Software Testing Project Report

Ehraz Hasan **MT2021045**
Md Zainul Haque **MT2021073**

Prof - Meenakshi D'Souza

GitHub link - https://github.com/zaindol/Software-Testing.git

INDEX -

1) Algorithms

- Anagram

- Basic Block
- CFG
- Prime path coverage
- Edge pair coverage

- Binary Search

- Basic Block
- CFG
- Prime path coverage
- Edge pair coverage

- Selection Sort

- Basic Block
- CFG
- Prime path coverage
- Edge pair coverage

2) Mathematical procedure

- Contruct 2D Matrix

- Basic Block
- CFG
- Prime path coverage
- Edge pair coverage

- Pythagorean Triplet

- Basic Block
- CFG
- Prime path coverage
- Edge pair coverage

- **Area**

    - Basic Block
    - CFG
    - Prime path coverage
    - Edge pair coverage

- **Square Root**

    - Basic Block
    - CFG
    - Prime path coverage
    - Edge pair coverage

- **Roman to Int**

    - Basic Block
    - CFG
    - Prime path coverage
    - Edge pair coverage

3) Data Structure

- **Remove Linked List Element value**

    - Basic Block
    - CFG
    - Prime path coverage
    - Edge pair coverage

Problem - Control flow Graph Based

Tool used - JUnit

We are slowly increasing the difficulty of testing

Algorithms

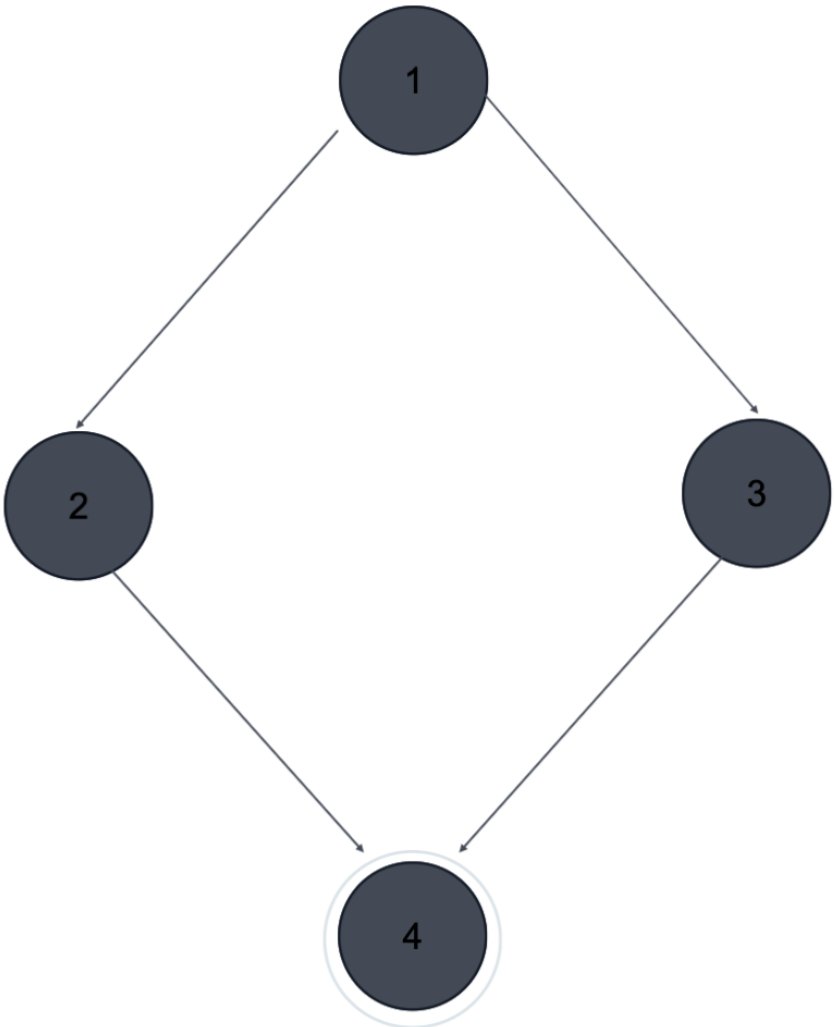
Anagram Code -

```java

import java.util.Arrays;

public class Anagram {
    public static boolean anagram(String str1 , String str2){
    //        removing the space
    String s1 = str1.replaceAll( regex: "\\s", replacement: "");
    String s2 = str2.replaceAll( regex: "\\s", replacement: "");
    boolean status = true;
    //          checking the length
        if (s1.length() != s2.length()) {
        status = false;
    } else {
            char[] ArrayS1 = s1.toLowerCase().toCharArray();
            char[] ArrayS2 = s2.toLowerCase().toCharArray();
            Arrays.sort(ArrayS1);
            Arrays.sort(ArrayS2);
            status = Arrays.equals(ArrayS1, ArrayS2);
        }
        return status;
    }
```

- **Basic Block -**

Lines	Block Number	remark
6 - 11	**1**	We have called all the function
12 - 13	**2**	If condition
14 - 20	**3**	Else condition
21	**4**	Final statement

CFG



2 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,4]	[1,2,4]
[1,3,4]	[1,3,4]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,4]	None
[1,3,4]	None

Infeasible prime paths are:
None

2 test paths are needed for Edge-Pair Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,4]	[1,2,4]
[1,3,4]	[1,3,4]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,4]	None
[1,3,4]	None

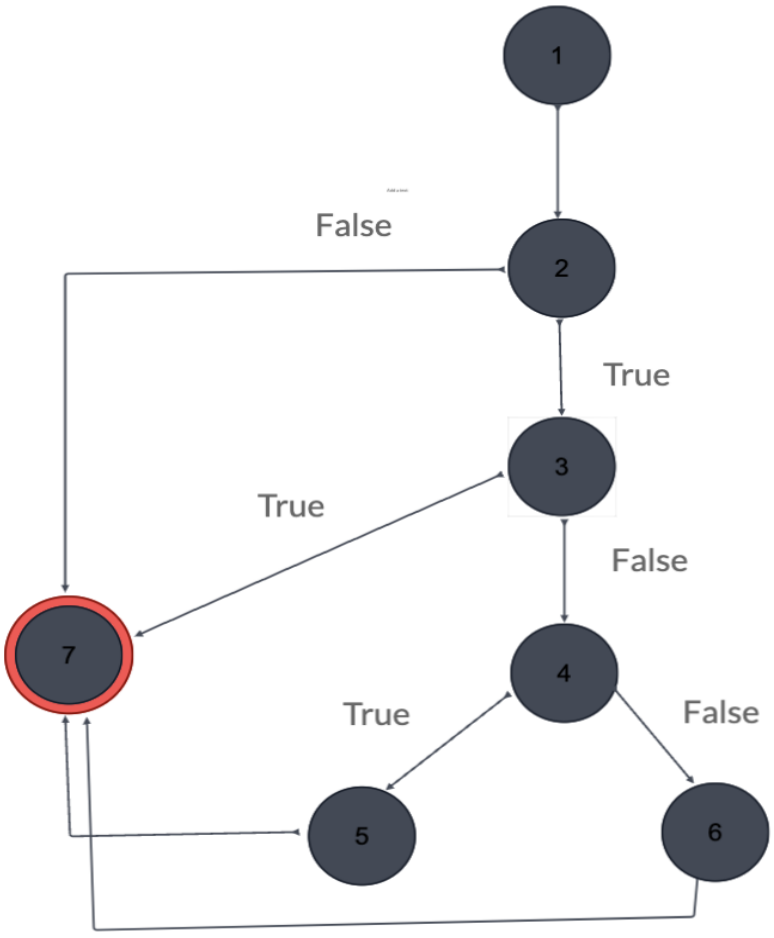
Binary Search Code -

```java
package Methods.Algorithm;

public class BinarySearch {
    public static  int binarysearch (int arr[], int x)
    {
        int l = 0, r = arr.length - 1, res=-1;
        while (l <= r) {
            int m = l + (r - l) / 2;
            if (arr[m] == x) {
                res = m;
                break;
            }
            if (arr[m] < x)
                l = m + 1;
            else
                r = m - 1;
        }
        return res;
    }
}
```

