SE2205: Algorithms and Data Structures for Object-Oriented Design

Recursion

Instructor: Pirathayini Srikantha

Western University

January 10, 2019

Readings/References

- Recursion: Goodrich (5)

Table of Contents

1. Recursion Basics
2. Example: Sum of Squares
3. Example: Reversing a String
4. Example: Towers of Hanoi

Table of Contents

1. Recursion Basics
2. Example: Sum of Squares
3. Example: Reversing a String
4. Example: Towers of Hanoi

Recursion

- Has the potential of offering **simple** and **elegant** solutions to **hard** problems
- Consists of a **circular** definition: define something in terms of itself
- More specifically, define a **big** version of problem X in terms of **simpler** versions of X

A Divide and Conquer Approach

- **Divide and conquer**: Breakdown a **large** problem into **simpler** sub-problems until these are simple enough to solve **directly**
- **Sub-problems** are smaller than the overall problem but of the **same kind**
- **Smaller problems** can either be further divided into even smaller problems or solved directly
- Solutions of sub-problems can be **combined** with one another to obtain solution for the overall problem

General Implementation

- **Recursive call**: Function calls itself inside itself
- **Base case**: Simplest sub-problem with a **direct solution**
  - Recursive calls stop here!
- **Infinite regress**: Function keeps on calling itself (**no stop point**)
  - Problem division doesn't move closer to the base case at each recursive step

Tools to Aid with Implementation

- **Call tree**: Shows how function is being called recursively
- **Call trace**: Trace of function calls

Table of Contents

1. Recursion Basics
2. Example: Sum of Squares
3. Example: Reversing a String
4. Example: Towers of Hanoi

Sum of Squares: Problem Statement

Problem: Find the **sum of squares** $S(n, m)$ of integers that fall within the range $[n, m]$

$$S(n, m) = n^2 + (n+1)^2 \ldots + m^2$$

$$S(n, m) = \sum_{i=n}^{m} i^2$$

- We will look at two recursive implementations of this problem

Sum of Squares: Implementation #1

It is possible to express the sum of squares problem as follows:

$$S(n,m) = \sum_{i=n}^{m} i^2$$

$$S(n,m) = m^2 + \sum_{i=n}^{m-1} i^2$$

$$S(n,m) = m^2 + S(n,m-1)$$

- $S(n,m-1)$ is a **smaller** problem than $S(n,m)$ when $n,m > 0$ but of the **same kind**
- $S(i,i)$ is the smallest problem (i.e. the base case)
- $S(i,i)$ can be evaluated directly (i.e. $S(i,i) = i^2$)

Sum of Squares: Implementation #1

```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```

$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```

Definition of recursive function

$$S(n, m) = m^2 + S(n, m - 1)$$

Sum of Squares: Implementation #1

```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```

Base case

$$S(n, m) = m^2 + S(n, m - 1)$$

Sum of Squares: Implementation #1

```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```

Recursive definition

$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```

Recursive call to solve a simpler problem

$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```

Combination of solutions to simpler sub-problems

$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```

First call to recursive function where according to definition $S(7,10)=7^2+8^2+9^2+10^2=294$

$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

Call Tree

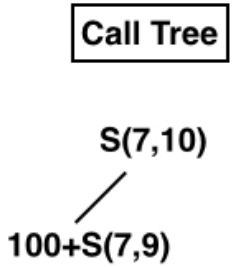
S(7,10)

```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```

$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

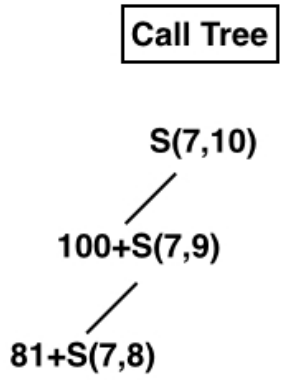
```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```



$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

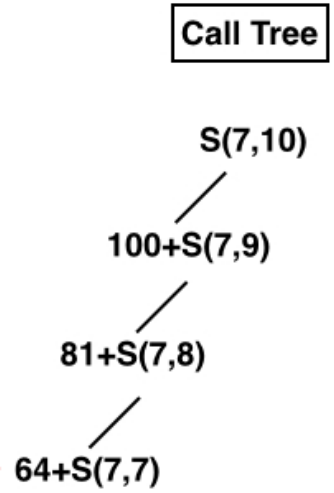
```java
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```



$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

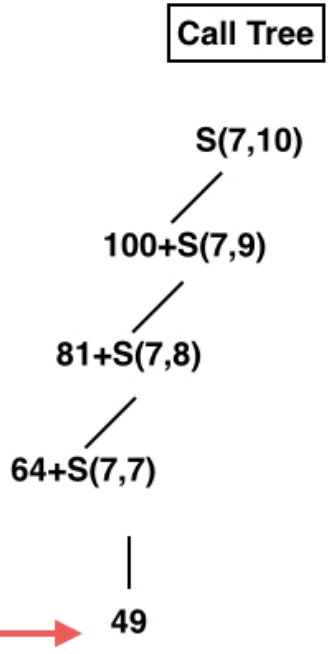
```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```



$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

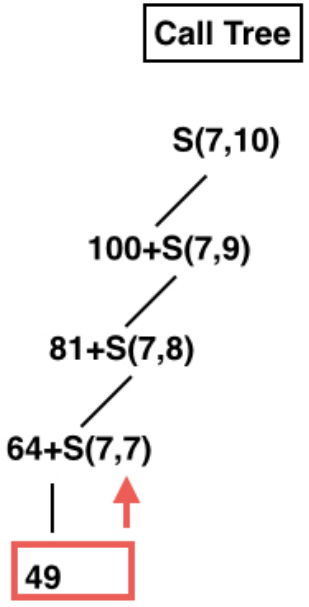
```java
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```



