

COMP 250

INTRODUCTORY SCIENCE

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Week 9-2: Re Add WeChat edu_assist_pro

Giulia Alberini, Fall 2020

WHAT ARE WE GOING TO DO IN THIS VIDEO?



- Recursive algorithms Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

EXAMPLE

```
public static void countdown(int n) {  
    if (n == 0) {  
        System.out.println("Go!");  
    } else {  
        System.out.println(n);  
        countdown(n-1);  
    }  
}
```

- What prints if we call `countdown(3)`?

➤ 3 2 1 Go!

EXAMPLE – EXECUTION

```
public static void countdown(int n) {  
    if (n == 0) {  
        System.out.print("Go!");  
    } else {  
        System.out.print(n + " ");  
        countdown(n-1);  
    }  
}
```

Execution of `countdown(3)`.

- The execution of `countdown` starts with `n==3`. Since it is not 0, 3 is printed and `countdown` is called with input 2
 - The execution of `countdown` starts with `n==2`. It is not 0, thus 2 is printed and `countdown` is called with input 1.
 - The execution of `countdown` starts with `n==1`. Since `n` is not 0, 1 is printed and `countdown` is called with input 0.
 - The execution of `countdown` starts with `n==0`. Since `n` is 0, `Go!` is printed and the execution ends.
- ❖ The execution of `countdown(1)` ends.
 - The execution of `countdown(2)` ends.
- The execution of `countdown(3)` ends and we are back in `main`.

RECURSIVE – DEFINITION

Recursive functions/methods consists of the following

- **Base Case(s):** f terminating scenario that does not use recursion to produce an answer.
- **Recursive or Inductive step(s):** rules that determine how to produce an answer from simpler cases.

BASE CASE

Note that if there is no base case in a recursive method, or if the base case is never reached, the execution will never end.

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

```
public static void f (int n) {  
    forever (n) ;  
}
```

COMING UP

Several examples of algorithms that can be implemented recursively:

- Factorial function
- Fibonacci numbers
- reverseList
- sortList
- towerOfHanoi

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

EXAMPLE 1 – FACTORIAL

The factorial of a number is defined as follows:

$$0! = 1$$

$$1! = 1$$

$$2! = 1 * 2 = 2$$

$$3! = 3 * 2 * 1 = 6$$

...

$$n! = n * (n - 1) * (n - 2) * (n - 3) * ... * 1$$

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

FACTORIAL: RECURSIVE DEFINITION

- Notice that:

$$\begin{aligned} n! &= n * (n-1) * (n-2) * (n-3) * \dots * 1 \\ &= n * \end{aligned}$$

- Thus, the following definition determines the factorial:

Base case: $0! = 1$

Recursive step: $n! = n * (n-1)!$

FACTORIAL (ITERATIVE)

Assignment Project Exam Help

```
public static int factorial (int n) {  
    int result = 1;  
    for(int i=2; i<=n; i++)  
        result = result * i;  
    return result;  
}
```

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

FACTORIAL (RECURSIVE)

Let's use its recursive definition to write a method that computes the factorial function:

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

```
public static int factorial(int n) {  
    if (n == 0) {  
        return 1;  
    }  
    return n * factorial(n-1);  
}
```

Base case

Induction step

FACTORIAL: AN EXAMPLE

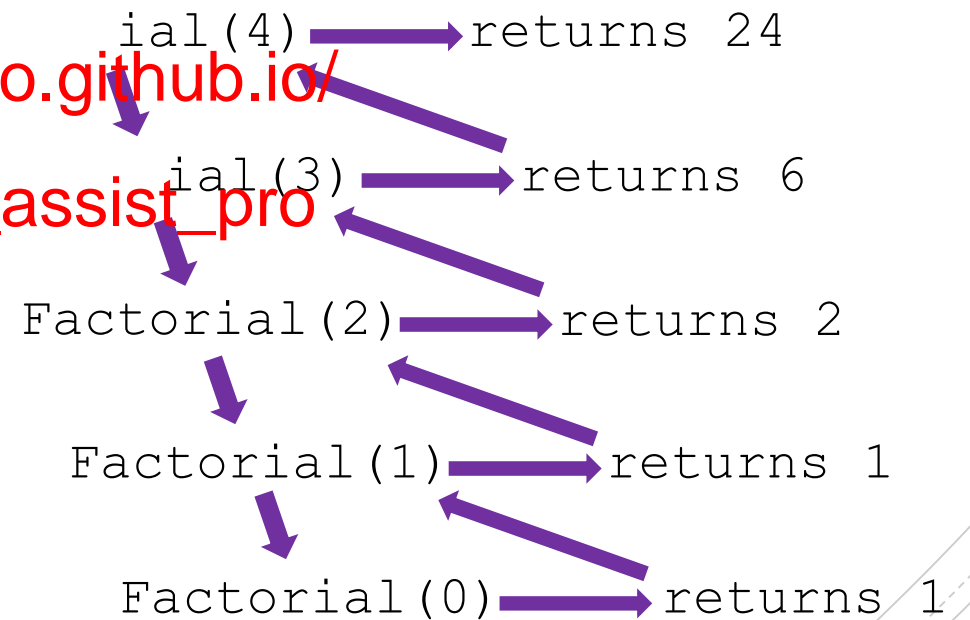
- What happens when the method call `factorial(4)` is executed?

