Algorithm performance

Searching analysis

By the end of this video you will be able to…

- State and justify the asymptotic performance for
  - linear search,
  - binary search

in the best case and in the worst case

	Best case	Worst case
Linear Search		
Binary Search*		

* Assuming data is sorted

Linear Search: Basic Algorithm

Start at the first **index** in the array

while **index < length** of the array:
    if toFind matches current value,
        return true
    increment index by 1

return false

Linear Search: Basic Algorithm

Start at the first **index** in the array

while **inde**

E.g. `hasLetter(String word, char letter)`

value,

increment index by 1

return false

	Best case	Worst case
Linear Search	**O(1)**	
Binary Search*		

* Assuming data is sorted

	Best case	Worst case
Linear Search	**O(1)**	
Binary Search*		

* Assuming data is sorted

Linear Search: Basic Algorithm

Start at the first **index** in the array

while **inde**

E.g. `hasLetter(String word, char letter)`

value,

increment index by 1

return false

	Best case	Worst case
Linear Search	**O(1)**	**O(n)**
Binary Search*		

* Assuming data is sorted

	Best case	Worst case
Linear Search	**O(1)**	**O(n)**
Binary Search*		

* Assuming data is sorted

Binary Search: Basic Algorithm

Initialize **low = 0, high = length of list**

while low <= high:
    mid = (high+low)/2
    if toFind matches value at mid,
        return true
    if toFind < value at mid
        high = mid-1
    else low = mid+1

return false

Binary Search: Basic Algorithm

Initialize **low = 0, high = length of list**

while low <= high:
    mid = (high+low)/2
    if toFind matches value at mid,
        return true
    if toFind < value at mid
        high = mid-1
    else low = mid+1

return false

Binary Search: Basic Algorithm

Initialize **low = 0, high = length of list**

while low <= high:
    mid = (high+low)/2
    if toFind matches value at mid,
        return true
    if toFind < value at mid
        high = mid-1
    else low = mid+1

return false

	Best case	Worst case
Linear Search	**O(1)**	**O(n)**
Binary Search*	**O(1)**	

* Assuming data is sorted

	Best case	Worst case
Linear Search	**O(1)**	**O(n)**
Binary Search*	**O(1)**	

* Assuming data is sorted

Binary Search: Basic Algorithm

Worst case : don't find!

Initialize **low = 0, high = length of list**

while low <= high:
    mid = (high+low)/2
    if toFind matches value at mid,
        return true
    if toFind < value at mid
        high = mid-1
    else low = mid+1

return false

Binary Search: Basic Algorithm

Worst case : don't find!

Initialize **low = 0, high = length of list**

```
while low <= high:
    mid = (high+low)/2
    if toFind matches value at mid,
        return true
    if toFind < value at mid
        high = mid-1
    else low = mid+1

return false
```

Binary Search: Basic Algorithm

Worst case : don't find!

Initialize **low = 0, high = length of list**

```
while low <= high:
    mid = (high+low)/2
    if toFind matches value at mid,
        return true
    if toFind < value at mid
        high = mid-1
    else low = mid+1

return false
```

 # times to half size?

	Best case	Worst case
Linear Search	**O(1)**	**O(n)**
Binary Search*	**O(1)**	**O(log n)**

* Assuming data is sorted

	Best case	Worst case
Linear Search	**O(1)**	**O(n)**
Binary Search*	**O(1)**	**O(log n)**

* Assuming data is sorted