$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

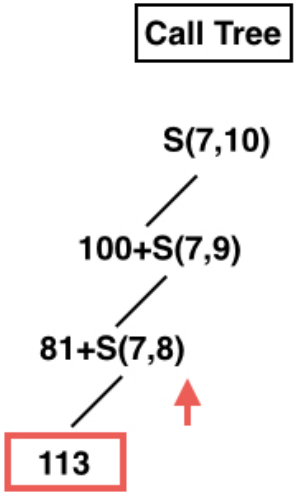
```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```



$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

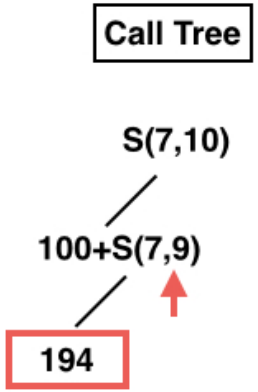
```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```



$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

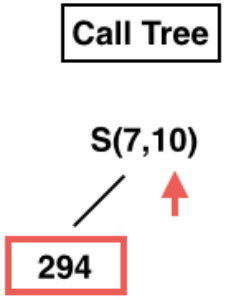
```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```



$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```



$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```

Call Tree

S(7,10)=294

$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #1

```
public class SumOfSquaresV1{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return m*m+S(n,m-1);
  }
}

public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV1 obj1 = new SumOfSquaresV1();
    System.out.println("S("+n+","+m+"):"+obj1.S(n,m));
  }
}
```

Program Output:

S(7,10): 294

$$S(n,m) = m^2 + S(n,m-1)$$

Sum of Squares: Implementation #2

The sum of squares can be re-written to another equivalent form:

$$
\begin{aligned}
S(n,m) = & \sum_{i=n}^{m} i^2 \\
= & n^2 + \ldots + (\left\lfloor \frac{(n+m)}{2} \right\rfloor)^2 + (\left\lfloor \frac{(n+m)}{2} \right\rfloor + 1)^2 + \ldots + m^2 \\
= & \sum_{i=n}^{\lfloor \frac{m+n}{2} \rfloor} i^2 + \sum_{j=\lfloor \frac{m+n}{2} \rfloor + 1}^{m} j^2 \\
= & s(n, \left\lfloor \frac{m+n}{2} \right\rfloor) + s(\left\lfloor \frac{m+n}{2} \right\rfloor + 1, m)
\end{aligned}
$$

Sum of Squares: Implementation #2

```
public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV2 obj2 = new SumOfSquaresV2();
    System.out.println("S("+n+","+m+"):"+obj2.S(n,m));
  }
}
```

Call Tree

S(7,10)

```
public class SumOfSquaresV2{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return S(n,(m+n)/2)+S((m+n)/2+1,m);
  }
}
```

Call Trace

S(7,10)

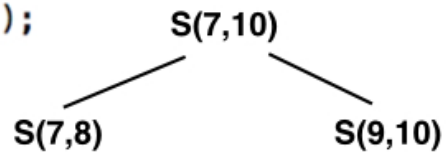
$$S(n,m) = s(n, \left\lfloor \frac{m+n}{2} \right\rfloor) + s(\left\lfloor \frac{m+n}{2} \right\rfloor + 1, m)$$

Sum of Squares: Implementation #2

```
public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV2 obj2 = new SumOfSquaresV2();
    System.out.println("S("+n+","+m+"):"+obj2.S(n,m));
  }
}
```

```
public class SumOfSquaresV2{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return S(n,(m+n)/2)+S((m+n)/2+1,m);
  }
}
```

Call Tree



Call Trace

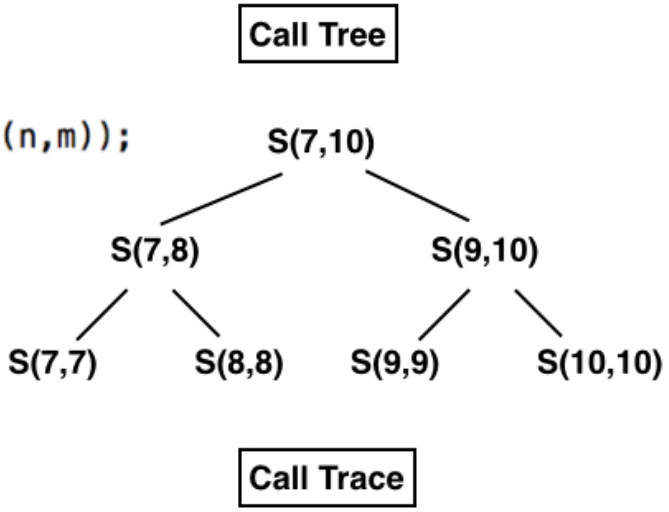
S(7,10) = S(7,8) + S(9,10)

$$S(n,m) = s(n, \left\lfloor \frac{m+n}{2} \right\rfloor) + s(\left\lfloor \frac{m+n}{2} \right\rfloor + 1, m)$$

Sum of Squares: Implementation #2

```java
public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV2 obj2 = new SumOfSquaresV2();
    System.out.println("S("+n+","+m+"):"+obj2.S(n,m));
  }
}
```

