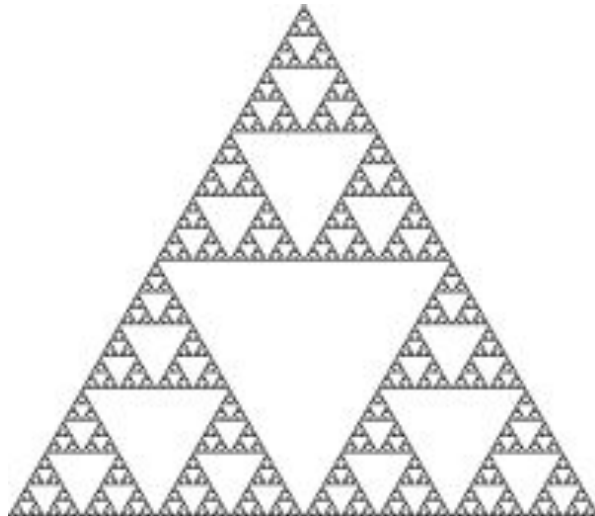


Recursividad

Esta presentación proporciona una exploración en el concepto de recursividad, y su uso en la creación de algoritmos. Cubre ejemplos y técnicas.

¿Qué es la recursión?

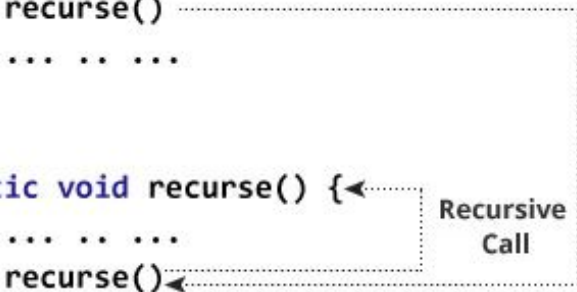
La recursión o recursividad es la posibilidad que tiene un cierto tipo de unidad o proceso de contenerse o aplicarse a sí mismo indefinidamente.



Recursividad en programación

La recursión es la técnica de hacer que una función se llame a sí misma. Esta técnica proporciona una manera de dividir problemas complicados en problemas simples que son más fáciles de resolver.

```
public static void main(String[] args) {  
    ... ..  
    recurse()  
    ... ..  
}  
  
static void recurse() {  
    ... ..  
    recurse()  
    ... ..  
}
```



Normal Method Call

Recursive Call

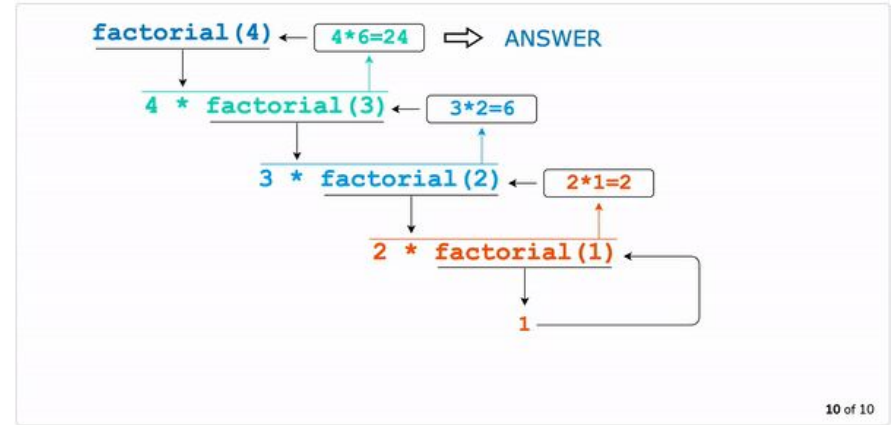
La recursividad puede ser un poco difícil de entender. La mejor manera de descubrir cómo funciona es experimentar con ella.

