</talentlabs>

Lecture 1
Introduction to Algorithm

</talentlabs>

Agenga

- Algorithm
- Main types of Algorithms
  - Brute Force
  - Recursion

Introduction to Algorithm

Algorithm

- A methodology to solve problem, a description of the steps to solve a problem
- The focus is not about "coding", but the "thinking"
- There are no formulas, but some common way of thinking about a problem
- Practice more and read more is the best way to improve your skills

</talentlabs>

Algorithm Example

- Question: How do you find the biggest numbers in an array
- Answer:
    - Create a variable “`tempMaxNumber`”
    - Go through the numbers in the array one by one, if the number is bigger than `tempMaxNumber`, then update `tempMaxNumber` to that number
    - After going through the whole array, `tempMaxNumber` is the biggest number in the array

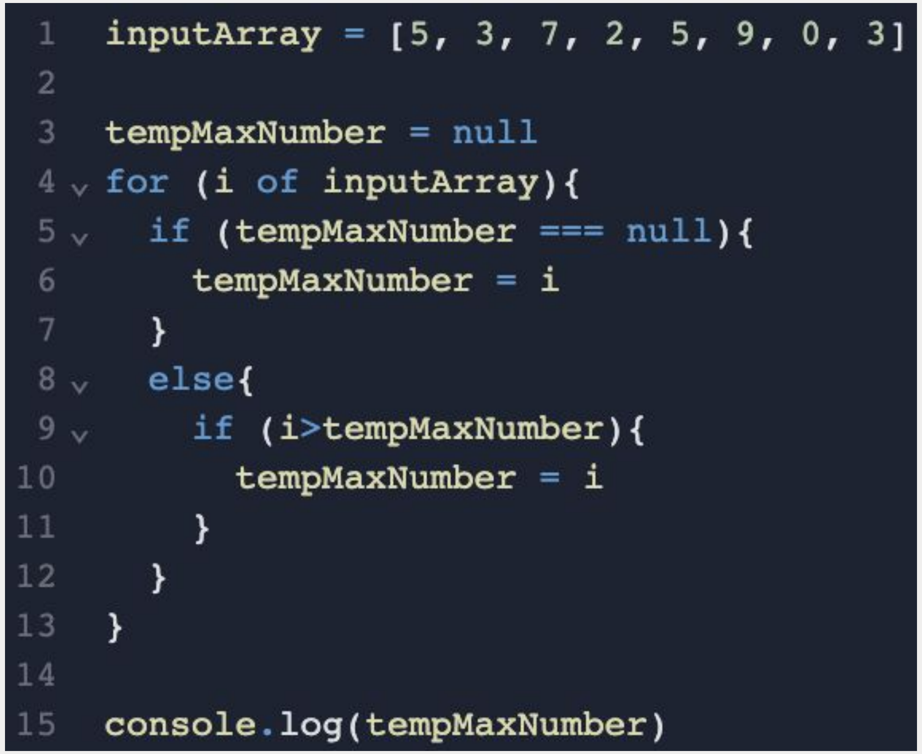
</talentlabs>

Algorithm Example Illustration

Input Array: `[5, 3, 7, 2, 5, 9, 0, 3]`

`tempMaxNumber`:

Translating the Algorithm to Code



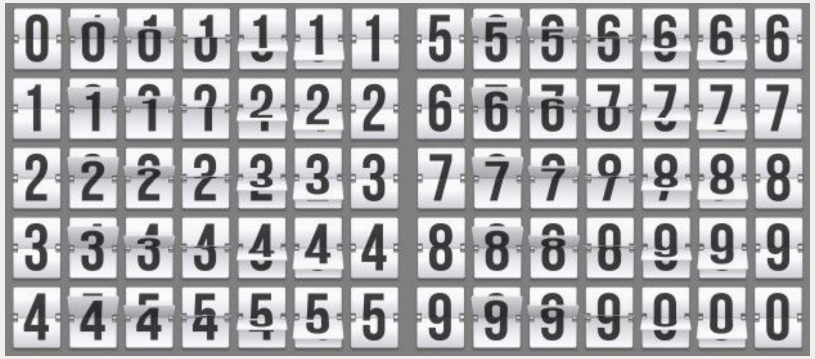
```
inputArray = [5, 3, 7, 2, 5, 9, 0, 3]

tempMaxNumber = null
for (i of inputArray){
  if (tempMaxNumber === null){
    tempMaxNumber = i
  }
  else{
    if (i>tempMaxNumber){
      tempMaxNumber = i
    }
  }
}

console.log(tempMaxNumber)
```