```java
public class SumOfSquaresV2{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return S(n,(m+n)/2)+S((m+n)/2+1,m);
  }
}
```



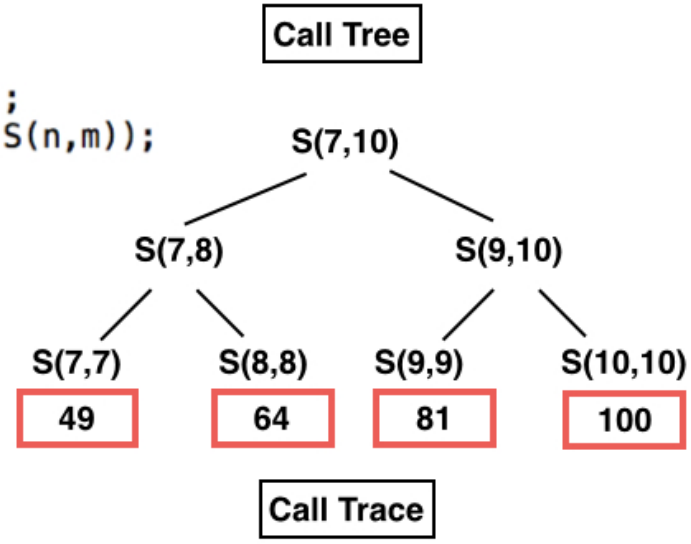
S(7,10) = S(7,8) + S(9,10)
= S(7,7) + S(8,8) + S(9,9) + S(10,10)

$$S(n,m) = s(n, \left\lfloor \frac{m+n}{2} \right\rfloor) + s(\left\lfloor \frac{m+n}{2} \right\rfloor + 1, m)$$

Sum of Squares: Implementation #2

```java
public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV2 obj2 = new SumOfSquaresV2();
    System.out.println("S("+n+","+m+"):"+obj2.S(n,m));
  }
}
```

```java
public class SumOfSquaresV2{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return S(n,(m+n)/2)+S((m+n)/2+1,m);
  }
}
```



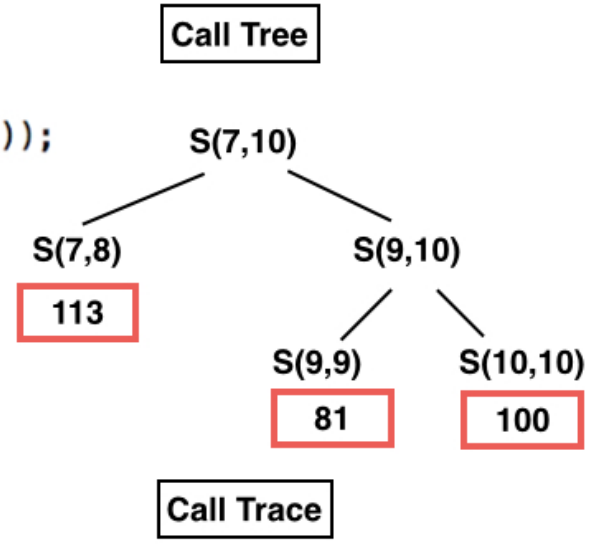
S(7,10) = S(7,8) + S(9,10)
= S(7,7) + S(8,8) + S(9,9) + S(10,10)
= (49+64) + (81+100)

$$S(n,m) = s(n, \left\lfloor \frac{m+n}{2} \right\rfloor) + s(\left\lfloor \frac{m+n}{2} \right\rfloor + 1, m)$$

Sum of Squares: Implementation #2

```
public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV2 obj2 = new SumOfSquaresV2();
    System.out.println("S("+n+","+m+"):"+obj2.S(n,m));
  }
}
```

```
public class SumOfSquaresV2{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return S(n,(m+n)/2)+S((m+n)/2+1,m);
  }
}
```



$$
\begin{aligned}
S(7,10) &= S(7,8) + S(9,10) \\
&= S(7,7) + S(8,8) + S(9,9) + S(10,10) \\
&= (49+64) + (81+100) \\
&= 113 + (81+100)
\end{aligned}
$$

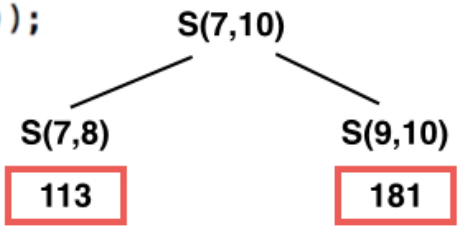
$$S(n,m) = s(n, \left\lfloor \frac{m+n}{2} \right\rfloor) + s(\left\lfloor \frac{m+n}{2} \right\rfloor + 1, m)$$

Sum of Squares: Implementation #2

```java
public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV2 obj2 = new SumOfSquaresV2();
    System.out.println("S("+n+","+m+"):"+obj2.S(n,m));
  }
}
```

```java
public class SumOfSquaresV2{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return S(n,(m+n)/2)+S((m+n)/2+1,m);
  }
}
```

Call Tree



Call Trace

S(7,10) = S(7,8) + S(9,10)
= S(7,7) + S(8,8) + S(9,9) + S(10,10)
= (49+64) + (81+100)
= 113 + 181

$$S(n,m) = s(n, \left\lfloor \frac{m+n}{2} \right\rfloor) + s(\left\lfloor \frac{m+n}{2} \right\rfloor + 1, m)$$

Sum of Squares: Implementation #2

```
public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV2 obj2 = new SumOfSquaresV2();
    System.out.println("S("+n+","+m+"):"+obj2.S(n,m));
  }
}
```

```
public class SumOfSquaresV2{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return S(n,(m+n)/2)+S((m+n)/2+1,m);
  }
}
```

Call Tree

S(7,10)

294

Call Trace

S(7,10) = S(7,8) + S(9,10)
= S(7,7) + S(8,8) + S(9,9) + S(10,10)
= (49+64) + (81+100)
= 113 + 181
= 294

$$S(n,m) = s(n, \left\lfloor \frac{m+n}{2} \right\rfloor) + s(\left\lfloor \frac{m+n}{2} \right\rfloor + 1, m)$$

Sum of Squares: Implementation #2