Assignment Project Exam Help

```
public static int factorial(int n) {  
    if (n == 0) {  
        return 1;  
    }  
    return n * factorial(n-1);  
}
```

<https://eduassistpro.github.io/>

Add WeChat `edu_assist_pro`



CORRECTNESS

Claim: the recursive `factorial(n)` algorithm returns $n!$.

Proof (by mathematical induction):

- **Base case:** `factorial(0)`

- **Induction step:**

- **IH:** Assume `factorial(k)` returns k

- **To prove:** `factorial(k+1)` returns $(k + 1)!$

`factorial(k+1)` returns $\text{factorial}(k) * (k + 1)$
 $= k! * (k + 1)$, by IH
 $= (k + 1)!$

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

EXAMPLE 2 – FIBONACCI NUMBERS

- Fibonacci sequence: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Assignment Project Exam Help

- Base cases:

$f_1 = 1$
 $f_2 = 1$

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

- Recursive/Inductive Step:

$$f_n = f_{n-1} + f_{n-2}$$

FIBONACCI (ITERATIVE)

```
public static int fibonacci(int n) {  
    if(n==0 || n==1) {  
        return 1;  
    }  
    fib0 = 1;  
    fib1 = 1;  
    for(int i=2; i<=n; i++)  
    {  
        fib2 = fib0 + fib1;  
        fib0 = fib1;  
        fib1 = fib2;  
    }  
    return fib2;  
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

FIBONACCI (RECURSIVE)

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

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

This is much simpler to express than the iterative version.

CORRECTNESS

Claim: the recursive Fibonacci algorithm is correct.

Assignment Project Exam Help

Proof(sketch): (by strong induction)

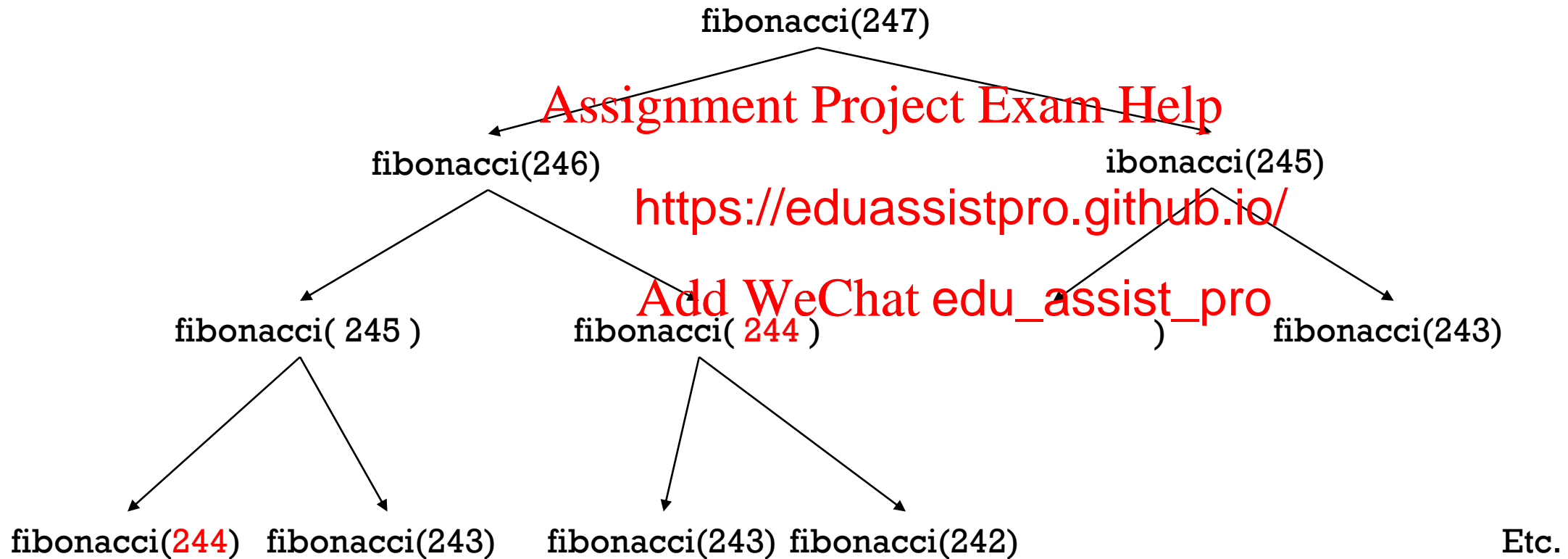
<https://eduassistpro.github.io/>

■ Induction step:

- IH: Let $k \geq 0$, Assume $\text{fibonacci}(i)$ returns f_i for every $0 \leq i < k$
- To prove: $\text{fibonacci}(k)$ returns f_k

Add WeChat: edu_assist_pro

However, the recursive Fibonacci algorithm is very inefficient. It computes the same quantity many times, for example:



EXAMPLE 3: REVERSING A LIST

input

$\{a, b, c, d, e, f, g, h\}$

Assignment Project Exam Help

output

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

EXAMPLE 3: REVERSING A LIST

input

$\{a, b, c, d, e, f, g, h\}$

Assignment Project Exam Help

output

$\}$
<https://eduassistpro.github.io/>

Idea of recursion: Add WeChat edu_assist_pro

$a \quad \{b, c, d, e, f, g, h\}$

$\{h, g, f, e, d, c, b\} \quad a$

REVERSING A LIST (RECURSIVE)

