Análise de Eficácia de Testes com Teste de Mutação

Disciplina: Teste de Software
Trabalho: Análise de Eficácia de Testes com Teste de Mutação
Nome: Matheus Dias Mendes
Matrícula:
Data: 02/11/2025

Análise Inicial

Cobertura de Código Inicial: 98.64%
Pontuação de Mutação Inicial (Mutation Score): 73,71%

Quantidade Mutantes Morto: 154
Quantidade Mutante Sobreviventes: 44
Sem cobertura: 12
Time out: 3

Discussão: Pelo que deu para ver o quase todas as linhas do código foram verificadas com os testes, mas mesmo assim o número de mutantes mortos não foi tão alto, isso mostra que nem todos os testes verificaram corretamente o comportamento esperado, mostrando que falta mais profundidade aos testes

Análise de Mutantes Críticos

Mutante 1: if (n === 0 || n === 1) return 1;

- **Mutação:** A condição ||
- **Por que sobreviveu:** O mutante que substituiu || por && não foi morto porque os casos de teste para n=0 e n=1 ainda passam, já que a função trata cada valor separadamente, sendo esse um mutante equivalente

Mutante 2: return numeros.reduce((acc, val) => acc * val, 1);

- **Mutação:** acc * val
- **Por que sobreviveu:** Por conta que a mutação ela trocou a ordem de acc * val para val * acc, e como na multiplicação a ordem dos fatores não altera o resultado, o teste não consegue tratar isso sendo esse um mutante equivalente.

Mutante 3:

- **Mutação:**if (valor < min) return min; if (valor > max) return max***;***
- **Por que sobreviveu:** O mutante é equivalente porque alterar < para <= ou > para >= não muda o resultado para nenhum dos casos de teste existentes

```
Ran 2.68 tests per mutant on average.
-------------|------------------|----------|----------|-----------|------------|----------|
             | % Mutation score |          |          |           |            |          |
File         |  total | covered | # killed | # timeout | # survived | # no cov | # errors |
-------------|--------|---------|----------|-----------|------------|----------|----------|
All files    |  98.15 |   98.15 |      208 |         4 |          4 |        0 |        0 |
 operacoes.js |  98.15 |   98.15 |      208 |         4 |          4 |        0 |        0 |
-------------|--------|---------|----------|-----------|------------|----------|----------|
```

Soluções Implementadas

Alteração Casos de Teste Criados:

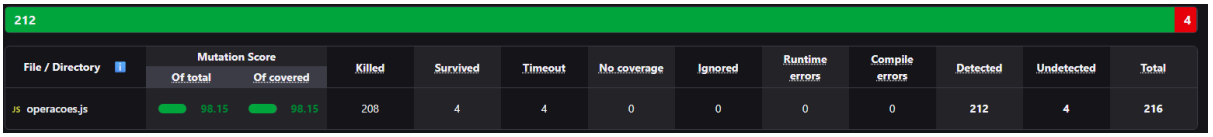
```
function produtoArray(numeros) {
  if (numeros.length === 0) return 1;
  let resultado = numeros[0];
  for (let i = 1; i < numeros.length; i++) {
    resultado = resultado * numeros[i];
  }
  return resultado;
}
  function fatorial(n) {
  if (n < 0) throw new Error('Fatorial não é definido para números negativos.');
  if (n === 0) return 1;
  if (n === 1) return 1;

  let resultado = 1;
  for (let i = 2; i <= n; i++) { resultado *= i; }
  return resultado;
}
```

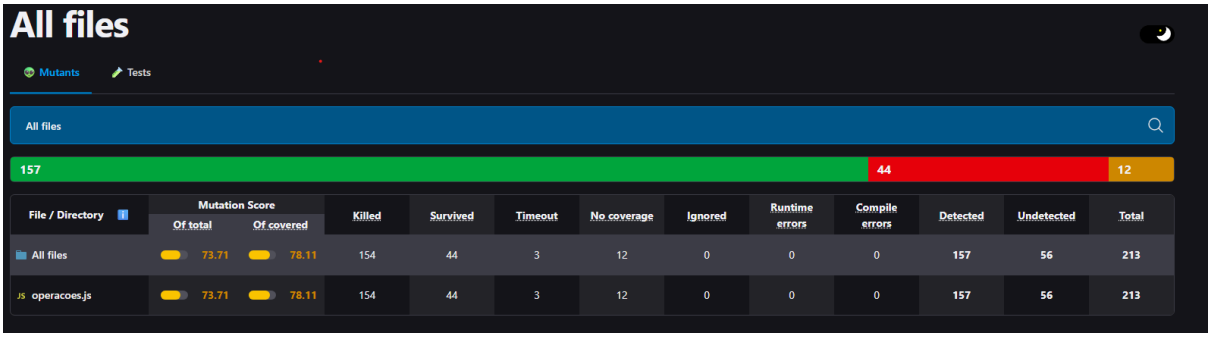
Justificativa:
Eu alterei o fatorial para poder evitar o mutante do || que trocava ele por && e não tinha como matá-lo e o produtoArray eu modifiquei ele por conta que se o problema era a ordem, eu troquei a forma que era feito para que a ordem fique fixa

Resultados Finais

Pontuação de Mutação Após Melhorias: 98.15 %
Comparação com a Pontuação Inicial: 73.71 %



File / Directory	Mutation Score		Killed	Survived	Timeout	No coverage	Ignored	Runtime errors	Compile errors	Detected	Undetected	Total
	Of total	Of covered										
operacoes.js	98.15	98.15	208	4	4	0	0	0	0	212	4	216



File / Directory	Mutation Score		Killed	Survived	Timeout	No coverage	Ignored	Runtime errors	Compile errors	Detected	Undetected	Total
	Of total	Of covered										
All files	73.71	78.11	154	44	3	12	0	0	0	157	56	213
operacoes.js	73.71	78.11	154	44	3	12	0	0	0	157	56	213

Conclusão

Bem, minha conclusão é que os testes de mutação são muito bons para identificar certos pontos cegos nas suítes de teste originais, ajudando a mostrar também que mesmo com 100% de cobertura de código não reflete 100% de eficácia. Após a melhoria dos testes, a pontuação de mutação aumentou substancialmente, comprovando que os novos casos são capazes de detectar mutações sutis no código.