- **Basic Block -**

Lines	Block Number	Remark
4 - 6	1	Starting
7 - 8	2	While loop
9 - 11	3	If statment
13	4	Second if
14	5	If part
15 - 17	6	Else part

CFG



4 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,3,4,5,7]	[1,2,3,4,5,7]
[1,2,3,4,6,7]	[1,2,3,4,6,7]
[1,2,3,7]	[1,2,3,7]
[1,2,7]	[1,2,7]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,3,4,5,7]	None
[1,2,3,4,6,7]	None
[1,2,3,7]	None
[1,2,7]	None

Infeasible prime paths are:
None

4 test paths are needed for Edge-Pair Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,3,7]	[1,2,3], [2,3,7]
[1,2,7]	[1,2,7]
[1,2,3,4,6,7]	[1,2,3], [2,3,4], [3,4,6], [4,6,7]
[1,2,3,4,5,7]	[1,2,3], [2,3,4], [3,4,5], [4,5,7]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,3,7]	None
[1,2,7]	None
[1,2,3,4,6,7]	None
[1,2,3,4,5,7]	None

Infeasible Edge-Pairs are:
None

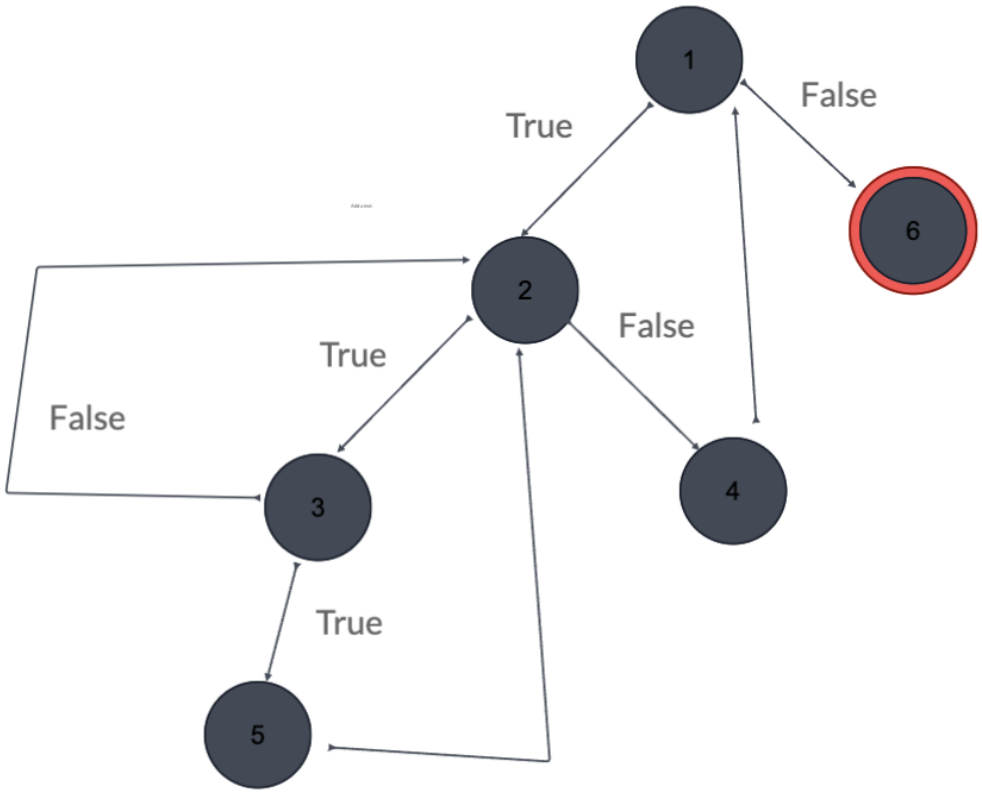
Selection sort Code -

```java
package Methods.Algorithm;

public class SelectionSort {
    public static int[] selectionSort(int[] arr , int n) {
        for (int i = 0; i < n; i++) {
            int index = i;
            for (int j = i + 1; j < n; j++) {
                if (arr[j] < arr[index]) {
                    index = j;//searching for lowest index
                }
            }
            int smallerNumber = arr[index];
            arr[index] = arr[i];
            arr[i] = smallerNumber;

        }
        return arr;
    }


}
```

Basic Block -

Lines	Block number	remark
5	**1**	Starting For loop
6 - 7	**2**	Starting of 2nd For
8	**3**	If condition
9	**5**	After True if condition
12 - 15	**4**	After false for condition
17	**6**	Final statement

CFG



5 test paths are needed for Edge-Pair Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,4,1,6]	[1,2,4], [1,6], [2,4,1], [4,1,6]
[1,2,3,5,2,4,1,6]	[1,2,3], [1,6], [2,3,5], [2,4,1], [3,5,2], [5,2,4], [4,1,6]
[1,2,3,2,3,2,4,1,6]	[1,2,3], [1,6], [2,3,2], [2,4,1], [3,2,3], [3,2,4], [4,1,6]
[1,2,3,5,2,3,2,4,1,6]	[1,2,3], [1,6], [2,3,5], [2,3,2], [2,4,1], [3,5,2], [3,2,4], [5,2,3], [4,1,6]
[1,2,4,1,2,4,1,6]	[1,2,4], [1,6], [2,4,1], [4,1,2], [4,1,6]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,4,1,6]	None
[1,2,3,5,2,4,1,6]	None
[1,2,3,2,3,2,4,1,6]	[1,2,3], [2,3,2], [3,2,4]
[1,2,3,5,2,3,2,4,1,6]	[3,5,2], [5,2,4]
[1,2,4,1,2,4,1,6]	None

