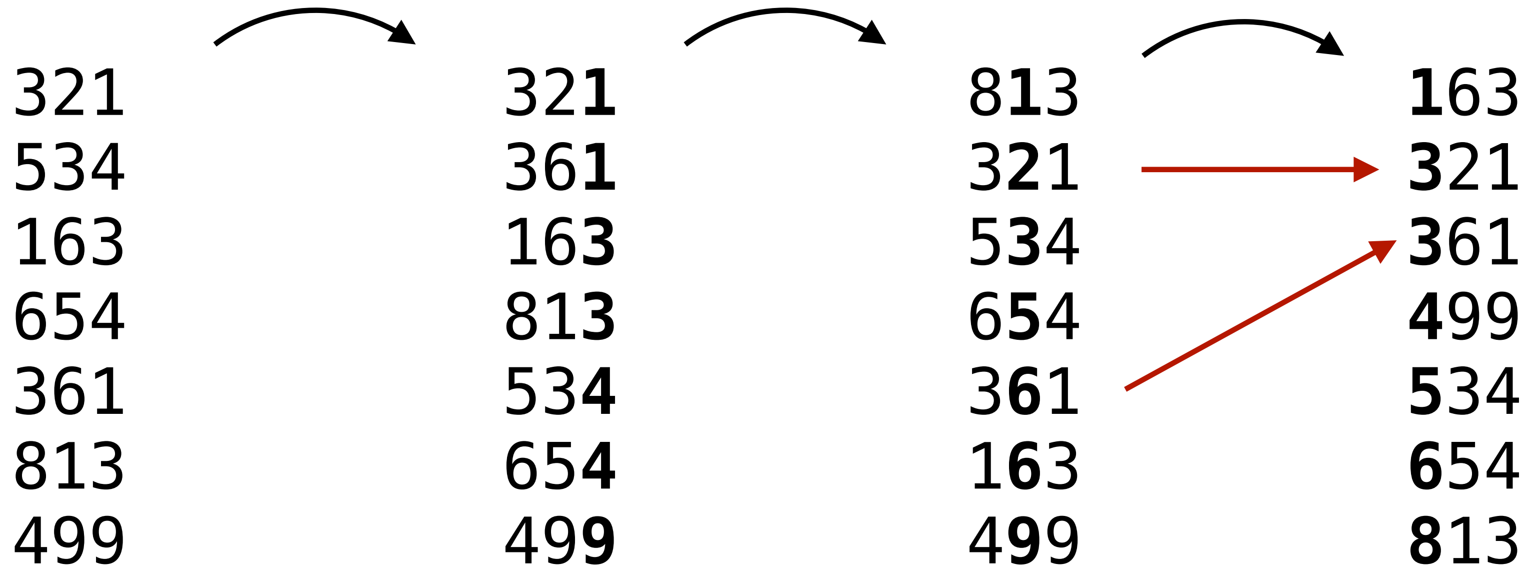


321	32**1**	8**1**3	**1**63
534	36**1**	3**2**1	**3**21
163	16**3**	5**3**4	**3**61
654	81**3**	6**5**4	**4**99
361	53**4**	3**6**1	**5**34
813	65**4**	1**6**3	**6**54
499	49**9**	4**9**9	**8**13

Runtime? Let D be max # of digits and K be # of distinct possible digits.

Least Significant Digit Radix Sort

- Let's sort from least significant digit to most significant digit.
- We'll use counting sort, but only using one digit at a time as our search key.

