Recursion [Day 8]

10 September 2024 19:03

Recursion - [Day 8]

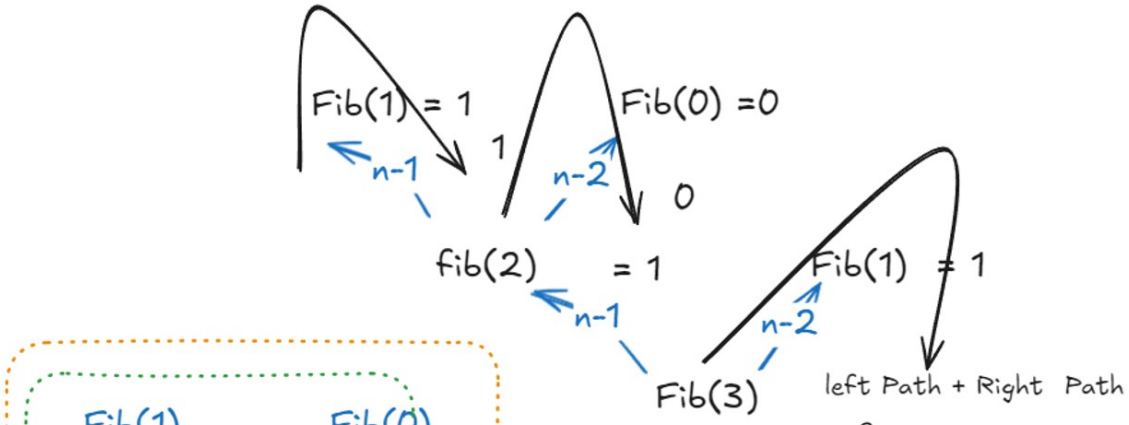
Generate n-th Fibonacci Number using Recursion.

