International Institute of Information Technology, Bangalore
(IIIT Bangalore)

Software Testing CS 731
Project Report

Rahul Modak (MT2020014)

Akshay Nagpal (MT2020016)

Sounak Dey (MT2020081)

Problem: Dataflow Graph based Testing

A data-flow graph (DFG) is a graph that represents control flow of a function in which every node is labeled with definitions(def) and uses(use) of variables in that basic block.

In this test paths which are DU-paths are generated for each variable which which covers both a def and use of that variable.So basically in this method we use test paths of a program according to the locations of definitions and uses of variables in the program

It is concerned with:

- Statements where variables receive values,
- Statements where these values are used or referenced.

DEF(S) = {X | statement S contains the definition of X}

USE(S) = {X | statement S contains the use of X}

Then, we write Test Cases for every unique DU Path.

About our Project Code:

(https://github.com/rahul166/Software-Testing-)

Our source code is of a “Basic Algorithm Util”, in which we have implemented a command Line based app for running various algorithms.

Tools Used for Testing:

- **Data Flow Graph Coverage Web Application:** (https://cs.gmu.edu:8443/offutt/coverage/DFGraphCoverage)

  We used this web tool to generate all the DU path's for our Data Flow Graph of each function

- **JUnit:** (http://junit.org/junit5/)
  It is a unit testing tool for java based applications, used for automating the execution of the Test Cases.

How to run project and test cases:

Project->

It is a maven project made in intellij.Main code is located in src/main/java/Main.java file that can be simply executed just like any other java program or by clicking on run button in intellij.

Tests->

We have used junit for test automation,junit is added inside maven as a dependency(pom.xml).
To run test cases
first execute maven lifecycle commands->
`mvn clean` -> `mvn install` -> `mvn compile`