Recursividad

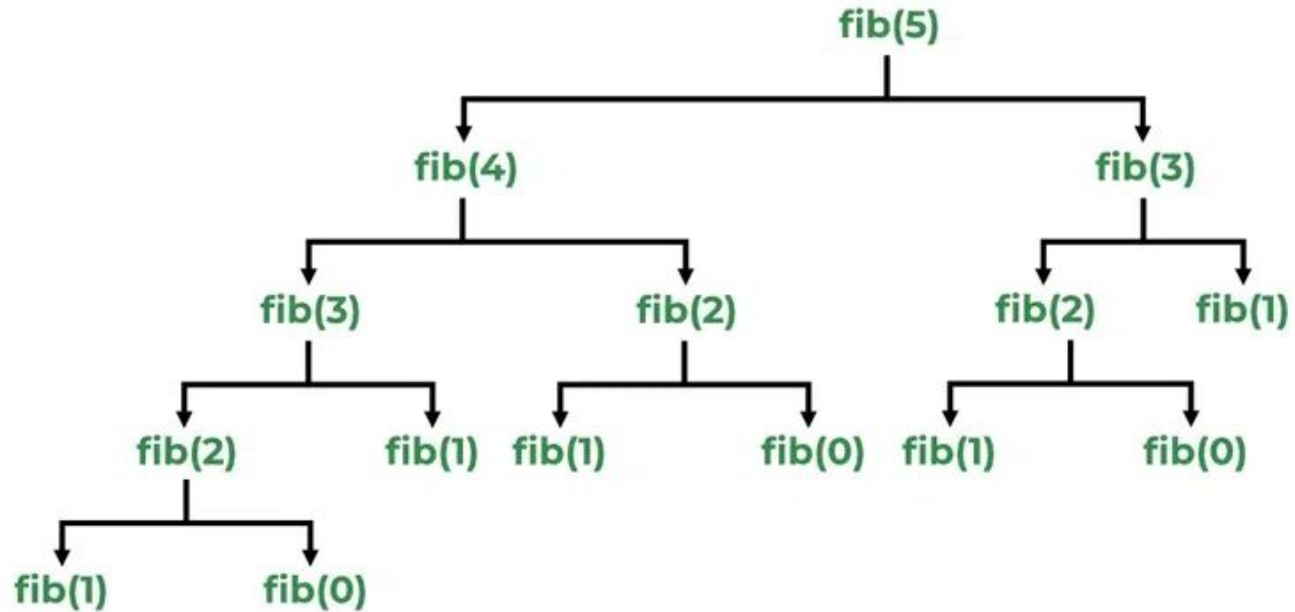
```
let getFactorial = (num) => {
  if (num == 1)
    return 1;

  else
    return num * getFactorial(num-1);
}
```

return



Recursividad



Recursividad en programación

En el programa recursivo, se proporciona la solución al caso base y la solución al problema mayor se expresa en términos de problemas más pequeños.

```
int fact(int n)
{
    if (n <= 1) // caso base
        return 1;
    else
        return n*fact(n-1); //caso recursivo
}
```

Problemas recursivos

1. Calcular el factorial de un número, Calcular la serie de fibonacci hasta un número n , Calcular la potencia de un número.
2. Poner una cadena de texto al revés.
3. Poner un arreglo al revés, Recorrer una lista.
4. Búsqueda binaria.
5. Resolver las torres de hanoi.
6. Otros

Recursividad - Notas importantes

1. Una solución recursiva es mucho más sencilla de leer y requiere menos tiempo de escritura, depuración y mantenimiento.
2. Sin embargo, cuando se realiza una llamada recursiva, se asignan nuevas ubicaciones de almacenamiento para variables en la pila de memoria. A medida que regresa cada llamada recursiva, las variables y parámetros antiguos se eliminan de la pila. Por lo tanto, la recursividad generalmente utiliza más memoria y suele ser lenta.

Recursividad - Notas importantes

Así como los bucles pueden encontrarse con el problema de los bucles infinitos, las funciones recursivas pueden encontrarse con el problema de la recursividad infinita.

La recursividad infinita es cuando la función nunca deja de llamarse a sí misma. Cada función recursiva debe tener una condición de parada, que es la condición en la que la función deja de llamarse a sí misma. Esta condición está asociada al caso base.

Recursividad - Búsqueda binaria

```
int binarySearch(int arr[], int left, int right, int x){  
    if (r >= 1) {  
        int mid = left + (right - left) / 2;  
        if (arr[mid] == x){ // Caso Base  
            return mid;  
        }  
        if (arr[mid] > x){  
            return binarySearch(arr, left, mid - 1, x); // Caso recursivo 1  
        }  
        return binarySearch(arr, mid + 1, right, x); // Caso recursivo 2  
    }  
    return -1;  
}
```

Recursos para práctica

<https://www.w3resource.com/java-exercises/recursive/index.php>

<https://www.geeksforgeeks.org/recursion-practice-problems-solutions/>

<https://codingbat.com/java/Recursion-1>

Gracias por su tiempo y atención 😊