Kerry Nordstrom 12/11/17

Restate. Find intersections of two arrays

Example/Pseudo: Declare empty array variable

```
Create for loop to iterate through one array
Check index of second array to be greater than zero
If so, push to empty array
```

A: [1,2,3,4]
B: [0,1,3,7] > [1,3]

A: " "
B: [ ] > Null

Edge

```
if(!A.isArray || !B.isArray)
return new TypeError
```

Code:

```
let A = [1,2,3,4];
let B= [0,3,5,6];
let intersection = [];

Function findIntersection = (A,B) => {
    for (let i in B){
        if((B[i].indexOf(A)) > 0){
            intersection.push(B[i]);
        };
    };
};
findIntersection(A, B);
```

O(N)

intersection = [3];

Approach: Hashmap

Cameron

```
const findIntersections = (arr1, arr2) => {
    if (!Array.isArray(arr1) || !Array.isArray(arr2)) {
        return null;
    }
    if (arr1.length === 0 || arr2.length === 0) {
        return null;
    }
    const Dictionary = new Set();
    for (let i = 0; i < arr1.length; i++) {
        Dictionary.add(arr1[i]);
    }
    const intersections = [];
    for (let i=0; i < arr2length; i++) {
        if (Dictionary.has(arr2[i])) {
            intersections.push(arr2[i]);
        }
    }
    return intersections;
};
```

- - - - - - - - -

Big O:

Space: O(n) where n is equal to # of elements in arr1.

Time: O(n+m) where n is equal to # of elements in arr1 and m is equal to # of elements in arr2.

Approach: Curry w/ Toggleable Closure

Cameron

```
const once = someFunction => {
    let hasBeenCalled = false;

    return (...args) => {
        if(!hasBeenCalled) {
            hasBeenCalled = true;
            someFunction(...args);
        }
        return;
    }
};
```

Ex:

```
const logOnce = once(console.log);
logOnce('Hello', 'World!'); // Hello World!
logOnce( "    ,    "    ); // undefined
```

Big O:

Space: O(1) constant

Time : O(1) constant

⇒ Both depend on speed of someFunction ... but once is still itself O(1) ...?

Jeff Kusowski

input = a1 , a2
arrays of number

output = [numbers that are in both]

[1,2,3][3,4,5] = [3]
[4,4] [4,4] = [4,4]

loop through a1
a1[i].i

O(n²)

```
let intersect = (a1, a2) => {
    if (!a1.length || !a2.length) return null
    let answer = []
    for (i = 0; i < a1.length; i++){
        if (a1[i].indexof(a2) > -1)
            answer.push(a2.splice(a1[i].indexof(a2)))
    }
    return answer
```

---

once

```
let once = (callback) => {
    let alreadyRan = folse

    return () => {
        if (!alreadyRan){
            already Ran = true
            callback()
        }
    }
```

```
test let loaded = once(console.log('Hi'))
         loaded()    // Hi
         loaded()    //
```