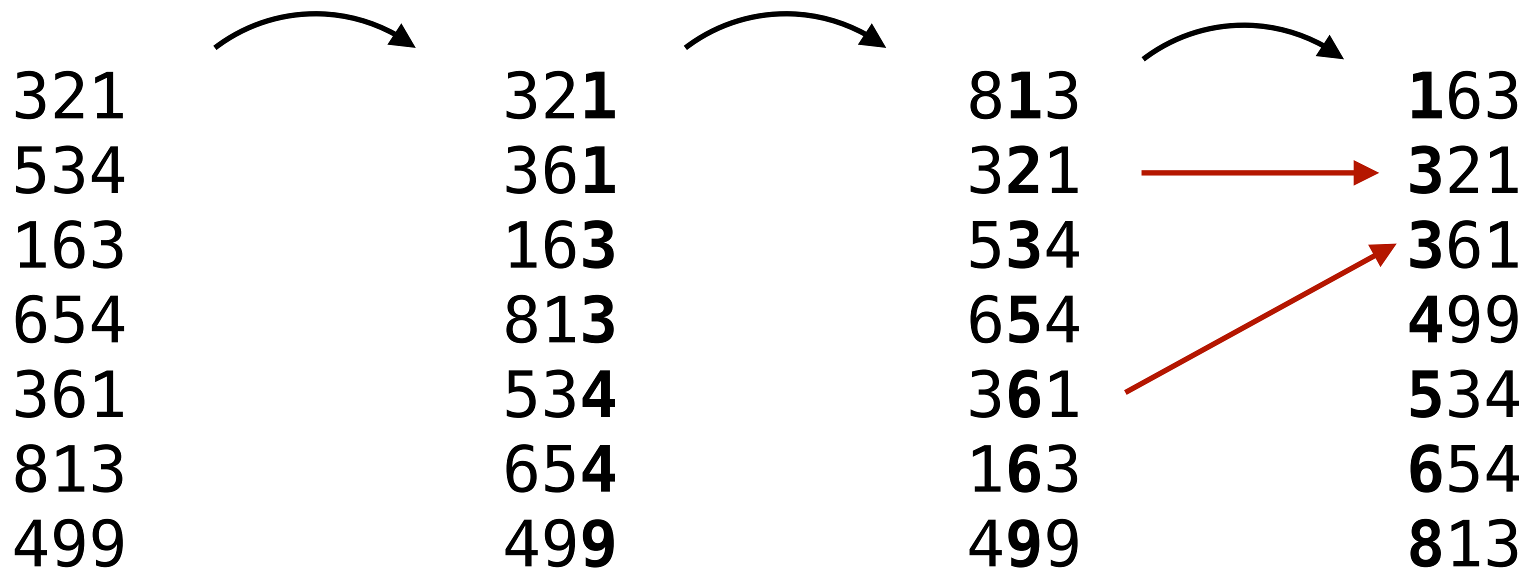


321	32**1**	8**13**	**163**
534	36**1**	3**21**	**321**
163	16**3**	5**34**	**361**
654	81**3**	6**54**	**499**
361	53**4**	3**61**	**534**
813	65**4**	1**63**	**654**
499	49**9**	4**99**	**813**

Runtime? Let D be max # of digits and K be # of distinct possible digits. E.g. for base 10, K = 10.

Least Significant Digit Radix Sort

- Let's sort from least significant digit to most significant digit.
- We'll use counting sort, but only using one digit at a time as our search key.



321	32**1**	8**13**	**163**
534	36**1**	3**21**	**321**
163	16**3**	5**34**	**361**
654	81**3**	6**54**	**499**
361	53**4**	3**61**	**534**
813	65**4**	1**63**	**654**
499	49**9**	4**99**	**813**

Runtime? Let D be max # of digits and K be # of distinct possible digits. E.g. for base 10, K = 10.

$\Theta\big(D(N+K)\big)$

Most Significant Digit Radix Sort

- What about from most significant digit to least significant digit?
- We can't do the same thing as before, since the most-recently applied sort takes priority.

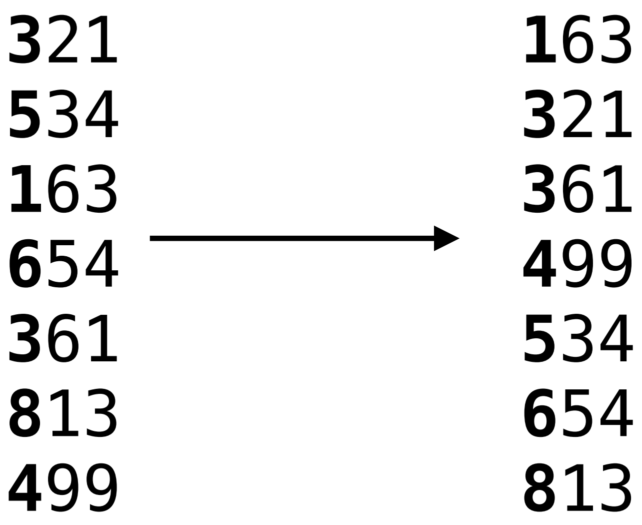
Most Significant Digit Radix Sort

- What about from most significant digit to least significant digit?
- We can't do the same thing as before, since the most-recently applied sort takes priority.