```
public class TestSumOfSquares{
  public static void main(String[] args) {
    int n=7,m=10;
    SumOfSquaresV2 obj2 = new SumOfSquaresV2();
    System.out.println("S("+n+","+m+"):"+obj2.S(n,m));
  }
}

public class SumOfSquaresV2{
  public int S(int n, int m){
    if(n==m)
      return n*n;
    else
      return S(n,(m+n)/2)+S((m+n)/2+1,m);
  }
}
```

Program Output:

S(7,10): 294

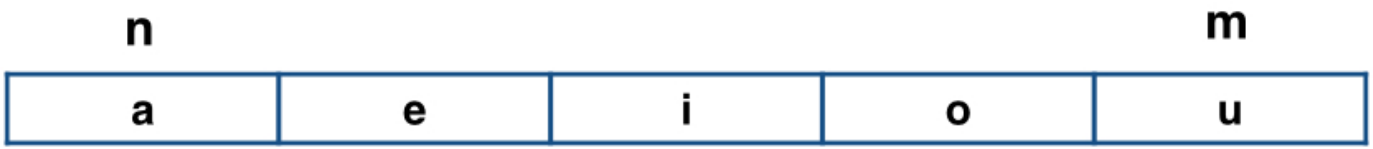
$$S(n,m) = s(n, \left\lfloor \frac{m+n}{2} \right\rfloor) + s(\left\lfloor \frac{m+n}{2} \right\rfloor + 1, m)$$

Table of Contents

1. Recursion Basics
2. Example: Sum of Squares
3. Example: Reversing a String
4. Example: Towers of Hanoi

Reversing a String

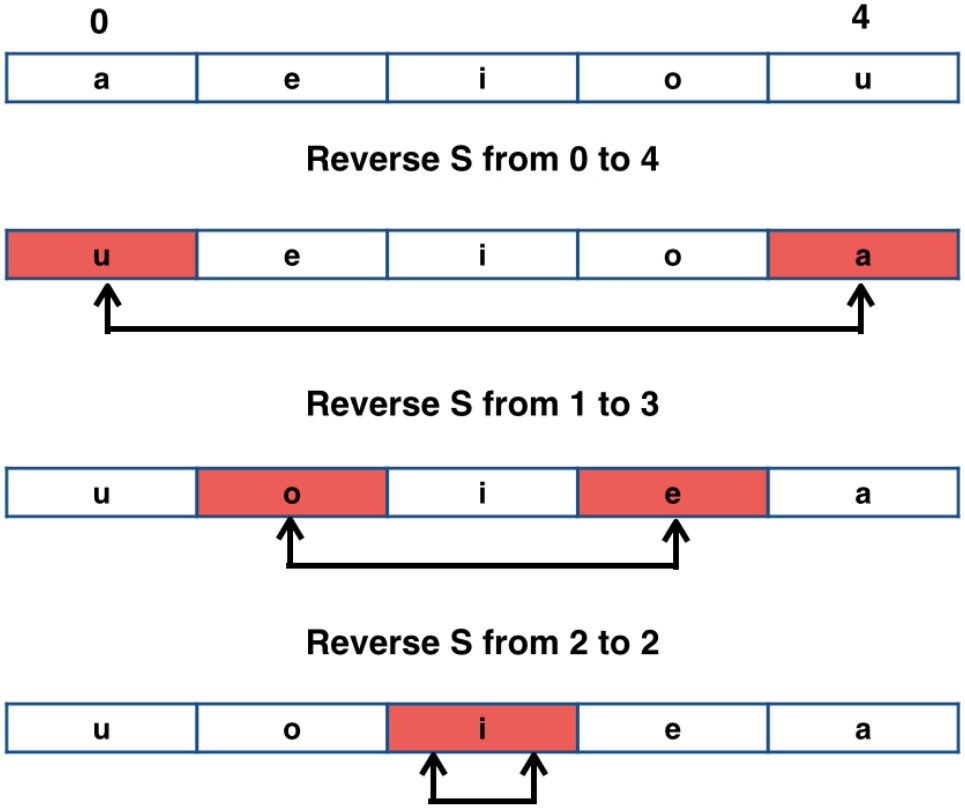
Reverse the following string S:



Recursive algorithm:

- Exchange $S[n]$ and $S[m]$
- Reverse string from index $[n+1]$ up to $[m-1]$
- **Base case:** when $n == m$ do nothing

Reversing a String

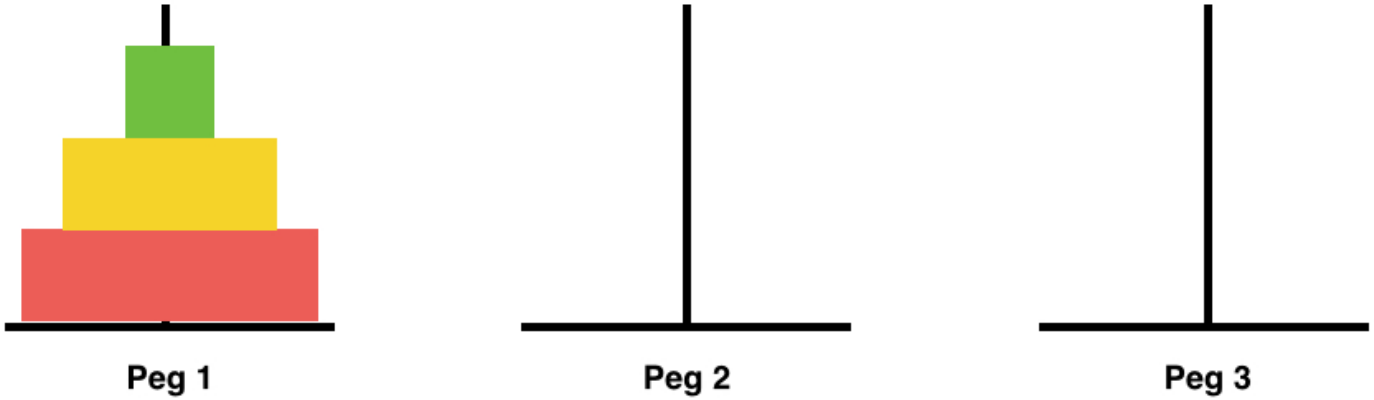


- Exchange $S[n]$ and $S[m]$
- Reverse string from index $[n+1]$ up to $[m-1]$
