```
def sumaRecursionLineal(l:List[Int]):Int = {
  l match {
    case Nil => 0
    case x::xs => x + sumaRecursionLineal(xs)
  }
}
```

```
sumL(List(1,2,3,4,5)) --> 1 + sumL(List(2,3,4,5)) <1>
sumL(List(2,3,4,5)) --> 2 + sumL(List(3,4,5)) <2>
sumL(List(3,4,5)) --> 3 + sumL(List(4,5)) <3>
sumL(List(4,5)) --> 4 + sumL(List(5)) <4>
sumL(List(5)) --> 5 + sumL(nil) <5> = 5
```

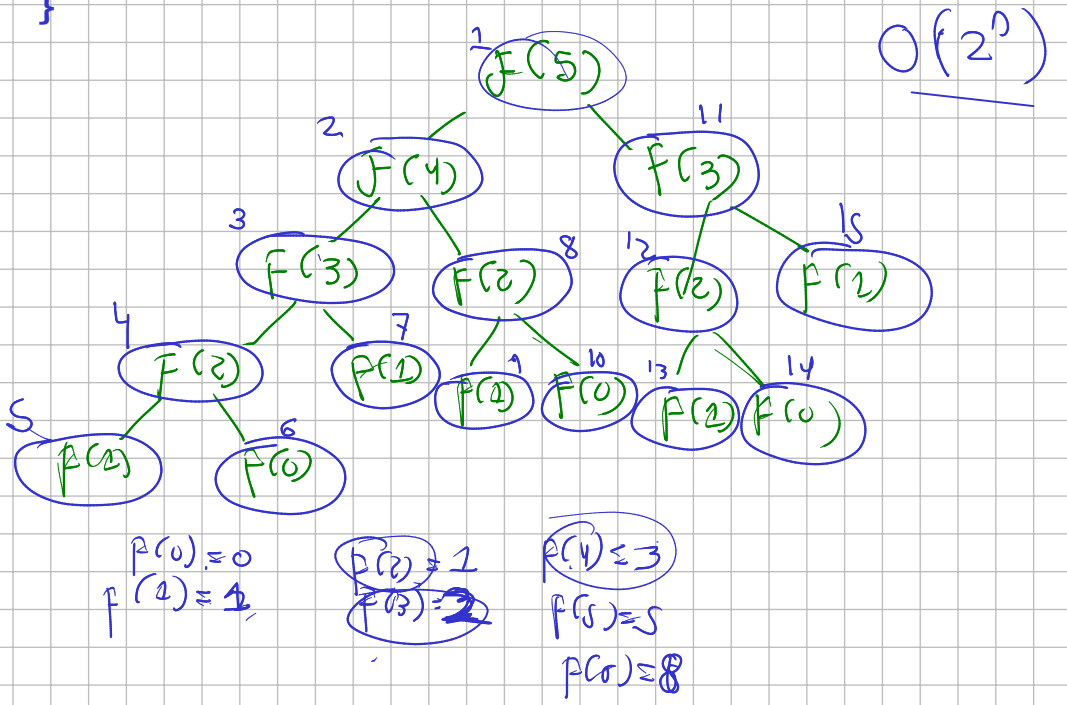
14+1=15
12+2=14
3+9=12
4+5=9

```
@tailrec
final def sumaRecursionCola(l:List[Int], acc:Int=0):Int = {
  l match {
    case Nil => acc
    case x::xs => sumaRecursionCola(xs, acc+x)
  }
}
```

```
sumaRC(List(1,2,3,4,5),0)
sumaRC(List(2,3,4,5), 1)
sumaRC(List(3,4,5),3)
sumaRC(List(4,5),6)
sumaRC(List(5), 10)
sumaRC(Nil, 15)
```

Recursión de árbol

```
def fibunnacci(n:Int):Int = {
  n match {
    case 0 => 0
    case 1 => 1
    case _ => fibunnacci(n-1) + fibunnacci(n-2)
  }
}
```



O(2^n)

f(0) = 0
f(1) = 1
f(2) = 1
f(3) = 2
f(4) = 3
f(5) = 5
f(6) = 8