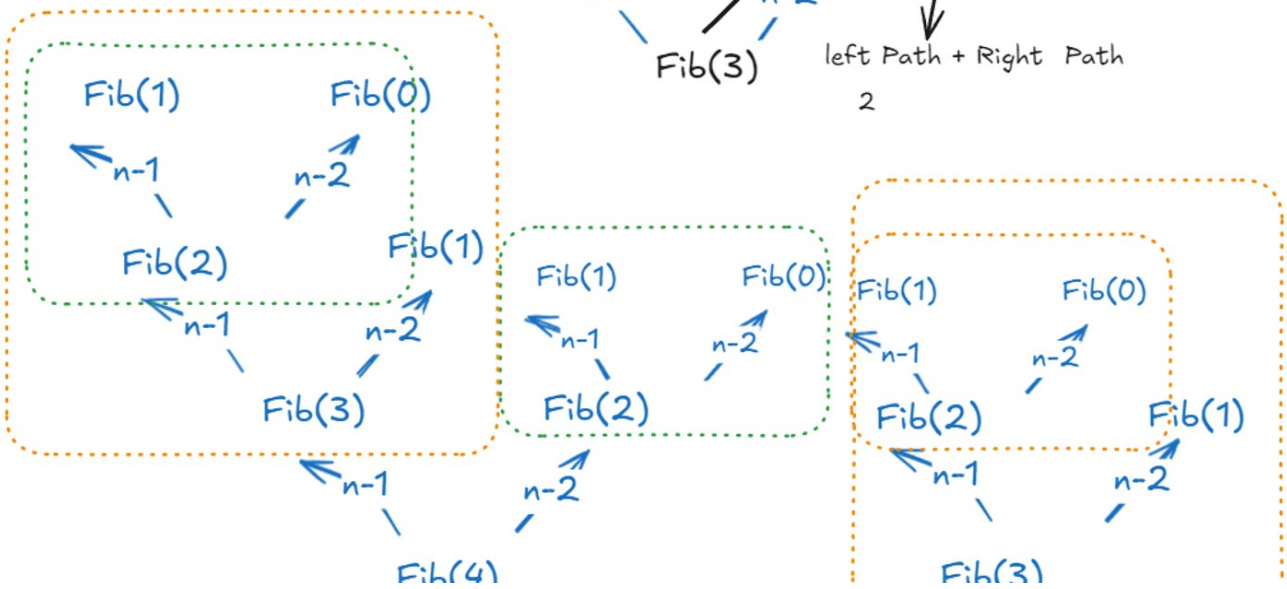
0 1 1 2 3 5 8 13 21 34

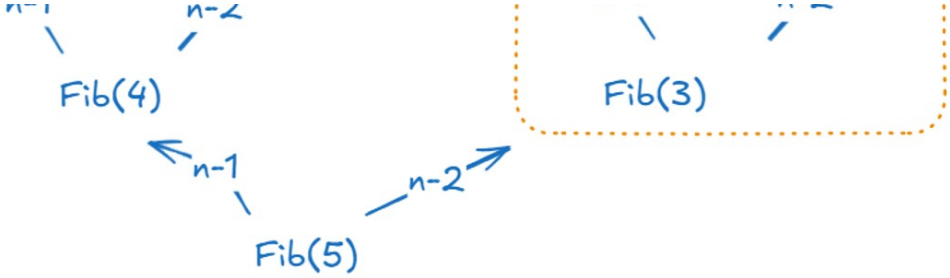
Fib(0) = 0 , Fib(1) = 1

Fib(n) = Fib(n-1) + Fib(n-2)

Fib(9) = Fib(8) + Fib(7) = 21 + 13 = 34
Fib(8) = Fib(7) + Fib(6) = 13 + 8 = 21
Fib(7) = Fib(6) + Fib(5) = 8 + 5 = 13
Fib(6) = Fib(5) + Fib(4) = 5 + 3 = 8
Fib(5) = Fib(4) + Fib(3) = 3 + 2 = 5
Fib(4) = Fib(3) + Fib(2) = 2 + 1 = 3
Fib(3) = Fib(2) + Fib(1) = 1 + 1 = 2
Fib(2) = Fib(1) + Fib(0) = 0+1 = 1








```java
import java.util.Scanner;
public class Fibonacci {
    public static int fibonacci(int num){
        if(num ==0){
            return 0;
        }else if(num == 1){
            return 1;
        }else{
            return fibonacci(num - 1) + fibonacci(num - 2);
        }
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the non-negative Integers : ");
        int num = scn.nextInt();
        int result = fibonacci(num);
        System.out.println(result);
    }
}
```

Problem Statement

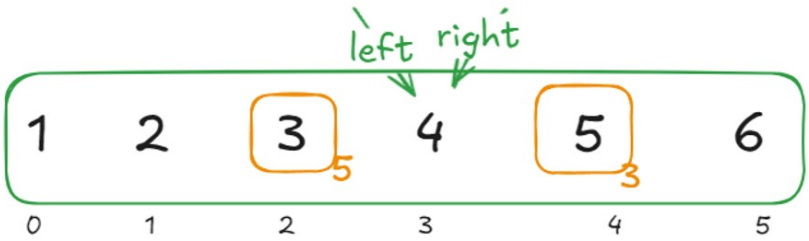
Problem: Reverse a Subarray

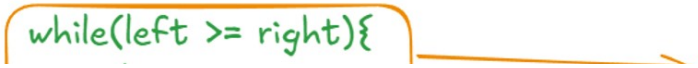
Given an array and two indices, `left` and `right`, reverse the elements of the array between these indices (inclusive).

Example

Example 1:

- **Input:**
  - Array: `[1, 2, 3, 4, 5, 6]`
  - Left index: `2`
  - Right index: `4`
- **Output:**
  - Array after reversing the subarray from index `2` to `4`: `[1, 2, 5, 4, 3, 6]`





```
while(left >= right){
    return;
}
```