Infeasible Edge-Pairs are:
None

5 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,4,1,2,3,5,2,4,1,6]	[3,5,2,4,1,6], [4,1,2,3,5], [2,4,1,2], [2,3,5,2], [1,2,4,1]
[1,2,4,1,2,4,1,6]	[2,4,1,2], [1,2,4,1], [4,1,2,4]
[1,2,3,5,2,3,2,4,1,6]	[3,2,4,1,6], [3,5,2,3], [2,3,5,2], [2,3,2]
[1,2,3,5,2,3,5,2,4,1,6]	[3,5,2,4,1,6], [3,5,2,3], [2,3,5,2], [5,2,3,5]
[1,2,3,2,3,2,4,1,6]	[3,2,4,1,6], [2,3,2], [3,2,3]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,4,1,2,3,5,2,4,1,6]	[2,4,1,2], [1,2,4,1], [4,1,2,4]
[1,2,4,1,2,4,1,6]	[1,2,4,1]
[1,2,3,5,2,3,2,4,1,6]	[3,5,2,4,1,6], [2,3,5,2], [2,3,2]
[1,2,3,5,2,3,5,2,4,1,6]	[3,5,2,4,1,6], [2,3,5,2]
[1,2,3,2,3,2,4,1,6]	[3,2,4,1,6], [2,3,2]

Infeasible prime paths are:
None

Mathematical Problem

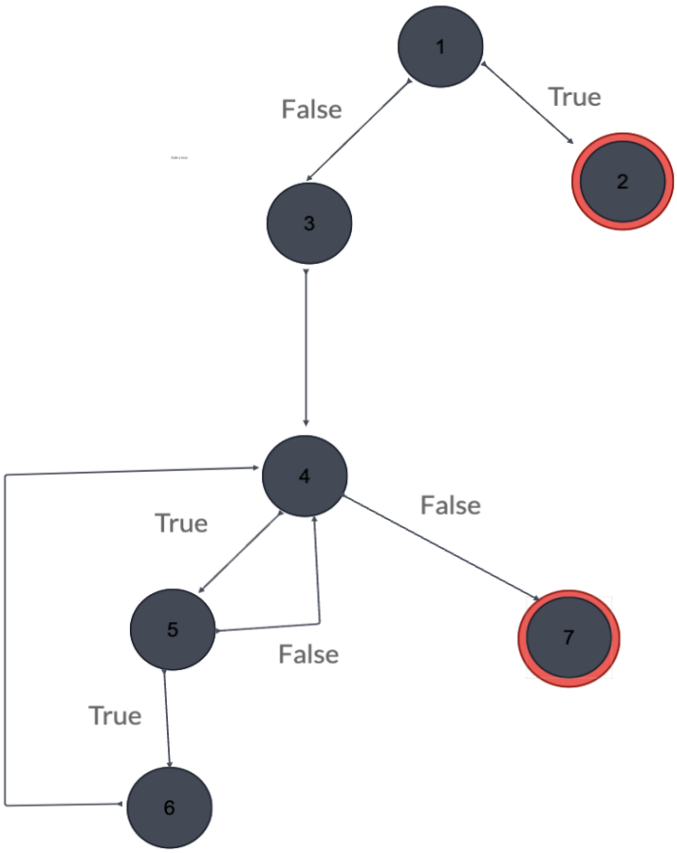
Construct 2D Array -

```java
package Methods.Mathematical;

public class Construct2D {
    public static int[][] construct2DArray(int[] original, int m, int n) {
        if(m * n != original.length) {
            return new int[0][0];
        }
        int[][] answer = new int[m][n];
        int rCount = 0, cCount = 0, len = original.length;
        for(int i=0;i<len;i++){
            answer[rCount][cCount++] = original[i];
            if(cCount == n) {
                rCount++;
                cCount = 0;
            }
        }

        return answer;
    }
}

```

Basic block -

Lines	Block number	remark
5	**1**	Starting if
6 - 7	**2**	Edge case
8 - 9	**3**	Statement
10	**4**	For loop
11 - 12	**5**	If condition
13 - 14	**6**	Inner statements
18	**7**	Exit

CFG



5 test paths are needed for Edge-Pair Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2]	[1,2]
[1,3,4,7]	[1,3,4], [3,4,7]
[1,3,4,5,6,4,7]	[1,3,4], [3,4,5], [4,5,6], [5,6,4], [6,4,7]
[1,3,4,5,4,5,4,7]	[1,3,4], [3,4,5], [4,5,4], [5,4,5], [5,4,7]
[1,3,4,5,6,4,5,4,7]	[1,3,4], [3,4,5], [4,5,6], [4,5,4], [5,6,4], [5,4,7], [6,4,5]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2]	None
[1,3,4,7]	None
[1,3,4,5,6,4,7]	None
[1,3,4,5,4,5,4,7]	[1,3,4], [3,4,5], [4,5,4], [5,4,7]
[1,3,4,5,6,4,5,4,7]	[5,6,4], [6,4,7]

Infeasible Edge-Pairs are:
None

5 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,3,4,7]	[1,3,4,7]
[1,3,4,5,6,4,5,6,4,7]	[1,3,4,5,6], [4,5,6,4], [6,4,5,6], [5,6,4,7], [5,6,4,5]
[1,3,4,5,6,4,5,4,7]	[1,3,4,5,6], [4,5,6,4], [5,6,4,5], [4,5,4], [5,4,7]
[1,3,4,5,4,5,4,7]	[4,5,4], [5,4,7], [5,4,5]
[1,2]	[1,2]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,3,4,7]	None
[1,3,4,5,6,4,5,6,4,7]	[1,3,4,5,6], [4,5,6,4], [5,6,4,7]
[1,3,4,5,6,4,5,4,7]	[4,5,6,4], [5,6,4,7], [4,5,4]
[1,3,4,5,4,5,4,7]	[4,5,4], [5,4,7]
[1,2]	None