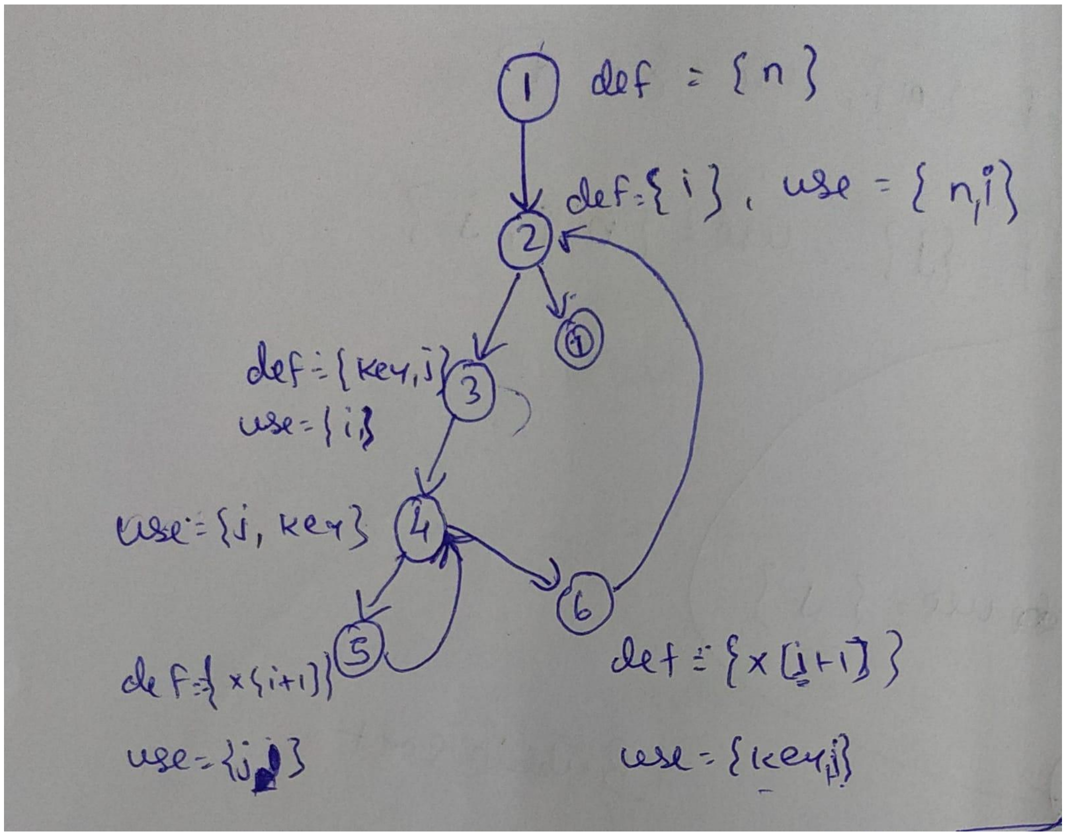
Finally run->
`mvn test`

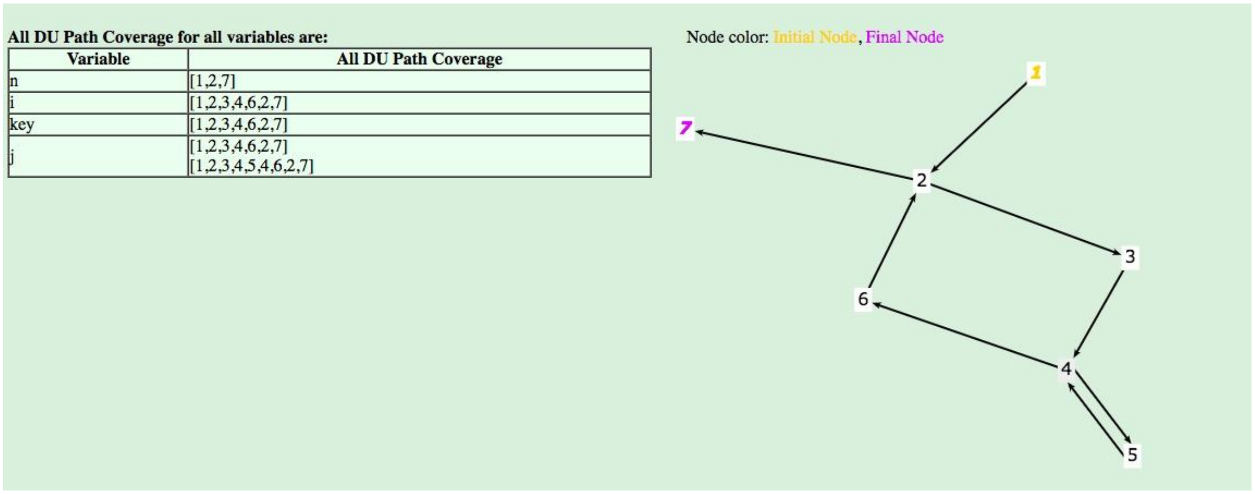
Contributions:

1. **Rahul Modak** - Contributed to full source code and DFG & test cases of Insertion Sort & Binary Search.
2. **Akshay Nagpal** - Contributed to full source code and DFG & test cases of Get Inverse Count & Bubble Sort.
3. **Sounak Dey** - Contributed to full source code and DFG & test cases of Power & Selection Sort.

Data Flow Graphs of a few selected functions along with all their generated DU paths(using the web tool) respectively:

1. Insertion Sort:

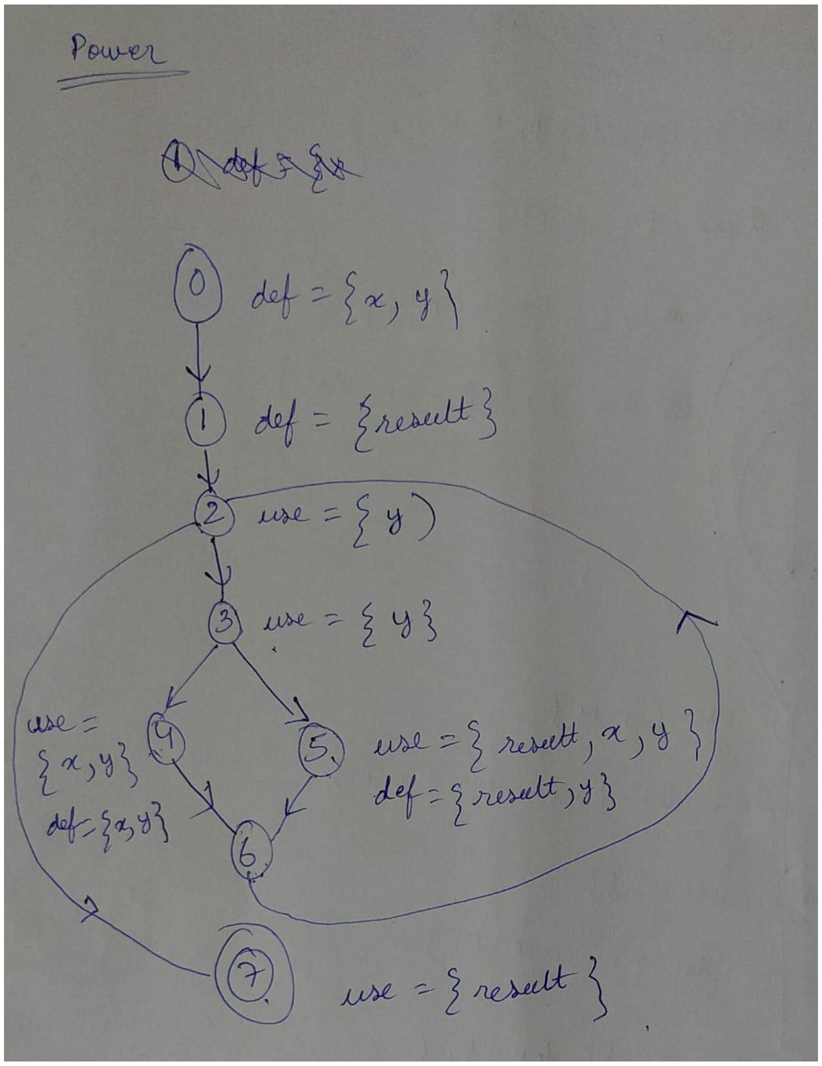


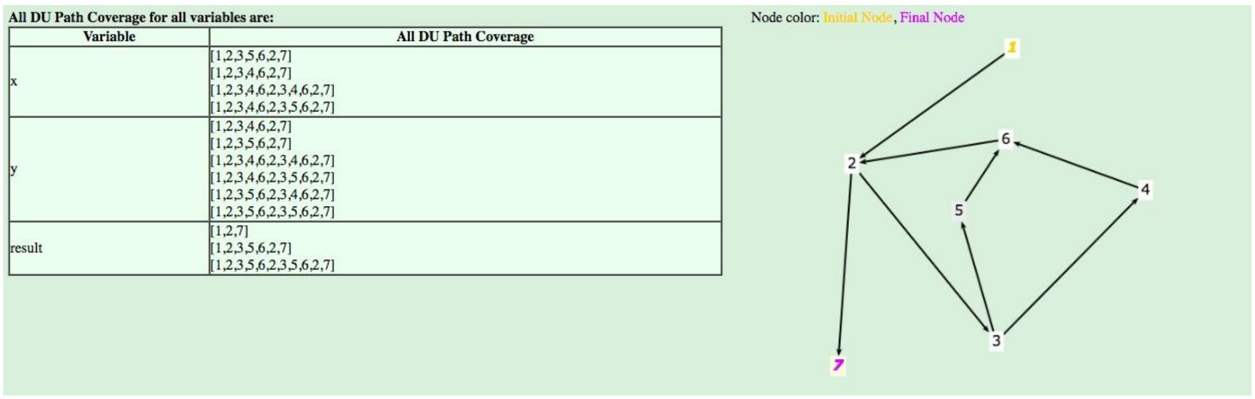

All DU Path Coverage for all variables are:

Variable	All DU Path Coverage
n	[1,2,7]
i	[1,2,3,4,6,2,7]
key	[1,2,3,4,6,2,7]
j	[1,2,3,4,6,2,7] [1,2,3,4,5,4,6,2,7]

Node color: Initial Node, Final Node

2. Power Function:

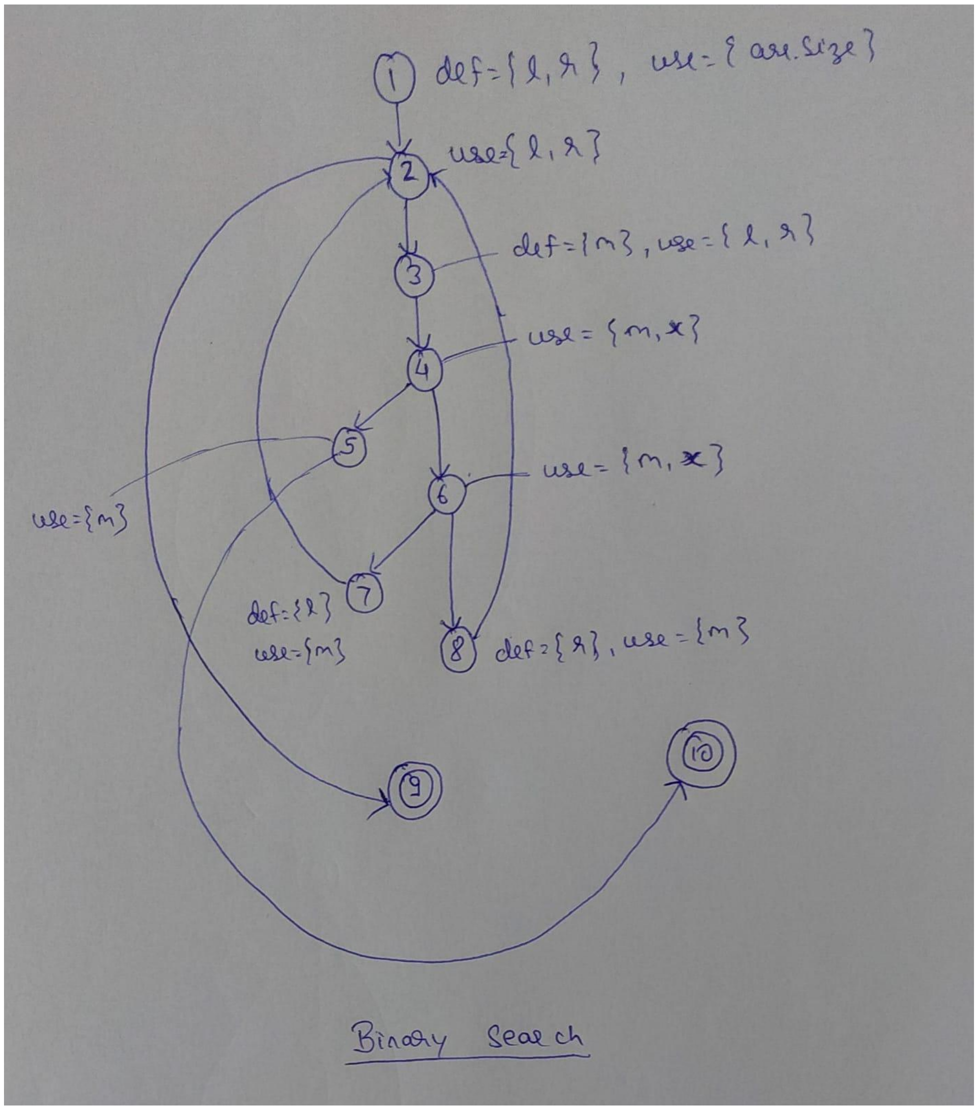


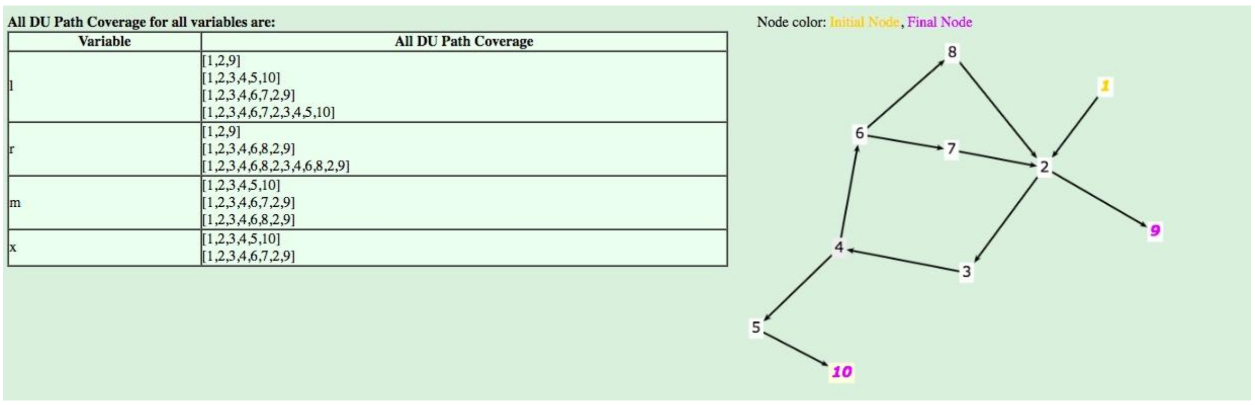

All DU Path Coverage for all variables are:

Variable	All DU Path Coverage
x	[1,2,3,5,6,2,7] [1,2,3,4,6,2,7] [1,2,3,4,6,2,3,4,6,2,7] [1,2,3,4,6,2,3,5,6,2,7]
y	[1,2,3,4,6,2,7] [1,2,3,5,6,2,7] [1,2,3,4,6,2,3,4,6,2,7] [1,2,3,4,6,2,3,5,6,2,7] [1,2,3,5,6,2,3,4,6,2,7] [1,2,3,5,6,2,3,5,6,2,7]
result	[1,2,7] [1,2,3,5,6,2,7] [1,2,3,5,6,2,3,5,6,2,7]

Node color: Initial Node, Final Node

3. Binary Search:

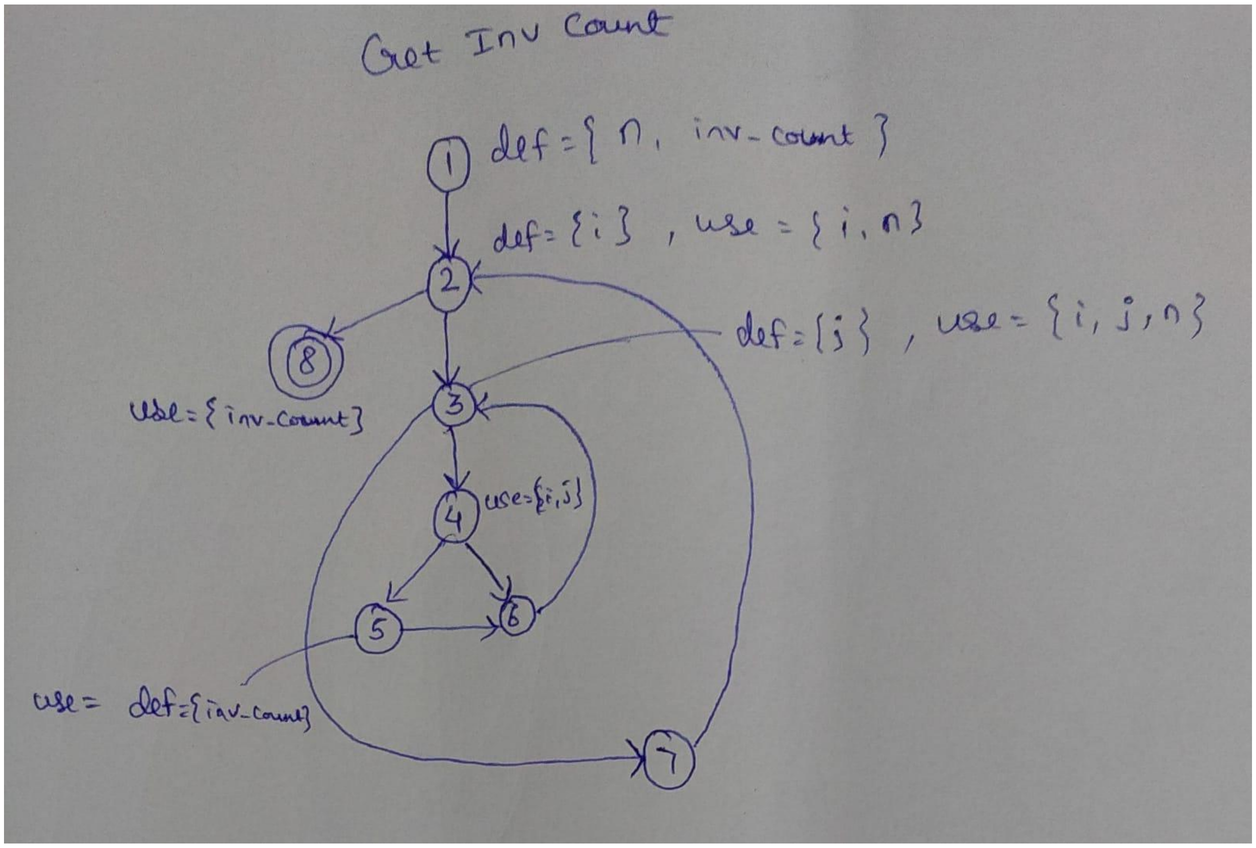


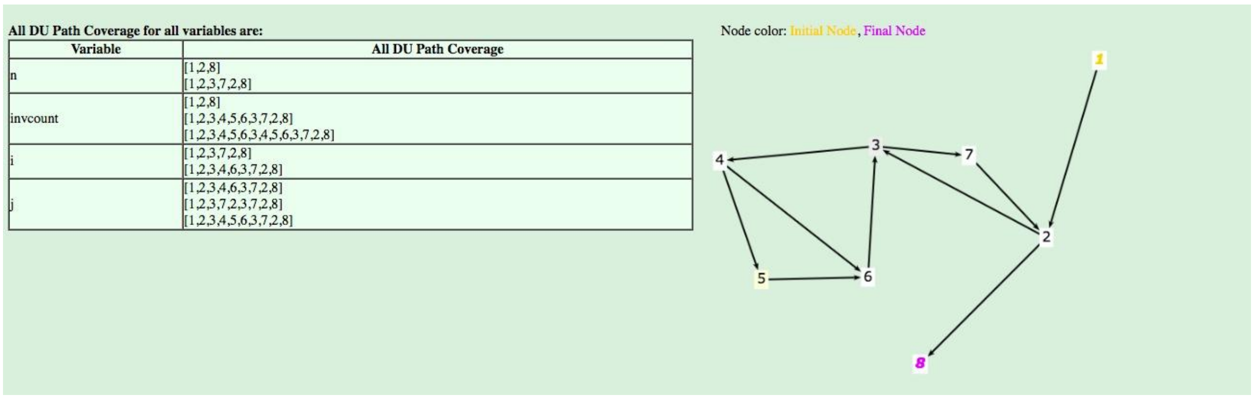

All DU Path Coverage for all variables are:

Variable	All DU Path Coverage
l	[1,2,9] [1,2,3,4,5,10] [1,2,3,4,6,7,2,9] [1,2,3,4,6,7,2,3,4,5,10]
r	[1,2,9] [1,2,3,4,6,8,2,9] [1,2,3,4,6,8,2,3,4,6,8,2,9]
m	[1,2,3,4,5,10] [1,2,3,4,6,7,2,9] [1,2,3,4,6,8,2,9]
x	[1,2,3,4,5,10] [1,2,3,4,6,7,2,9]

Node color: Initial Node, Final Node

4. Get Inverse Count:

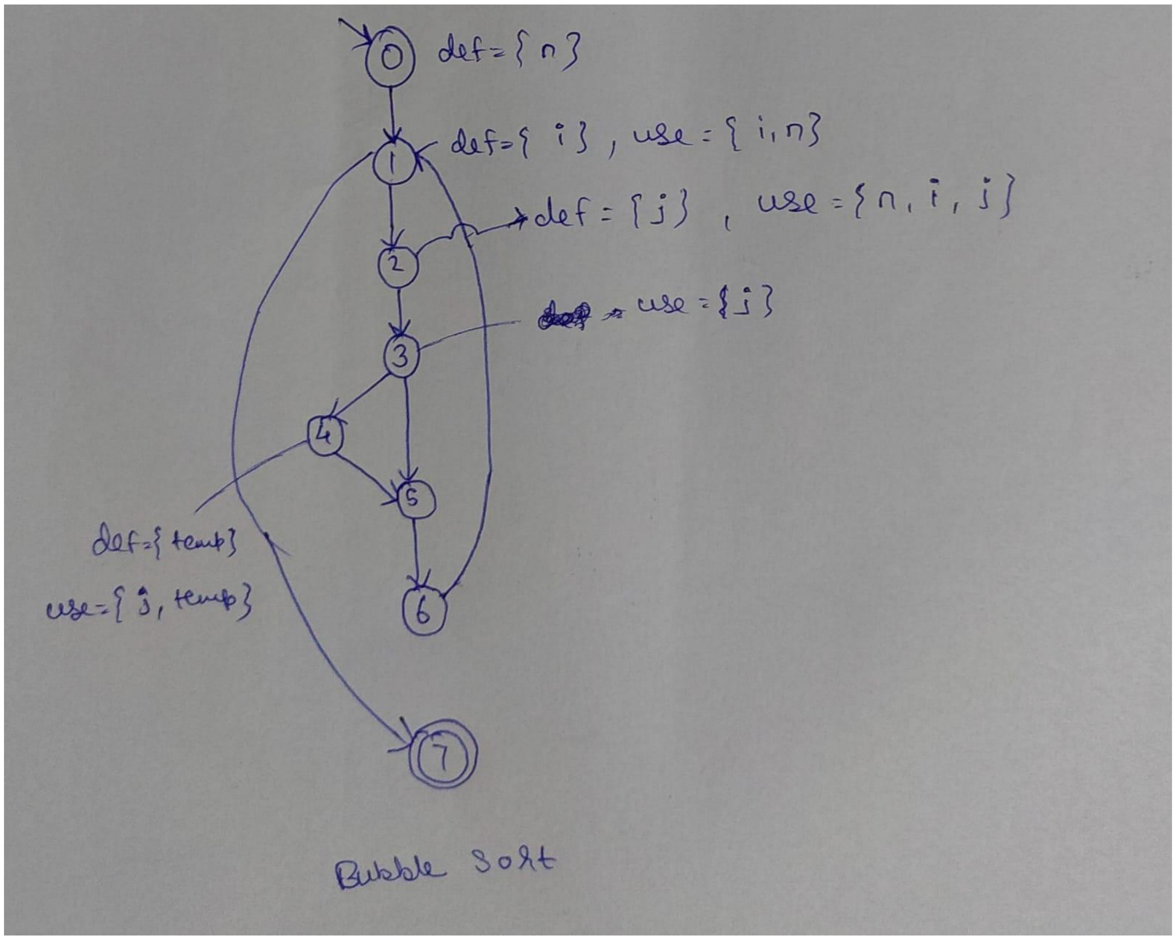


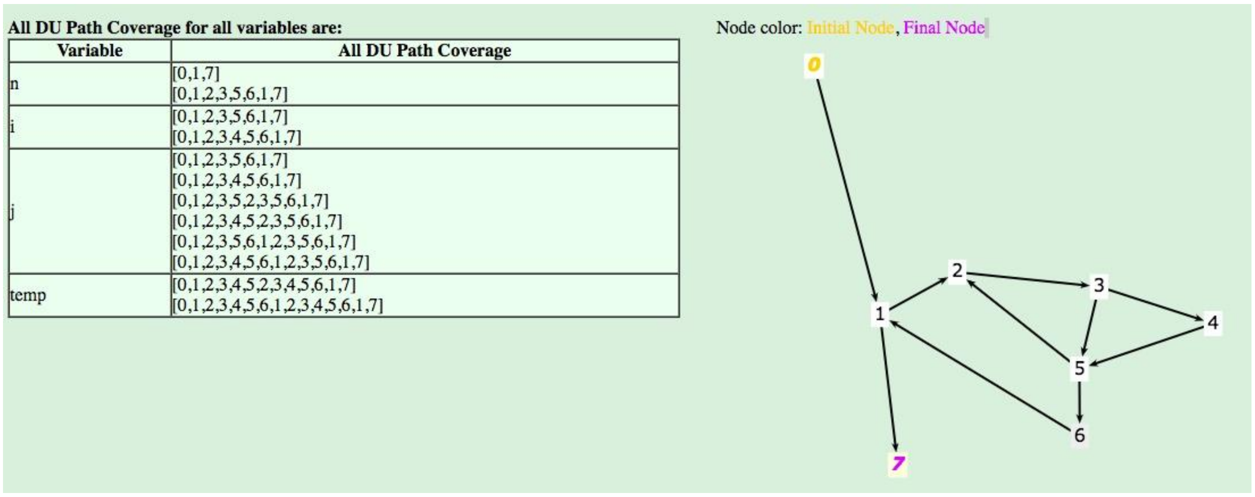

All DU Path Coverage for all variables are:

Variable	All DU Path Coverage
n	[1,2,8] [1,2,3,7,2,8]
invcount	[1,2,8] [1,2,3,4,5,6,3,7,2,8] [1,2,3,4,5,6,3,4,5,6,3,7,2,8]
i	[1,2,3,7,2,8] [1,2,3,4,6,3,7,2,8]
j	[1,2,3,4,6,3,7,2,8] [1,2,3,7,2,3,7,2,8] [1,2,3,4,5,6,3,7,2,8]

