# EXERCISE #6

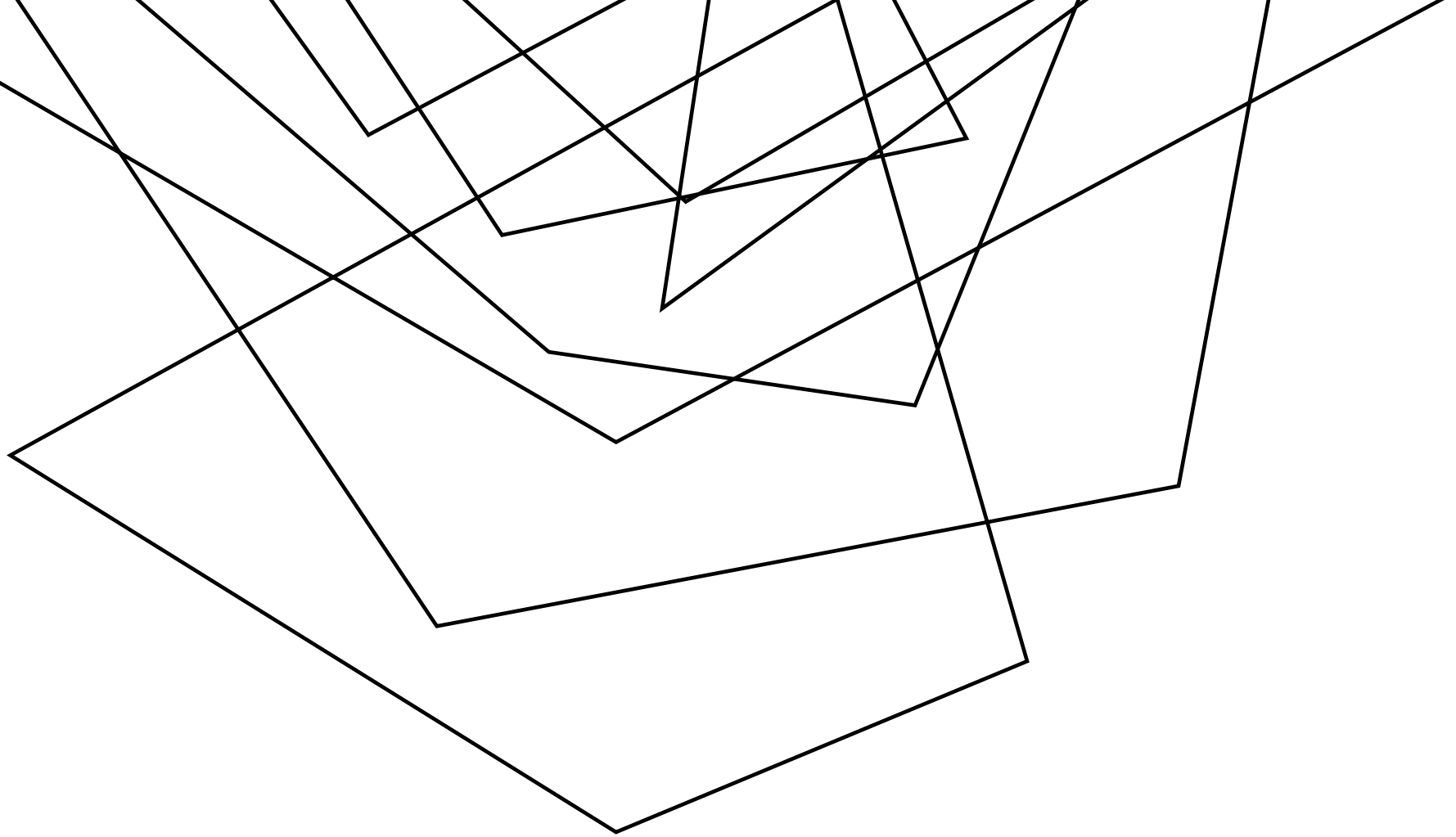
*DATAFLOW REVIEW*

## Write your name and answer the following on a piece of paper

- Show the control-flow graph of the following function and indicate the value-sets at the beginning and end of each basic block according to the method discussed in class

```
y = 0;
if ( g ) {
  x = 1;
  x += y;
} else {
  x = 3;
  if (g2) {
    x = y;
  }
  x = 4;
}
z = x;
```

