**QUESTÃO III – Dado o código que implementa a função de Fibonnaci, construa casos de testes que garantam que o código seja coberto no que diz respeito a métodos, expressões, ramo, decisão, caminhos (1.0 PTO).**

*1: public class Matematica {*

*2: public int fibonacci(int numero) {*

*3: if (numero >= 0) {*

*4 : if (numero == 0) return 0;*

*5 : if (numero == 1) return 1;*

*6 : return fibonacci(numero-1) + fibonacci(numero-2);*

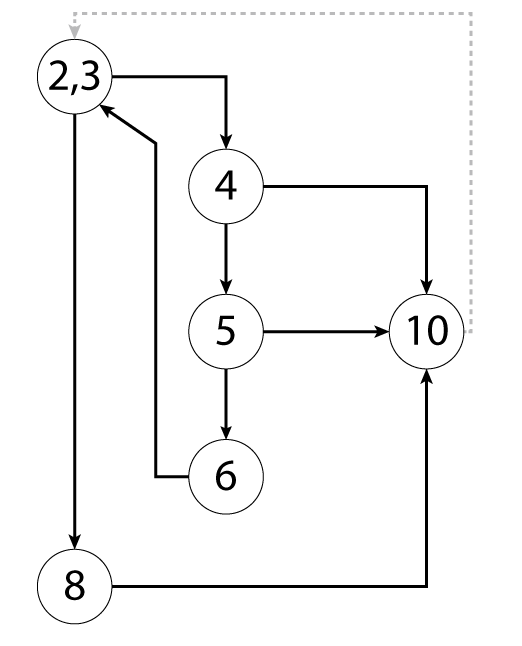
*7: }*

*8: throw new IllegalArgumentException();*

*9: }*

*10: }*

Do código acima podemos extrair o seguinte grafo de execução que nos auxiliará na avaliação da cobertura do código:



Dessa forma começaremos pelos nós:

C0: Cobertura de Expressões:

1. Numero = -1 -> Exceção!
2. Numero = 0 -> 0
3. Numero = 1 -> 1
4. Numero = 2 -> 2

Com esses quatro testes garantimos que todos os nós serão chamados, ou seja, garantimos a cobertura de expressões.

C1: Cobertura de Ramos:

1. Numero = -1 -> Exceção!
2. Numero = 0 -> 0
3. Numero = 1 -> 1
4. Numero = 2 -> 2

Com esses quatro testes também garantimos que todos os ramos serão percorridos, ou seja, garantimos a cobertura de ramos.

C1+: Cobertura de Decisão:

Como o código não apresenta if’s com mais de uma condição, não passamos por nenhum caso a mais de teste por cobertura de decisão.

C2: Cobertura de Caminhos:

Como temos recursão, precisamos considerar a análise em passeios e não apenas a análise em caminhos.

Para isso precisaríamos analisar qual o valor máximo que poderíamos colocar na função para que de forma recursiva ela calculasse o valor de Fibonacci desse número. Para tal análise, sabemos que um número int apresenta o valor máximo de 2147483647 (4 bytes – 32 bit). Assim para que possamos efetuar todos os passeios possíveis do nosso grafo, basta que encontremos o número que tenha o valor de Fibonacci mais próximo desse sem que o ultrapasse. Com o auxílio de uma calculadora, encontramos: 46 cujo número Fibonacci é aproximadamente 1,86\*10^9, o maior número de Fibonacci que cabe num int.

Assim com o intuito de garantir a cobertura de caminhos – passeios – podemos apenas testar o programa para o caso numero = 46.

Com todos os requisitos garantidos, podemos em uma análise final, para simplificar os casos de teste, testar apenas o caso com 46, pois, além de garantir todos os passeios, ele também garante todos os outros casos, por ser recursivo.

Dessa forma:

ANÁLISE DEFINITIVA:

CASO ÚNICO DE TESTES: numero = 46 -> F(46) = 1.86\*10^9