```
public static void reverse(List list) {  
    if(list.size() == 1)  
        return;  
    firstElement = list.remove(0);  
    reverse(list);  
    list.add(firstElement);  
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

EXAMPLE 4 – SORTING A LIST (RECURSIVE)

Assignment Project Exam Help

```
public static void sort(List list) {
```

```
    if(list.size()==1
```

```
        return;
```

```
    }
```

```
    minElement = removeMinElement(
```

```
    sort(list); // now the list has n-1 elements
```

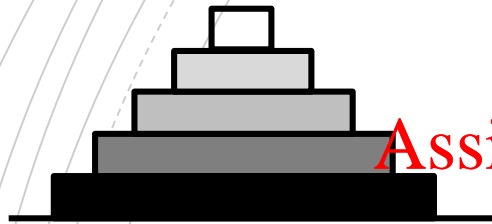
```
    list.add(0, minElement); // insert at the beginning of list
```

```
}
```

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

EXAMPLE 5 – TOWER OF HANOI



Tower A
(start)

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat [edu_assist_pro](#)

Tower C

(finish)

Problem: Move n disks from start tower to finish tower such that:

- move one disk at a time
- you can have a smaller disk on top of bigger disk (but you can't have a bigger disk onto a smaller disk)

EXAMPLE - n=1

Assignment Project Exam Help

start

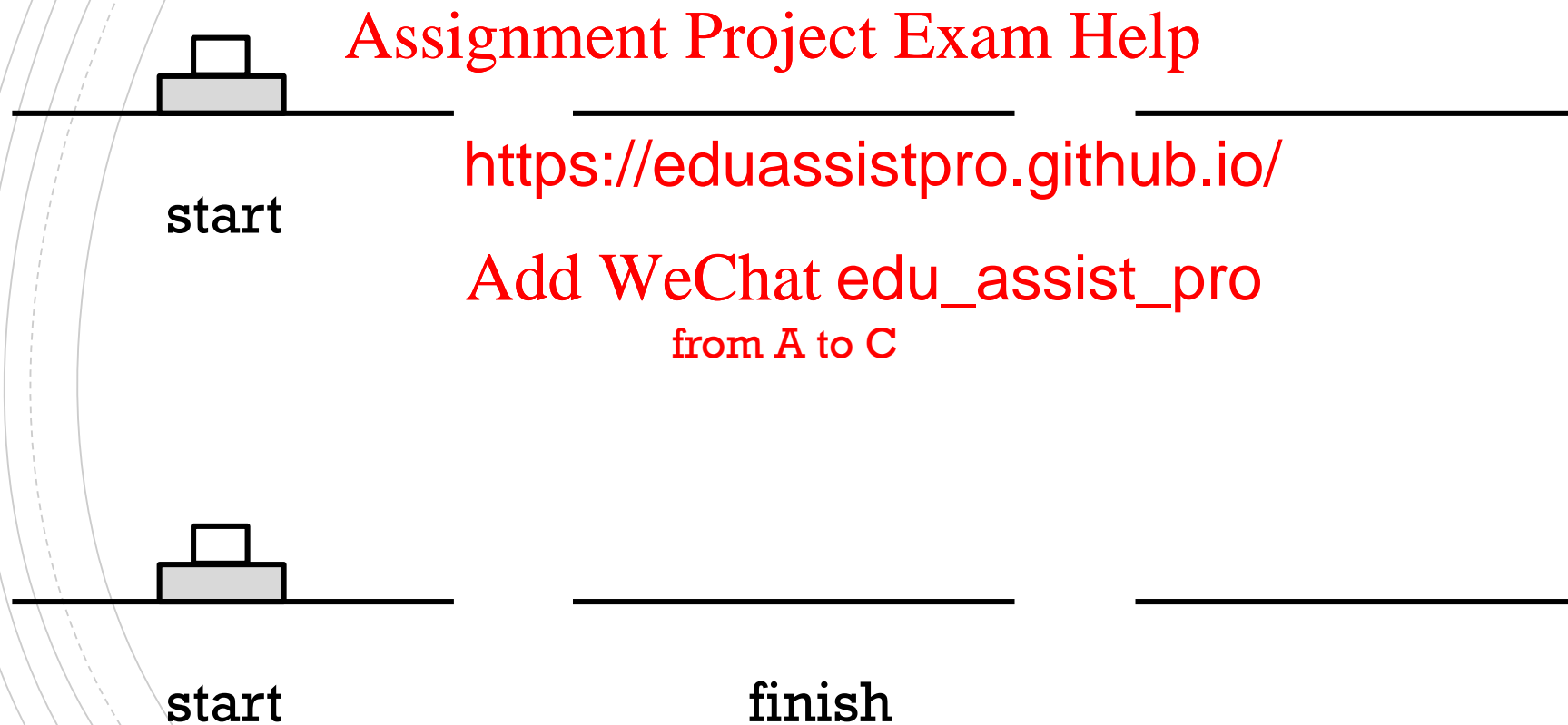
<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

start

finish

EXAMPLE - $n=2$



EXAMPLE - n=2

from A to B

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

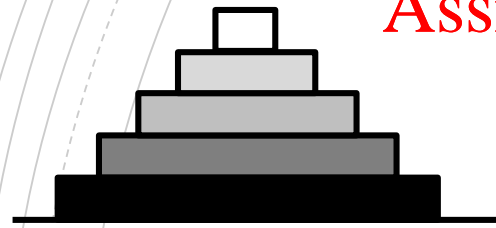
from C to B

start

start

finish

HOW SHOULD WE MOVE 5 DISKS FROM A TO B?



start

Assignment Project Exam Help

<https://eduassistpro.github.io/>

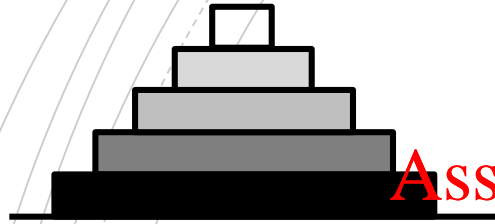
Add WeChat `edu_assist_pro`

finis

➤ Let's think about it recursively!

IDEA!