Base Condition.

```
Swap the value ;
reverseSubarray(arr , left++ , right--);

Main(){
    reverseSubarray(arr , left , right)
}
```

```java
import java.util.Arrays;
import java.util.Scanner;
public class ReverseSubarray {
    public static void reverseSubarray(int[] arr, int left , int right){
        if(left >= right){
            return;
        }
        int temp = arr[left];
        arr[left] = arr[right];
        arr[right] = temp;

        reverseSubarray(arr, left + 1, right - 1);

    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the size of an array");
        int n = scn.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array");
        for(int i =0 ; i<n; i++){
            arr[i] = scn.nextInt();
        }
        System.out.println("Enter the Left Index : ");
        int left = scn.nextInt();
        System.out.println("Enter the Right Index : ");
        int right = scn.nextInt();

        if(left < 0 || right >=n || left > right){
            System.out.println("Invalid Indices");
        }else{
            reverseSubarray(arr , left , right);
            System.out.println("After Reversing the subarray");
            System.out.println(Arrays.toString(arr));
        }
    }
}
```

```
Enter the size of an array
6
Enter the elements of the array
1
2
3
```

```
1
2
3
4
5
6
Enter the Left Index :
2
Enter the Right Index :
4
After Reversing the subarray
[1, 2, 5, 4, 3, 6]
```

Problem Statement

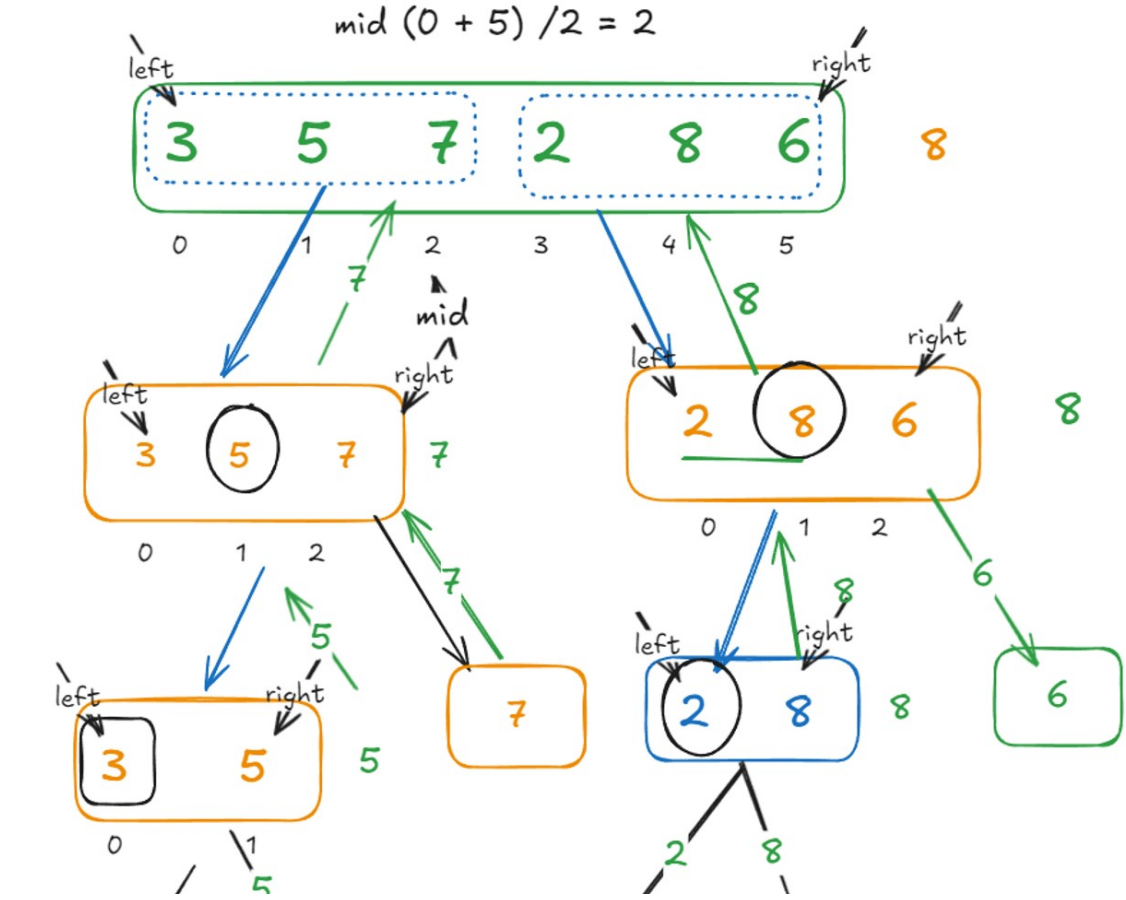
Problem: Find the Maximum Element in an Array