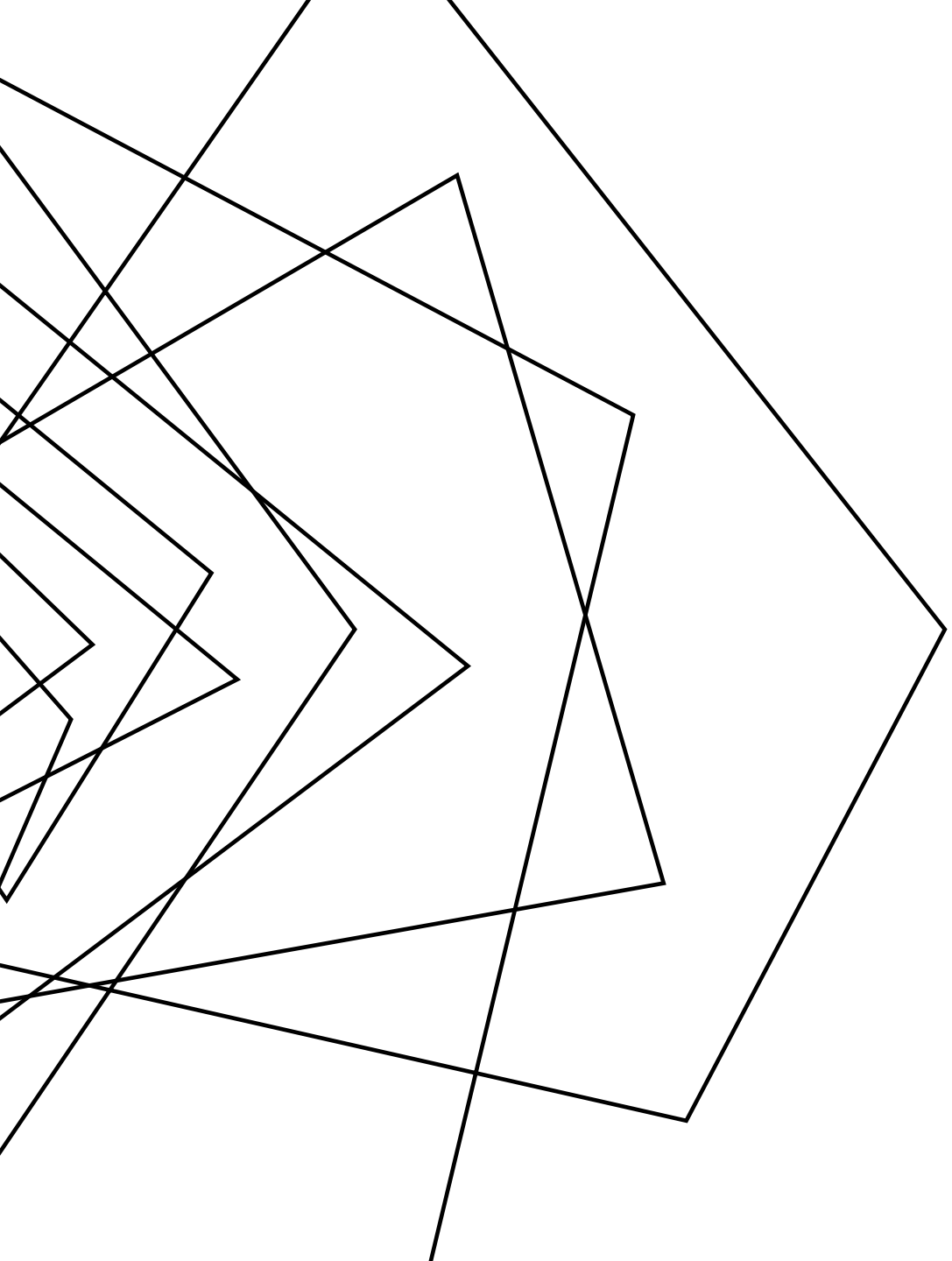
# ADMINISTRIVIA AND ANNOUNCEMENTS



# DATAFLOW FIXPOINTS

EECS 677: Software Security Evaluation

Drew Davidson



# CLASS PROGRESS

EXPLORING A FORM OF STATIC ANALYSIS THAT SUMMARIZES HOW CONTROL AND DATA FLOWS ACROSS A PROGRAM

- MANIFEST A COMPLETE ANALYSIS BY DENOTING SETS OF ALL VALUES MEMORY MIGHT CONTAIN (**NB – THIS WILL END UP BEING CUMBERSOME!**)

# LAST TIME: CONTROL-FLOW GRAPHS

## REVIEW: STATIC ANALYSIS

### Set rules for Partitioning Functions into Basic Blocks

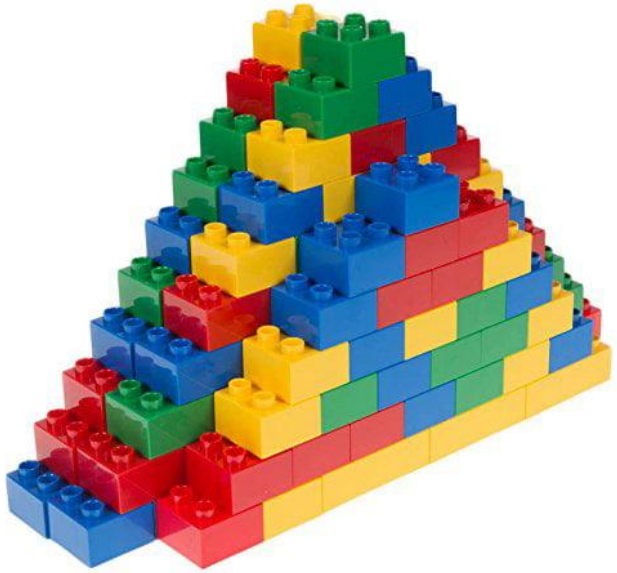
- Leader instructions
- Terminator instructions

### Connected the Basic Blocks into a Control-Flow Graph

- Edges indicate control flow transfers

### Explored the "Big Idea" of Dataflow Anlaysis

- Treat each instruction as a transfer function
- Compose transfer functions to model blocks data flow
- Merge block effects to get function's data flow



***Building something bigger out of basic blocks***

# LAST TIME: FLOW SENSITIVE VALUE SETS

## REVIEW: DATAFLOW

### ACCOUNT FOR PROGRAM FLOW, NOT PATHS

- Necessarily Over-approximating states

### LET'S DO AN EXAMPLE

```
1. int x = 0;
2. int y = 0;
3. if (g == true){
4.   x = 10;
5. }
6. if (g == false){
7.   y = 1 / (x - 10);
8. }
9. return;
```

