2637. Promise Time Limit

Description

Given an asynchronous function `fn` and a time `t` in milliseconds, return a new **time limited** version of the input function. `fn` takes arguments provided to the **time limited** function.

The **time limited** function should follow these rules:

- If the `fn` completes within the time limit of `t` milliseconds, the **time limited** function should resolve with the result.
- If the execution of the `fn` exceeds the time limit, the **time limited** function should reject with the string `"Time Limit Exceeded"`.

Example 1:

```
Input:
fn = async (n) => {
  await new Promise(res => setTimeout(res, 100));
  return n * n;
}
inputs = [5]
t = 50
Output: {"rejected":"Time Limit Exceeded","time":50}
Explanation:
const limited = timeLimit(fn, t)
const start = performance.now()
let result;
try {
   const res = await limited(...inputs)
   result = {"resolved": res, "time": Math.floor(performance.now() - start)};
} catch (err) {
   result = {"rejected": err, "time": Math.floor(performance.now() - start)};
}
console.log(result) // Output

The provided function is set to resolve after 100ms. However, the time limit is set to 50ms. It rejects at t=50ms because the time limit was
reached.
```

Example 2:

```
Input:
fn = async (n) => {
  await new Promise(res => setTimeout(res, 100));
  return n * n;
}
inputs = [5]
t = 150
Output: {"resolved":25,"time":100}
Explanation:
The function resolved 5 * 5 = 25 at t=100ms. The time limit is never reached.
```

Example 3:

```
Input:
fn = async (a, b) => {
  await new Promise(res => setTimeout(res, 120));
  return a + b;
}
inputs = [5,10]
t = 150
Output: {"resolved":15,"time":120}
Explanation:
The function resolved 5 + 10 = 15 at t=120ms. The time limit is never reached.
```

Example 4:

```
Input:
fn = async () => {
  throw "Error";
}
inputs = []
t = 1000
Output: {"rejected":"Error","time":0}
Explanation:
The function immediately throws an error.
```

Constraints:

- `0 <= inputs.length <= 10`
- `0 <= t <= 1000`
- `fn` returns a promise