Big O Notation

Time complexity of an algorithm

"How much time it takes to run a function as the size of the input grows."

{ Runtime

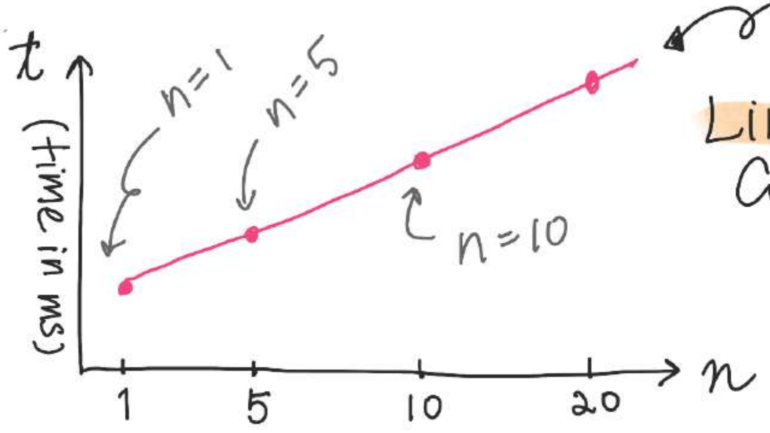
array

Const array1 = [🐱, 🌸, 🎩, 💩, 🪡] number of elements n = 5

Let's see if there is a needle in the haystack!

```
Const numNeedles = (haystack, needle) ⇒ {
  let count = 0
  for (let i = 0 ; haystack.length ; i++) {
    if (haystack[i] ≡ needle) count += 1;
  }
  return count;
}
```

How long does it take to execute when the number of elements (n) is:



♥ execution time grows linearly as array size increases!

this is Linear time Complexity = O(n)

@girlie_mac

@girlie_mac

Big O Notation

JS Let's see if we have some function that doesn't actually loop the array:

```
const alwaysTrueNoMatterWhat = (haystack) => {
  return true;
```

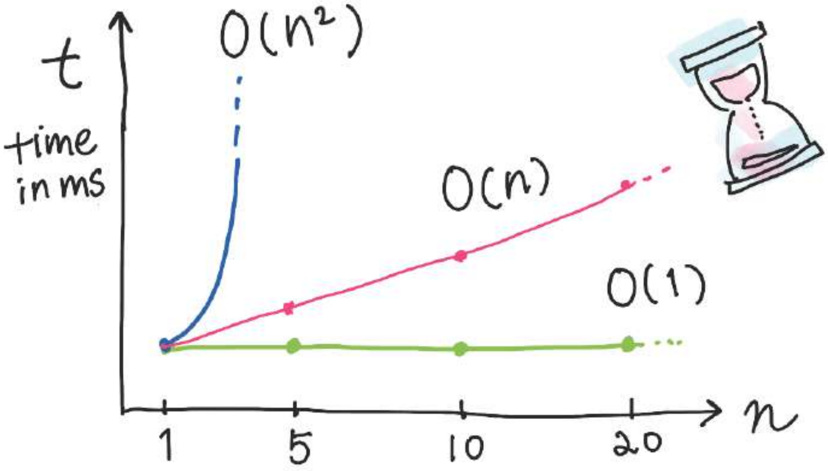
n=5
n=10
n=20
⋮

← Array size has no effect on the runtime

☆ Constant time

=

O(1)



☆ Quadratic time = O(n^2)

n=5, however the runtime is proportional to n^2

Const array2 = [ ];

```
JS Const hasDuplicates = (arr) => {
  for (let i = 0; i < arr.length; i++)
    let item = arr [i];
    if(arr.slice (i+1).indexOf(item) !== -1) {
      return true;
    }
  }
  return false;
}
```

① Loop thru the array

② Another array lookup w/ indexOf method