- **Base case:** when $n == m$ do nothing

Table of Contents

1. Recursion Basics
2. Example: Sum of Squares
3. Example: Reversing a String
4. Example: Towers of Hanoi

Towers of Hanoi



Problem:

- Move all disks to the last peg
- Move disks one at a time
- Larger disk cannot be placed above a smaller one

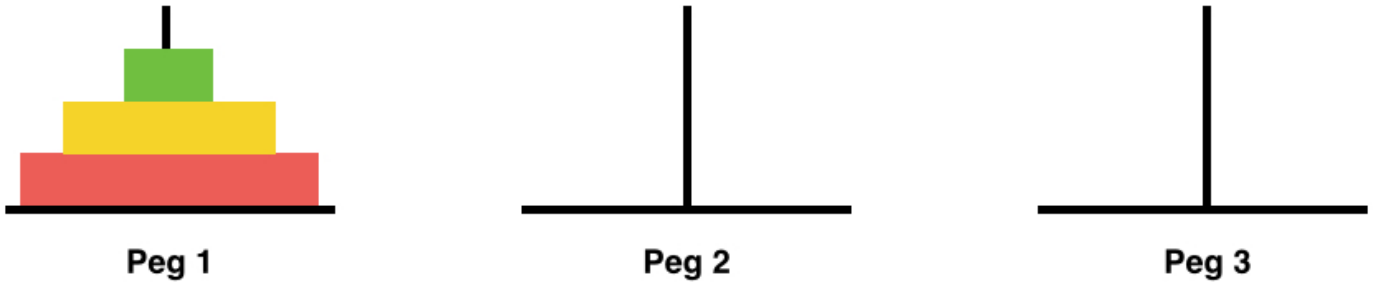
Recursive Implementation:

- Move $N-1$ disks from start peg to spare peg
- Move N^{th} disk from start peg to finish peg
- Move $N-1$ disks from spare peg to finish peg

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree

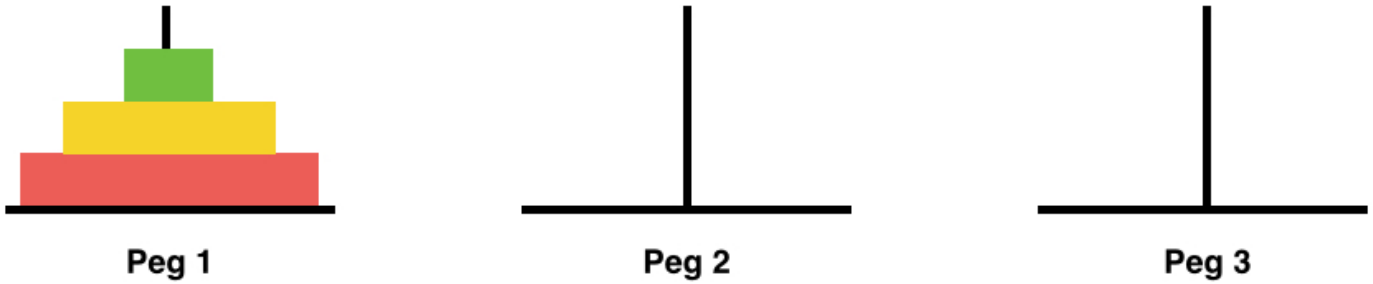


Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree

mT(1,2,3,3)



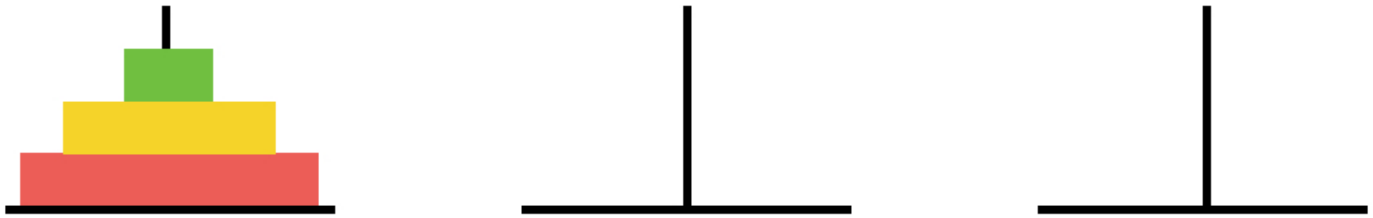
Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree

mT(1,2,3,3)

mT(1,3,2,2)



Peg 1 **Peg 2** **Peg 3**

Towers of Hanoi

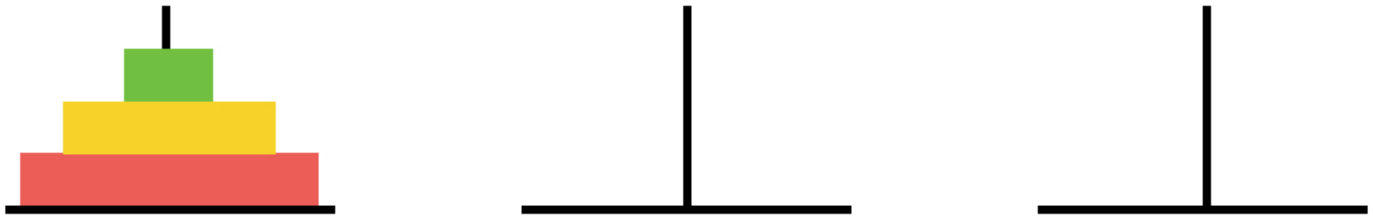
```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree

mT(1,2,3,3)

mT(1,3,2,2)

mT(1,2,3,1)