321
534
163
654
361
813
499

Most Significant Digit Radix Sort

- What about from most significant digit to least significant digit?
- We can't do the same thing as before, since the most-recently applied sort takes priority.

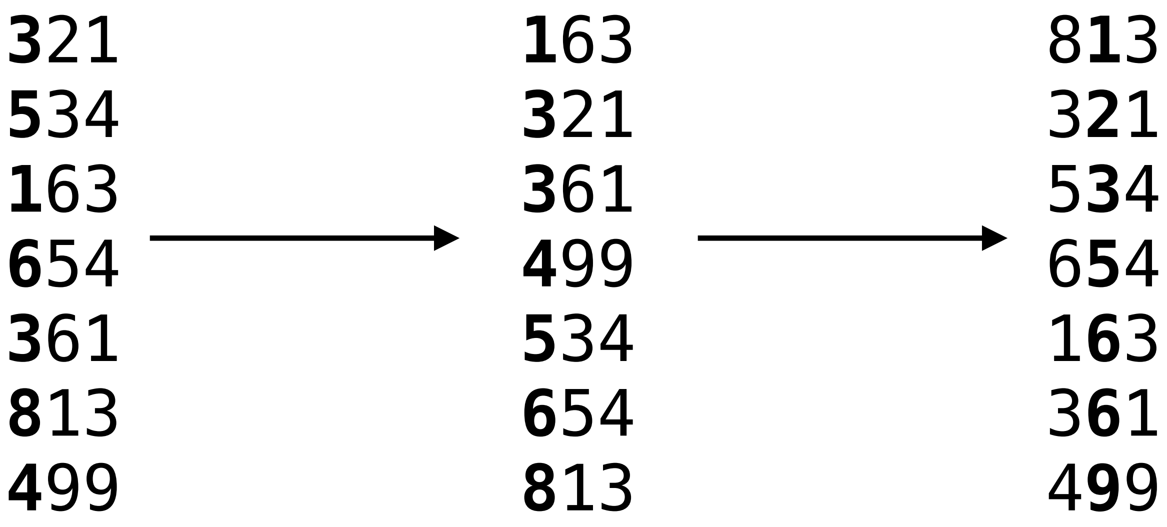


321
534
163
654
361
813
499

163
321
361
499
534
654
813

Most Significant Digit Radix Sort

- What about from most significant digit to least significant digit?
- We can't do the same thing as before, since the most-recently applied sort takes priority.



