Tracing Recursion

Given a piece of recursive code, answer the following questions:

- What is the return result?
- What is the maximum depth of the call stack?
- How many times was the recursive method called?

To date we have traced code with memory diagrams

These become too unwieldy with multiple calls

Tracing recursion (cont)

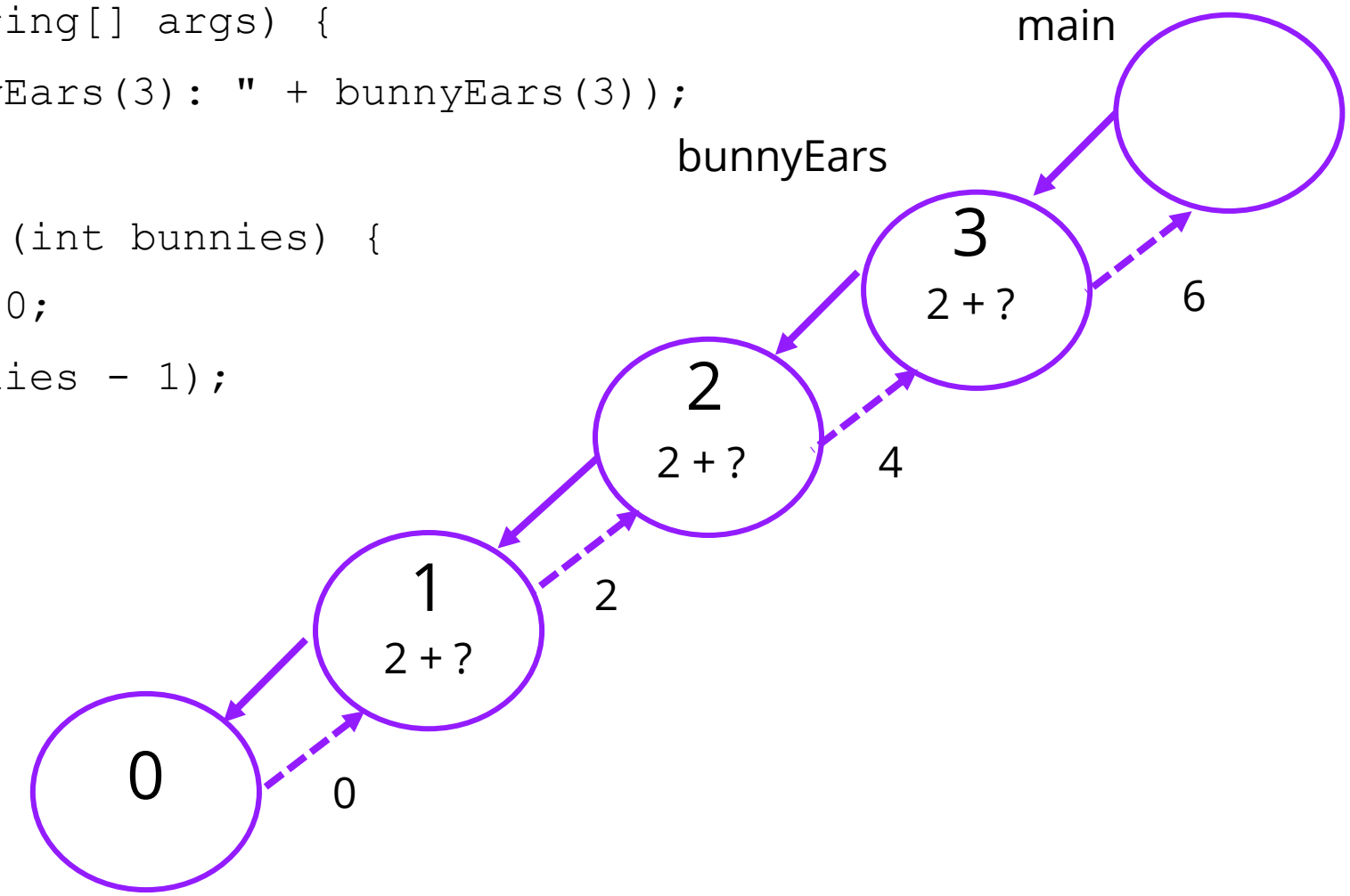
Characteristics of tracing mechanism

- Visible representation of each call
- Tracking of arguments for each call
- Tracking of return value for each call

Use a circle for each dynamic call with arguments
Small computation inside if desired
Arrow for new call with solid line
Arrow for return with value and dotted line

Example 1

```
public static void main(String[] args) {
  System.out.println("bunnyEars(3): " + bunnyEars(3));
}
public static int bunnyEars(int bunnies) {
  if (bunnies == 0) return 0;
  return 2 + bunnyEars(bunnies - 1);
}
```



-What is the return result?

