Poblem statement

Please write code to change **`seconds_batches`** to be **`sorted_uniq_seconds`** in simplest way in Javascript

```
const seconds_batches = [
  [1, 2, 3, 4],
  [1, 2, 3, 4, 5, 6, 7],
  [30, 31, 32, 33],
  [1, 2, 3, 32, 33, 34, 35, 36, 37, 38, 39, 40],
]


const sorted_uniq_seconds = [1, 2, 3, 4, 5, 6, 7, 30, 31, 32, 33, 34,
35, 36, 37, 38, 39, 40]
```

Solution

Using **`Set`** as main data structure and flatten array using **`O(1)`** as space complexity.

Within optimal the flatten method to be **O(1)** as space complexity using several approach.

Here's the details of the implementation **on next page**:

```
/**
 * Original Creator: Abdul Salam
 * Github Account: @abdulsalam01
 */

/**
 * Problem statements
 * Make the seconds_batches to be sorted_uniq_seconds in simplest way
 * With given input: `seconds_batches` and output: `sorted_uniq_seconds`
 */

// Input:
const seconds_batches = [
  [1, 2, 3, 4],
  [1, 2, 3, 4, 5, 6, 7],
  [30, 31, 32, 33],
  [1, 2, 3, 32, 33, 34, 35, 36, 37, 38, 39, 40],
]
// Output:
const sorted_uniq_seconds = [1, 2, 3, 4, 5, 6, 7, 30, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40]

/**
 * Here's the solution sections
 */

// There's many approach to make the input to be output one on the simplest way.
// But the things that we need to keep in mind is the process complexity of
remove duplication and append to array.
// It should be an important things about time complexity and space complexity to
make sure:
// - if the array of input is growth
// - even growth exponentially
// - or there's another issue if the input has an array of array.
// So, let's see which one is the best and optimization solution for this case.

// Let's try :)
// Basically, we can merge the array into one using `flat` method
// And then, use data structure that named `Set`, `Set` is special data structure
that keep only unique data inside it.
// The implementation should be like this:
const flatten = seconds_batches.flat()
const sets = new Set(flatten)

// On oneline version should be:
```

```
const uniqueSort = new Set(seconds_batches.flat())

// It's done? Not yet
// But how about time complexity of `flat` method
// Refer to this docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/flat
// `Flat` using copying method and then looping for every array inside input, make new variable and append to that new variables.
// So:
// `Flat` take O(n) for time complexity and since it's using `shallow copy` the space complexity should be O(n)
// And `Set` it's quite great data structure to remove duplicate date with O(1), just great!

// So, we need improve our `flat` method with own, to make the AT LEAST time complexity O(n) and space complexity O(1)
// I will use `reduce` function from js and concact it together each iteration.
// I just need to modify itself of input and store to new variables.
// Let's try.
const flattenImprArray = seconds_batches.reduce((acc, curr) => acc.concat(curr), [])
const uniqueSortImprArr = new Set(flattenImprArray)

// So, yeah I guess we're good to go with this technique, it will cost
// O(n) as total of time complexity
// O(1) as total of space complexity (Don't count the new variable of flattenImprArray as new shallow copy :) )

/**
 * Print section
 * Comparison with real data
 */
console.log(sets) // Method 1 (Flat and Set): [1, 2, 3, 4, 5, 6, 7, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]
console.log(uniqueSort) // Method 2 (Oneline of Method 1): [1, 2, 3, 4, 5, 6, 7, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]
console.log(uniqueSortImprArr) // Method 3 (Improvement of Method 1 & 2): [1, 2, 3, 4, 5, 6, 7, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]

// Compare the results with output given:
console.log(isEquals(sets, sorted_uniq_seconds)) // True
console.log(isEquals(uniqueSort, sorted_uniq_seconds)) // True
console.log(isEquals(uniqueSortImprArr, sorted_uniq_seconds)) // True
```

```
/**
 * Helper function for check the equality of 2 given data.
 * @param {Should be `Set`} sets
 * @param {Should be an `Array`} arr
 * @returns true/false
 */
function isEquals(sets, arr) {
  const arrFromSet = Array.from(sets)
  const arrFromOutput = arr

  return JSON.stringify(arrFromSet) === JSON.stringify(arrFromOutput)
}
```