Quiz 1

- You can find your grade for Quiz 1 at `https://www.eecs.yorku.ca/~roumani/ePost/server/ep.cgi?year=2018-19&term=W&course=4315`.
- You received an email with feedback.

Space Exploration

EECS 4315

wiki.eecs.yorku.ca/course/4315/

Nondeterminism

Nondeterministic code is code that, even for the same input, can exhibit different behaviours on different runs, as opposed to deterministic code.

- Randomization and
- concurrency

both give rise to nondeterminism.

Testing nondeterministic code

```
public class RandomFraction {
  public static void run() {
    Random random = new Random(System.currentTimeMillis());
    System.out.print(1 / random.nextInt(1000000));
  }
}
```

Question

If we run the above app 1,000,000 times, what is the probability that it does not throw an exception in any of those runs?

Testing nondeterministic code

Answer

- The probability of choosing zero is

Testing nondeterministic code

Answer

- The probability of choosing zero is $\frac{1}{1{,}000{,}000}$.
- The probability of not choosing zero is

Testing nondeterministic code

Answer

- The probability of choosing zero is $\frac{1}{1{,}000{,}000}$.
- The probability of not choosing zero is $1 - \frac{1}{1{,}000{,}000} = \frac{999{,}999}{1{,}000{,}000}$.
- The probability of not choosing zero one million times in a row is

Testing nondeterministic code

Answer

- The probability of choosing zero is $\frac{1}{1{,}000{,}000}$.
- The probability of not choosing zero is $1 - \frac{1}{1{,}000{,}000} = \frac{999{,}999}{1{,}000{,}000}$.
- The probability of not choosing zero one million times in a row is $(\frac{999{,}999}{1{,}000{,}000})^{1{,}000{,}000} \approx 0.37$.

Limitations of testing

Limitations of testing of nondeterministic code include

- no guarantee that all different behaviours have been checked, and
- errors may be difficult to reproduce.

Alternatives to testing

To detect bugs in nondeterministic code, testing needs to be supplemented with other approaches.

Question

How to tackle the limitations of testing of nondeterministic code?

Alternatives to testing

To detect bugs in nondeterministic code, testing needs to be supplemented with other approaches.

Question

How to tackle the limitations of testing of nondeterministic code?

Answer

Control the nondeterminism: this allows us to

- systematically check all different behaviours and
- reproduce errors.

Exercises

Solve the following exercises. Fill in the body of the following main method.

```
public class Exercise {
  public static void main(String[] args) {
    Random random = new Random();
  }
}
```

using only the `nextBoolean` method of the `Random` class.

Exercises

1. The app prints either 1 or 2, both with probability 0.5.
2. The app prints 1, 2, 3, or 4, each with probability 0.25.
3. The app prints any integer, each with positive but not necessarily equal probability.

Executions

- The first app has two different executions.
- The second app has four different executions

Question

How many different executions has the third application?

Executions

- The first app has two different executions.
- The second app has four different executions

Question

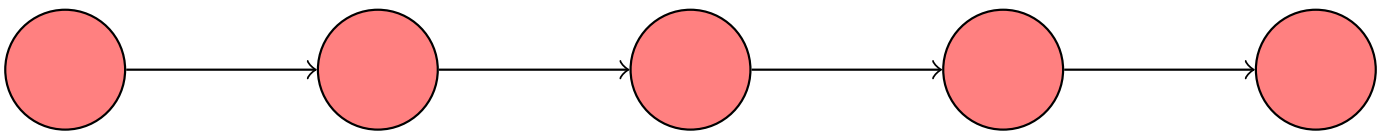
How many different executions has the third application?

Answer

$2^{32} = 4,294,967,296.$

Execution

An execution consists of states connected by transitions.



State

A state of a Java virtual machine (JVM) includes

- the heap,
- for each thread
  - its state (runnable, waiting, terminated, . . .),
  - its stack,
  - etc,
- etc.

https://docs.oracle.com/javase/8/docs/platform/jvmti/jvmti.html

Transition

A transition of a JVM takes the JVM from one state to another by executing a bytecode instruction.