# LOOPS ARE TOUGH TO HANDLE!

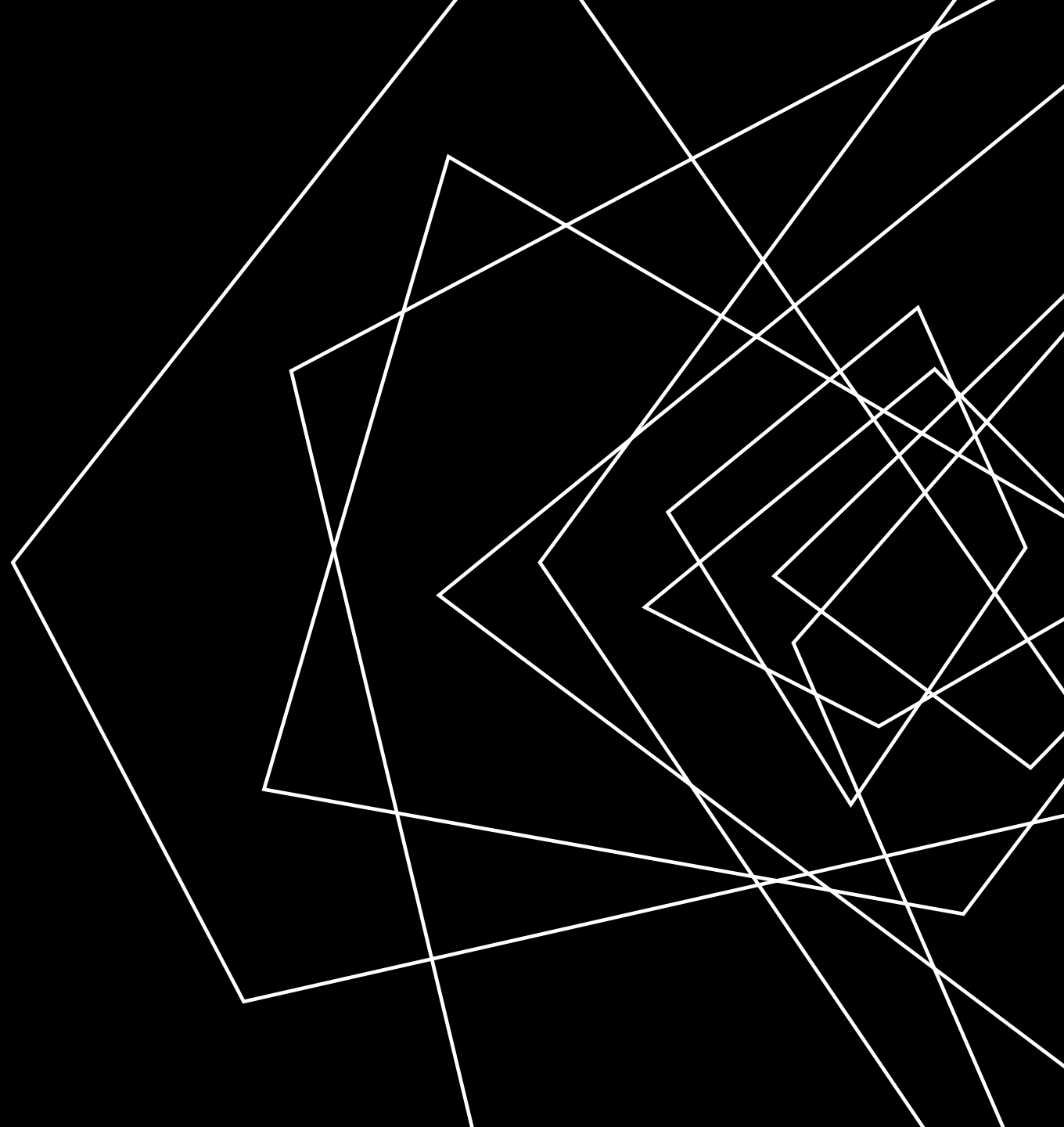
## REVIEW: DATAFLOW ANALYSIS

### Issues with Loops

- Generate lots of paths
- Cyclic data dependency

# LECTURE OUTLINE

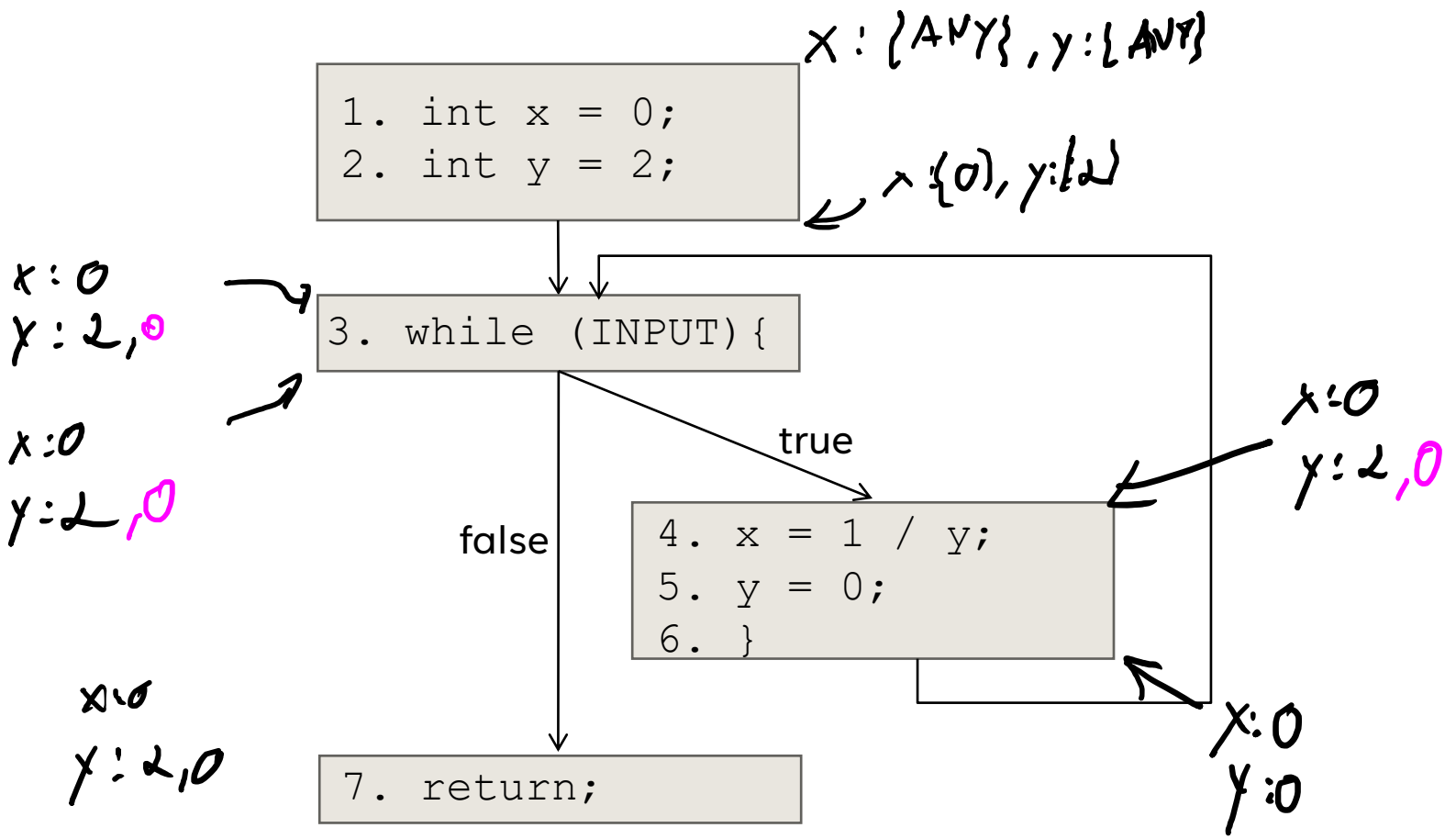
- Breaking cyclic dependency
- Termination
- Handling large value sets