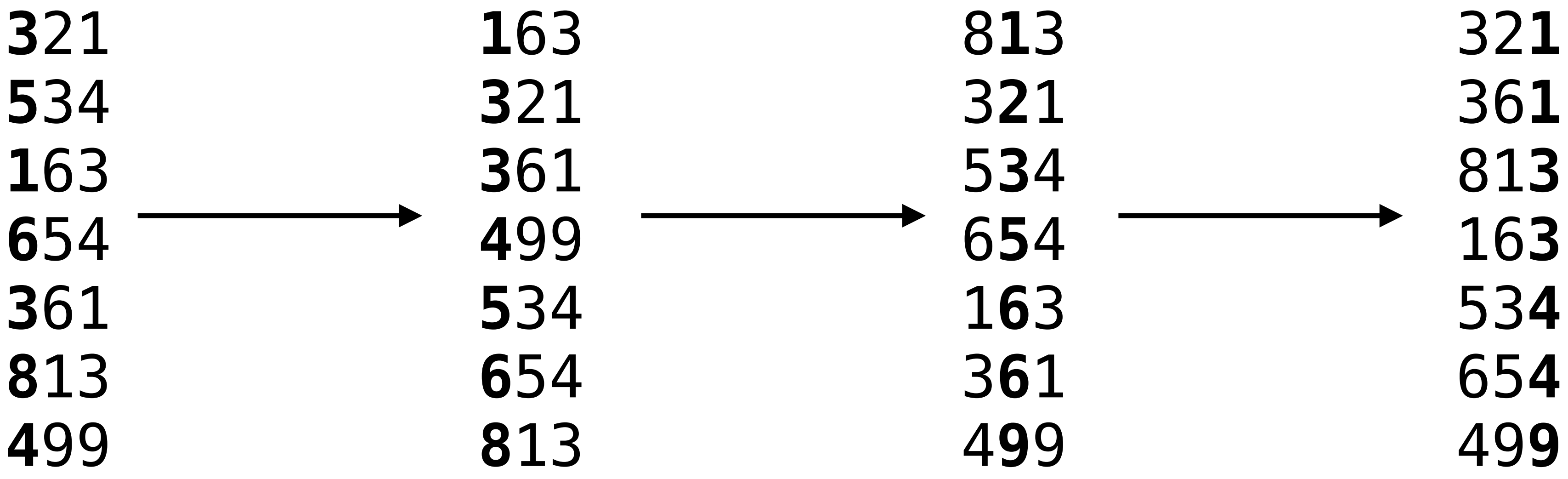
321
534
163
654
361
813
499

163
321
361
499
534
654
813

8**1**3
3**2**1
5**3**4
6**5**4
1**6**3
3**6**1
4**9**9

Most Significant Digit Radix Sort

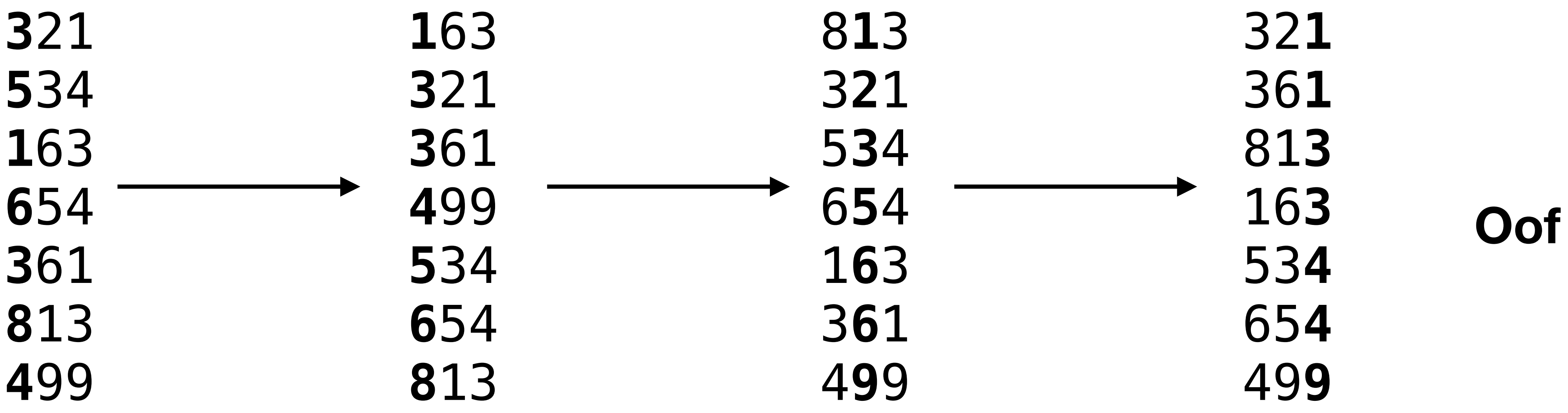
- What about from most significant digit to least significant digit?
- We can't do the same thing as before, since the most-recently applied sort takes priority.



321		**1**63		8**1**3		32**1**
534		**3**21		3**2**1		36**1**
163		**3**61		5**3**4		81**3**
654	→	**4**99	→	6**5**4	→	16**3**
361		**5**34		1**6**3		53**4**
813		**6**54		3**6**1		65**4**
499		**8**13		4**9**9		49**9**

Most Significant Digit Radix Sort

- What about from most significant digit to least significant digit?
- We can't do the same thing as before, since the most-recently applied sort takes priority.



