Recursion - Fibonacci

- Let's examine how this would work for another classic recursive problem.
  - The Fibonacci sequence:

    Fib(0) = 1
    Fib(1) = 1
    Fib(n) = Fib(n-2) + Fib(n-1)
  - How can we code this?
  - What parts are the base case?
  - What parts are the recursive step?

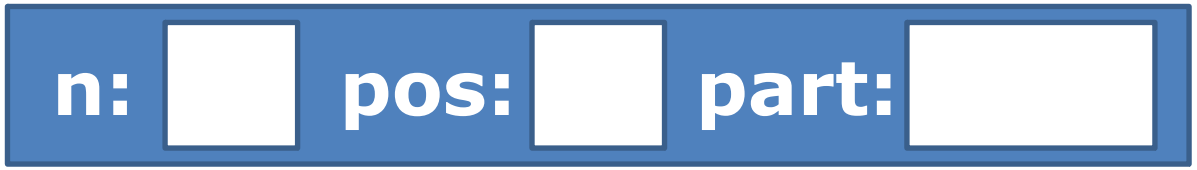
Recursion - Fibonacci

```
int fibonacci(int n)
{
   if(n == 0 || n == 1)
      return 1;
   else
      return fibonacci(n-2) +
       fibonacci(n-1);
}
```

Recursion - Fibonacci

```
int fibonacci(int n)
{
   if(n == 0 || n == 1)
   return 1;
   else
A: return fibonacci(n-2) +
B:          fibonacci(n-1);
}
```

We'll use the below graphics to aid our analysis of this

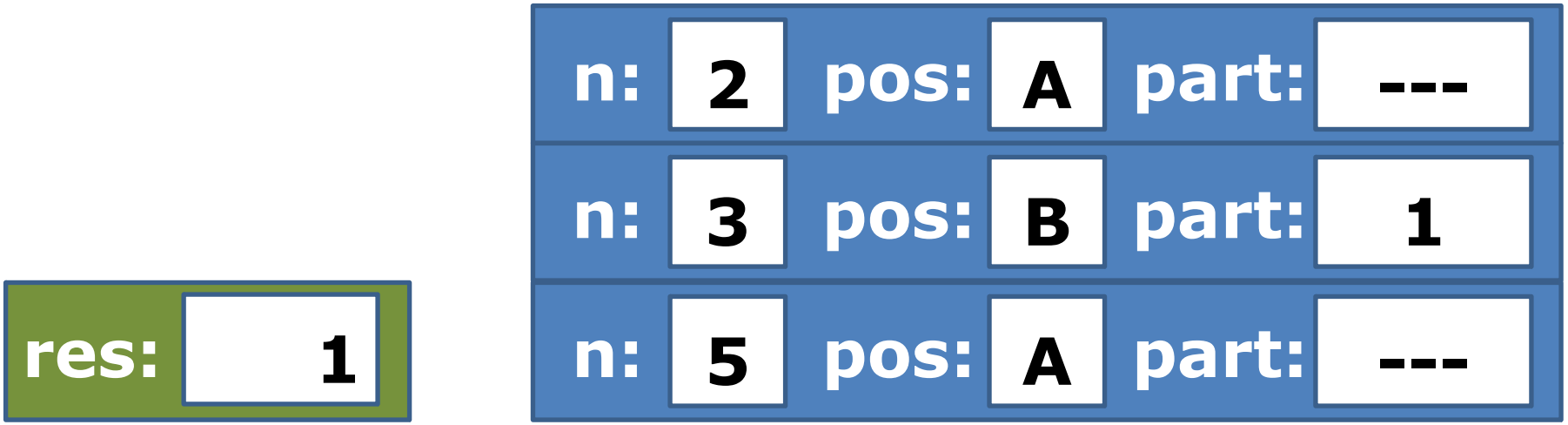


Recursion - Fibonacci

```
if(n == 0 || n == 1)
    return 1;
else
A: return fibonacci(n-2) +
B:           fibonacci(n-1);
```

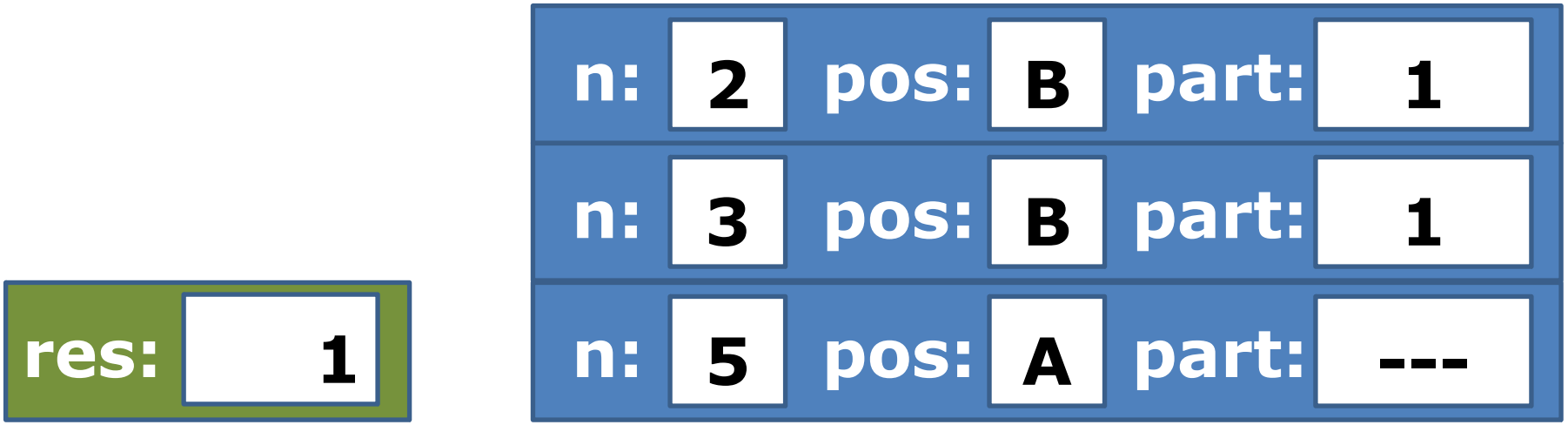
Recursion - Fibonacci

```
if(n == 0 || n == 1)
    return 1;
else
A: return fibonacci(n-2) +
B:           fibonacci(n-1);
```