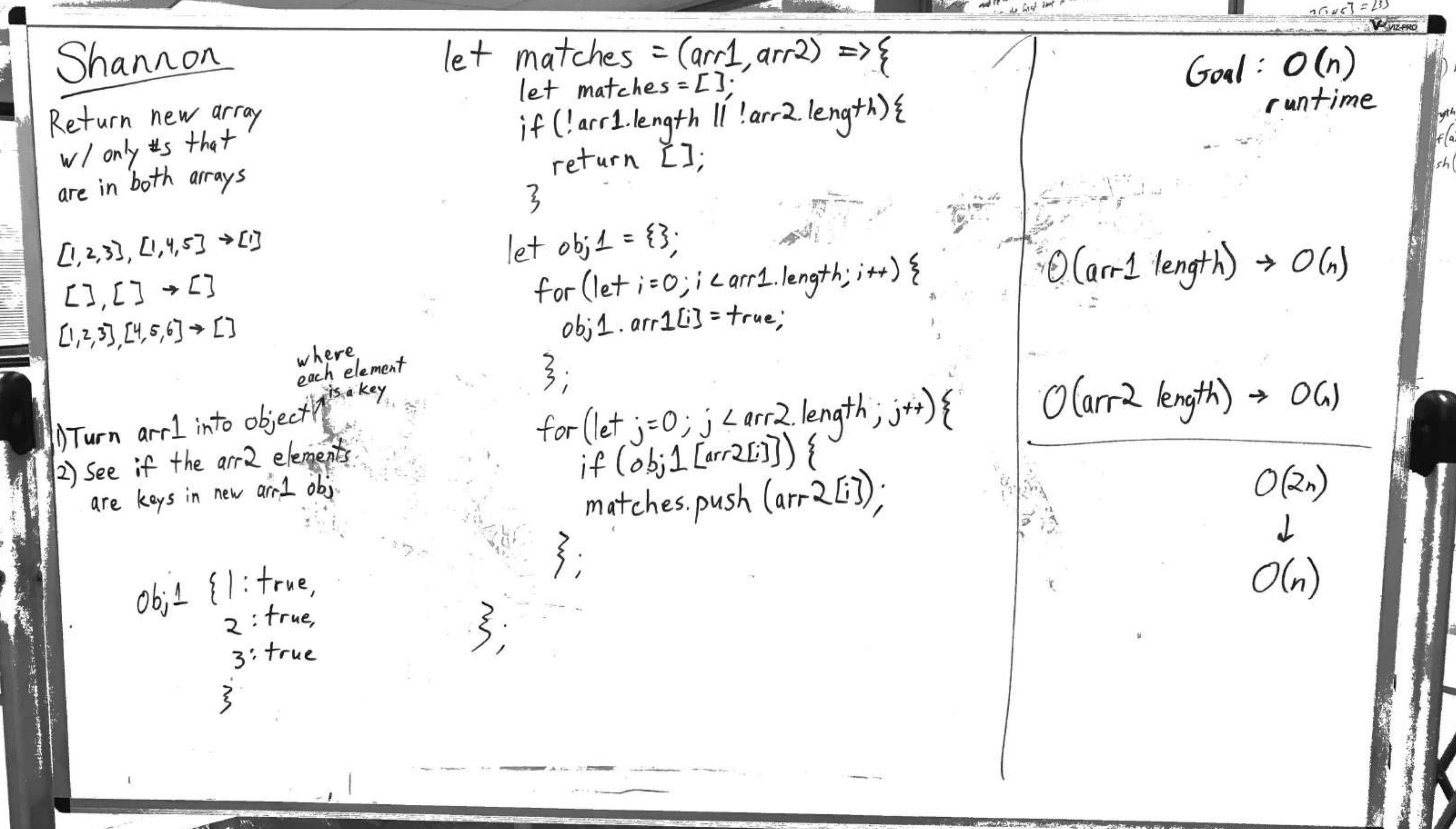
Kerr

Restate: I

Example/Pseu

let c

let ha



Shannon

Return new array w/ only #s that are in both arrays

[1,2,3], [1,4,5] → [1]
[], [] → []
[1,2,3], [4,5,6] → []

1) Turn arr1 into object where each element is a key
2) See if the arr2 elements are keys in new arr1 obj

```
obj1 {1: true,
      2: true,
      3: true
     }
```

```
let matches = (arr1, arr2) => {
  let matches = [];
  if (!arr1.length || !arr2.length) {
    return [];
  }

  let obj1 = {};
   for (let i = 0; i < arr1.length; i++) {
     obj1.arr1[i] = true;
   };

   for (let j = 0; j < arr2.length; j++) {
     if (obj1[arr2[i]]) {
     matches.push (arr2[i]);
   };

};
```

Goal: O(n) runtime

O(arr1 length) → O(n)

O(arr2 length) → O(n)

O(2n)
↓
O(n)

1) Problem: Write a function that intersects two arrays.
Assume you are given two valid arrays as argument.
Return the intersection as an array.