Common Algorithm Type 1 - Brute Force

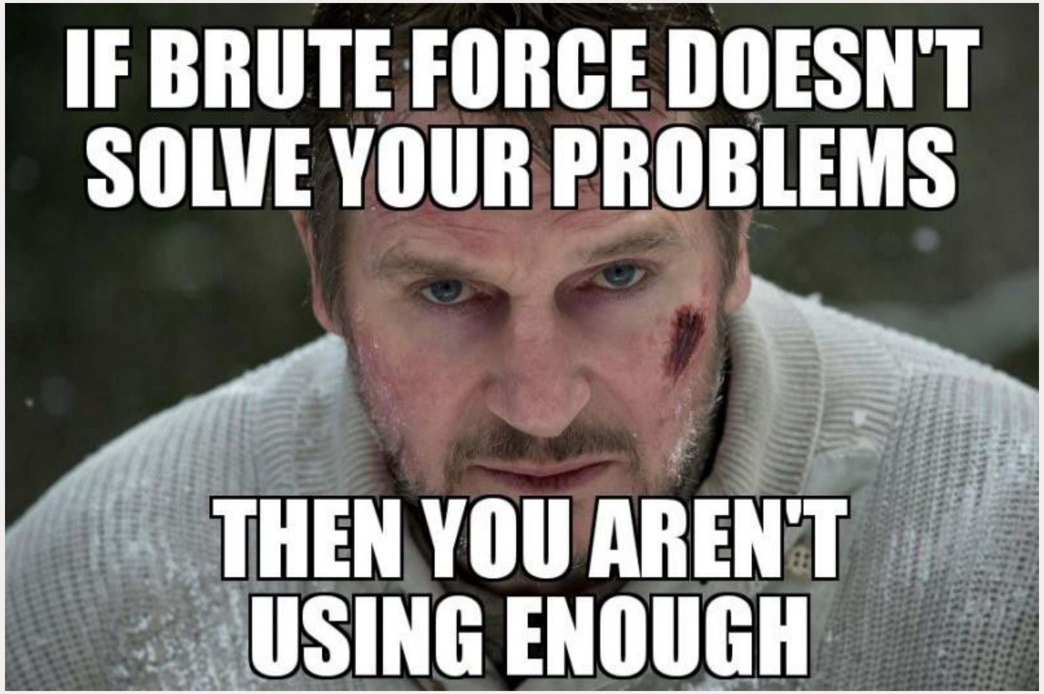
</talentlabs>

What is Brute force algorithm?

Brute Force Algorithms are straightforward methods of solving a problem that rely on computing power and trying every possibility rather than advanced techniques to improve efficiency.



What is Brute Force Algorithm?


IF BRUTE FORCE DOESN'T SOLVE YOUR PROBLEMS
THEN YOU AREN'T USING ENOUGH

Brute Force Algorithm Example 1

For example, imagine you have a small padlock with 3 digits, each from 0-9. You forgot your combination, but you don't want to buy another padlock.

Since you can't remember any of the digits, you have to use a brute force method to open the lock.

Brute Force Algorithm for Padlock

So you set all the numbers back to 0 and try them one by one: 000, 001, 002, 003, and so on until it opens. In the worst case scenario, it would take a maximum of 1000 tries to find your password.

</talentlabs>

It’s Not Stupid at all...

