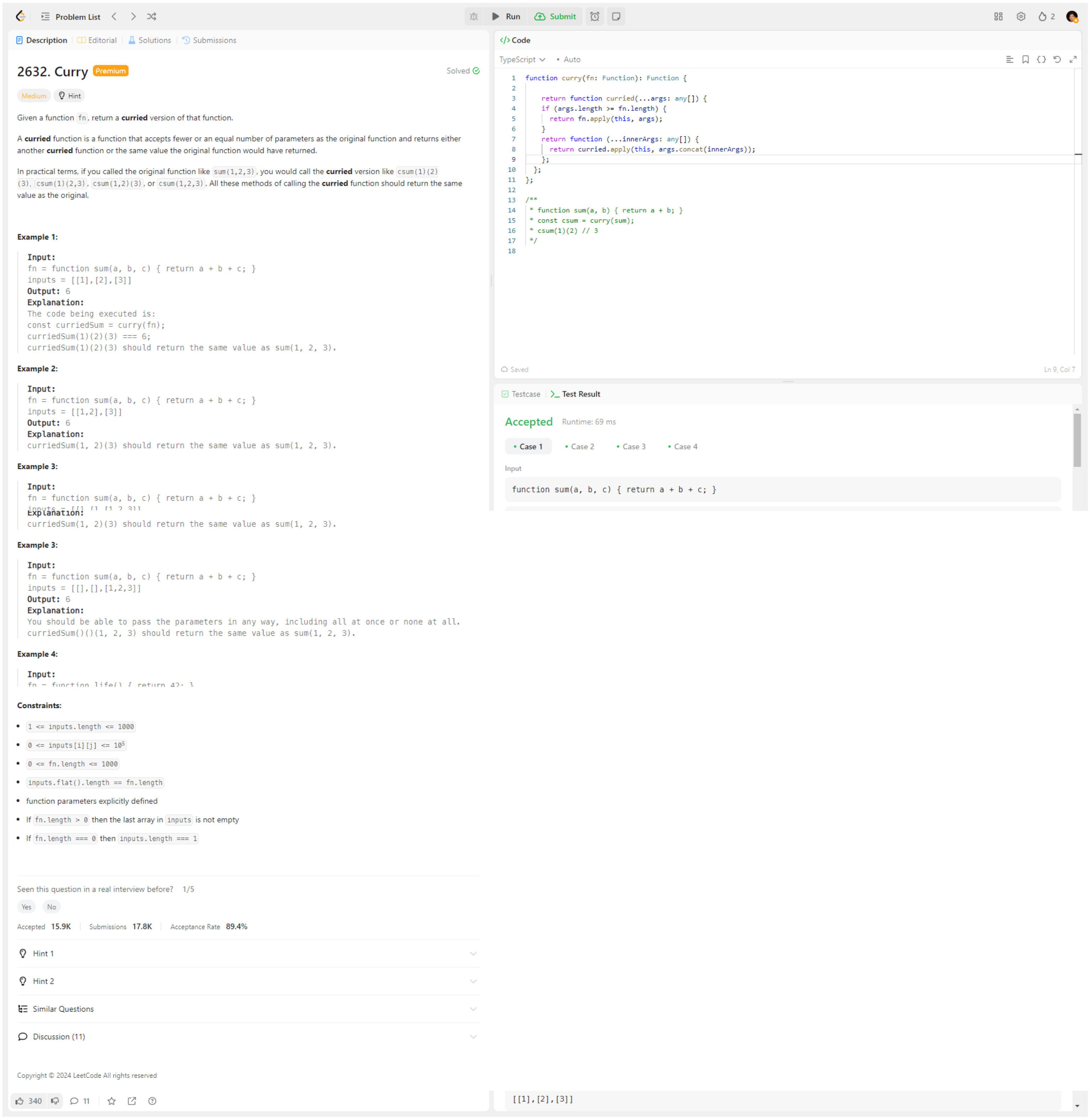

2632. Curry

Given a function `fn`, return a **curried** version of that function.

A **curried** function is a function that accepts fewer or an equal number of parameters as the original function and returns either another **curried** function or the same value the original function would have returned.

In practical terms, if you called the original function like `sum(1,2,3)`, you would call the **curried** version like `csum(1)(2)(3)`, `csum(1)(2,3)`, `csum(1,2)(3)`, or `csum(1,2,3)`. All these methods of calling the **curried** function should return the same value as the original.

Example 1:

```
Input:
fn = function sum(a, b, c) { return a + b + c; }
inputs = [[1],[2],[3]]
Output: 6
Explanation:
The code being executed is:
const curriedSum = curry(fn);
curriedSum(1)(2)(3) === 6;
curriedSum(1)(2)(3) should return the same value as sum(1, 2, 3).
```

Example 2:

```
Input:
fn = function sum(a, b, c) { return a + b + c; }
inputs = [[1,2],[3]]
Output: 6
Explanation:
curriedSum(1, 2)(3) should return the same value as sum(1, 2, 3).
```

Example 3:

```
Input:
fn = function sum(a, b, c) { return a + b + c; }
inputs = [[],[],[1,2,3]]
Explanation:
curriedSum(1, 2)(3) should return the same value as sum(1, 2, 3).
```

Example 3:

```
Input:
fn = function sum(a, b, c) { return a + b + c; }
inputs = [[],[],[1,2,3]]
Output: 6
Explanation:
You should be able to pass the parameters in any way, including all at once or none at all.
curriedSum()()(1, 2, 3) should return the same value as sum(1, 2, 3).
```

Example 4:

```
Input:
fn = function life() { return 42; }
```

Constraints:

- `1 <= inputs.length <= 1000`
- `0 <= inputs[i][j] <= 10^5`
- `0 <= fn.length <= 1000`
- `inputs.flat().length == fn.length`
- function parameters explicitly defined
- If `fn.length > 0` then the last array in `inputs` is not empty
- If `fn.length === 0` then `inputs.length === 1`

Seen this question in a real interview before? 1/5

Yes No

Accepted 15.9K | Submissions 17.8K | Acceptance Rate 89.4%

Hint 1

Hint 2

Similar Questions

Discussion (11)

```typescript
function curry(fn: Function): Function {

    return function curried(...args: any[]) {
    if (args.length >= fn.length) {
      return fn.apply(this, args);
    }
    return function (...innerArgs: any[]) {
      return curried.apply(this, args.concat(innerArgs));
    };
  };
};

/**
 * function sum(a, b) { return a + b; }
 * const csum = curry(sum);
 * csum(1)(2) // 3
 */
```

Accepted Runtime: 69 ms

Input

```
function sum(a, b, c) { return a + b + c; }
```

```
[[1],[2],[3]]
```