321
534
163
654
361
813
499

163
321
361
499
534
654
813

8**1**3
3**2**1
5**3**4
6**5**4
1**6**3
3**6**1
4**9**9

32**1**
36**1**
81**3**
16**3**
53**4**
65**4**
49**9**

Oof

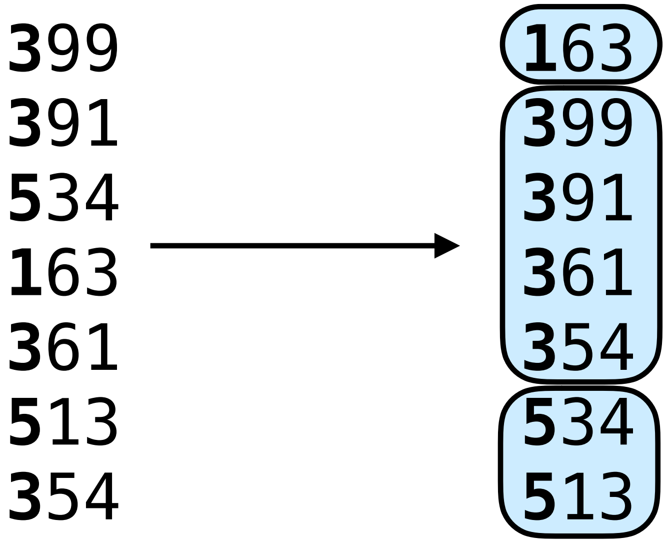
Most Significant Digit Radix Sort

- We have to group items with the same digit into buckets and then sort the buckets recursively.

399
391
534
163
361
513
354

Most Significant Digit Radix Sort

- We have to group items with the same digit into buckets and then sort the buckets recursively.



399
391
534
163
361
513
354

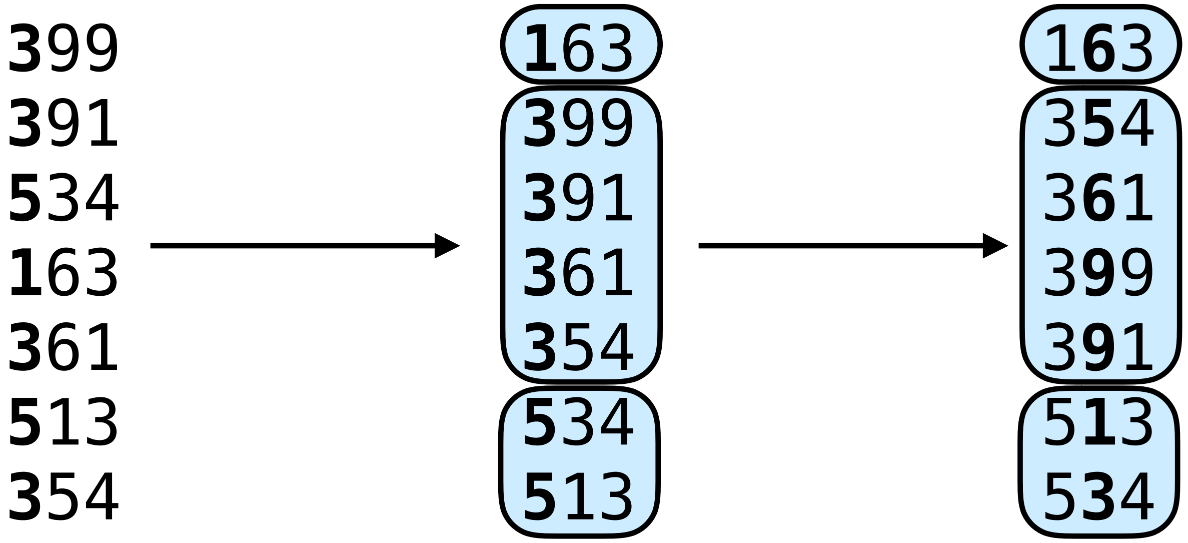
163

399
391
361
354

534
513

Most Significant Digit Radix Sort

- We have to group items with the same digit into buckets and then sort the buckets recursively.