Recursion - Fibonacci

```
if(n == 0 || n == 1)
    return 1;
else
A: return fibonacci(n-2) +
B:           fibonacci(n-1);
```



Recursion - Fibonacci

```
if(n == 0 || n == 1)
    return 1;
else
A: return fibonacci(n-2) +
B:           fibonacci(n-1);
```

Recursion - Fibonacci

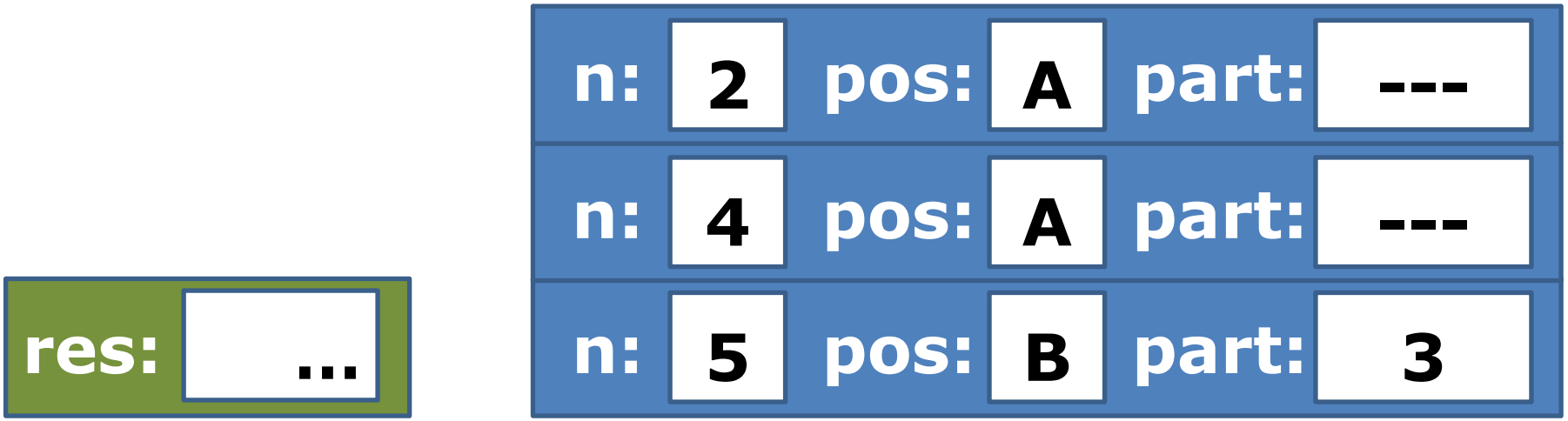
```
if(n == 0 || n == 1)
    return 1;
else
A: return fibonacci(n-2) +
B:          fibonacci(n-1);
```

res:	3

n:	5	pos:	A	part:	---

Recursion - Fibonacci

```
if(n == 0 || n == 1)
    return 1;
else
A: return fibonacci(n-2) +
B:          fibonacci(n-1);
```



Recursion - Fibonacci

Didn’t we already get an answer for n = 2?

Yep. So I’ll save us some time.

n:	2	pos:	A	part:	---
n:	4	pos:	A	part:	---
n:	5	pos:	B	part:	3

res: ...

Recursion - Fibonacci

```
if(n == 0 || n == 1)
    return 1;
else
A: return fibonacci(n-2) +
B:           fibonacci(n-1);
```

Recursion - Fibonacci

Didn’t we already get an answer for n = 3?

Yep. So I’ll save us some time.

n:	3	pos:	A	part:	---
n:	4	pos:	B	part:	2
n:	5	pos:	B	part:	3

Recursion - Fibonacci

Didn't we already get an answer for n = 3?

Yep. So I'll save us some time.

Recursion - Fibonacci

```
if(n == 0 || n == 1)
   return 1;
else
A: return fibonacci(n-2) +
B:          fibonacci(n-1);
```

res:	5

n:	5	pos:	B	part:	3

Recursion - Fibonacci

```
if(n == 0 || n == 1)
   return 1;
else
A: return fibonacci(n-2) +
B:          fibonacci(n-1);
```

res: 8

Recursion - Fibonacci

- Can this be done more efficiently?
  - You betcha! First off, note that we had had to recalculate some of the intermediate answers.
  - What if we could have saved those answers?
  - It's possible, and the corresponding technique is called *dynamic programming*.
  - We'll not worry about that for now.