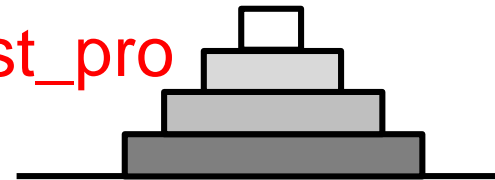
Somehow move 4 disks from A to C



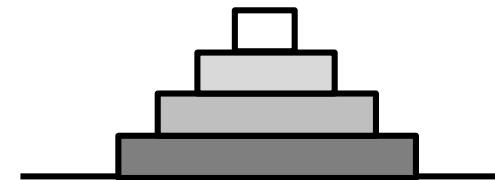
Assignment Project Exam Help

move <https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Somehow move 4 disks from C to B



ALGORITHM

```
tower(n, start, finish, other) { // e.g. tower(5, A, B, C)
    if(n==1) {
        move from start to finish
    } else {
        tower(n-1, start, other, finish)
        tower(1, start, finish, other)
        tower(n-1, other, finish, start)
    }
}
```

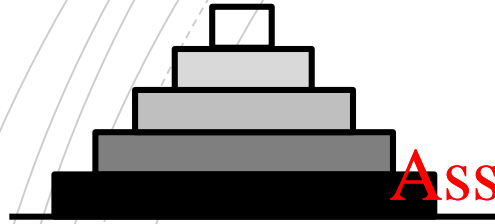
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat: edu_assist_pro

EXAMPLE

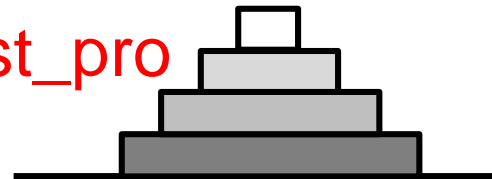
tower(4,A,C,B)



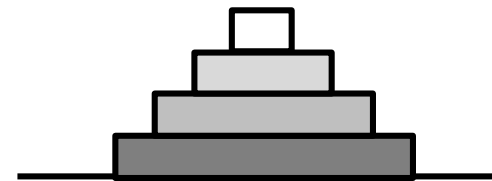
Assignment Project Exam Help

tower <https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



tower(4,C,B,A)



CORRECTNESS

Claim: the `tower()` algorithm is correct, namely it moves the blocks from start to finish without breaking the two rules (one at a time, and can't put bigger one onto smaller one).

Assignment Project Exam Help

Proof: (sketch)

<https://eduassistpro.github.io/>

- **Base case:** `tower(1, *, *, *)` is correct
- **Induction step:**
 - for any $k > 1$, assume `tower(k, *, *, *)` is correct
 - **Prove** `tower(k+1, *, *, *)` is correct.

Add WeChat edu_assist_pro

HOW MANY MOVES?

`tower(1, start, finish, other)`

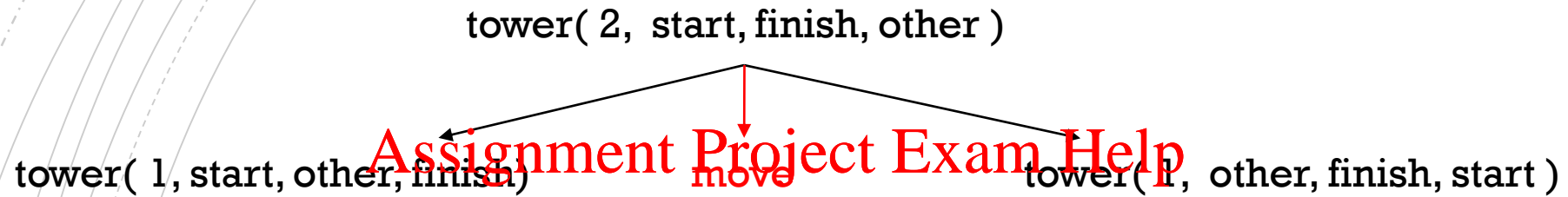
Assignment Project Exam Help
↓
move from

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Answer: 1

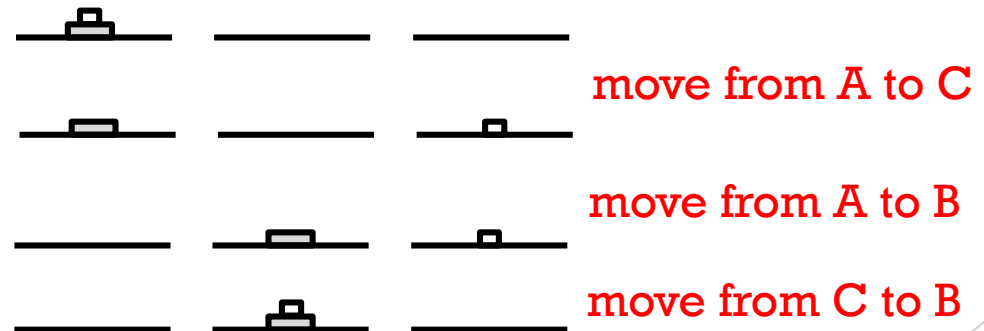
HOW MANY MOVES?