# WHERE TO START ANALYSIS?

## BREAKING CYCLIC DEPENDENCY

```
1. int x = 0;
2. int y = 2;
3. while (INPUT){
4.    x = 1 / y;
5.    y = 0;
6. }
7. return;
```



# CHAOTIC ITERATION

## STATIC ANALYSIS: CONTROL FLOW GRAPHS

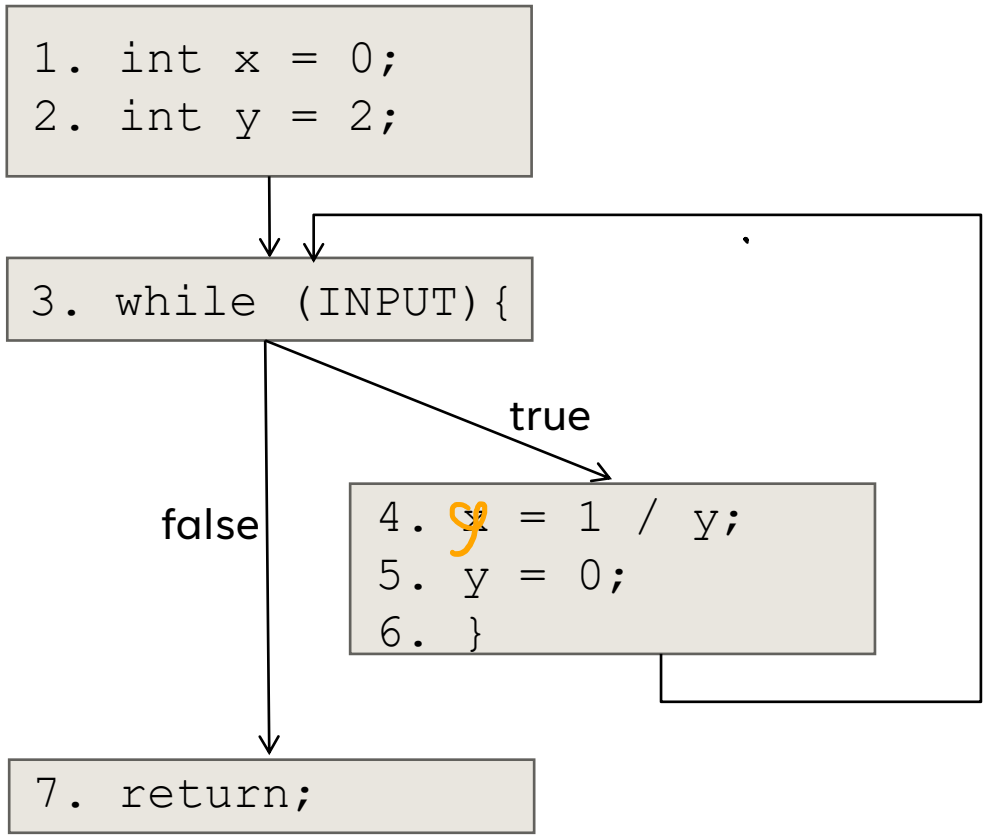
### A WORKLIST ALGORITHM

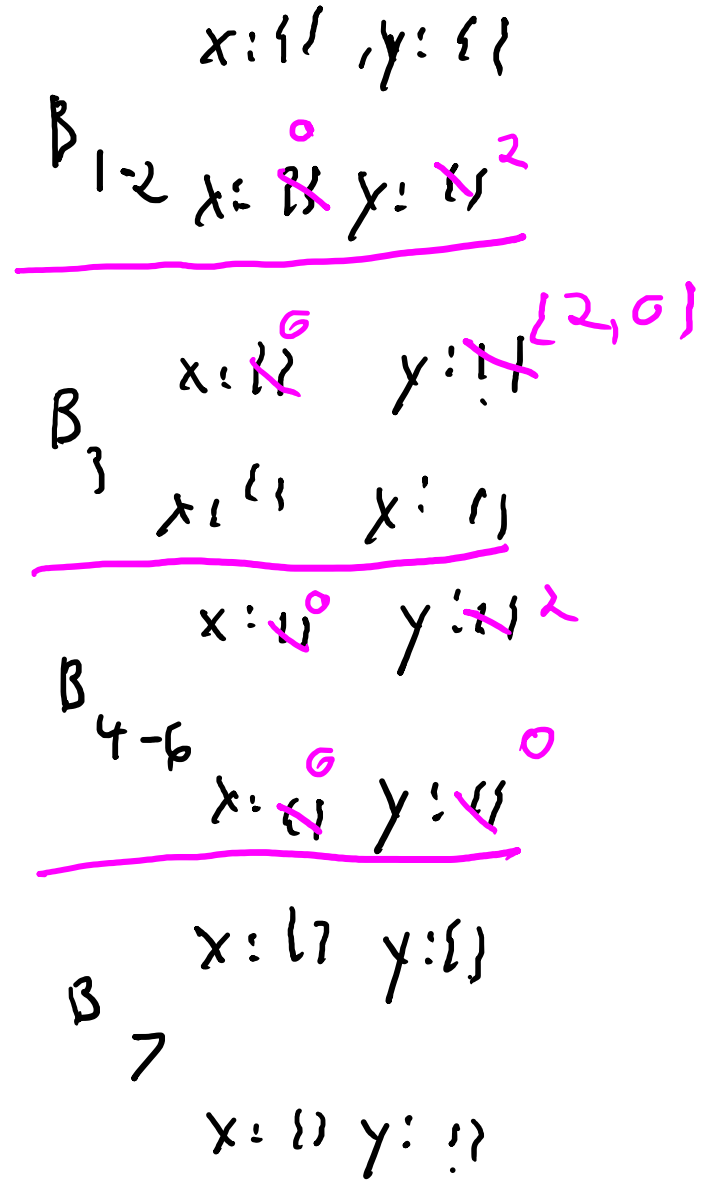
- Select the next worklist item in any order
- Necessarily assumes progress towards some goal


***Surprisingly, not a band with merch at Hot Topic***

# WHERE TO START ANALYSIS?

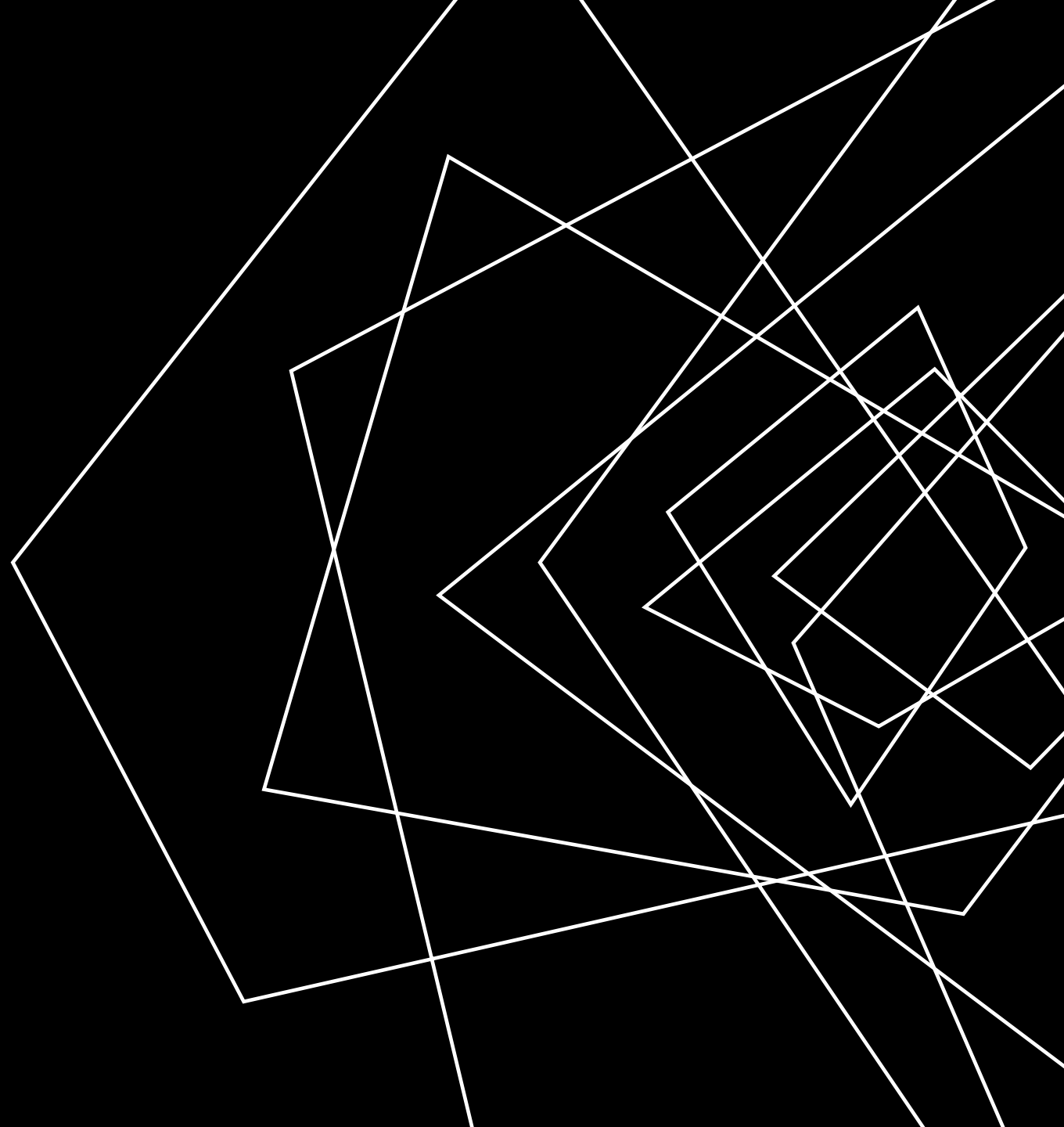
## STATIC ANALYSIS: CONTROL FLOW GRAPHS



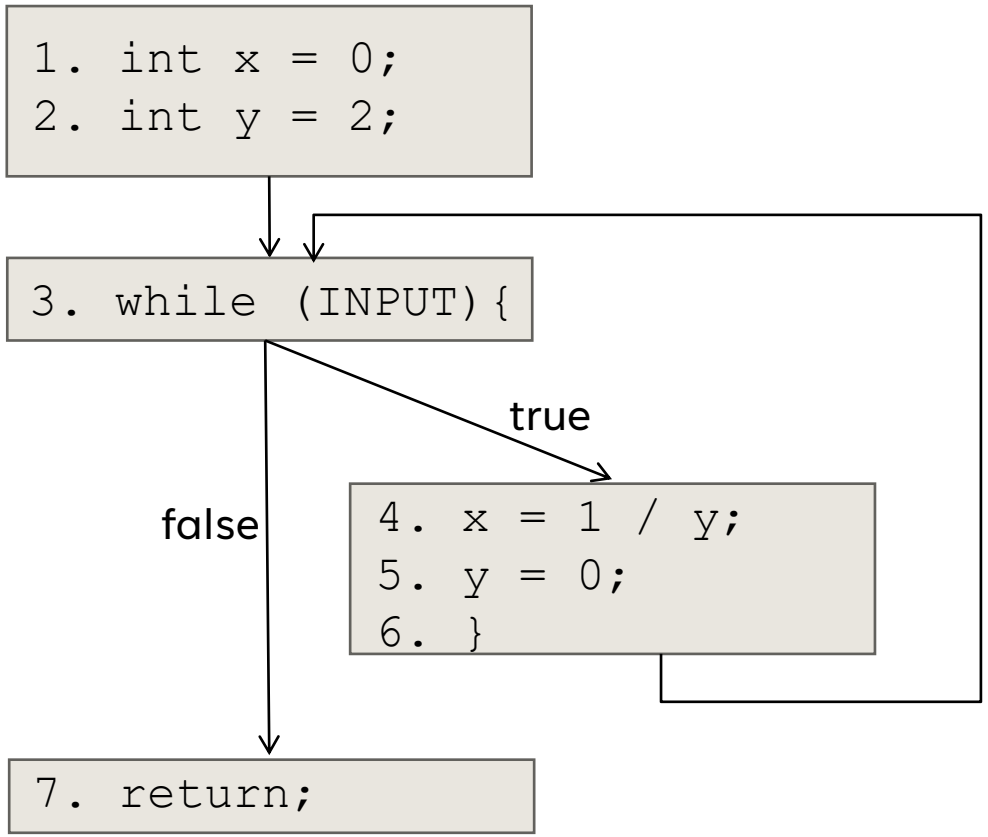
# LECTURE OUTLINE

- Breaking cyclic dependency
- Termination
- Handling large value sets



# WHERE TO STOP ANALYSIS?

## STATIC ANALYSIS: CONTROL FLOW GRAPHS



# ANALYSIS PROGRESS

## STATIC ANALYSIS: CONTROL FLOW GRAPHS

### ANALYSIS ENDS WHEN THE FACT SETS REACH ***SATURATION***

- No additional elements will ever be added