Infeasible prime paths are:
None

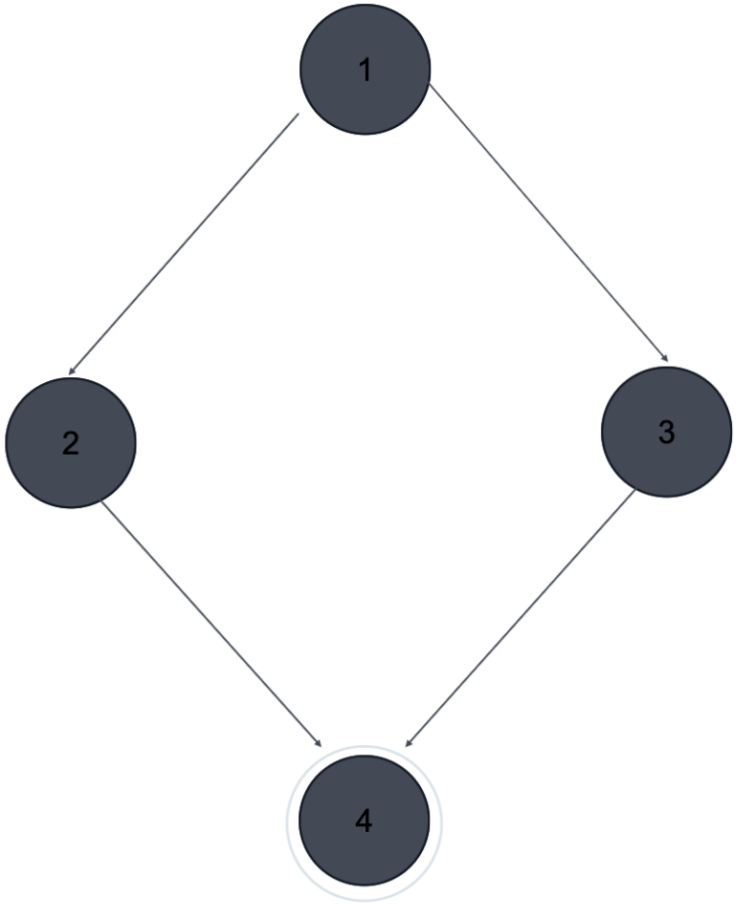
Pythagorean Triplet -

```java
package Methods.Mathematical;

public class PythagoreanTriple {
    public static boolean pythagoreantriple(int a,int b,int c) {
        boolean res = true;
        int max = Math. max(a, Math.max(b, c));
        int min = Math.min(a, Math.min(b, c));
        int mid = a + b + c - max - min;
        if (min <=0 || mid <= 0 ||max <=0) {
            res = false;
        } else {
            res = (min * min) + (mid * mid) == (max * max);
        }
        return res;
    }
}

```

Basic Block -

Lines	Block number	Remark
4 - 9	**1**	Entry statement
10	**2**	Inner if
11 - 13	**3**	Inner else
14	**4**	**Exit**

CFG -



2 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,4]	[1,2,4]
[1,3,4]	[1,3,4]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,4]	None
[1,3,4]	None

Infeasible prime paths are:
None

2 test paths are needed for Edge-Pair Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,4]	[1,2,4]
[1,3,4]	[1,3,4]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,4]	None
[1,3,4]	None

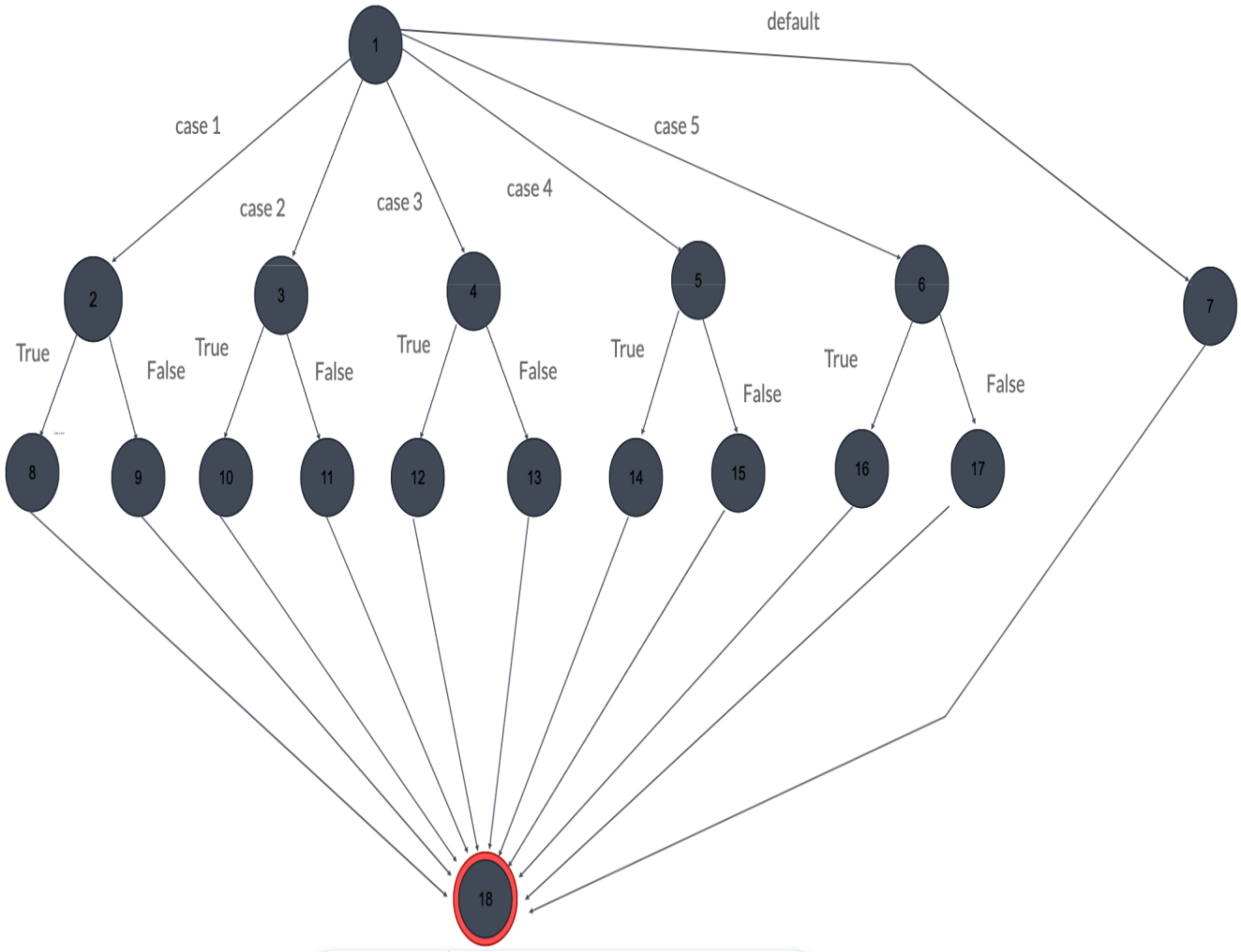
Area -

```java
package Methods.Mathematical;

public class Area {
    public double area(int input, double a, double b) {
        double ans = 0;
        switch (input) {
            case 1:
                if (a < 0) {
                    System.out.println("sides cannot be negative");
                    ans = -1;
                    break;
                }
                ans = 6 * a * a;
                break;
            case 2:
                if (a < 0) {
                    System.out.println("sides cannot be negative");
                    ans = -1;
                    break;
                }
                ans = 4 * Math.PI * a * a;
                break;
            case 3:
                if (a < 0 || b < 0) {
                    System.out.println("sides cannot be negative");
                    ans = -1;
                    break;
                }

                ans = Math.PI * a * (a + Math.pow((b * b + a * a), 0.5));
                break;
```

```
            case 4:
                if (a < 0 || b < 0) {
                    System.out.println("sides cannot be negative");
                    ans = -1;
                    break;

                }
                ans = 3 * Math.PI * a * a;
                break;
            case 5:
                if (a < 0 || b < 0) {
                    System.out.println("sides cannot be negative");
                    ans = -1;
                    break;
                }
                ans = 2 * (Math.PI * a * a + Math.PI * a * b);
                break;
            default:
                System.out.println("invalid input");
                ans = -1;
        }
        System.out.println("Result: " + ans);
        return ans;
    }
}