- In coding interviews, if you can solve the problem using brute force algorithm, you are already better than most of the junior developers!
- If you can’t think of a smarter way, just start with brute force.

</talentlabs>

Brute Force Algorithm Example 2

- Question: Given an array, how I can know if a particular letter exists in the array?
- Answer:
  - Check the elements in the array one by one in a for loop.
  - Terminate the loop and return True if there is a match.
  - If there are no match after looping through the array, return False.

</talentlabs>

Assignment Question 1

Brute Force Algorithm Example 2

Input Array: `[a, b, f, u, i, k, p, e, v, k, s]`

found:

</talentlabs>

Common Algorithm Type 2 - Recursion

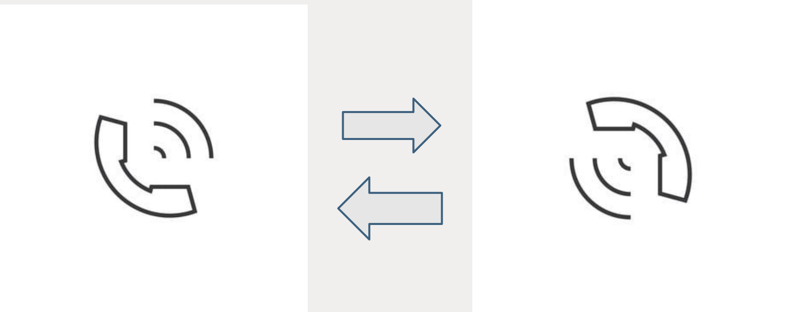
</talentlabs>

What is Recursive algorithm?

The process in which a function calls itself directly or indirectly is called recursion.

Recursion is useful for problems that can be represented by a simpler version of the same problem.

The smallest example of the same task has a non-recursive solution.