2) Example:

[1,3,7]
[1,3] > [1,

[2,4,7] > [] · [] = []
[]

Robert Reed
12/11/17
401d19

3) Pseudo:

```
0) Establish empty array called intersect
1) Establish empty object called inA
2) iterate through array A and add property to
   inA [illegible] A[i] and value = true.
3) iterate through second array (B)
    1) Check if B[i] is a property of inA
        1) If it is push B[i] into intersect
4) return intersect
```

4) code:

```
const intersection = (A,B) => {
  if(A.length===0 || B.length===0)
      return [];
  let intersect =[], inA = {};
  for(let element of A)
      inA[element] = true;
  for(let element of B) {
    if (inA[element])
        intersect.push(element);
  }
  return intersect;
};
```

Since there are no nested loops, but we iterate fully though both Arrays, so the Big O time complexity is $O(n)$, where n is the length of the sum of the array lengths.

Since we are making an array-object copy of Array A, our space complexity is $O(A)$, where A is the length of array A.

5) Test:

```
let A=[1,3,5,7], B=[3,5,8];
intersection(A,B);
↳ A.length ≠ 0, B.length ≠ 0, Continue
↳ intersect = [], inA = {}
↳ Loop A
    ↳ inA = {1: true, 3: true, 5: true, 7: true}
↳ loop B
    ↳ inA[3] → true → push 3 into intersect → [3]
    ↳ inA[5] → true → push 5 into intersect → [3,5]
    ↳ inA[8] → falsy → do nothing
↳ return [3,5]
```

Je

```
input = a1 , a2
        ^    ^
        arrays of

[1,2,3][3,4,5]
[4,4] [4,4]

let intersect = (a1,
      if(!a1.
      let an
      for (
          i

      }
      retur
```

once

```
let once = (callback)
      let alreadyRan

      return
         if (!alre
            alread
            callb
         }
      }

let  loaded =
     loaded()
     loaded()
```

Andrew

Write a Function that will intersect two arrays

e.g. intersect (arr1, arr2)

```
do input validation
    - if input not array,
    - if arrays have no length
    - members of array are #'s
create hash map →
    new array
    new object

for each item in array1,
create a new property on the obj
with a key of the value, + value: true

for each member of array2,
    if obj[value], push to new array.

return new array
```

```
const intersect = (arr1, arr2) => {
    // input validation here
  const intersection = [];
  const hashObj = {};
  arr1.forEach(value => heshObj.value = true);
  arr2.forEach(value => {
      if (heshObj[value]) {
          intersection.push(value)
      };
  };
  return intersection;
}
```

O(arr1.length)

O(arr2.length)

$O(n)$

in space + time

---

write a function which takes another function as input and returns the same function, but whic can only be called once

```
example: let onlyOnce = Once(console.log)
         onlyOnce('hello') // 'hello'
         onlyOnce('bye') // undefined
```

```
Define a new function
return a new function:
→ with a property unused to true
  if unused is true,
  set unused to false and run
   inner function

  when called again, function will
  not run because unused will be false
```

```
const once = callback => {
  let callback.unused = true;
    return (...args) => {
     if (unused) {
      callback.unused = false;
      callback(...args);
     };
   };
};
```