```

Basic block -

Lines	Block number	remarks
4 - 6	1	Starting
8	2	Case 1
15	3	Case 2
24	4	Case 3
32	5	Case 4
42	6	Case 5
49 - 51	7	default
9 - 11	8	Inside if
13	9	Outside if
17 - 19	10	Inside if
21	11	Outside if
25 - 27	12	Inside if
30	13	Outside if
33 - 36	14	Inside if
39	15	Outside if
43 - 45	16	Inside if
47	17	Outside if
53- 54	18	Final


1
default
case 1
case 2
case 3
case 4
case 5
2
3
4
5
6
7
True
False
True
False
True
False
True
False
True
False
8
9
10
11
12
13
14
15
16
17
18

11 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,3,11,18]	[1,3,11,18]
[1,4,12,18]	[1,4,12,18]
[1,3,10,18]	[1,3,10,18]
[1,2,8,18]	[1,2,8,18]
[1,2,9,18]	[1,2,9,18]
[1,4,13,18]	[1,4,13,18]
[1,6,16,18]	[1,6,16,18]
[1,6,17,18]	[1,6,17,18]
[1,5,15,18]	[1,5,15,18]
[1,5,14,18]	[1,5,14,18]
[1,7,18]	[1,7,18]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,3,11,18]	None
[1,4,12,18]	None
[1,3,10,18]	None
[1,2,8,18]	None
[1,2,9,18]	None
[1,4,13,18]	None
[1,6,16,18]	None
[1,6,17,18]	None
[1,5,15,18]	None
[1,5,14,18]	None
[1,7,18]	None

Infeasible prime paths are:
None

11 test paths are needed for Edge-Pair Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,8,18]	[1,2,8], [2,8,18]
[1,2,9,18]	[1,2,9], [2,9,18]
[1,3,10,18]	[1,3,10], [3,10,18]
[1,3,11,18]	[1,3,11], [3,11,18]
[1,4,12,18]	[1,4,12], [4,12,18]
[1,4,13,18]	[1,4,13], [4,13,18]
[1,5,14,18]	[1,5,14], [5,14,18]
[1,5,15,18]	[1,5,15], [5,15,18]
[1,6,16,18]	[1,6,16], [6,16,18]
[1,6,17,18]	[1,6,17], [6,17,18]
[1,7,18]	[1,7,18]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,8,18]	None
[1,2,9,18]	None
[1,3,10,18]	None
[1,3,11,18]	None
[1,4,12,18]	None
[1,4,13,18]	None
[1,5,14,18]	None
[1,5,15,18]	None
[1,6,16,18]	None
[1,6,17,18]	None
[1,7,18]	None

Infeasible Edge-Pairs are:
None

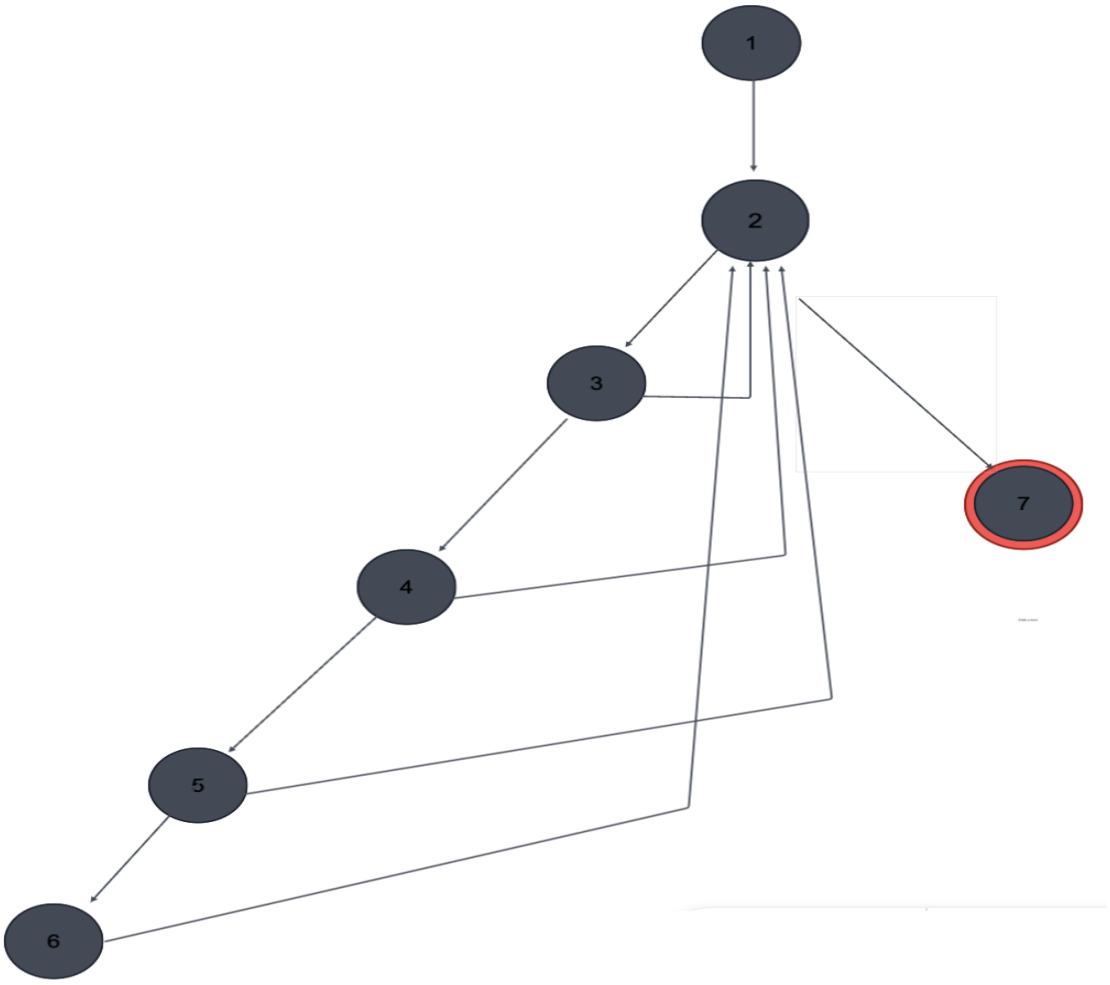
Roman to Int -

```java
public static int romanToInt(String s) {
    Map<Character, Integer> romanToNumber  = new HashMap<>();
    romanToNumber.put('I', 1);
    romanToNumber.put('V', 5);
    romanToNumber.put('X', 10);
    romanToNumber.put('L', 50);
    romanToNumber.put('C', 100);
    romanToNumber.put('D', 500);
    romanToNumber.put('M', 1000);

    int result = 0;
    for(int i = 0; i <= s.length()-1; i++){
        char romanNumb = s.charAt(i);

        if(romanNumb == 'I' || romanNumb == 'X' || romanNumb == 'C') {
            if(i != s.length()-1) {
                char next = s.charAt(i+1);

                if((romanNumb == 'I' && (next == 'V' || next == 'X')) ||
                        (romanNumb == 'X' && (next == 'L' || next == 'C')) ||
                        (romanNumb == 'C' && (next == 'D' || next == 'M'))) {
                    result -= romanToNumber.get(romanNumb);
                    continue;
                }
            }
        }

        result += romanToNumber.get(romanNumb);

    };
```

