Recursion

递归算法三条重要的定律：

1、递归算法必须有个基本结束条件；

2、递归算法必须递归地调用自身 。

3、递归算法必须改变自己的状态、并向基本结束条件演进；

1. **一般递归：**

- 在一般递归中，函数在调用自身之前可能会执行其他操作。
- 每个递归调用都需要在调用栈上占用空间，因此可能导致栈溢出（stack overflow）的问题，特别是在递归深度较大的情况下。

2. **线性递归：**

- 线性递归是一种特殊的递归形式，其中函数在递归调用之前不执行其他操作，直接调用自身。
- 典型的线性递归形式是朝一个方向递归，最终到达基本情况。

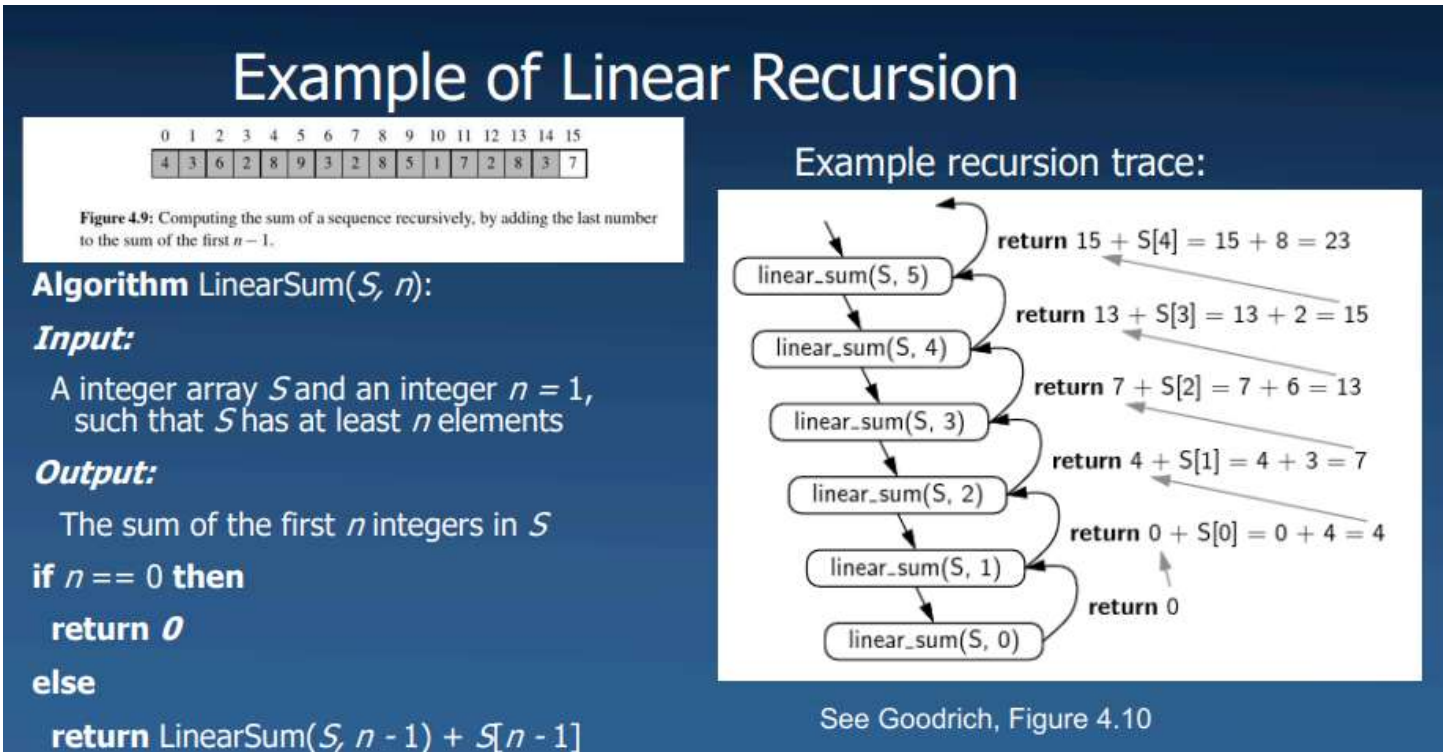
3. **尾部递归：**

- 尾部递归是一种特殊的线性递归，其递归调用是函数的最后一条语句。
- 尾部递归对编译器来说有优化的可能，因为尾部递归可以通过一些编译器优化技术（如尾调用优化）转化为迭代形式，减少栈的使用。

简而言之，线性递归是递归的一种特殊形式，而尾部递归是线性递归的特殊形式，具有更好的优化可能性。在编写递归函数时，尽量将递归调用设计成尾部递归形式，以提高性能并减少栈空间的使用。

Linear Recursion:线性递归

每个可能的递归调用链最终必须到达一个不使用递归的基本情况



线性递归是指在函数调用过程中，函数直接或间接地调用自身的过程。

尾部递归：

尾部递归是一种特殊的线性递归形式，在函数的最后一个操作是对自身的递归调用，并且没有其他操作或表达式需要在递归调用之后执行。在尾部递归中，递归调用是整个函数的最后一步操作。

Tail Recursion can be resource intensive尾部递归是资源密集性的

尾部递归可以转化成循环当中进行