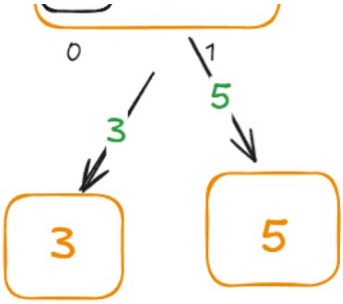
Given an array of integers, find the maximum element in the array using recursion.

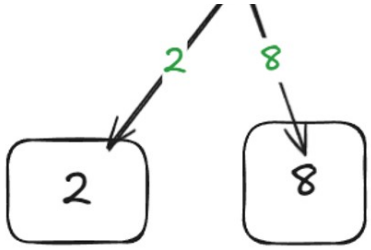
Example

Example 1:

- **Input:**
  - Array: `[3, 5, 7, 2, 8, 6]`
- **Output:**
  - Maximum Element: `8`



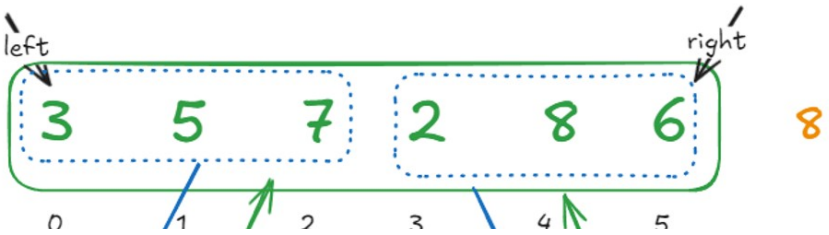


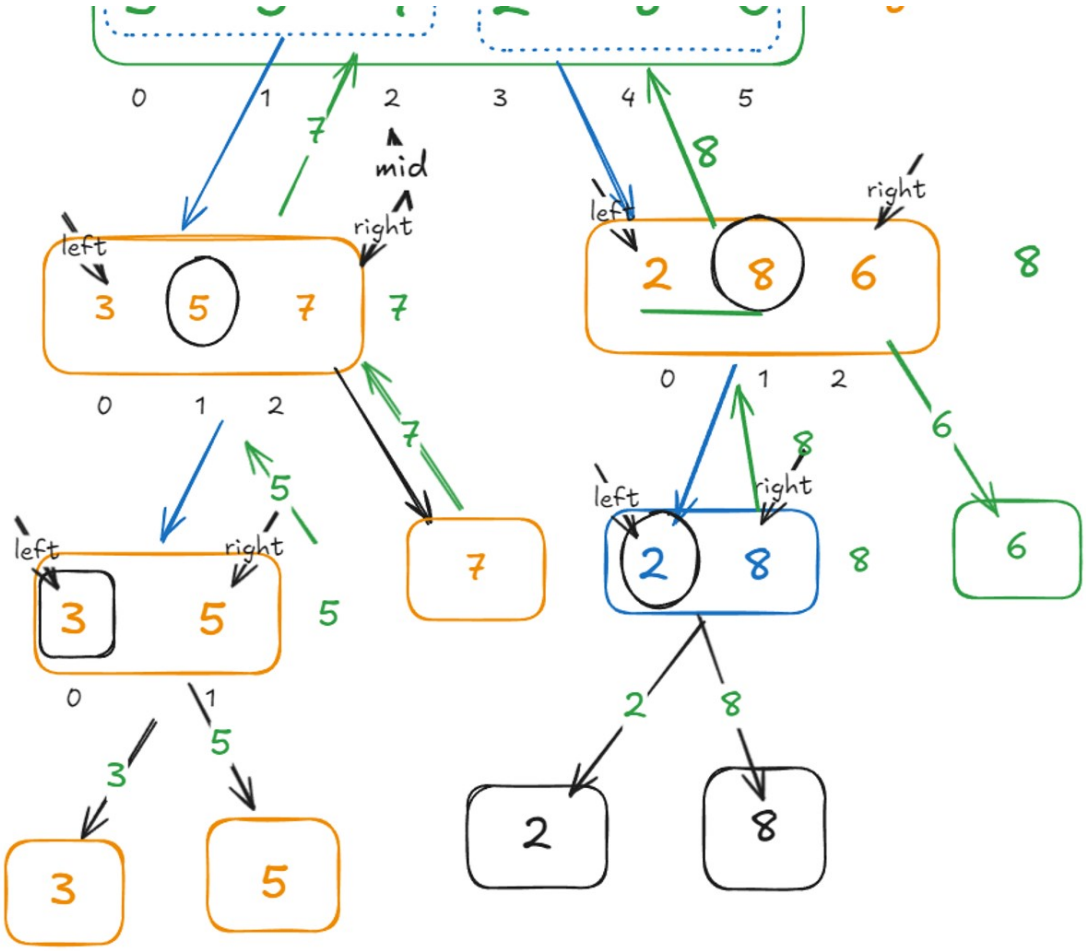



```java
import java.util.Scanner;
public class MaxElementRecursion {

    public static int FindMax(int[] arr , int left , int right){
        if(left == right){
            return arr[left]; // or arr[right] both are pointing to same value.
        }
        int mid = (left + right) /2;
        int leftMax = FindMax(arr, left , mid);
        int rightMax = FindMax(arr, mid + 1, right);
        return Math.max(leftMax, rightMax);
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the size of an array : ");
        int n = scn.nextInt();

        int[] arr = new int[n];
        System.out.println("Enter the elements of an array:");
        for(int i = 0 ; i< n; i++){
            arr[i] = scn.nextInt();
        }

        int max = FindMax(arr , 0 , n -1);
        System.out.println("The maximum element of an array is : " + max);
    }
}
```

```
Enter the size of an array :
6
Enter the elements of an array:
4
5
7
1
9
3
The maximum element of an array is : 9
```

```java
if(left == right){
    return arr[left]; // or arr[right] both are
}
int mid = (left + right) /2;
int leftMax = FindMax(arr, left , mid);
int rightMax = FindMax(arr, mid + 1, right);
return Math.max(leftMax, rightMax);
```





Problem Statement

Problem: Print the string "ALGOTUTOR" using recursion.

A L G O T U T O R

A L G O T U T O R