Node color: Initial Node, Final Node

5. Bubble Sort:

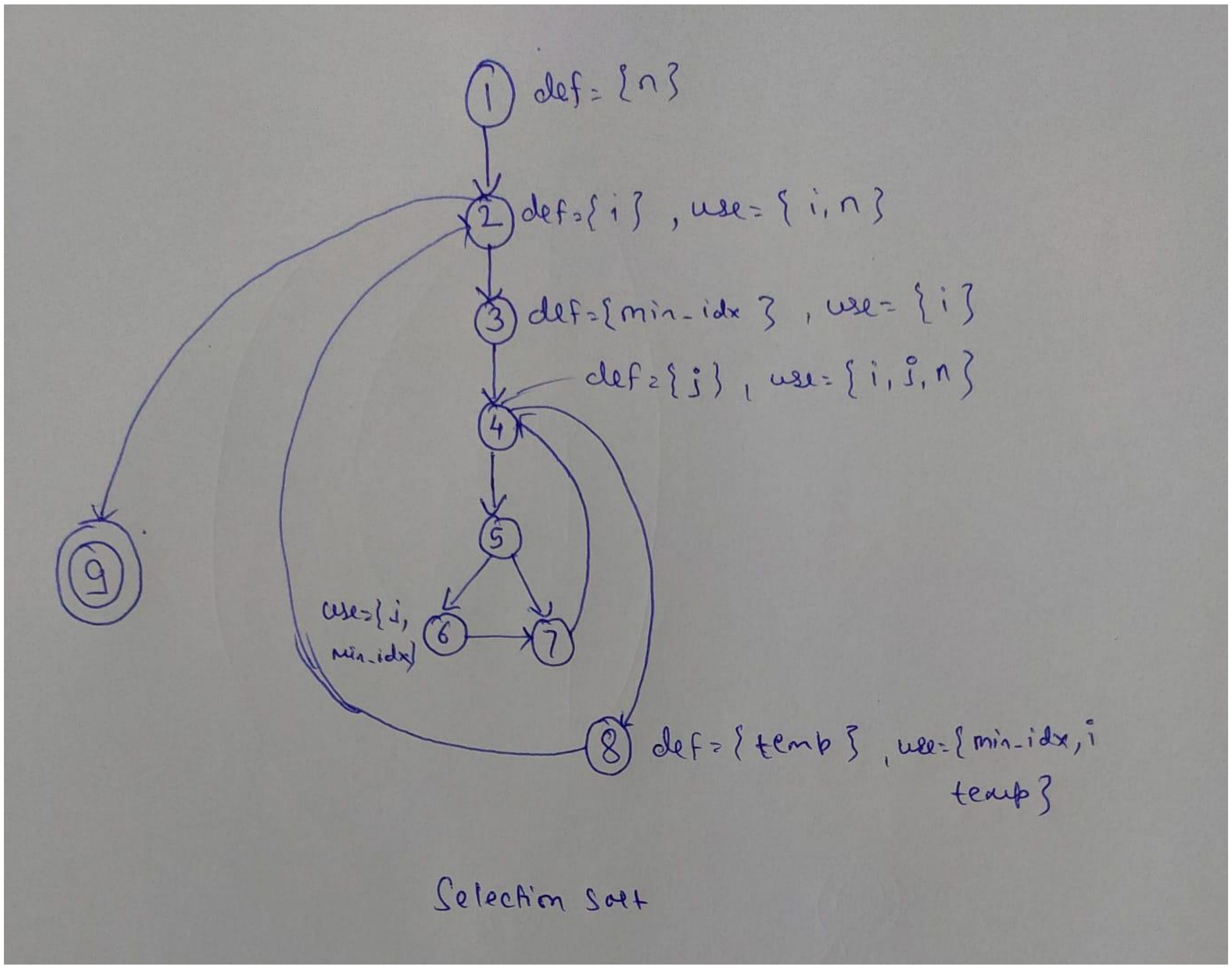




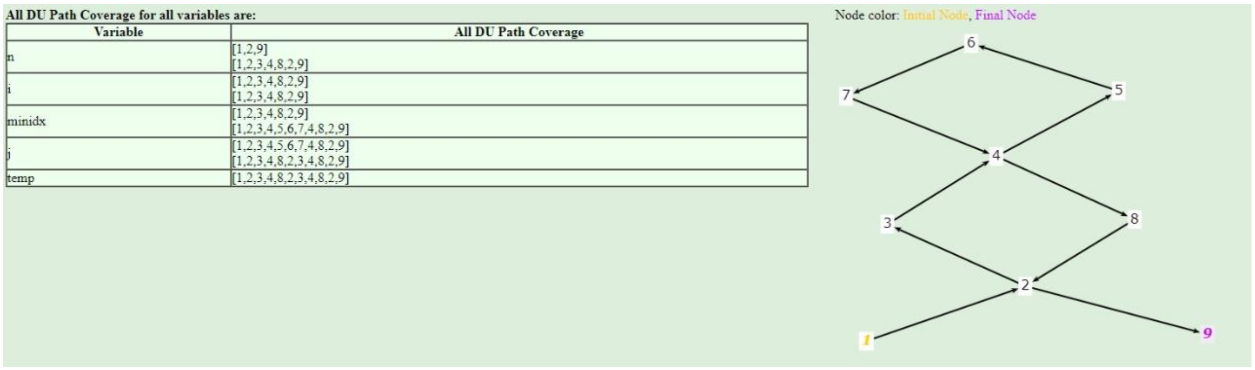
All DU Path Coverage for all variables are:

Variable	All DU Path Coverage
n	[0,1,7] [0,1,2,3,5,6,1,7]
i	[0,1,2,3,5,6,1,7] [0,1,2,3,4,5,6,1,7]
j	[0,1,2,3,5,6,1,7] [0,1,2,3,4,5,6,1,7] [0,1,2,3,5,2,3,5,6,1,7] [0,1,2,3,4,5,2,3,5,6,1,7] [0,1,2,3,5,6,1,2,3,5,6,1,7] [0,1,2,3,4,5,6,1,2,3,5,6,1,7]
temp	[0,1,2,3,4,5,2,3,4,5,6,1,7] [0,1,2,3,4,5,6,1,2,3,4,5,6,1,7]

6. Selection Sort



1 def = {n}
2 def = {i} , use = {i, n}
3 def = {min_idx} , use = {i}
def = {j} , use = {i, j, n}
4
5
use = {j, min_idx}
6
7
8 def = {temp} , use = {min_idx, i temp}
9
Selection sort


All DU Path Coverage for all variables are:

Variable	All DU Path Coverage
n	[1,2,9] [1,2,3,4,8,2,9]
i	[1,2,3,4,8,2,9] [1,2,3,4,8,2,9]
minidx	[1,2,3,4,8,2,9] [1,2,3,4,5,6,7,4,8,2,9]
j	[1,2,3,4,5,6,7,4,8,2,9] [1,2,3,4,8,2,3,4,8,2,9]
temp	[1,2,3,4,8,2,3,4,8,2,9]

Node color: Initial Node, Final Node