```
def pascal_triangle(n)
    if n==1
        return [[1]]
    end

    result = pascal_triangle(n-1)
    new_row = [1]
    previous_row = result[-1]

    (0... previous_row.length-1).each_with_index do |idx|
        sum = 0
        sum = result[-1][idx] + result[-1][idx+1]
        new_row.push(sum)
    end

    new_row.push(1)
    result.push(new_row)
end
```

pascal_triangle(4)

result is being memoized. Memoized, saving recursive calls.

result = [[1],[1,1],[1,2,1]]

result = [[1], [1,1], [1,2,1], [1,3,3,1]]

```
def pascal_triangle(n)
    if n==1
        return [[1]]
    end

    result = pascal_triangle(n-1)
    new_row = [1]
    previous_row = result[-1]

    (0... previous_row.length-1).each_with_index do |idx|
        sum = 0
        sum = result[-1][idx] + result[-1][idx+1]
        new_row.push(sum)
    end

    new_row.push(1)
    result.push(new_row)
end
```

pascal_triangle(3)

result = [[1], [1,1]]

result = [[1],[1,1],[1,2,1]]

```
def pascal_triangle(n)
    if n==1
        return [[1]]
    end

    result = pascal_triangle(n-1)
    new_row = [1]
    previous_row = result[-1]

    (0... previous_row.length-1).each_with_index do |idx|
        sum = 0
        sum = result[-1][idx] + result[-1][idx+1]
        new_row.push(sum)
    end

    new_row.push(1)
    result.push(new_row)
end
```

pascal_triangle(2)

result = [[1]]

result = [[1], [1, 1]]

```
def pascal_triangle(n)
    if n==1
        return [[1]]
    end

    result = pascal_triangle(n-1)
    new_row = [1]
    previous_row = result[-1]

    (0... previous_row.length-1).each_with_index do |idx|
        sum = 0
        sum = result[-1][idx] + result[-1][idx+1]
        new_row.push(sum)
    end

    new_row.push(1)
    result.push(new_row)
end
```

pascal_triangle(1)