$O(1)$

Cath

PRoble

• WRite

that int

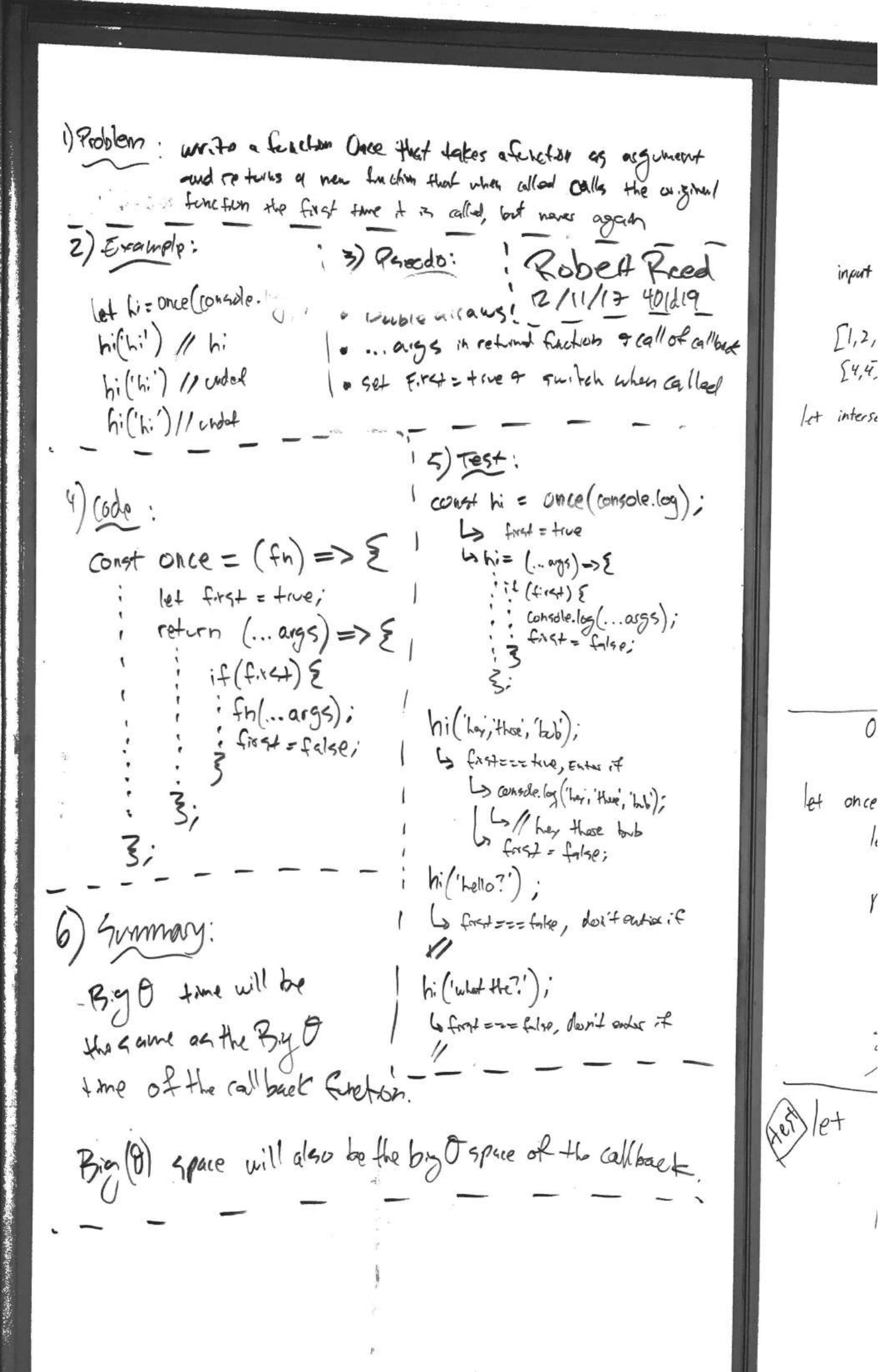
array

arr1 [0, 1,

arr2 [3, 4

Return newArr

```
for (let i
  for (let j
    if (ar
```



1) Problem: write a function Once that takes a function as argument and returns a new function that when called calls the original function the first time it is called, but never again

2) Example:

```
let hi = once(console.
hi('hi') // hi
hi('hi') // undef
hi('hi') // undef
```

3) Pseudo:

- Double arrows!
- ...args in returned function & call of callback
- set First = true & switch when called

Robert Reed
12/11/17 401d19

4) Code:

```
Const once = (fn) => {
    let first = true;
    return (...args) => {
        if(first) {
            fn(...args);
            first = false;
        }
    };
};
```

5) Test:

```
const hi = once(console.log);
  ↳ first = true
  ↳ hi = (...args) => {
      if (first) {
        console.log(...args);
        first = false;
      }
    };
hi('hey', 'there', 'bob');
  ↳ first === true, Enter if
    ↳ console.log('hey', 'there', 'bob');
      ↳ // hey there bob
    ↳ first = false;
hi('hello?');
  ↳ first === false, don't enter if
//
hi('what the?');
  ↳ first === false, don't enter if
//
```

6) Summary:

- Big O time will be the same as the Big O time of the callback function.

Big(O) space will also be the big O space of the callback.

input

[1,2,
[4,4,

let interse

O

let once

l

r

Hest let

Jacob Evans

arr1= [1,2,3,4,5,6,7,8]

arr2= [4,5]

$O(n^2)$ ↓ breaking out loops could change it to $O(n)$

```
Const intersectArr =(arr1,arr2) =>{

  return arr1.filter(ele => !arr1.includes(arr2))
                  ↓                        ↓
              iterating                iterating
};
```

array1[1,2,3]
array2[1,4,5]

Dalton #1

Pveko. Filter through different arrays and find the intersect point

```
let intersect = (array1, array2) =>{
    return array1. filter (a =>(array2.includes(a)):
```

#2

Write function that takes another function
and returns function once

```
Let once =  function => {
    let function. notlled = true
```

intersect two arrays

Matthew LeBlanc
12/11/07

[0, 2, 4 6, 9]
[1, 2, 3, 4]   O(n)
=> [2, 4]

---

loop

normally filter 1 with 2

edge
- not array
- duplicates
- String vs integer

```
    min / max
for (let i = 0;  i < arr 1.length; i++)
    if ( arr 2 [0 || length-1]
        break;
    if (min > max) break
    min ++
    max --
```

```
function intersect(arrOne, arrTwo) {
    let min = 0;    let max = arrTwo.length;
    let  bject  = {};  let newarray = []                      '5'
    for (let i = 0;  i < arrOne.length;  i++) {
            Object[arrOne[i].to strigg()] = true;              [0-4
        }
                                                         { '0' = true
    for (let i = 0; i < arrTwo.length;  i++) {                '2' = true
        if (Object[arrTwo[i].toString()])                    '4' = true
            newarray.push(arrTwo[i])                          '6' = true
                                                           '9' = true }


    }

    return newarray;

}
```

Write a function that will intersect two arrays ⇒ try to make an improvement over last exercise's version.

Assumptions:

Unique values in both input arrays. Both inputs are Arrays. Input arrays have 0+ entries.

Example:

[], [] ⇒ []   [1, 2, 3] ['foo'] ⇒ []   [1, 4, 'bar'], [2, 3, 'foo', 'bar'] = [bar].

Pseudocode:

```
→ Convert A to objects: {value, exists}.
→ Go through B → A[B.val].exists = true.
→ Return A. true.
```

Code:

```
const intersect = (a, b) => {
    for (val of b) {
        a[val].exists = true;
    }
    return a.filter (x => a.exists === true);
};
```

Questionable syntax; can add line before to arr.map() copy object into a if this doesn't work.

Theta:

O(Cn), where C is the per-iteration burden of the for...of code, and n is the size of the 'b' dataset.

Phelan

nnnn);

SETH DONOHUE

① Write a function that intersects two arrays

② [1,2,3] * [2,2,3] → [2,3]
[1,2] * [2,1] → [1,2]
[1,2,3] * [] → []
[5,5,5] * [5] → [5]

use object as a checker

③
```
const intersection = (arr 1, arr 2)=> {
    arr1. filter (element, index) => {
    return if (arr 2.includes [index])
      }
  };
```

$O(n^2)$

```
const intersection = (arr1, arr2) => {
    let found = { };
    for (let i in arr1){
        if(arr2[i]===arr1[i]){
          found.push(arr2[i]);
      }
```

```
for (let i in arr1)
    if (arr1[i]===arr2[i])
       found = ?
```

Fredric

Write a function that i ntersects two arrays O(n) time/s?