- It sure would be nice if we could guarantee that this will happen!


***When your fact sets couldn't possibly hold any more data***

# FIXED-POINTS

## STATIC ANALYSIS: CONTROL FLOW GRAPHS

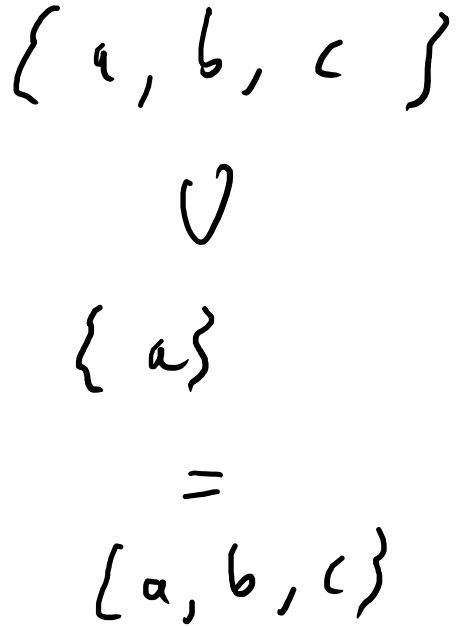
### A Fixed-Point (AKA fixpoint, fixed point)

- A value that does not change under a given transformation

### Our value-set analysis WILL have Facts that reach a Fixed-point

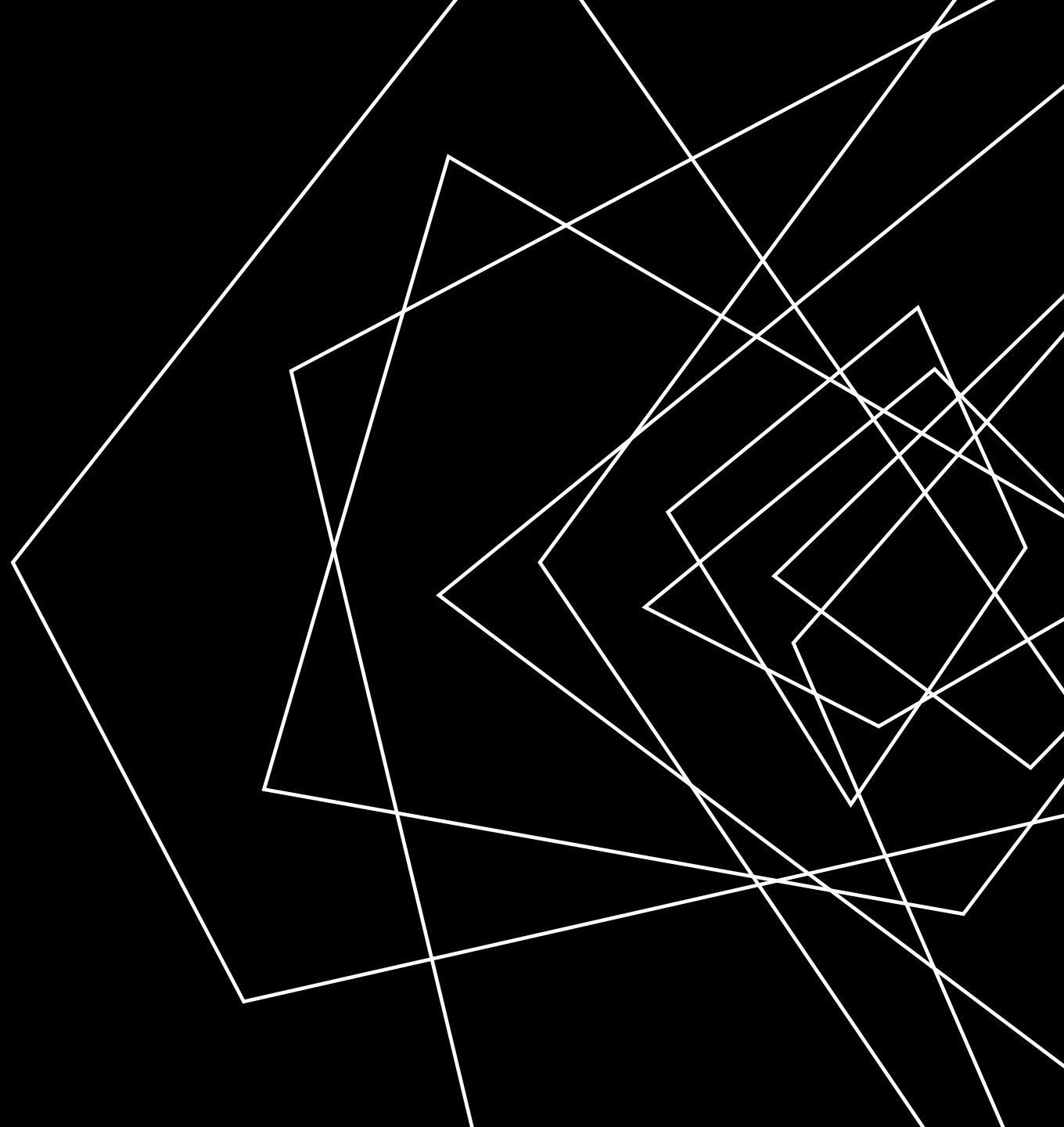
Why?