6

-What is the maximum depth of the call stack of recursive calls?

4

-How many times was the recursive method called?

4

Example 2

```
public static void main(String[] args) {
  System.out.println("fibonacci(4): " + fibonacci(4));
}
public static int fibonacci(int n) {
  if (n == 0)return 0;
  if (n == 1)return 1;
  return fibonacci(n - 1) + fibonacci(n - 2);
}
```

-What is the return result?

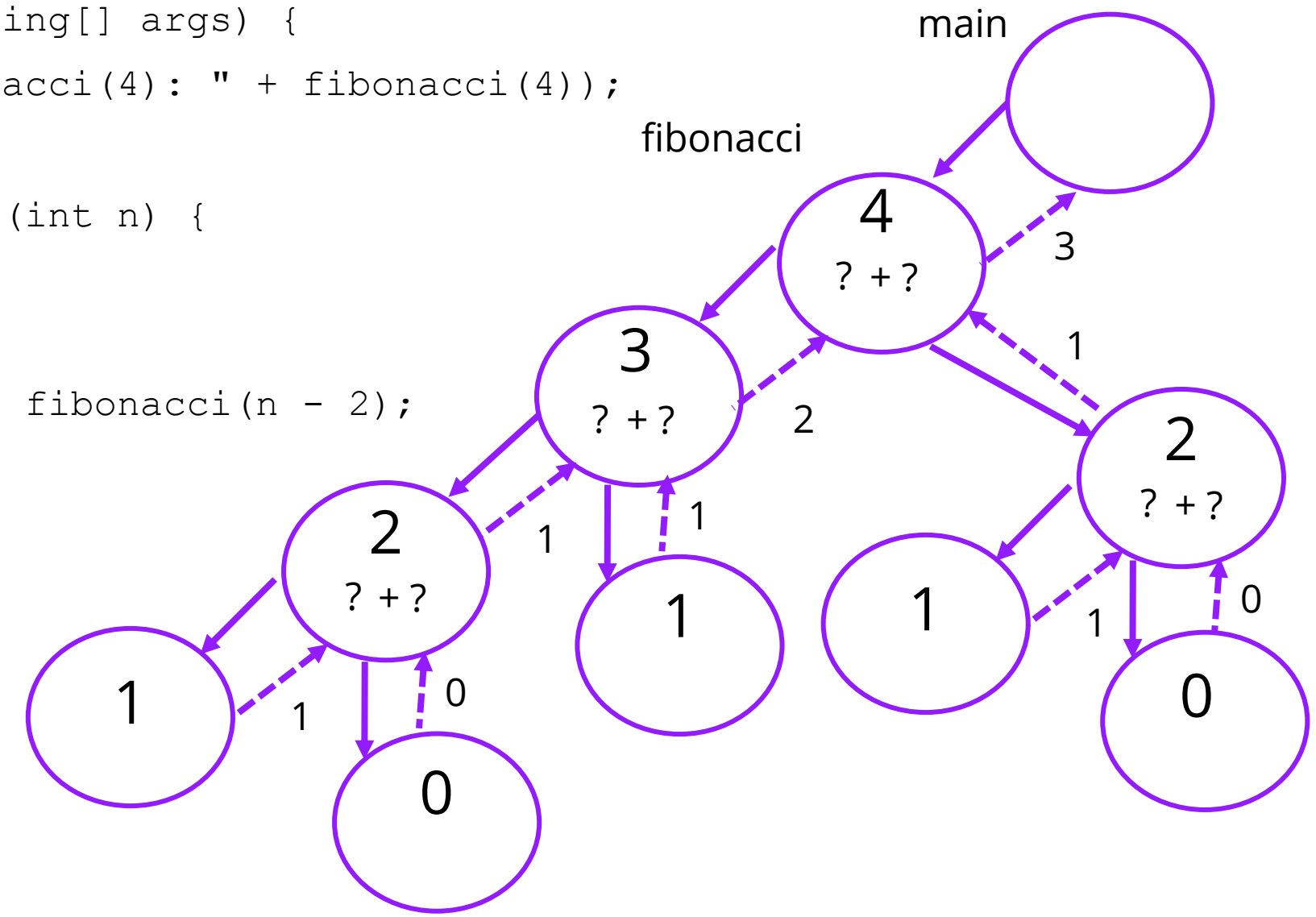
3

-What is the maximum depth of the call stack of recursive calls?

