**ALGORITHM:**

1. **Hash table**

\_Putting in an item into a table using keys and hash algorithm

\_Have constant O(1) lookup time b/c it uses keys

\_No concept of order

\_Collision

* are **unordered** (the keys are not guaranteed to stay in the same order)
* can use **many types of objects as keys** (commonly strings)

Q: Hash vs Array?

A: Arras are fixed in size. Hash tables have no size limit

\_Insertion in hash is O(1). Array it’s O(n)

\_

**II- Reduce: Reduce all the numbers to 1 number (like average)”**

\_Arr.reduce(callback{currentValue, initialValue})

Ex:

var arr = [0,1,2,3]

var x = arr.reduce(function(initial,total){

return initial + total}, 3);

* Reduce takes in an initial value, a total (something that adds together), and then return the two, as well as an additional number to plus together

\_For loop: Separate by ; not ,

## \_The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

do {  
*code block to be executed*}  
while (*condition*);

\_What Objects lok like:

var car = {type:"Fiat", model:"500", color:"white"};

Q: To replace an item in an array, use splice:

A: Ex:

var arr =[1,2,3]

arr.splice(0,1,2)

// arr = [2,2,3]

//splice(start,length, what to replace with)

\_FOR/IN LOOP:

var text = "";  
var x;  
for (x in person) {  
    text += person[x];  
}

CANNOT use “return” in LOOPs

\_FOR/OF loop:

**const** iterable = ['a', 'b'];

**for** (**const** x **of** iterable) {

console.log(x);

}

//Objects not iterable, so can’t use For..Ops

Q: SAMPLE TEST:

/\*

\* anagramPalindrome

\*

\* Write a function which accepts an input word and returns true or false if there exists

\* some anagram (permutation) of that input word that is a palindrome.

\*

\* "cat" => "tac", "tca"

\*/

A:

//HIGH LEVEL:

BASICALLY PUT EACH LETTER OF THE WORD INTO A HASH TABLE.

HASH TABLE: ALPHABETICAL ORDER: VALUE

SO HASH TABLE IS BASICALLY AN {}

SO YOU STORE HOW MANY TIMES EACH LETTER APPEARS AS A KEY

//SO SINCE A PALINDROME IS EITHER IN THIS FORM: ABA or AA. But remember this is a permutation so it’s not in that order

//SO STRATEGY IS, IF THERE’S ONLY ONE LETTER, AND EVERYTHING ELSE’S VALUE IS DIVIDED BY 2, THEN THAT’S A PALINDROM! (ABA case)

//ALSO, OF COURSE, IF EVERTHING’S VALUE IS DIVIDED BY 2 (AA case) then it’s a palindrom

// else, nope

var anagramPalindrome = function(word){

let frequencyOfLetters={};

for(var i =0; i<word.length; i++){

let count = 0;

//So that you can reset count to 0 every time

//and why you need to? Because you want to increment the value count, not the count itself by more than 1

if(frequencyOfLetters[word[i]]){

frequencyOfLetters[word[i]] ++;

}

else{

count =1;

frequencyOfLetters[word[i]]= count

}

}

let check = true;

let frequencyCount = 0;

//outside so that value of count is tallied up each time and not set to 0

for(let letter in frequencyOfLetters){

//if values of all keys is divided by 2, return true

//+) if only one key is false, also return true

//if values of all but one key is divided by 2, return true

//otherwise, false

if ((frequencyOfLetters[letter] %2) ===1 ){

frequencyCount ++;

}

if ((frequencyOfLetters[letter] %2) ===0 || frequencyCount ===1) {

check;

}

else{

check= false;

}

}

return check;

};

console.log(anagramPalindrome("carrace")); // true

console.log(anagramPalindrome("cutoo")); // false

console.log(anagramPalindrome("an")); // false

console.log(anagramPalindrome("dddaaaayyyy")); // true

console.log(anagramPalindrome("anna")); //true

console.log(anagramPalindrome("bbd")); //true