Java code

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

Java bytecode

The command

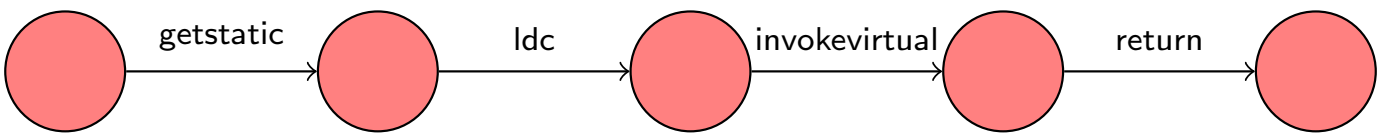
```
javap -c HelloWorld.class
```

produces

```
0: getstatic
// of attribute System.out of class PrintStream
3: ldc
// String "Hello World"
5: invokevirtual
// of method println with argument String
8: return
```

Java code and execution

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```



Java code

```
public class RedOrGreen {
  public static void main(String[] args) {
    Random random = new Random();
    if (random.nextBoolean()) {
      System.out.println("Red");
    } else {
      System.out.println("Green");
    }
  }
}
```

Java bytecode

```
0: new
3: dup
4: invokespecial
7: astore_1
8: aload_1
9: invokevirtual
12: ifeq
15: getstatic
18: ldc
20: invokevirtual
23: goto
26: getstatic
29: ldc
31: invokevirtual
34: return
```

Executions

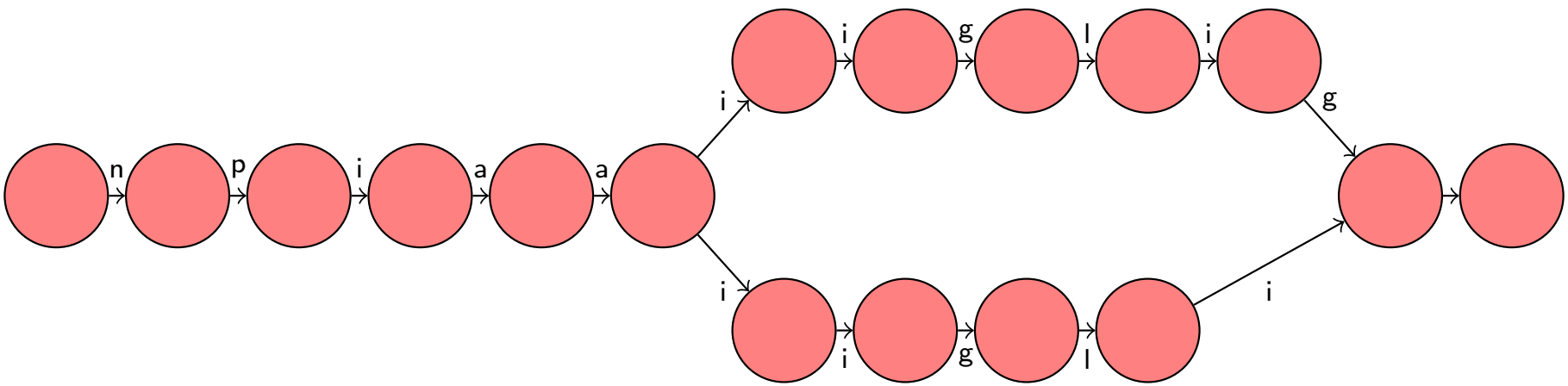
Question

Draw the state-transition diagram.

Executions

Question

Draw the state-transition diagram.