```
public class PrintingAlgoTutor {
    public static void printString(String str , int index){
        if(index == str.length()){
            return;
        }
        System.out.println(str.charAt(index));
        printString(str, index + 1);
    }
    public static void main(String[] args) {
        String str = "ALGOTUTOR";

        printString(str , 0);
    }
}
```

```
        printString(str , 0);
    }
}
```

Problem Statement:

Write a recursive function to calculate the power of a number in a linear manner. Given two integers `x` (the base) and `n` (the exponent), you need to compute `x^n` using recursion.

```
Input:
x = 2, n = 5

Output:
32
Explanation:
2^5 = 2 * 2 * 2 * 2 * 2 = 32
```

2^5 = 2 * 2^4 .
2^4 = 2 * 2^3.
2^3 = 2 * 2^2.
2^2 = 2 * 2^1
2^1 = 2 * 2^0
[x ^ n] , n ==0 return 1]

```
import java.util.Scanner;
public class PowerLinear {

    public static int power(int x , int n){
        if(n ==0){
            return 1;
        }
        int result = power(x, n-1);
        return x * result;
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the base (x) : ");
        int x = scn.nextInt();
        System.out.println("Enter the exponent (n) : ");
        int n = scn.nextInt();

        int result = power(x , n);
        System.out.println( x + " ^ " + n + " = " + result);
    }

}
```

```
ceptionMessages   -cp   C:\U
Enter the base (x) :
2
Enter the exponent (n) :
5
2 ^ 5 = 32
```

Problem Statement:

Write a recursive function to calculate the power of a number in a logarithmic manner. Given two integers `x` (the base) and `n` (the exponent), compute `x^n` using recursion, optimizing the solution to have logarithmic time complexity.