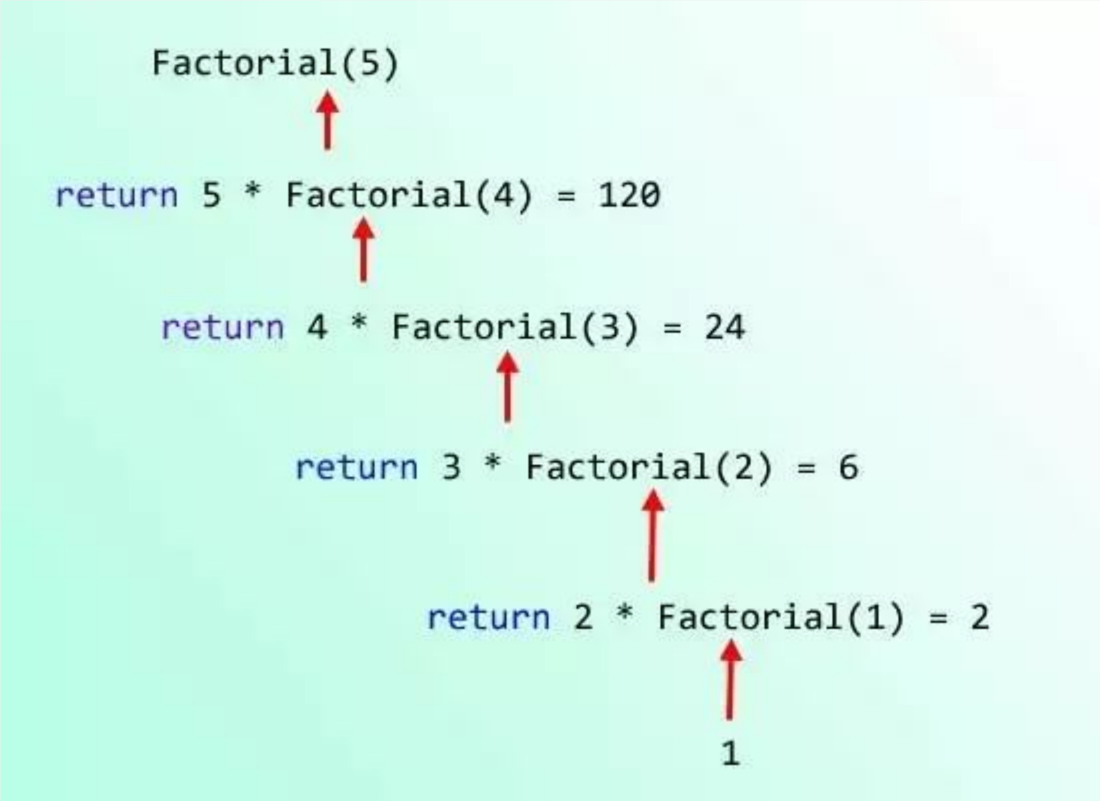
Factorial

5 Factorial = 5! = 5 * 4 * 3 * 2 *1

3! = 3 * 2 * 1 = 6

0! = 1
1! = 1

Example of Recursion algorithm



```
     Factorial(5)

return 5 * Factorial(4) = 120

     return 4 * Factorial(3) = 24

            return 3 * Factorial(2) = 6

                   return 2 * Factorial(1) = 2

                                1
```

JavaScript Recursive Function Exercise

The factorial function

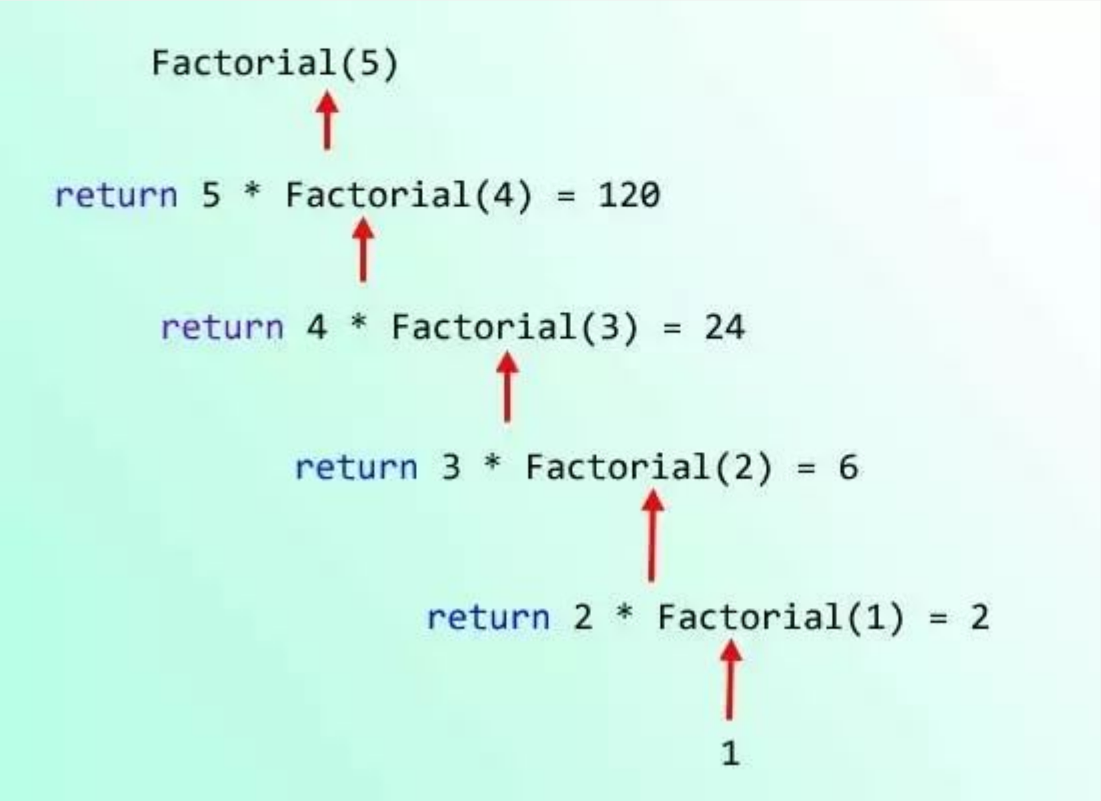
6! = 6*5*4*3*2*1

6! = 6 *5!