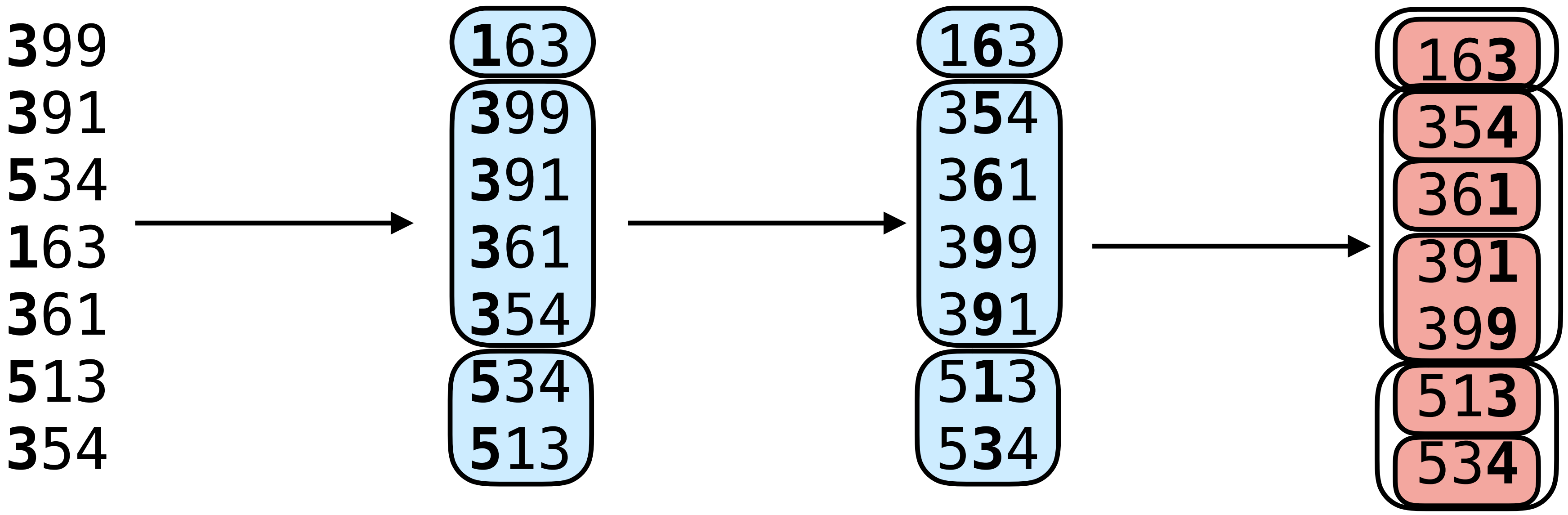
399
391
534
163
361
513
354

163
399
391
361
354
534
513

1**6**3
3**5**4
3**6**1
3**9**9
3**9**1
5**1**3
5**3**4

Most Significant Digit Radix Sort

- We have to group items with the same digit into buckets and then sort the buckets recursively.