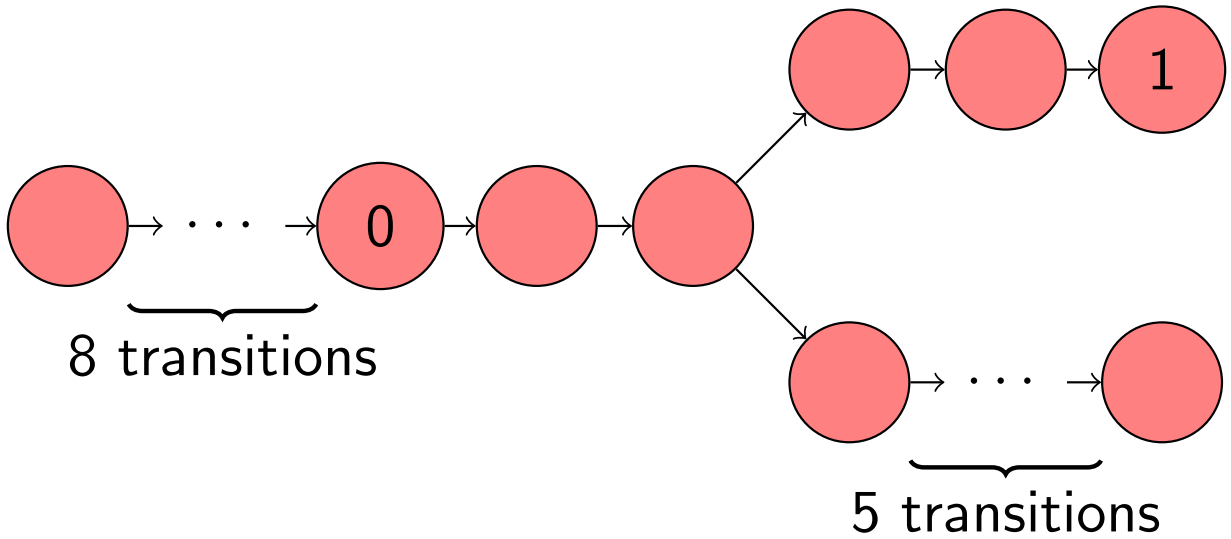
Executions

Question

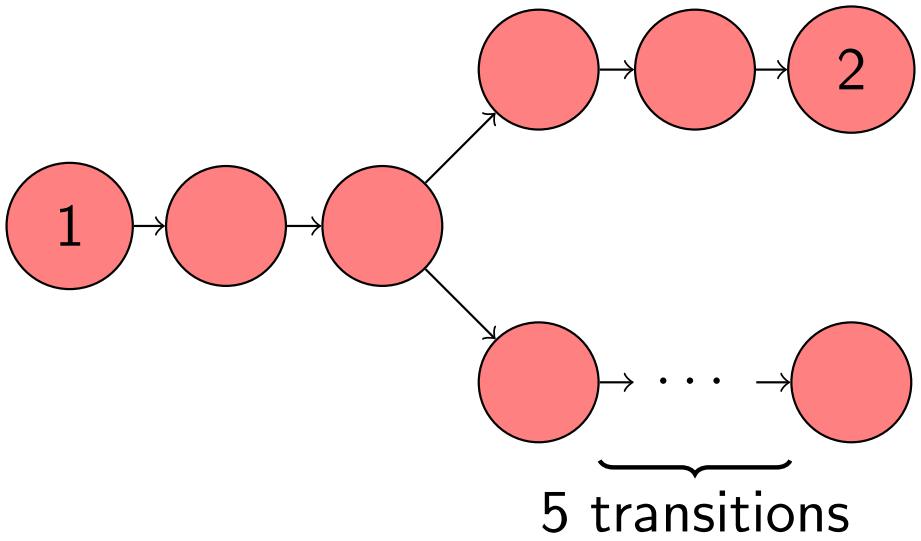
Draw the state-transition diagram corresponding to

```
Random random = new Random();
int value = 0;
while (random.nextBoolean()) {
  value++;
}
System.out.println(value);
```