n! = n * (n-1)!

n! = 1 (if n = 0 or 1)

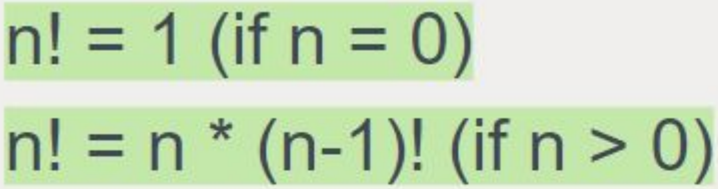
n! = n * (n-1)! (if n > 1)



```
   Factorial(5)

return 5 * Factorial(4) = 120

      return 4 * Factorial(3) = 24

            return 3 * Factorial(2) = 6

                  return 2 * Factorial(1) = 2

                              1
```

JavaScript Recursive Function Exercise

```
function factorial(n){
    if(n == 0 || n == 1){
        return 1;
    }else{
        return ????;
    }
}
```

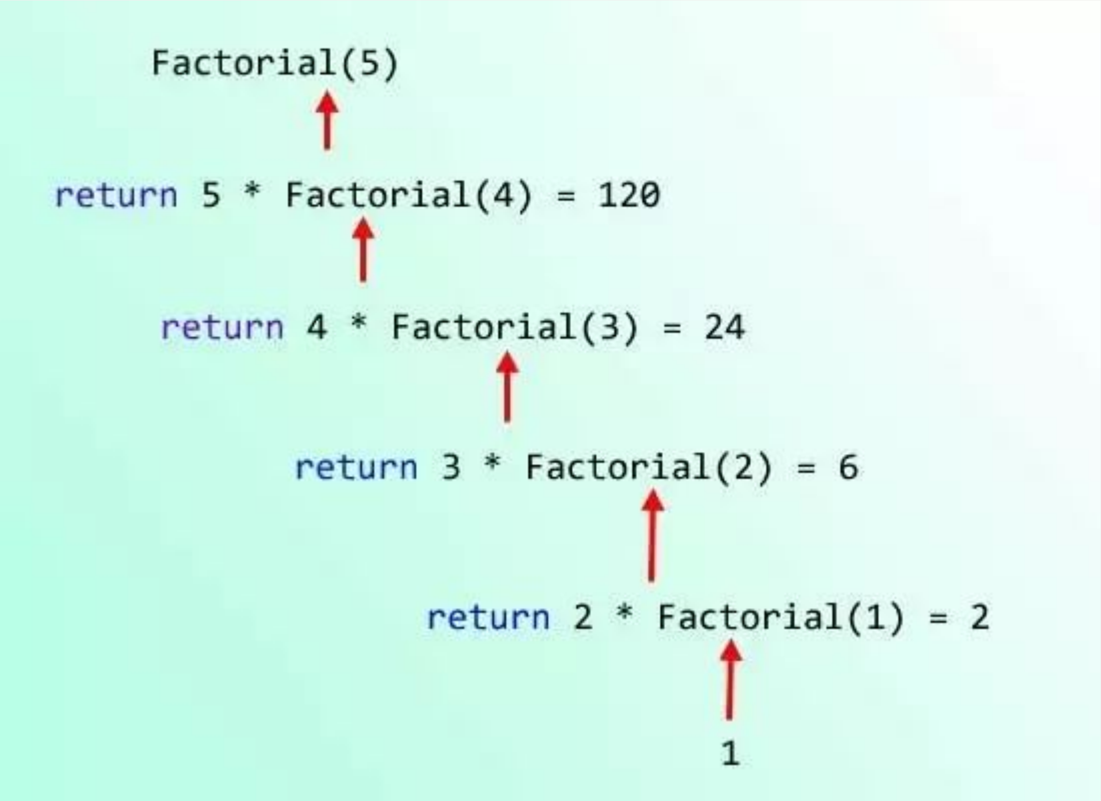


n! = 1 (if n = 0)

n! = n * (n-1)! (if n > 0)

