CAPA

## Passo-a-passo de cada requerimento

1. Line and Branch Coverage for methods contains and equals;

Para a criação dos testes para o line e branch coverage de cada método foram criados 2 ficheiros: “TestLCAndBCContains.java” e “TestLCAndBCEquals.java”. Para verificar o line e branch coverage foi utilizada uma ferramenta do eclipse, o uso da ferramenta basea-se na execução dos testes utilizando a opção Coverage As -> JUnit Test no respetivo ficheiro a executar.

Após a execução dos testes podemos concluir que a primeira condição no método *equalTrees*, *if (one == other)*,na linha 301, é impossível ser concretizada pois essa condição já é verificada no início do *equals* e quando verdadeira o resultado é logo retornado, nem sendo executada a função *equalTrees*, logo a linha 302 é impossível ser atingida.   
Para além desse branch ser impossível de atingir, existem mais 2 branch que não são possíveis de ser atingidos, na condição *if (one != null && other != null)*, os branches: *one == null && other != null* e *one != null && other != null*. Estes branches são impossíveis de serem atingidos pois o other é brevemente verificado se é null ou não quando executa o programa verifica o seguinte, *other instanceof NTree*. Caso other seja null, o resultado do operador binário, &&, é de imediato false, não sendo executado o equalTrees.

1. Edge-Pair Coverage and at least 50% coverage for Prime Path Coverage for method equals;

Antes de tudo, é preciso construir o grafo para o método equals e este foi o grafo obtido:

PRINT DO GRAFO  
Após a criação do grafo foi utilizado o website …, e lá foram introduzidas as informações que podemos obter analisando o grafo. Utilizando o website podemos obter automaticamente os testes para o Edge-Pair Coverage e para o Prime Path Coverage.  
Estes foram os resultados obtidos:

PRINT DAS 2 CENAS

1. All-Coupling-Use Coverage for method delete and its private methods;

Para perceber os testes necessários para o All-Coupling-Use Coverage sobre o método delete e os outros métodos privados que o delete utiliza, foi construída a seguinte tabela.

Inserir tabela

Através da tabela foram analisadas as variáveis que provêm de outras funções e o primeiro uso delas e a última definição das mesmas. Através da análise foi possível obter a seguinte tabela:

|  |  |  |
| --- | --- | --- |
| Function | First use | Last definition |
| **delete** | Variável **position** em “position < 0” – edge (7,9) e (7,10) | Última definição na função **proposePosition** nonodeA, G ou I |
| **compact** | Variável **children** em “children[i].isEmpty()” – edge (d,e) e (d,f) | Última definição na função **delete** nonode 8 ou 12 |
| **proposePosition** | Variável **elem** em “children[i].data.compareTo(elem)==0” – edge (c,e),(c,f) | Última definição na função **delete** nonode 1 |

1. Select and apply one Logic-based test coverage for method equals, justify your option.

O teste Logic-based escolhido foi o RACC, que da coverage a … e a …

Foi escolhido este método pois é o método que mais se aproxima à opcao de realizar teste para todas as opções, **ver slides e escrever o que o stor diz sobre o RACC over ACC**,para os aplicar o RACC foram criadas tabelas para cada conjunto de consições existentes no método equals e equalTrees, as tabelas criadas foram as seguintes: