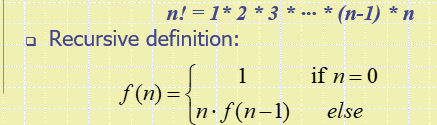
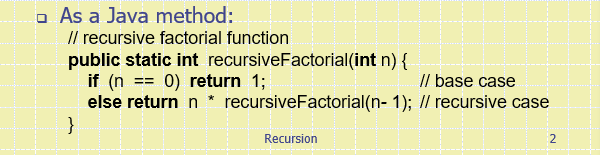
Recursion：指一个method重新call自己





首先要设置一个Base case/stopping case，当没有更多recursive calls的时候会进行这些case

每个recursive call必须要最终达到base case,不然就会无限循环

Recursive calls：call method本身，

Recursion1.标准recursion。

Recursion2. 没有设置好base case，导致无限循环

运行一个JAVA程序会激活一个private memory area: stack栈。

Stack的作用是跟踪正在被激活的mmethod

当一个method被激活时，他的所有信息（参数，当地变量，PC。。。）都被存在一个栈的小框架里（frame）.

当一个method return以后，这个frame就会被stack移除。

Heap：堆，是另外一个memory area，它用来在运行时动态分配内存。（比如我们用new 来创造一个新object的时候）

通常栈和堆在内存中相互竞争。

因此，recursion有可能快速消耗所有游泳空间来导致stack的溢出

RECURSION3.如果超多recursive call（heap过多消耗内存）导致stack overflow(分别输入了1 100 800 7000)

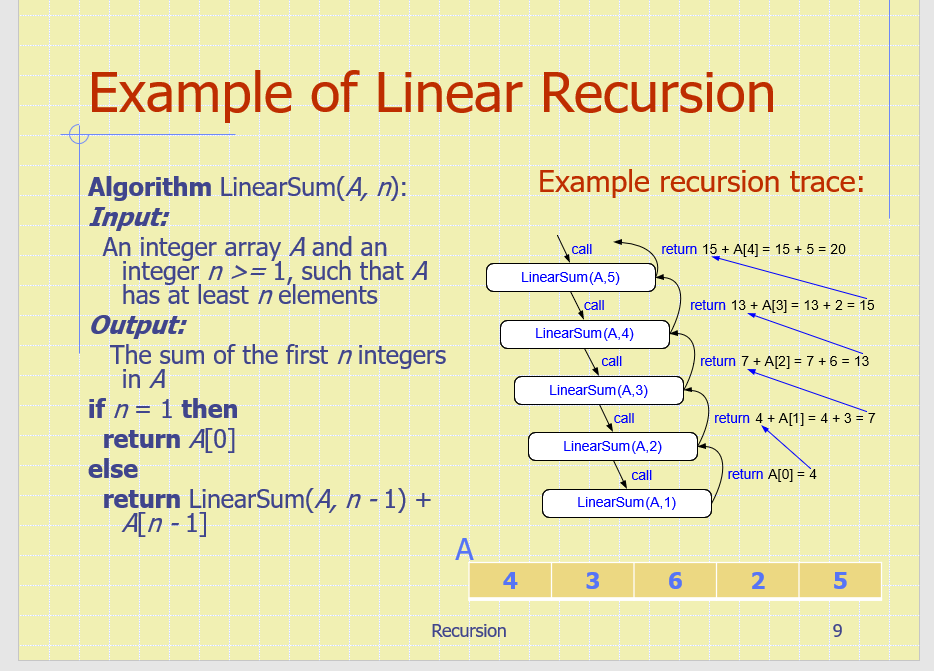
Linear：在一个method里只call自己一次

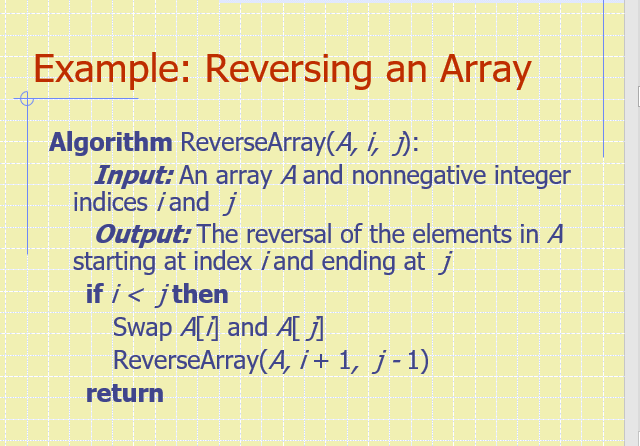
Linear Recursion:最简单的一种recursion，Method最多制造一个recursive call当他被激活的时候