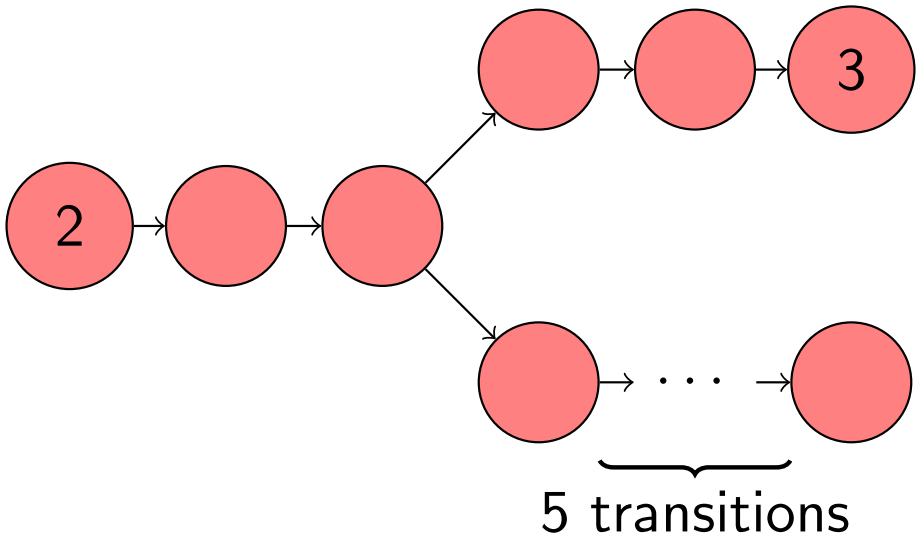
Executions



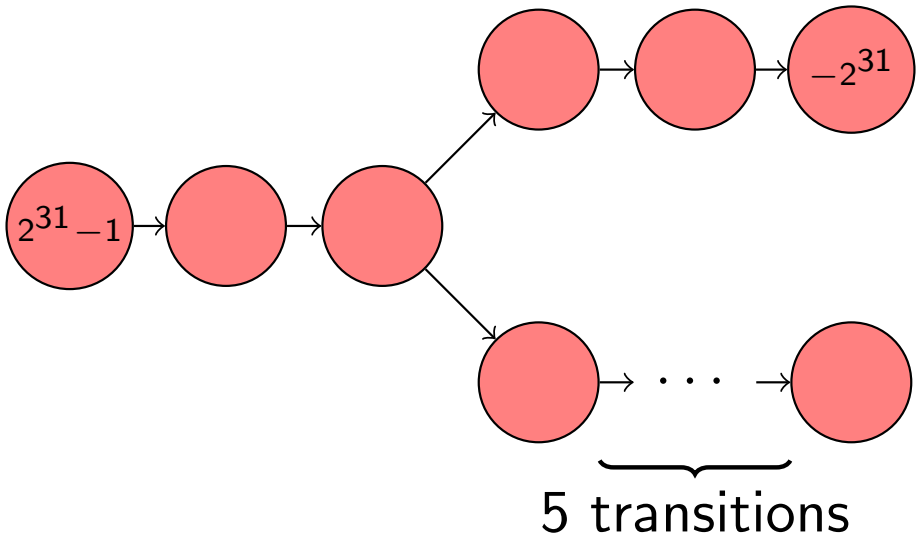
Executions



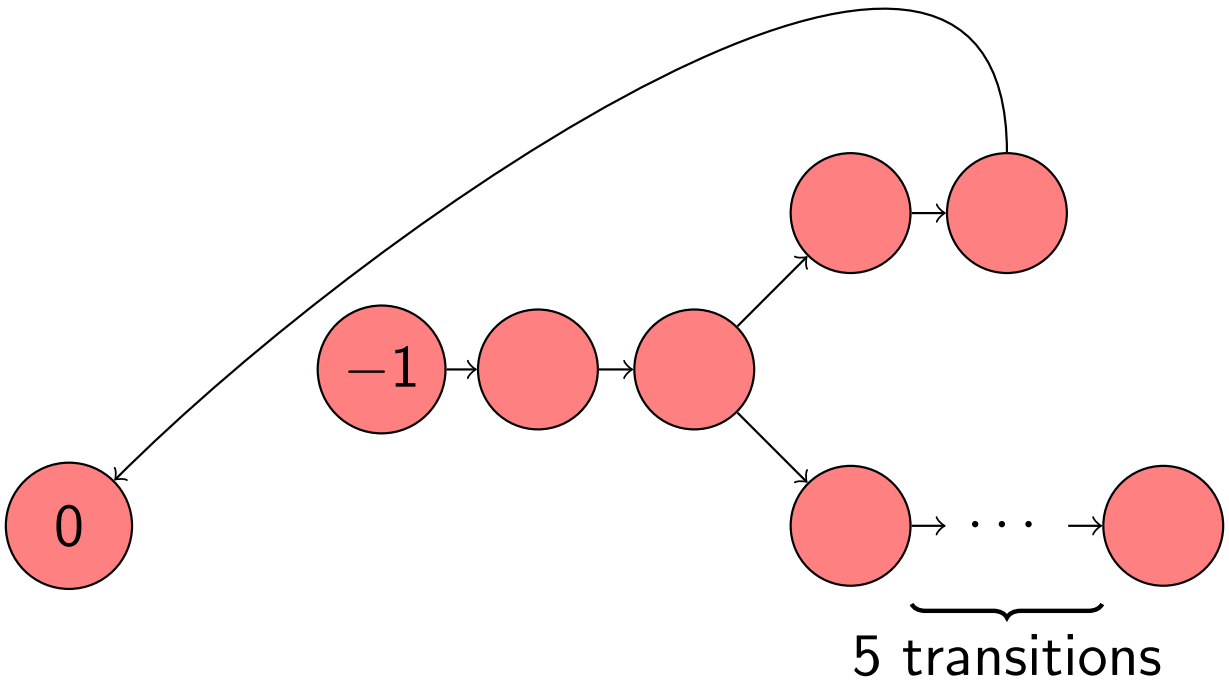
Executions



Executions



Executions



The state space explosion problem

Problem

The size of the state space, that is, the number of states, may become very large.

The state space explosion problem

Problem

The size of the state space, that is, the number of states, may become very large.

This is one of the major challenges in model checking.