Basic block -

Links	Block number	remarks
8 - 17	**1**	Starting
18	**2**	For loop
19 - 21	**3**	If condition
22	**4**	Inner if condition
23 - 25	**5**	If condition
28 - 30	**6**	Body
34	**7**	**Exit**

CFG -



7 test paths are needed for Edge-Pair Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,3,4,2,7]	[1,2,3], [2,3,4], [3,4,2], [4,2,7]
[1,2,3,4,5,2,7]	[1,2,3], [2,3,4], [3,4,5], [4,5,2], [5,2,7]
[1,2,3,2,3,2,7]	[1,2,3], [2,3,2], [3,2,3], [3,2,7]
[1,2,3,4,2,3,2,7]	[1,2,3], [2,3,4], [2,3,2], [3,4,2], [3,2,7], [4,2,3]
[1,2,3,4,5,6,2,7]	[1,2,3], [2,3,4], [3,4,5], [4,5,6], [5,6,2], [6,2,7]
[1,2,3,4,5,2,3,2,7]	[1,2,3], [2,3,4], [2,3,2], [3,4,5], [3,2,7], [4,5,2], [5,2,3]
[1,2,3,4,5,6,2,3,2,7]	[1,2,3], [2,3,4], [2,3,2], [3,4,5], [3,2,7], [4,5,6], [5,6,2], [6,2,3]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,3,4,2,7]	None
[1,2,3,4,5,2,7]	None
[1,2,3,2,3,2,7]	[1,2,3], [2,3,2], [3,2,7]
[1,2,3,4,2,3,2,7]	[3,4,2], [4,2,7]
[1,2,3,4,5,6,2,7]	None
[1,2,3,4,5,2,3,2,7]	[4,5,2], [5,2,7]
[1,2,3,4,5,6,2,3,2,7]	[5,6,2], [6,2,7]

Infeasible Edge-Pairs are:
[1,2,7]

11 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,3,4,5,6,2,3,2,7]	[3,4,5,6,2,3], [2,3,4,5,6,2], [1,2,3,4,5,6], [3,2,7], [2,3,2]
[1,2,3,4,5,6,2,3,4,5,6,2,7]	[3,4,5,6,2,3], [2,3,4,5,6,2], [1,2,3,4,5,6], [3,4,5,6,2,7], [6,2,3,4,5,6], [5,6,2,3,4,5], [4,5,6,2,3,4]
[1,2,3,4,5,6,2,3,4,5,2,7]	[3,4,5,6,2,3], [2,3,4,5,6,2], [1,2,3,4,5,6], [5,6,2,3,4,5], [4,5,6,2,3,4], [3,4,5,2,7], [2,3,4,5,2]
[1,2,3,4,5,6,2,3,4,2,7]	[3,4,5,6,2,3], [2,3,4,5,6,2], [1,2,3,4,5,6], [4,5,6,2,3,4], [3,4,2,7], [2,3,4,2]
[1,2,3,4,5,2,3,2,7]	[3,4,5,2,3], [2,3,4,5,2], [3,2,7], [2,3,2]
[1,2,3,4,5,2,3,4,2,7]	[3,4,5,2,3], [2,3,4,5,2], [4,5,2,3,4], [3,4,2,7], [2,3,4,2]
[1,2,3,4,5,2,3,4,5,2,7]	[3,4,5,2,3], [3,4,5,2,7], [2,3,4,5,2], [4,5,2,3,4], [5,2,3,4,5]
[1,2,3,4,2,3,2,7]	[3,4,2,3], [2,3,4,2], [3,2,7], [2,3,2]
[1,2,3,4,2,3,4,2,7]	[3,4,2,7], [3,4,2,3], [2,3,4,2], [4,2,3,4]
[1,2,3,2,3,2,7]	[3,2,3], [3,2,7], [2,3,2]
[1,2,7]	[1,2,7]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,3,4,5,6,2,3,2,7]	[2,3,4,5,6,2], [3,4,5,6,2,7], [2,3,2]
[1,2,3,4,5,6,2,3,4,5,6,2,7]	[2,3,4,5,6,2], [1,2,3,4,5,6], [3,4,5,6,2,7]
[1,2,3,4,5,6,2,3,4,5,2,7]	[2,3,4,5,6,2], [3,4,5,6,2,7], [3,4,5,2,7], [2,3,4,5,2]
[1,2,3,4,5,6,2,3,4,2,7]	[2,3,4,5,6,2], [3,4,5,6,2,7], [3,4,2,7], [2,3,4,2]
[1,2,3,4,5,2,3,2,7]	[3,4,5,2,7], [2,3,4,5,2], [2,3,2]
[1,2,3,4,5,2,3,4,2,7]	[3,4,5,2,7], [2,3,4,5,2], [3,4,2,7], [2,3,4,2]
[1,2,3,4,5,2,3,4,5,2,7]	[3,4,5,2,7], [2,3,4,5,2]
[1,2,3,4,2,3,2,7]	[3,4,2,7], [2,3,4,2], [2,3,2]
[1,2,3,4,2,3,4,2,7]	[3,4,2,7], [2,3,4,2]
[1,2,3,2,3,2,7]	[3,2,7], [2,3,2]
[1,2,7]	None

Infeasible prime paths are:

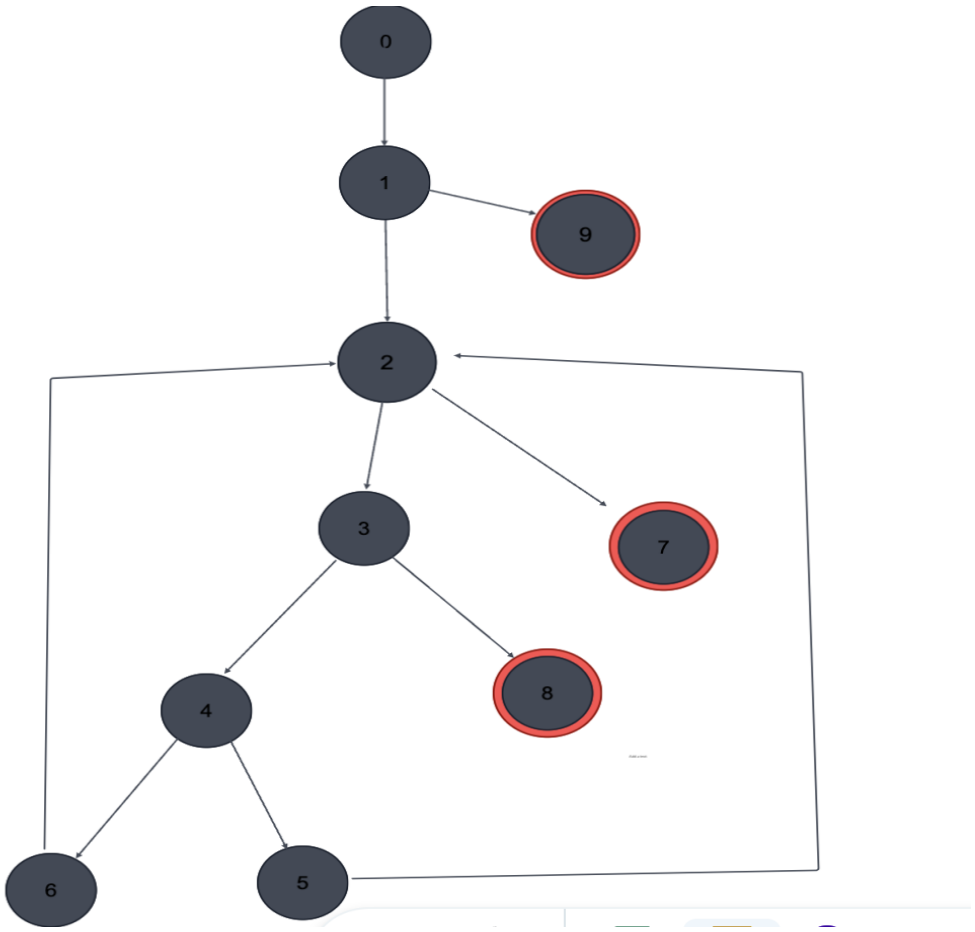
Square Root -

```java
package Methods.Mathematical;

public class SquareRoot {
    public static int squareroot(int x)
    {
        if(x <= 1) return x;
        int start = 1;
        int end = x/2;

        while(start < end) {
            int mid = (start + (end-start)/2) + 1;

            int div = x/mid;
            if(div == mid) return mid;
            if(div > mid) start = mid;
            else end = mid-1;
        }

        return start;
    }
}
```

Basic block -

Links	Block number	Remarks
6	**0**	**If condition**
7 - 8	**1**	Starting node
10	**2**	While loop
11 - 14	**3**	If condition
15	**4**	**Final**
16	**5**	If condition
17	**6**	Else
18	**7**	**Final**

CFG



6 test paths are needed for Edge-Pair Coverage

Test Paths	Test Requirements that are toured by test paths directly
[0,1,9]	[0,1,9]
[0,1,2,7]	[0,1,2], [1,2,7]
[0,1,2,3,4,5,2,7]	[0,1,2], [1,2,3], [2,3,4], [3,4,5], [4,5,2], [5,2,7]
[0,1,2,3,4,6,2,7]	[0,1,2], [1,2,3], [2,3,4], [3,4,6], [4,6,2], [6,2,7]
[0,1,2,3,4,6,2,3,8]	[0,1,2], [1,2,3], [2,3,4], [2,3,8], [3,4,6], [4,6,2], [6,2,3]
[0,1,2,3,4,5,2,3,8]	[0,1,2], [1,2,3], [2,3,4], [2,3,8], [3,4,5], [4,5,2], [5,2,3]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[0,1,9]	None
[0,1,2,7]	None
[0,1,2,3,4,5,2,7]	None
[0,1,2,3,4,6,2,7]	None
[0,1,2,3,4,6,2,3,8]	None
[0,1,2,3,4,5,2,3,8]	None

Infeasible Edge-Pairs are:
None

9 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[0,1,2,3,4,6,2,3,8]	[0,1,2,3,4,6], [3,4,6,2,3], [2,3,4,6,2], [4,6,2,3,8]
[0,1,2,3,4,5,2,3,8]	[0,1,2,3,4,5], [2,3,4,5,2], [3,4,5,2,3], [4,5,2,3,8]
[0,1,2,3,8]	[0,1,2,3,8]
[0,1,2,3,4,6,2,3,4,5,2,7]	[0,1,2,3,4,6], [3,4,6,2,3], [2,3,4,5,2], [2,3,4,6,2], [3,4,5,2,7], [6,2,3,4,5], [4,6,2,3,4]
[0,1,2,3,4,6,2,3,4,6,2,7]	[0,1,2,3,4,6], [3,4,6,2,3], [3,4,6,2,7], [2,3,4,6,2], [6,2,3,4,6], [4,6,2,3,4]
[0,1,2,3,4,5,2,3,4,6,2,7]	[0,1,2,3,4,5], [2,3,4,5,2], [3,4,6,2,7], [3,4,5,2,3], [2,3,4,6,2], [5,2,3,4,6], [4,5,2,3,4]
[0,1,2,3,4,5,2,3,4,5,2,7]	[0,1,2,3,4,5], [2,3,4,5,2], [3,4,5,2,3], [3,4,5,2,7], [5,2,3,4,5], [4,5,2,3,4]
[0,1,2,7]	[0,1,2,7]
[0,1,9]	[0,1,9]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[0,1,2,3,4,6,2,3,8]	[0,1,2,3,8]
[0,1,2,3,4,5,2,3,8]	[0,1,2,3,8]
[0,1,2,3,8]	None
[0,1,2,3,4,6,2,3,4,5,2,7]	[0,1,2,3,4,5], [2,3,4,5,2], [3,4,6,2,7], [2,3,4,6,2], [3,4,5,2,7]
[0,1,2,3,4,6,2,3,4,6,2,7]	[0,1,2,3,4,6], [3,4,6,2,7], [2,3,4,6,2]
[0,1,2,3,4,5,2,3,4,6,2,7]	[0,1,2,3,4,6], [2,3,4,5,2], [3,4,6,2,7], [2,3,4,6,2], [3,4,5,2,7]
[0,1,2,3,4,5,2,3,4,5,2,7]	[0,1,2,3,4,5], [2,3,4,5,2], [3,4,5,2,7]
[0,1,2,7]	None
[0,1,9]	None