399
391
534
163
361
513
354

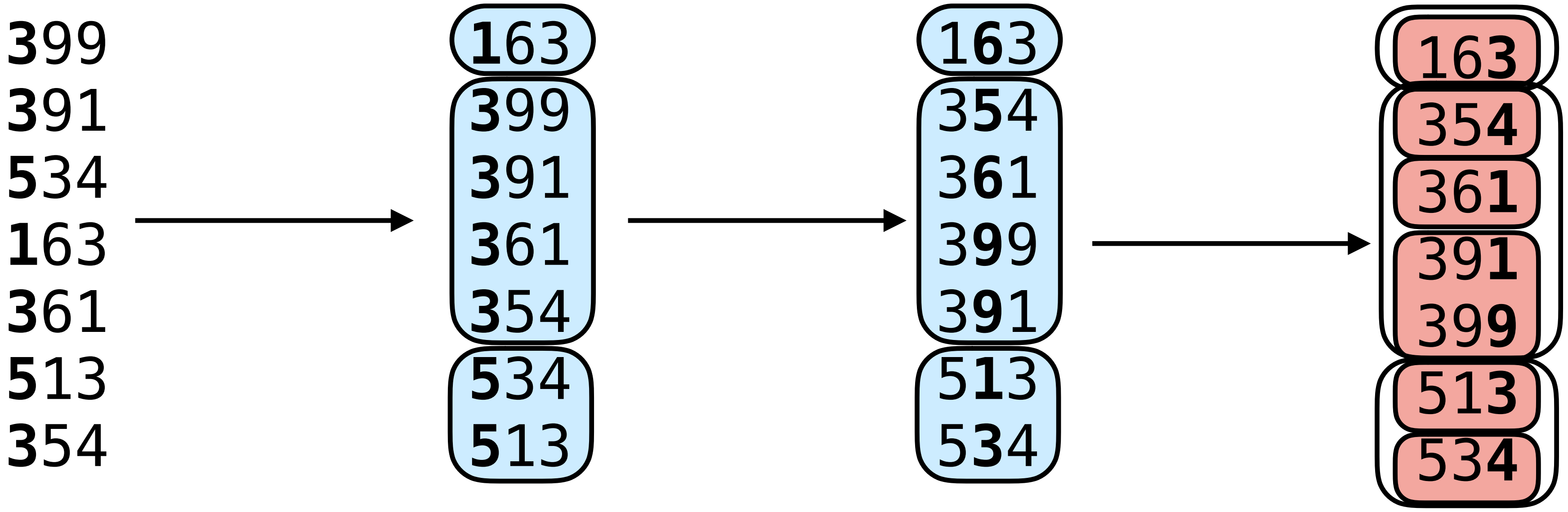
163
399
391
361
354
534
513

1**6**3
3**5**4
3**6**1
3**9**9
3**9**1
5**1**3
5**3**4

16**3**
35**4**
36**1**
39**1**
39**9**
51**3**
53**4**

Most Significant Digit Radix Sort

- We have to group items with the same digit into buckets and then sort the buckets recursively.



399
391
534
163
361
513
354

163
399
391
361
354
534
513

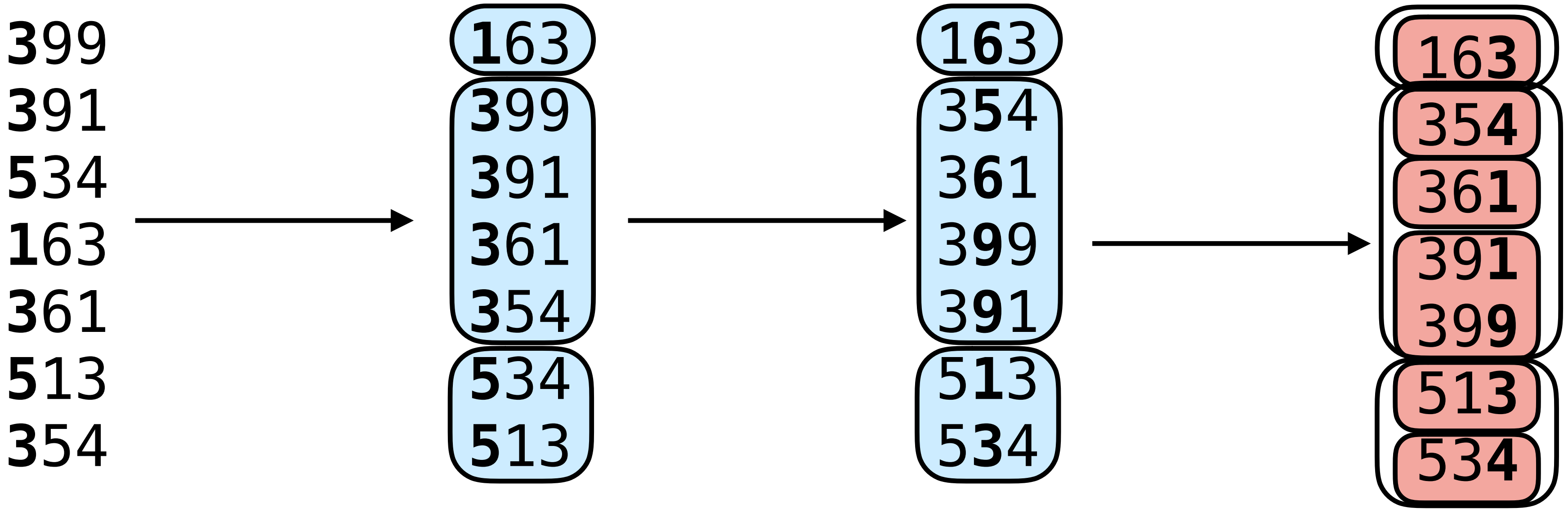
163
354
361
399
391
513
534

163
354
361
391
399
513
534

Runtime? Let D be max # of digits and K be # of distinct possible digits.

Most Significant Digit Radix Sort

- We have to group items with the same digit into buckets and then sort the buckets recursively.



Runtime? Let D be max # of digits and K be # of distinct possible digits.

It's possible that when we sort the left-most digit, all of the most significant digits are unique, so we're done in one pass, but it's also possible we have to recurse through all of the digits. So $\Omega(N+K), O\left(D(N+K)\right)$