How many times Do we Loop through an array to find the matching elements.

$\underbrace{A([1,2,3])B([0,3,5])}_{\text{Read}}$ (compare) A+B for(found elements)

Let found = {

4.4
16
6 = 122

2

DAVID LINDAHL
WHITEBOARDING 12-11

PROBLEM

- WRITE FUNCTION THAT INTERSECTS 2 ARRAYS
- FIND #S FROM BOTH ARRAYS THAT ARE =

EXAMPLE:

[0,1,2]
[1,2,3]
⇒ 1,2

- SPEEDFAKECODE
  - loop over Array A & compare to Array B
  - if true, pull out into new Array

PSUEDOCODE

```
newA = [ ];
for  A length
        if B.includes A[i]
        newA.pop A[i]

for B.length
     if ( A.includes B[i])
   newA. pop B[i]

return newA
```

REAL CODE   Big O(n^2)

```
function Fun ( A, B ) => {
     newArray = [ ];
     for (i=0, i<  A.length, i++) {
          if (B.includes (A[i])) {
               newArray. pop (A[i]);
     };
     for (i=0, i < B.length, i++) {
          if (A. includes (B[i])) {
               newArray. pop (A[i]);

     };

     return newArray;

};
```

test
[0,1,2]

for

```
let x = obj = { };

    x[     ] = true;
```

for

```
found[a[i]]=true
```

```
{...args}
OBJECT = { };
```

↓

0:1,10
1:2,11
2:3,12

Shannon

Function that has 1 param. (a function)

-should return a func. that can be called mult. times but will only execute the param. func. once.

```
let oneTimeFunc = (func) => {                         Time
    let done = false;                                 O(1)
    let innerFunc = (fn) => {
        if (!done) {
            done = true;                              O(1)
            fn();                                     O(1)

        };

    };                                                O(1)
    return innerFunc(func);                           ----
                                                      O(1)
};
```

function that will intersect 2 arrays

```
let interArray = (arrayOne, arrayTwo) => {
        current = {};
        for(let i in arrayOne){

            current.x = arrayOne[i];
            if(current[x] in arrayTwo)
              let item = current[x];
            newArray.push(item);


        return newArray;
```

Pedja 401d19

EDGE CASES:
- check if arrays are arrays
- check if arrays are empty
- doubles in array

```
loop thru first array
    loop thru second one if includes
        if so push to new intersect
                                ARRAY.
let a = [1,2,3]; b = [2,4,5]
 interArray(a, b)
   for (let i=0; i<a.length; i++){
        current = a[i]
        for (b.includes(current)
              push to new Array
   return newArray
```

OLD WAY ↑ $O(n^2)$

Using obj. ↓ $O(n)$

```
let x = obj = {}
 x[ ] = true
```

assing current value of arrayOne to object property.

if value of obj. property is in ArrayTwo -> push to newArray

(f

f(x

Catherine Looper

Problem Domain

- Write a function that intersects two arrays

```
arr1 [ 0, 1, 2, 3, 4 ]
arr2 [ 3, 4, 5, 6, 7 ]
Return newArray [ 3, 4 ]
```

```
for (let i = 0; i < arr1.length; i++) {
  for (let j = 0; j < arr2.length; j++) {
    if (arr1
```

Code

```
let arr 1 = [0, 1, 2, 3, 4]
let arr 2 = [3, 4, 5, 6]

const joinArrays = (arr1, arr2) {
  let intersect = { };
  if (arr1.includes(arr 2.value)) {
    intersect.value = arr2.value;

  }
  return intersect.value;
}
```

O(n)

```
const joinArrays = (arr1, arr2) {
  let intersect = { };
  for (let i = 0; i < arr1.length; i++) {
    intersect [value] = true;
    if (intersect [arr2.value]) {
      intersect. arr2.value;
  }
  for (let i = 0; i < arr2.length; i++) {
    if (intersect [value) {

  }
  return intersect;
}

}
```

true));

rr1.length)

rr2.length)

input

#2