Write a recursive function to calculate the power of a number in a logarithmic manner. Given two integers `x` (the base) and `n` (the exponent), compute `x^n` using recursion, optimizing the solution to have logarithmic time complexity.

Input:

x = 2, n = 5

Output:

32

Explanation:

2^5 = 32

x^ n = x ^ (n-1) * x ^ (n-2)

x ^ n = x ^ n/2 * x ^ n /2. - even
x ^ n = (x ^ n/2 * x ^ n /2) * x [odd]

2 ^ 5 = (2 ^2 * 2 ^2) * 2 = 16 * 2 = 32

2 ^ 4 = 2 ^2 * 2 ^2 = 4 * 4 = 16

```
import java.util.Scanner;
public class PowerLogarithmic {
    public static int power(int x , int n){
        if(n ==0){
            return 1;
        }
        int halfpower = power(x, n/2);
        if(n % 2 ==0){
            return halfpower * halfpower;
        }else{
            return x * halfpower * halfpower;
        }
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the base (x) : ");
        int x = scn.nextInt();
        System.out.println("Enter the exponent (n) : ");
        int n = scn.nextInt();

        int result = power(x , n);
        System.out.println( x + " ^ " + n + " = " + result);
    }

}
```

```
if(n ==0){
    return 1;
}
int halfpower = power(x, n/2);
if(n % 2 ==0){
    return halfpower * halfpower;
}else{
    return x * halfpower * halfpower;
}
```

2 ^ 8 = 2 ^ 4 * 2 ^ 4 = 16 * 16 = 256.
2 ^ 4 = 2 ^ 2 * 2 ^ 2 = 4 * 4 = 16
2 ^ 2 = 2 ^ 1 * 2 ^ 1 = 2 * 2 = 4

```
2 ^ 4 =  2 ^ 2  * 2 ^ 2 = 4 * 4 = 16
2 ^ 2 =  2 ^ 1  * 2 ^ 1 = 2 * 2 = 4
2 ^ 1 =  2 * 2 ^0   * 2 ^ 0 = 2 * 1 * 1 =2
```

$$2^a * 2^b = 2^{(a+b)};$$

$$2^4 * 2^3 = 2^{(4+3)} = 2^7.$$

Completed

Attendance Code: E782F53A

Problem Statement:

Write a recursive function that takes an array of integers and prints all its elements, one at a time. The function should print elements starting from the first element to the last element.

```
Input:
Array = [10, 20, 30, 40, 50]

Output:
10
20
30
40
50
```

```
import java.util.Scanner;
public class DisplayArray {

    public static void displayArray(int[] arr , int idx){
        if(idx == arr.length){
            return;
        }

        System.out.println(arr[idx]);
        displayArray(arr, idx+1);
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the number of Elements  in  the array : ");

        int n = scn.nextInt();
        int[] arr = new int[n];
```

```
        int n = scn.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of an array : ");
        for(int i = 0 ; i < n ; i++){
            arr[i] = scn.nextInt();
        }
        System.out.println("Array Elemets are : ");
        displayArray(arr, 0);
    }
}
```

"print the array element in reverse order."

```
import java.util.Scanner;
public class DisplayArrayReverse {

    public static void displayArrayReverse(int[] arr , int idx){
        if(idx < 0){
            return;
        }

        System.out.println(arr[idx]);
        displayArray(arr, idx - 1);
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the number of Elements  in  the array : ");

        int n = scn.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of an array : ");
        for(int i = 0 ; i < n ; i++){
            arr[i] = scn.nextInt();
        }
        System.out.println("Array Elemets are : ");
        displayArrayReverse(arr , n -1);
    }
}
```