JavaScript Recursive Function Exercise

```
function factorial(n){
    if(n == 0 || n == 1){
        return 1;
    }else{
        return n * factorial(n-1);
    }
}
```



```
Factorial(5)

return 5 * Factorial(4) = 120

return 4 * Factorial(3) = 24

return 3 * Factorial(2) = 6

return 2 * Factorial(1) = 2

1
```

Assignment Question 2

Take a look at this animation :D

Simple JavaScript Recursive Function Example

Suppose that you need to develop a function that counts down from a specified number to 1.

For example, to count down from 3 to 1:

5

4

3

2

1

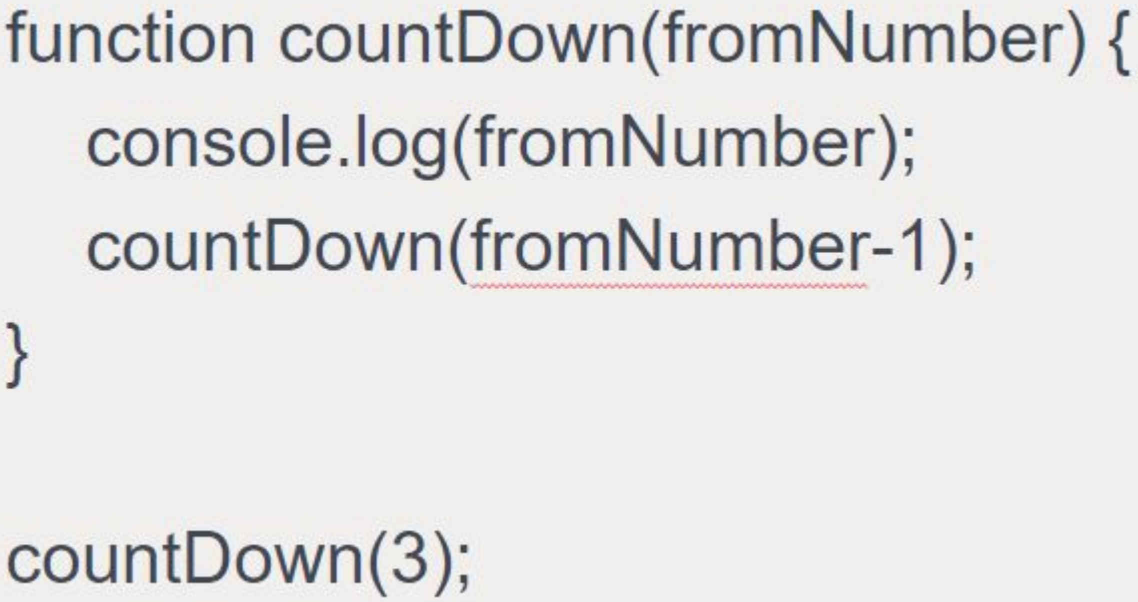
Simple JavaScript Recursive Function Example

```
function countDown(fromNumber) {
    console.log(fromNumber);
    countDown(fromNumber-1);
}

countDown(3);
```

Any Problem???

Simple JavaScript Recursive Function Example

That program doesn’t have the condition to stop calling itself !!!



```
function countDown(fromNumber) {
    console.log(fromNumber);
    countDown(fromNumber-1);
}

countDown(3);
```

Simple JavaScript Recursive Function Example

```
function countDown(fromNumber) {
    console.log(fromNumber);
    if (fromNumber === 0){
     return
     } else{
     countDown(fromNumber - 1)
     }
}
countDown(3);
```

The count down will stop when the next number is zero.

we can add an if condition to check this condition.

**The smallest example of the same task has a non-recursive solution(fromNumber = 0 is the non-recursive solution this time).