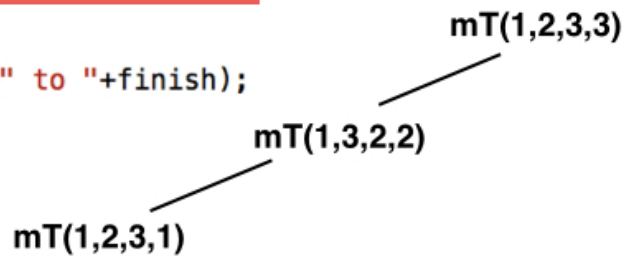


Peg 1 **Peg 2** **Peg 3**

Towers of Hanoi

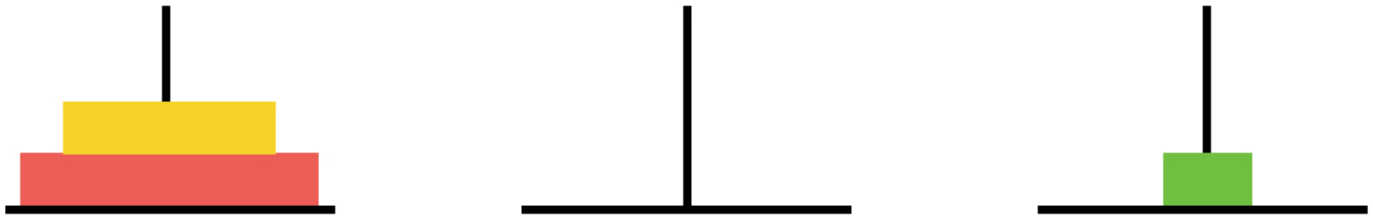
```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree



Program Output:

Move disk from 1 to 3



Peg 1 **Peg 2** **Peg 3**

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree

mT(1,2,3,3)

mT(1,3,2,2)

mT(1,2,3,1)

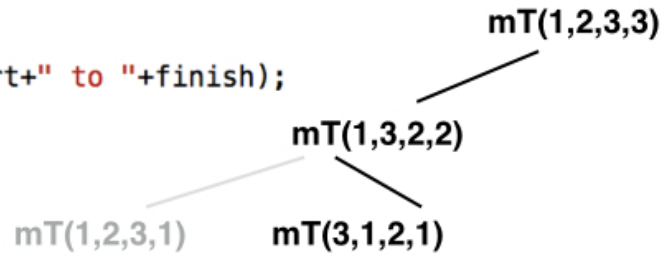
Program Output:
Move disk from 1 to 3
Move disk from 1 to 2

Peg 1 **Peg 2** **Peg 3**

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree

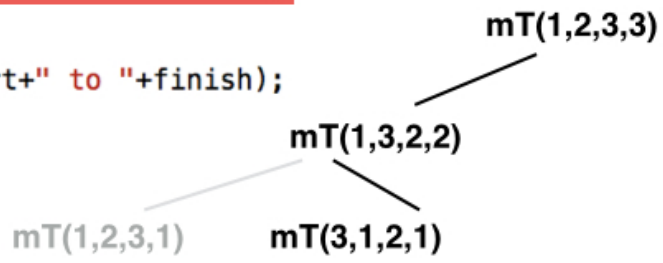


Program Output:
Move disk from 1 to 3
Move disk from 1 to 2

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree



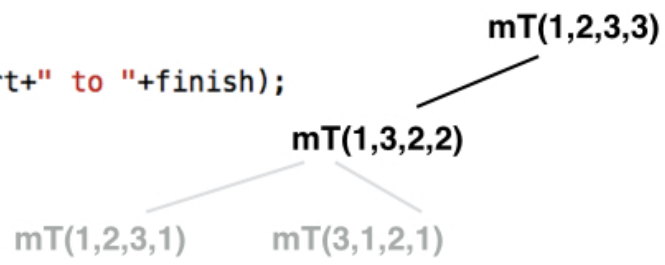
Program Output:
Move disk from 1 to 3
Move disk from 1 to 2
Move disk from 3 to 2

Peg 1 **Peg 2** **Peg 3**

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree



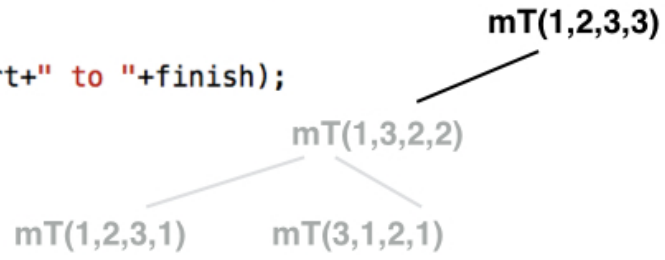
Program Output:
Move disk from 1 to 3
Move disk from 1 to 2
Move disk from 3 to 2

Peg 1 **Peg 2** **Peg 3**

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree



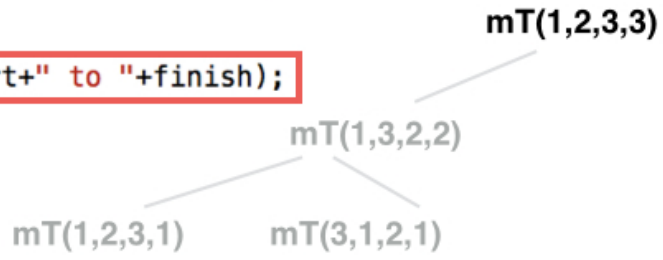
Program Output:
Move disk from 1 to 3
Move disk from 1 to 2
Move disk from 3 to 2

