Review 11: Week of April 7

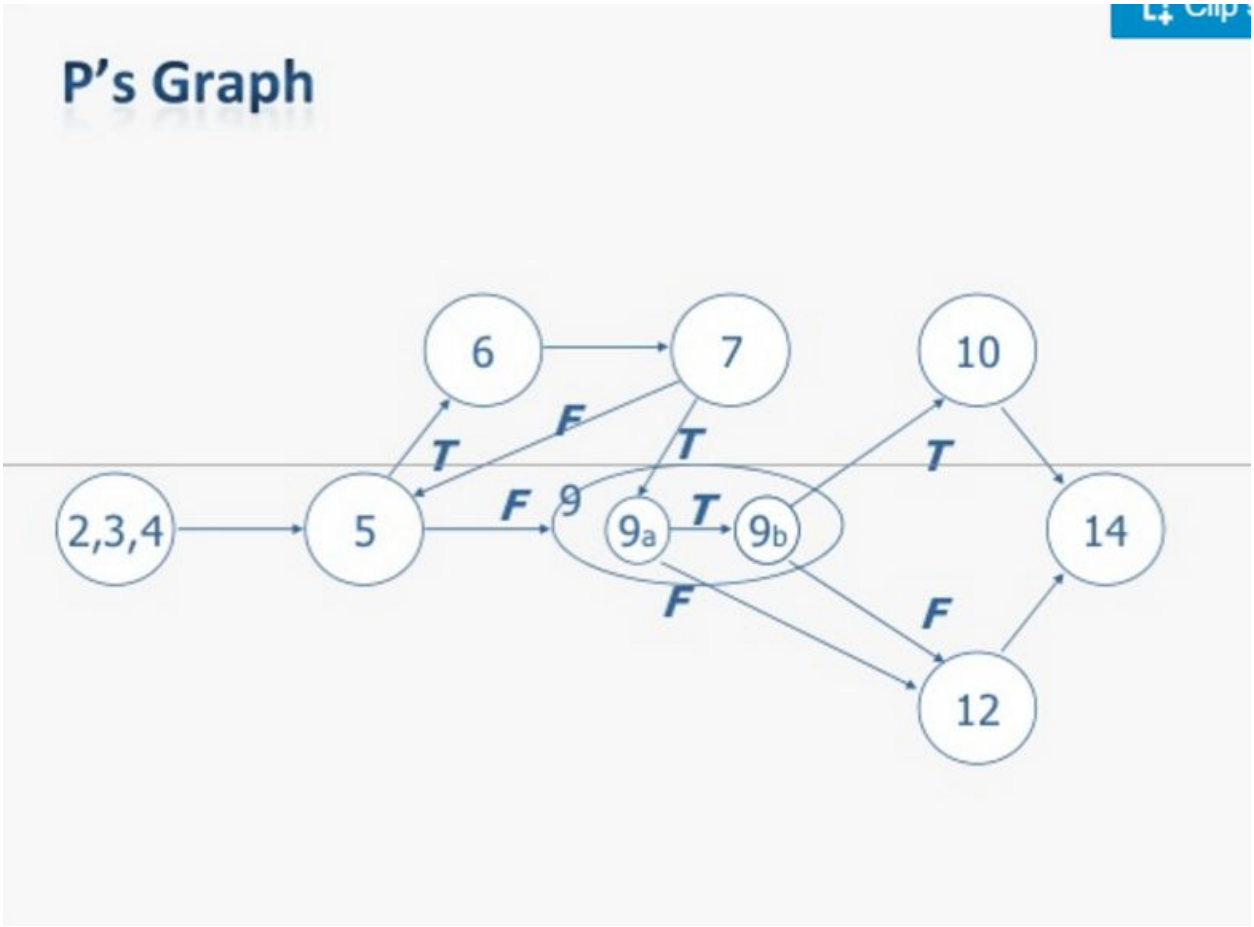
White-box Testing

1. Define the nodes in a control flow graph.
    a. Control Flow: the partial order of statement execution, as defined by the semantics of the language

    Each node in the control flow graph represents a basic block, a straight-line code sequence with no branches in except to the entry and no branches out except at the exit.
    Nodes are blocks of sequential statements.

2. Define the nodes in a data flow graph.
    Data flow: the flow of values from definitions of a variable to its uses
    Each node is a single function or a sub system.
3. Given the following program, draw the control flow graph for it.

```
1   function P return INTEGER is
2   begin
3      X, Y, INTEGER;
4      READ(X); READ(Y);
5      while (X > 10) loop
6         X := X - 10;
7         exit when X = 10;
8      end loop;
9      if (Y < 20 and then X mod 2 = 0) then
10        Y := Y + 20;
11     else
12        Y := Y - 20;
13     end if;
14     return 2*X + Y;
15  end P
```



4. Given the following program, provide tests that gets you greater than 50% and less than 100% statement coverage.

```
1   printSum(int a, int b) {
2       int result = a + b;
3       if (result > 0)
4           printCol("red", result);
5       else if (result < 0)
6           printCol("blue", result);
7       else
8           printCol("green", result);
9   }
```

5. Give an example where 100% statement coverage is required?

A == -5
B == -8
Printsum(int a ,int b){

```
In result = a+b;
If (result > 0)
        Printcol (“red”, result);
Else if (result < 0)
        Printcol (“blue”, result);
}
```

6. In the above program, what statement coverage is achieved from each of the following test cases:

- a = 2, b = 5
- a = -3, b = -6
- Both of the above.

(1). 7/9
(2). 7/9
(3). 11/11

7. Define branch coverage.

Test coverage criteria requires enough test cases such that each condition in a decision takes on all possible outcomes at least once, and each point of entry to a program or subroutine is invoked at least once. That is, every branch (decision) taken each way, true and false.

8. '100% Branch coverage ensures 100% statement coverage' - discuss.

Statement coverage
In order to execute every statement we need only one testcase which would set all conditions to true, every line of a code (statement) is touched.

shouldReturnInput(x, true, true, true) - 100% statement covered

But only half of branches are covered and only one path.

Branch coverage
You can visualize every "if-statment" as two branches (true-branch and false-branch). So it can clearly be seen that the above testcase follows only "true-branches" of every "if-statement". Only 50% of branches are covered.

In order to cover 100% of branches we would need to add following testcase:
shouldReturnInput(x, false, false, false)

With these two testcases we have 100% statements covered, 100% branches covered

9. Define semantic coverage. What kind of testing methods can be used to achieve semantic coverage?

Suppose you had some semantic property like "when re-entering from orbit, ensure the door is closed". simple code coverage measures would not address this property

can you think of other ways we might explore code looking for (e.g.) the above conjunction?