4

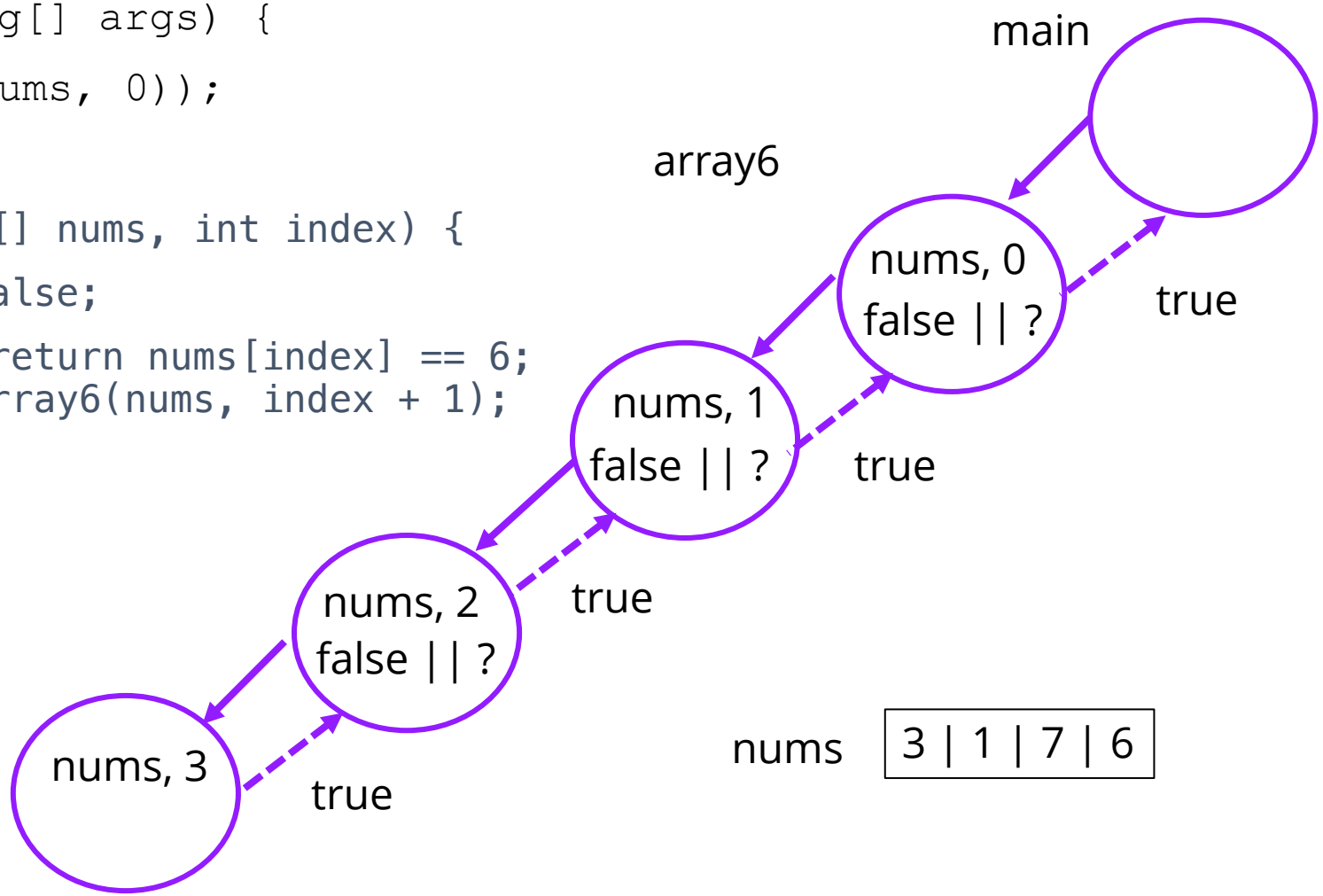
-How many times was the recursive method called?

9



Example 3

```
public static void main(String[] args) {
   System.out.println(array6(nums, 0));
}
public static boolean array6(int[] nums, int index) {
   if (nums.length == 0) return false;
   if (index == nums.length - 1) return nums[index] == 6;
   return (nums[index] == 6) || array6(nums, index + 1);
}
```



-What is the return result?

true

-What is the maximum depth of the call stack of recursive calls?

4

-How many times was the recursive method called?

4