Algorithms

Table of contents

Arrays

========== Arrays ==========

Recommendation

Отбить основной цикл и внутренний разделителями

```
function uniteUnique(arr) {
  arrOfArg.map((arr, id) => {
    console.log("++++++++++++++++++++++++++");
    console.log("------arr---------", arr);

    arr.map((el) => {
      console.log("======el=====", el);
    });
  });
}
uniteUnique([1, 3, 2, 2, 2], [5, 2, 1, 4]);
```

1. Smallest Common Multiple -> Сложность 5/5

Условие: Найти наименьший общий множитель между двумя или более числами

Info: Найти наименьший множитель между двумя числами на которые оба числа будут делиться одинаково. 1*3 = 3; 2*3 = 6 3*3 = 9....

1. 3 are 3, 6, 9, 12, 15, 18,

2. 4 are 4, 8, 12, 16, 20, ...
3. первое наименьшее число 12

Алгоритм:

1. определить минимальное число и максимальное входных данных
2. найти делитель двух чисел
3. найти верхнюю границу двух чисел (через цикл)
4.

```
function smallestCommons(arr) {
  // Setup
  const [min, max] = arr.sort((a, b) => a - b);
  const numberDivisors = max - min + 1;
  // Largest possible value for SCM
  let upperBound = 1;
  for (let i = min; i <= max; i++) {
    upperBound *= i;
  }
  // Test all multiples of 'max'
  for (let multiple = max; multiple <= upperBound; multiple += max) {
    // Check if every value in range divides 'multiple'
    let divisorCount = 0;
    for (let i = min; i <= max; i++) {
      // Count divisors
      if (multiple % i === 0) {
        divisorCount += 1;
      }
    }
    if (divisorCount === numberDivisors) {
      return multiple;
    }
  }
}

smallestCommons([1, 5]);
```

2. Drop it -> Сложность 2/5

Условие: По заданному условию отфильтровывать массив

Алгоритм:

1. перебрать массив
2. установить флаг в true если условие совпадает
3. возвращать все остальные данные с массива в новый массив

```
function dropElements(arr, func) {
  let flag = false;
```

```
  const newArr = arr.reduce((acc, el, id) => {
    if (func(el) && flag === false) {
      flag = true;
    }

    if (flag) {
      acc.push(el);
    }

    return acc;
  }, []);

  return newArr;
}

function dropElements(arr, func) {
  while (
    arr.length > 0
        &&
    !func(arr[0]) // 1 -> 2 -> 3
) {
    arr.shift(); // [ 2, 3, 7, 4 ] -> [ 3, 7, 4 ] -> [ 7, 4 ]
  }
  return arr;
}

dropElements([1, 2, 3, 4], function(n) {return n >= 3;});
```

3. Steamroller -> Сложность 3/5

Условие: Исключить вложенные массивы не применяя Array.prototype.flat() Array.prototype.flatMap(). И не использую глобальные переменную *Пример:* [[["a"]], [["b"]]] ==> ["a", "b"]

Info:

Алгоритм:

1. перебрать массив
2. проверить вложенные элемент массива на массив
3. применить рекурсию
4. если нет вложенного массив вернуться на следующий элемент родителя

```
    function steamrollArray(arr){
        let result = [];

        arr.forEach(item =>{
            if(Array.isArray(item)){
                result = result.concat(SteamrollArray(item));
            } else {
                result.push(item);
            }
```

```
        })

        return result;
    }

    function steamrollArray(arr){
        const flat = [].concat(...arr);
        return flat.some(Array.isArray)? SteamrollArray(flat) : flat;
    }
```

Strings

1. Базовый Алгоритм работы со словами в стоке "aasdsda adasd"
    1. разбить строку на массив через str.split(" ")
    2. сделать работу с каждой буквой for(), forEach()
    3. собрать назад все назад в строку arr.join("")

1. Compare strings -> Сложность 3/5

Условие: find the missing letter from a string and return it

```
// fiend uniq literals form alphabet
function fearNotLetter(str) {
  let currCharCode = str.charCodeAt(0);
  let missing = undefined;

  str.split("").forEach((letter) => {
    if (letter.charCodeAt(0) === currCharCode) {
      currCharCode++;
    } else {
      missing = String.fromCharCode(currCharCode);
    }
  });

  return missing;
}
```