Peg 1 **Peg 2** **Peg 3**

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree

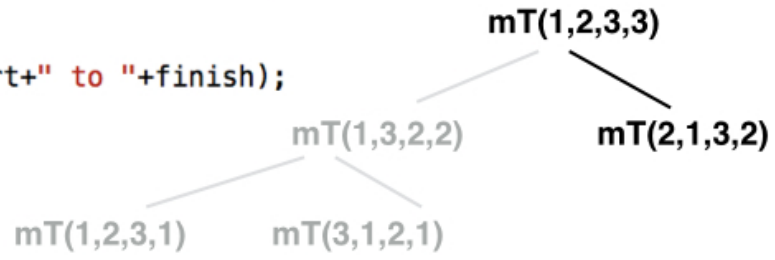


Program Output:
Move disk from 1 to 3
Move disk from 1 to 2
Move disk from 3 to 2
Move disk from 1 to 3

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree

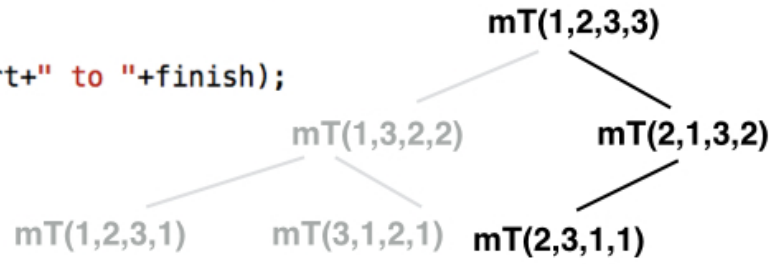


Program Output:
Move disk from 1 to 3
Move disk from 1 to 2
Move disk from 3 to 2
Move disk from 1 to 3

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree

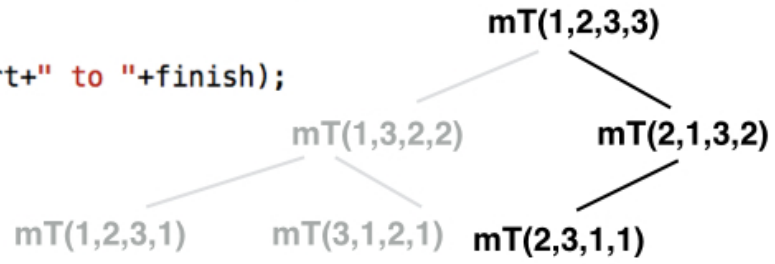


Program Output:
Move disk from 1 to 3
Move disk from 1 to 2
Move disk from 3 to 2
Move disk from 1 to 3

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree



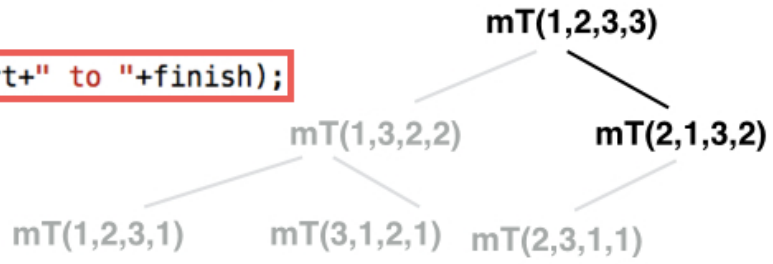
Program Output:

Move disk from 1 to 3
Move disk from 1 to 2
Move disk from 3 to 2
Move disk from 1 to 3
Move disk from 2 to 1

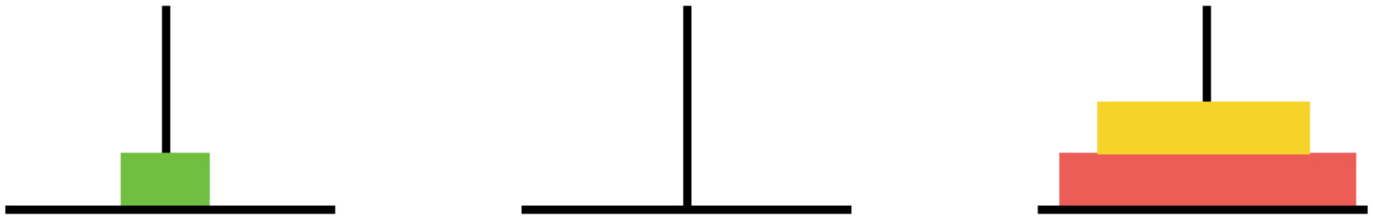
Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree



Program Output:
Move disk from 1 to 3
Move disk from 1 to 2
Move disk from 3 to 2
Move disk from 1 to 3
Move disk from 2 to 1
Move disk from 2 to 3



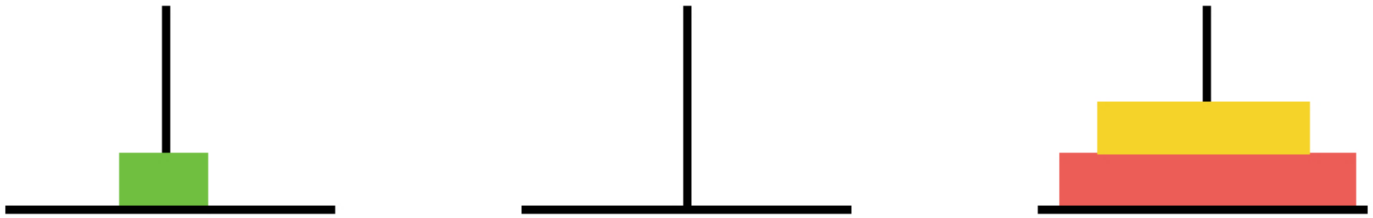
Peg 1 **Peg 2** **Peg 3**

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree

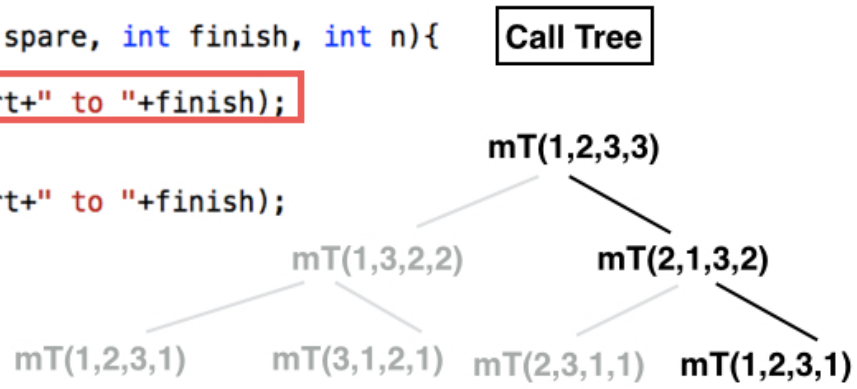
Program Output:
Move disk from 1 to 3
Move disk from 1 to 2
Move disk from 3 to 2
Move disk from 1 to 3
Move disk from 2 to 1
Move disk from 2 to 3



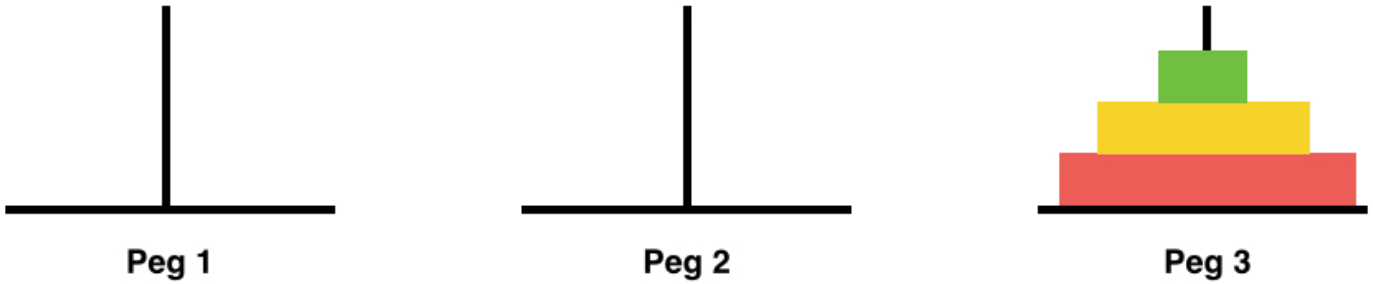
Peg 1 **Peg 2** **Peg 3**

Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```



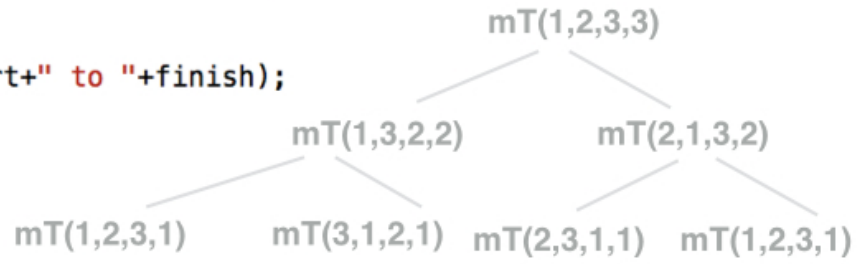
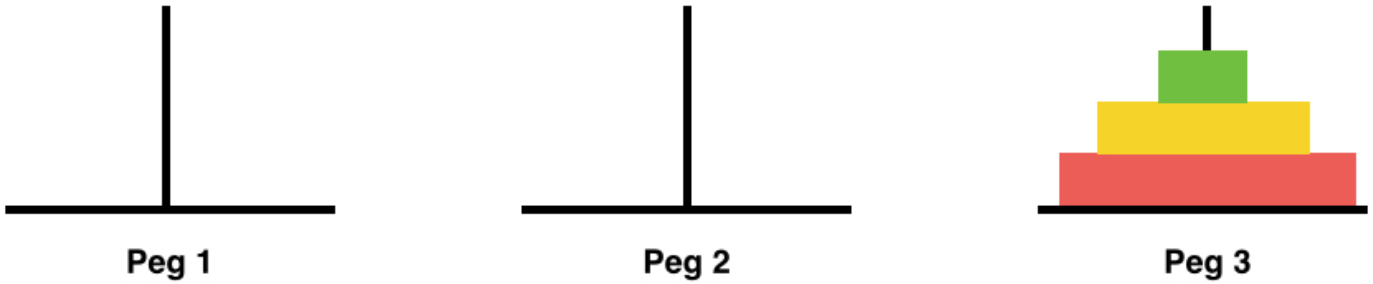
Program Output:
Move disk from 1 to 3
Move disk from 1 to 2
Move disk from 3 to 2
Move disk from 1 to 3
Move disk from 2 to 1
Move disk from 2 to 3
Move disk from 1 to 3



Towers of Hanoi

```java
public class TowersOfHanoi{
  public static void moveTowers(int start, int spare, int finish, int n){
    if(n==1)
      System.out.println("Move disk from "+start+" to "+finish);
    else{
      moveTowers(start, finish, spare, n-1);
      System.out.println("Move disk from "+start+" to "+finish);
      moveTowers(spare, start, finish, n-1);
    }
  }
  public static void main(String[] args){
    int start=1,spare=2,finish=3;
    moveTowers(start, spare, finish, finish);
  }
}
```

Call Tree



Program Output:
Move disk from 1 to 3
Move disk from 1 to 2
Move disk from 3 to 2
Move disk from 1 to 3
Move disk from 2 to 1
Move disk from 2 to 3
Move disk from 1 to 3