Infeasible prime paths are:
None

Datastructure Problem

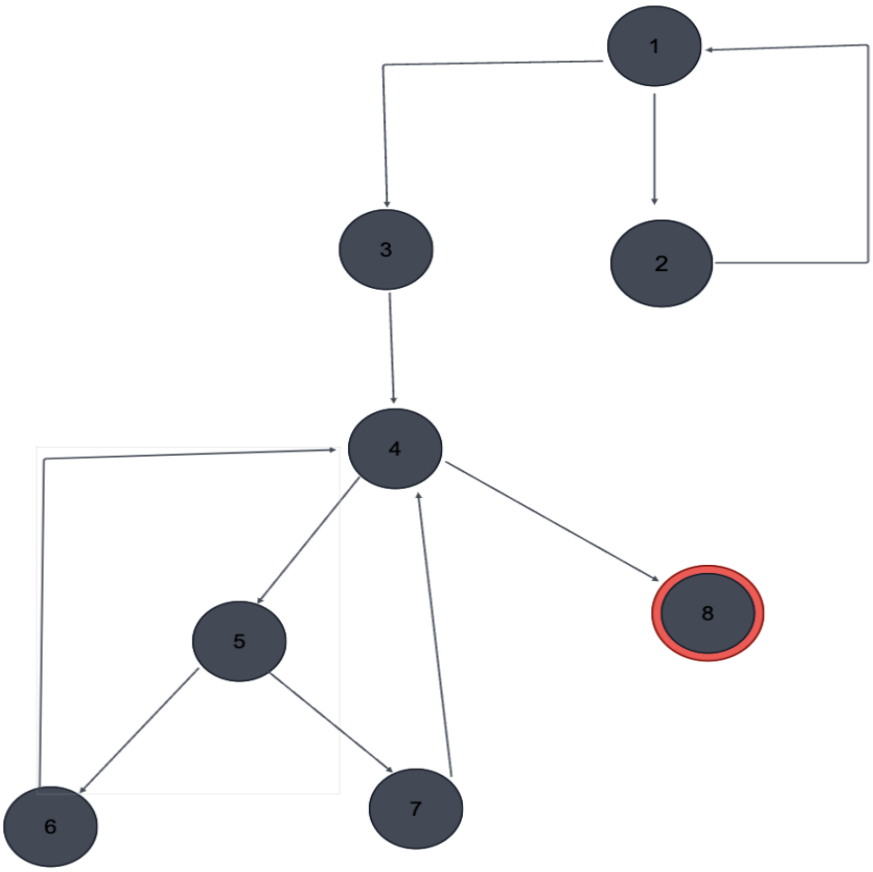
Remove Linked List Element -

```java
public class RemoveLinkedListElement {
    public static ListNode removeElements(ListNode head, int val) {

        while(head!=null && head.val == val){
            head = head.next;
        }

        ListNode current_node = head;
        while(current_node !=null && current_node.next!=null){
            if(current_node.next.val == val){
                current_node.next = current_node.next.next;
            }
            else{
                current_node = current_node.next;
            }
        }
        return head;
    }
}
```

Basic Block -

Links	Block number	Remarks
7	**1**	While loop
8	**2**	Inner
11	**3**	Statement
12	**4**	While loop
13	**5**	If condition
14	**6**	**statment**
17	**7**	**else**
20	**8**	**Final**

CFG



4 test paths are needed for Edge-Pair Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,1,2,1,3,4,8]	[1,2,1], [2,1,2], [2,1,3], [1,3,4], [3,4,8]
[1,3,4,5,6,4,8]	[1,3,4], [3,4,5], [4,5,6], [5,6,4], [6,4,8]
[1,3,4,5,6,4,5,7,4,8]	[1,3,4], [3,4,5], [4,5,6], [4,5,7], [5,6,4], [5,7,4], [6,4,5], [7,4,8]
[1,3,4,5,7,4,5,7,4,8]	[1,3,4], [3,4,5], [4,5,7], [5,7,4], [7,4,5], [7,4,8]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,1,2,1,3,4,8]	[1,2,1], [2,1,3]
[1,3,4,5,6,4,8]	None
[1,3,4,5,6,4,5,7,4,8]	None
[1,3,4,5,7,4,5,7,4,8]	None

Infeasible Edge-Pairs are:
None

9 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[0,1,2,3,4,6,2,3,8]	[0,1,2,3,4,6], [3,4,6,2,3], [2,3,4,6,2], [4,6,2,3,8]
[0,1,2,3,4,5,2,3,8]	[0,1,2,3,4,5], [2,3,4,5,2], [3,4,5,2,3], [4,5,2,3,8]
[0,1,2,3,8]	[0,1,2,3,8]
[0,1,2,3,4,6,2,3,4,5,2,7]	[0,1,2,3,4,6], [3,4,6,2,3], [2,3,4,5,2], [2,3,4,6,2], [3,4,5,2,7], [6,2,3,4,5], [4,6,2,3,4]
[0,1,2,3,4,6,2,3,4,6,2,7]	[0,1,2,3,4,6], [3,4,6,2,3], [3,4,6,2,7], [2,3,4,6,2], [6,2,3,4,6], [4,6,2,3,4]
[0,1,2,3,4,5,2,3,4,6,2,7]	[0,1,2,3,4,5], [2,3,4,5,2], [3,4,6,2,7], [3,4,5,2,3], [2,3,4,6,2], [5,2,3,4,6], [4,5,2,3,4]
[0,1,2,3,4,5,2,3,4,5,2,7]	[0,1,2,3,4,5], [2,3,4,5,2], [3,4,5,2,3], [3,4,5,2,7], [5,2,3,4,5], [4,5,2,3,4]
[0,1,2,7]	[0,1,2,7]
[0,1,9]	[0,1,9]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[0,1,2,3,4,6,2,3,8]	[0,1,2,3,8]
[0,1,2,3,4,5,2,3,8]	[0,1,2,3,8]
[0,1,2,3,8]	None
[0,1,2,3,4,6,2,3,4,5,2,7]	[0,1,2,3,4,5], [2,3,4,5,2], [3,4,6,2,7], [2,3,4,6,2], [3,4,5,2,7]
[0,1,2,3,4,6,2,3,4,6,2,7]	[0,1,2,3,4,6], [3,4,6,2,7], [2,3,4,6,2]
[0,1,2,3,4,5,2,3,4,6,2,7]	[0,1,2,3,4,6], [2,3,4,5,2], [3,4,6,2,7], [2,3,4,6,2], [3,4,5,2,7]
[0,1,2,3,4,5,2,3,4,5,2,7]	[0,1,2,3,4,5], [2,3,4,5,2], [3,4,5,2,7]
[0,1,2,7]	None
[0,1,9]	None

Infeasible prime paths are:
None