Gap Analyzer – Hackathon Submission👩‍💻

Created by:
Sumaya Haider Raheam Media & Communications Technology Engineering Student
Front-end Developer | Creative Thinker | Hackathon Enthusiast

Function code:

```
/**
 * Analyzes gaps in a sorted array of
numbers.
 * Returns all missing ranges, the longest
gap, and total missing count.
 * Created by Sumiiiiii 💫
 */
function sumi_analyzeGaps(numbers) {
  if (numbers.length === 0) {
    return { gaps: [], longestGap: null,
missingCount: 0 };
  }

  const unique = [...new Set(numbers)];
  let min = unique[0];
  let max = unique[0];
  for (let num of unique) {
```

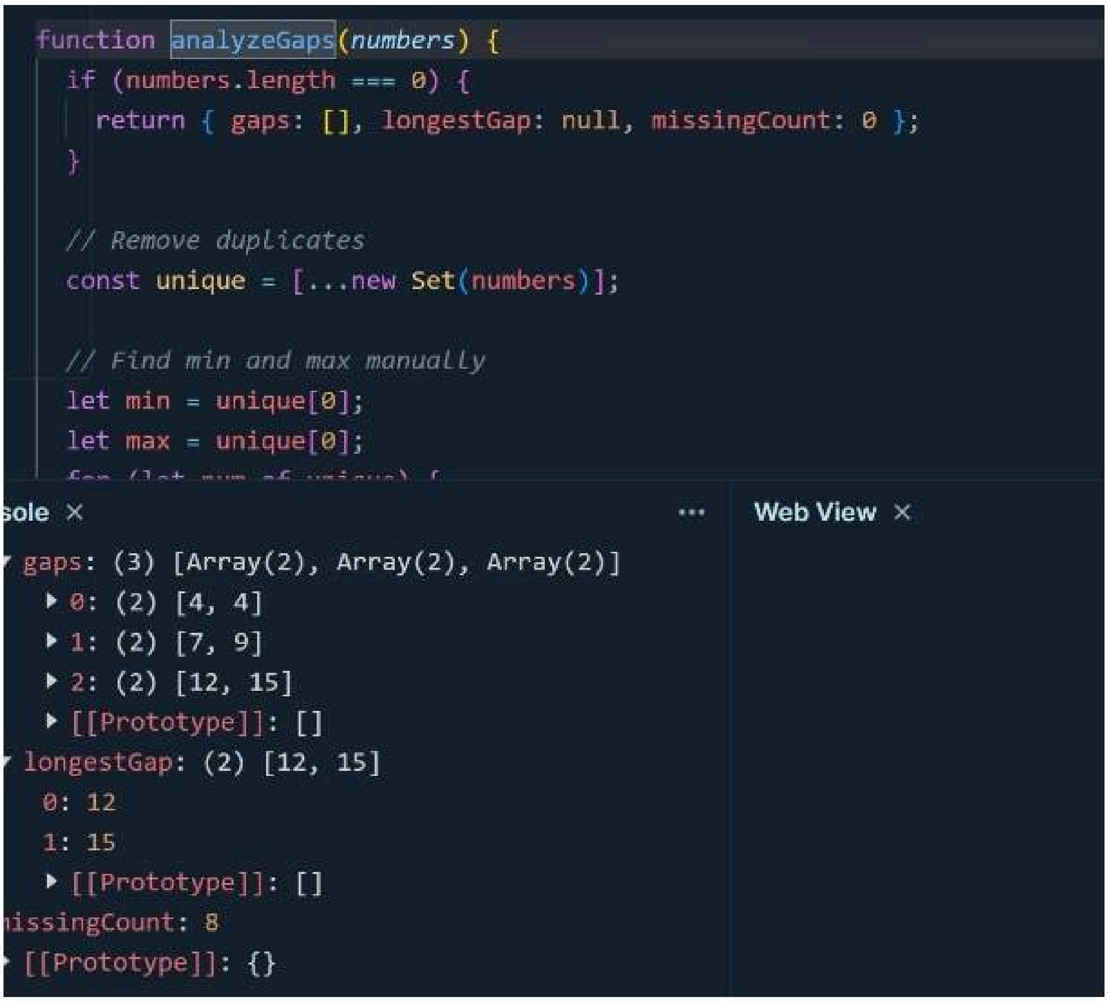
```
  if (num < min) min = num;
  if (num > max) max = num;
}

const numSet = new Set(unique);
let gaps = [];
let longestGap = null;
let maxLength = 0;
let missingCount = 0;
let start = null;

for (let i = min; i <= max; i++) {
  if (!numSet.has(i)) {
    if (start === null) start = i;
  } else {
    if (start !== null) {
      const end = i - 1;
      const length = end - start + 1;
      gaps.push([start, end]);
      missingCount += length;
      if (length > maxLength) {
        maxLength = length;
        longestGap = [start, end];
      }
      start = null;
    }
  }
}
```

```
  if (start !== null) {
    const end = max;
    const length = end - start + 1;
    gaps.push([start, end]);
    missingCount += length;
    if (length > maxLength) {
      maxLength = length;
      longestGap = [start, end];
    }
  }

  return { gaps, longestGap,
missingCount };
}

console.log(sumi_analyzeGaps([10, 2, 5, 1,
3, 11, 6, 16]));
```