2. Convert the characters -> Сложность 2/5

Условие: Convert the characters &, <, "", '' to name code

Main Алгоритм:

1. создать хранилище ключ значение
2. найти символ в строке
3. создать новую строку с новым значением

Алгоритм 1.:

1. разбить строку на массив элементов str.split("")

2. создать массив с characters &, <, "", ''
3. перебрать массив
4. найти элемент и заменить на нужный
5. собрать массив в кучу

Алгоритм 2.: Решение два: 1. создать объект ключ characters значение на что надо заменить 2. создать регулярное выражение и заменить по ключу

```
function convertHTML(str) {
  const characterEntries = {
    "&": "&amp;",
    "<": "&lt;",
    ">": "&gt;",
    '"': "&quot;",
    "'": "&apos;",
  };
  return str.replace(/([&<>\"'])/g, (match) => characterEntries[match]);
}
```

3. Binary Agents -> Сложность 1/5

Условие: На основе бинарной строки вернуть английскую строку "01000001 01110010 01100101..." -> "Aren't bonfires..."

Info: "1101".parseInt("1101", 2) -> 13 второй аргумент для определения основания счисления fromCharCode(13) -> "A" **Алгоритм:**

1. разбить на массив строк
2. создать массив аккумулятор
3. проходится по элементу массива
4. ковертировать двоичный коды в число parseInt("1101", 2) -> 13
5. конвертировать unicode в символ fromCharCode(code)
6. пушить все в аккумулятор
7. собрать аккумулятор в строку arr.join("")

```
    function binaryAgent(str){
        const biString = str.split(" "); // 1.
        let uniString = []; //2.

        for(let i = 0; i< biString.length; i++){
            let code,
                convertChar

            unicode = parseInt(biString[i],2)
            convertChar = String.fromCharCode(code)
            uniString.push(convertChar);
        }

        return uniString.join("");
```

```
    }
```

Numbers

1. Sum All Odd Fibonacci Numbers -> Сложность 4/5

Условие: нужно посчитать сумму фибоначчи чисел 0,1,1,2,3,5,8,13,21,34,55

Алгоритм:

1. получить предыдущее значение;
2. получить текущее значение;
3. (0) + (1) = (1) => 1 + 1 = (2) => 1 + 2 = (3)
4. сложить предыдущее с текущее -> (2)+(3)=5

```
function fibonacciShort(num) {
  let a = 1,
    b = 1;

  for (let i = 3; i <= num; i++) {
    [a, b] = [b, a + b];
  }
  return b;
}

function sumFibs(num) {
  let prev = 0;
  let current = 1;
  let fibonacci = 0;

  while (current <= num) {
    if (current % 2 !== 0) {
      fibonacci += current;
    }

    current += prev;
    prev = current - prev;
  }

  return fibonacci;
}

function fibonacci(num) {
  const result = [0, 1];
  for (let i = 2; i < num; i++) {
    const prevNum1 = result[i - 1];
    const prevNum2 = result[i - 2];
    result.push(prevNum1 + prevNum2);
```

```
    }

    return result[num];
  }
```

2. Sum all Primes -> Сложность 5/5

Условие: Нужно возвращать сумму всех простых чисел который меньше либо равны входному числу

Info:

1. простое число это натуральное число, единственным делителями которого являются только оно само и единица
    1. 2 это простое число которое делиться на 1 и 2
    2. 4 это не простое число которое делиться на 1,2,4
    3. натуральное число 1 не являются ни простым ни составным
2. разложением на простые множители - это если в натуральном числе все множители простые числа

Теорема: *Каждое натуральное число, отличное от 1, может быть разложено на простые множители, и притом единственным образом (если отожествлять разложения* pq и qp*, где p и q простые числа)*

Решето Эратосфена применимо к решению этой задачи.

Алгоритм:

1. разложить число начиная с 2 ...
2. исключение составных чисел (которые имею делители отличные от 1 и самого числа)
3. суммируем все оставшиеся просты числа

```
function sumPrimes(num) {
  let primes = [];

  for (let i = 2; i <= num; i++) {
    if (primes.every((prime) => i % prime !== 0)) {
      primes.push(i);
    }
  }

  return primes.reduce((sum, prime) => sum + prime, 0);
}
```