- Finite set of configurations over INT32s
- Data transforms only **add** data to fact sets



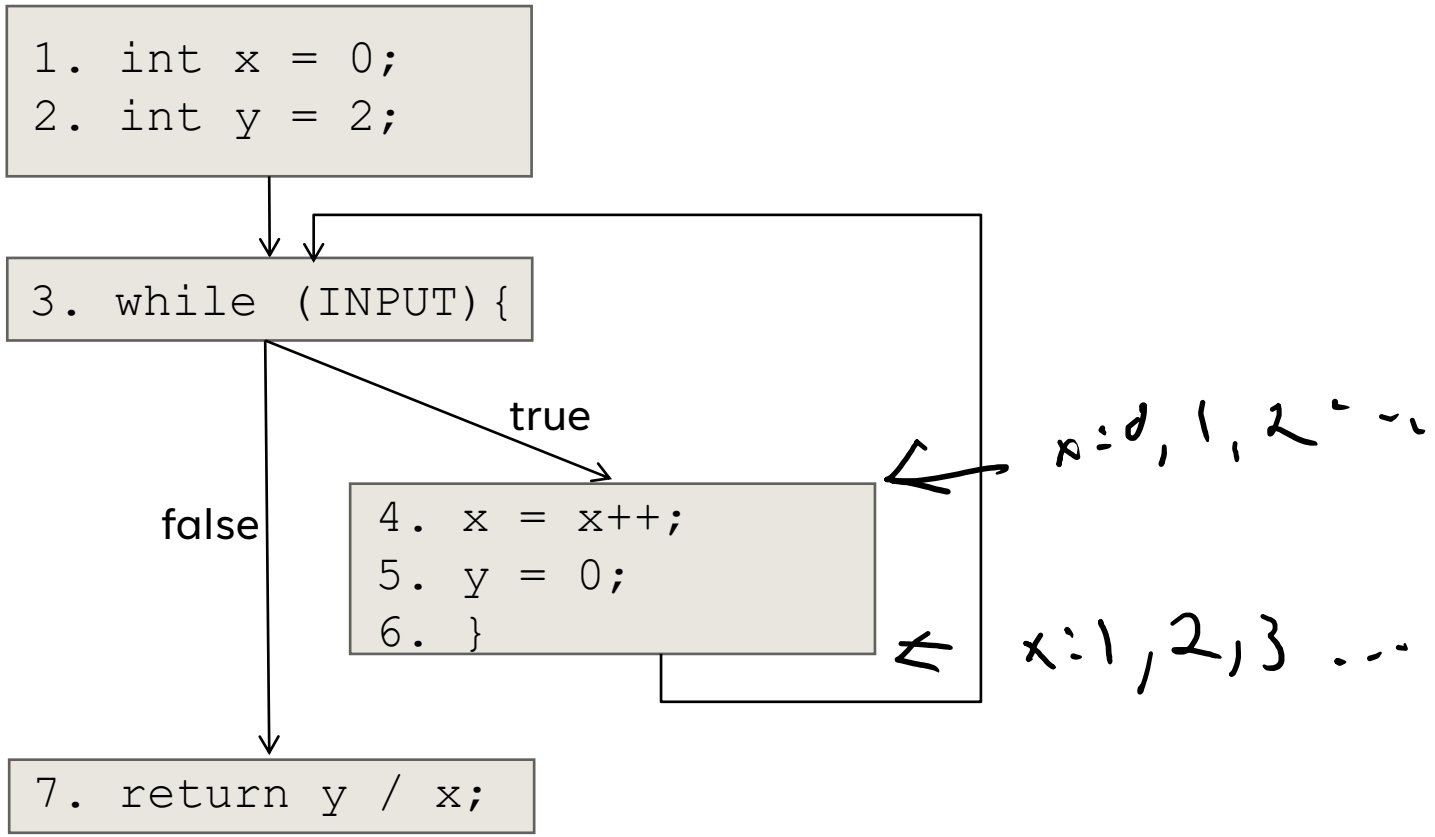
# LECTURE OUTLINE

- Breaking cyclic dependency
- Termination
- Handling large value sets

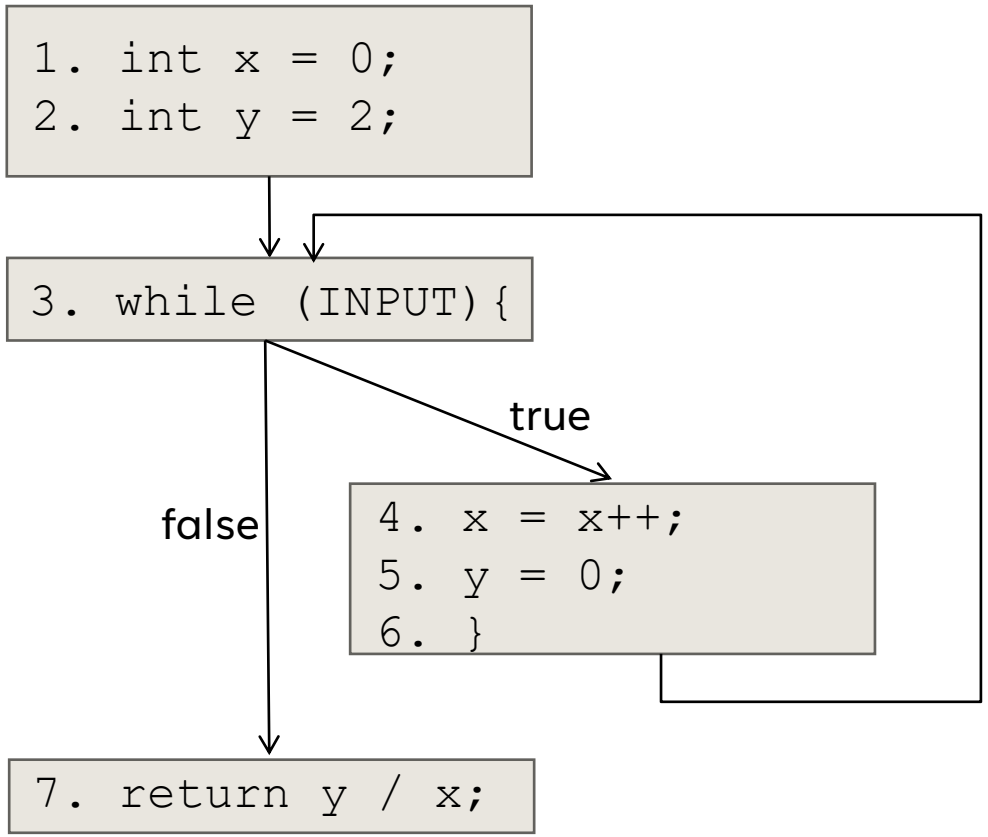


# WHERE TO STOP THIS ANALYSIS?

## ANALYSIS TERMINATION



# WIDENING
## ANALYSIS TERMINATION

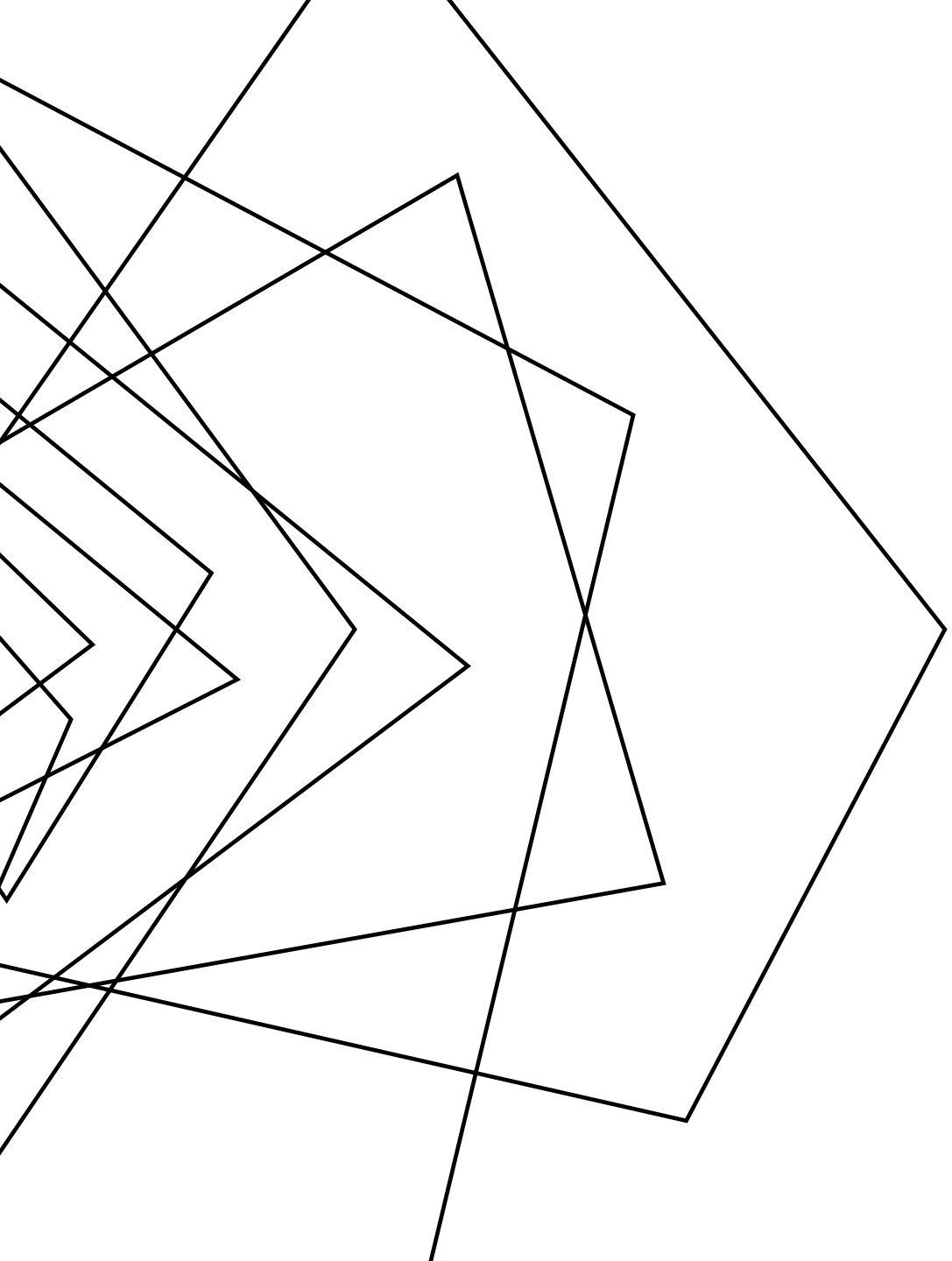



### *Accelerate progress towards Fix-Point*

- Lots of (over-approximate) ways to do this
- 1 simple idea: if we hit N values, immediately change the fact set to “All integers”

x: 0, 1, 2, 3 ⇒ x : ANY

x: 1, 2, ⇒ x : ANY



# LECTURE END!

*Described some of the issues and Fixes for dataflow in the presence of loops*