OUTPUT:

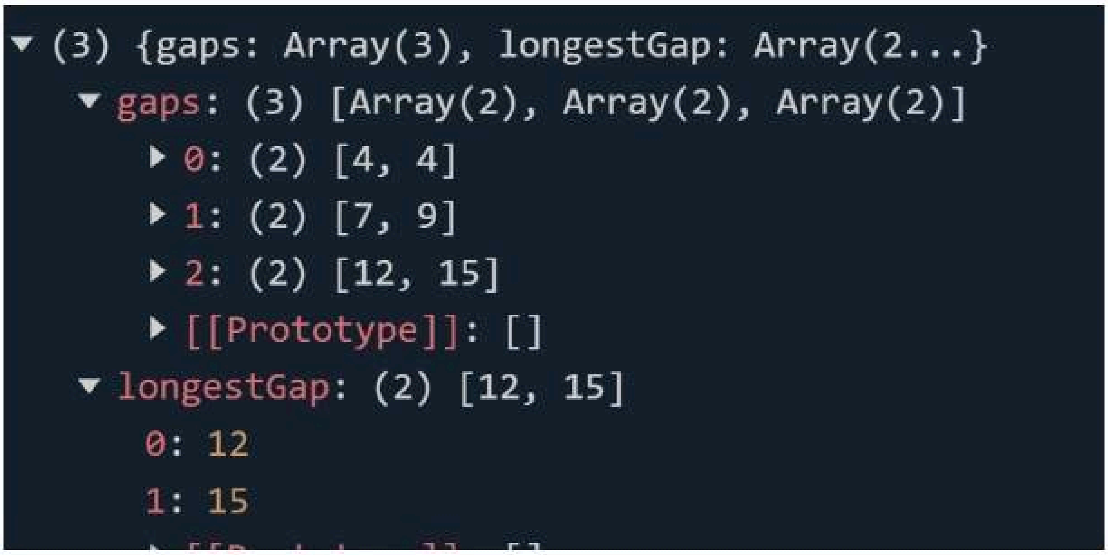
```
{
  gaps: [ [4, 4], [7, 9], [12, 15] ],
  longestGap: [12, 15],
  missingCount: 8
}
```

Function Output Verification:
The screenshot below shows the console output of the analyzeGaps() function executed on PlayCode.io. It confirms the correct behavior of the function, including:
This output validates the logic and structure of the function for submission.



```
function analyzeGaps(numbers) {
  if (numbers.length === 0) {
    return { gaps: [], longestGap: null, missingCount: 0 };
  }

  // Remove duplicates
  const unique = [...new Set(numbers)];

  // Find min and max manually
  let min = unique[0];
  let max = unique[0];
```

```
▾ gaps: (3) [Array(2), Array(2), Array(2)]
   ▸ 0: (2) [4, 4]
   ▸ 1: (2) [7, 9]
   ▸ 2: (2) [12, 15]
   ▸ [[Prototype]]: []
▾ longestGap: (2) [12, 15]
    0: 12
    1: 15
   ▸ [[Prototype]]: []
 issingCount: 8
▸ [[Prototype]]: {}
```



```
▾ (3) {gaps: Array(3), longestGap: Array(2...}
   ▾ gaps: (3) [Array(2), Array(2), Array(2)]
      ▸ 0: (2) [4, 4]
      ▸ 1: (2) [7, 9]
      ▸ 2: (2) [12, 15]
      ▸ [[Prototype]]: []
   ▾ longestGap: (2) [12, 15]
      0: 12
      1: 15
```