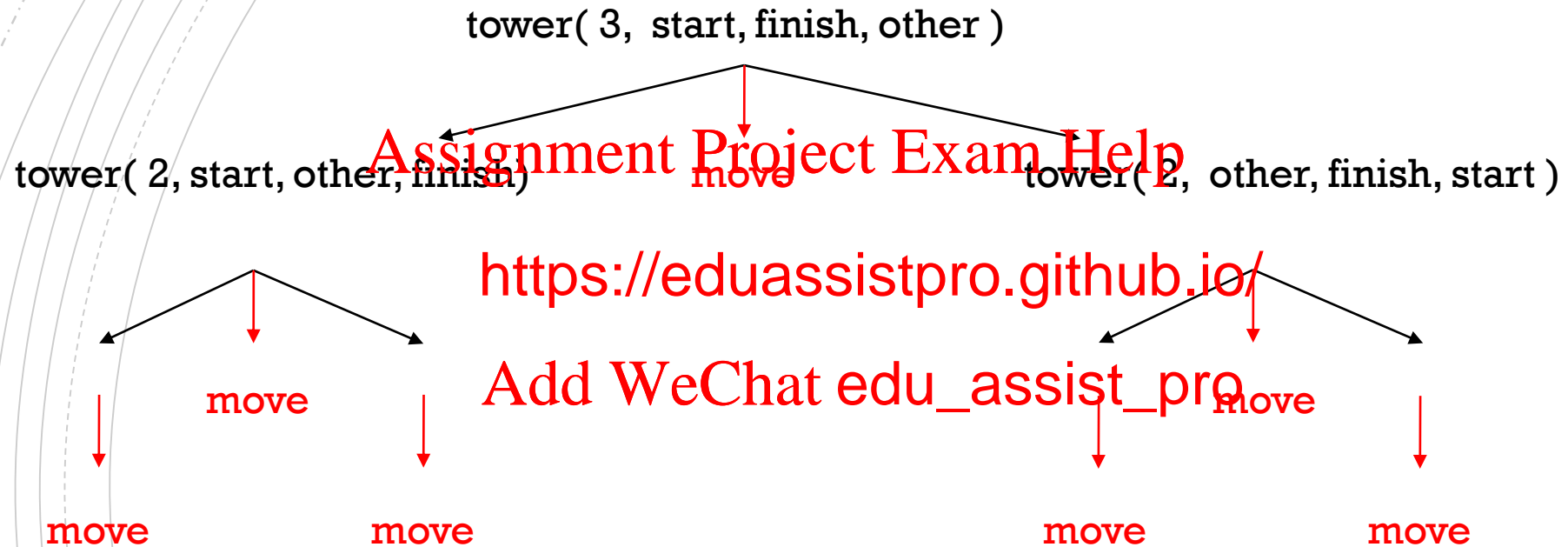
<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Answer: 1 + 2

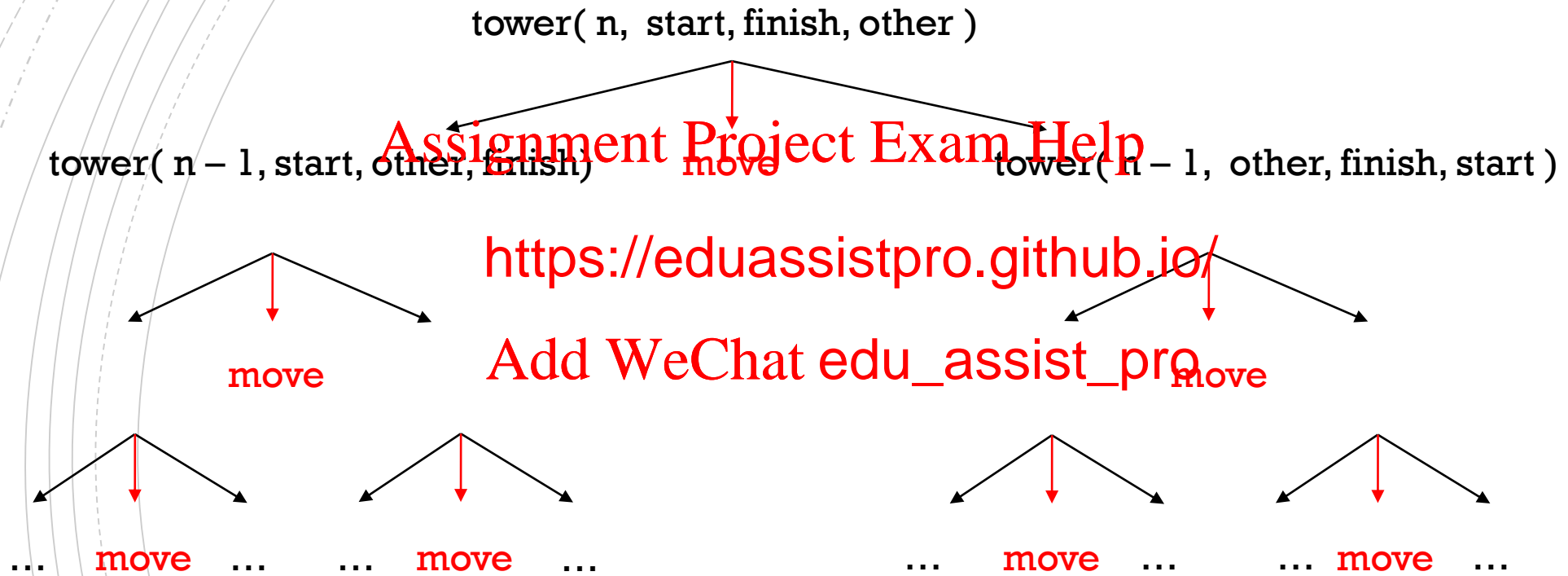


HOW MANY MOVES?



Answer: $1 + 2 + 4 = 2^0 + 2^1 + 2^2$

HOW MANY MOVES?



Answer: $1 + 2 + 4 + \dots + 2^{n-1} = 2^n - 1$

RECURSION AND ITERATION

Assignment Project Exam Help

- Recursion and iteration are usually expressive.
<https://eduassistpro.github.io/>
- Anything recursive can be done iteratively and do
- Anything iteration can do can be done recursively. Add WeChat edu_assist_pro

RECURSION VS ITERATION

- Which one to use?
 - Use the one is easier to think in terms of, for a specific problem.
 - For simple case <https://eduassistpro.github.io/> iter and faster.
 - For complex cases, recursion is elegant and simpler to code.
 - It is important to remember that when using one or the other, this decision might impact the performance of your program.

RECURSIVE DATA STRUCTURE

Assignment Project Exam Help

- We can recur structures.

<https://eduassistpro.github.io/>

- Let's consider arrays and let w we can recursively defined a list of items.

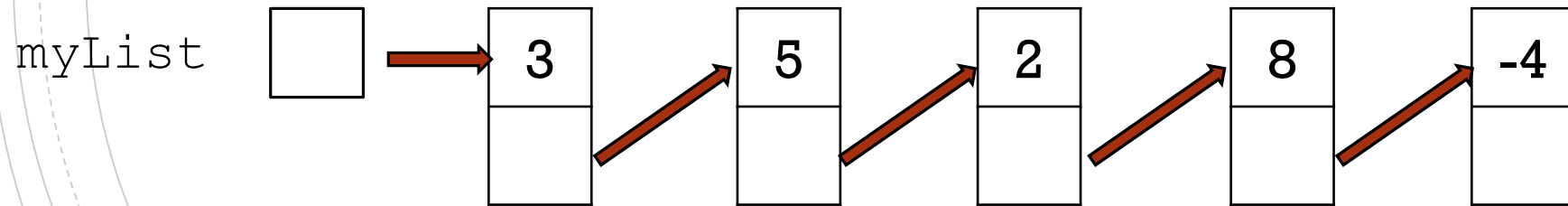
LINKEDLIST

- **LinkedList<E> class :**

```
private E val;
```

```
private LinkedList<https://eduassistpro.github.io/
```

Add WeChat edu_assist_pro



An orange paint roller with a red handle, positioned horizontally. The roller is partially covered in orange paint, which is dripping down the left side. The text "Coming Soon" is written in white on the orange surface of the roller.

Coming Soon

Assignment Project Exam Help

In the next

- Binary S <https://eduassistpro.github.io/>
Add WeChat edu_assist_pro