当问题是从头到尾或从尾到头并且相同数据结构时非常有用

比如：得到一个array里n个值的和，可以看成两步。1.得到n-1的和加上最后一个元素的值

2.如果一个array只有一个元素，那么和就是A[0]的值



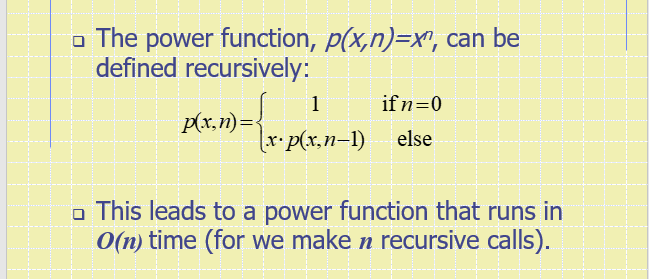


Defining Arguments for recursion描述recursion里的参数

在创造recursion的时候有时需要额外的参数

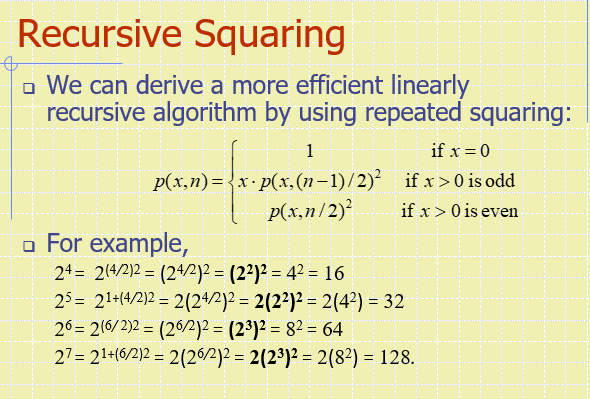
比如array reversal method,我们用了而不是

比如计算平方

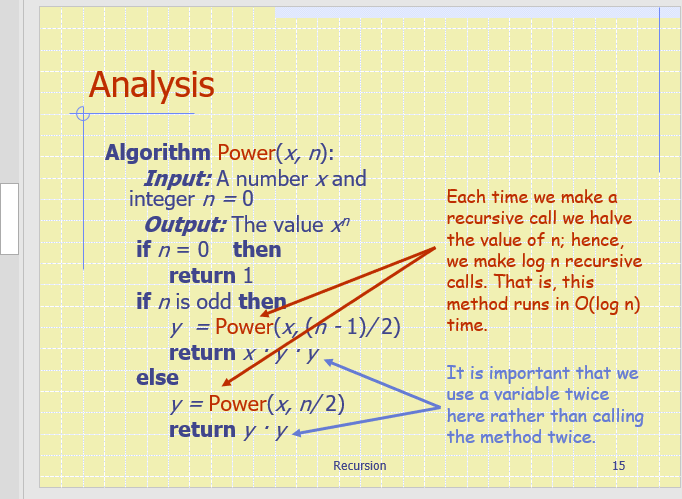


基本线性就是O（n）

我们可以通过重复平方打得到更有效率的线性算法



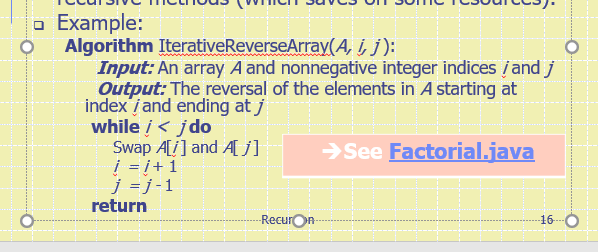
就是右上角是偶数，算他的一半然后平方，奇数，总的减一提出来单独乘然后一半再平方

  
奇数时，POWER=X乘以N-1/2的平方，偶数时，就等于除以2的平方

Tail Recursion

Tail Recursion 出现在线性recursive method把他的recursive call作为最后一步的时候，就比如说array reversal method

这类method可以轻松的转化成没有recursive的method(节省了资源)



Factorial.java：普通的recursion和tail recursion的区别，tail recursion不会在本体里面recursion，而是在本体里面用了另外一个辅助method，辅助method用了recursion: 意思是左边的每次小1，右边的每次×当前n,res指的是当前阶乘

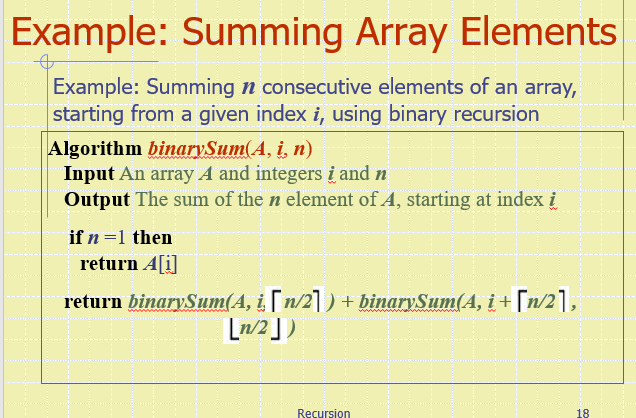
Binary recursion二元递归

二元递归出现在有且只有两个recursive call对应每个非base case(组成方式是base case+两种不同情况下的recursive call)

通过解决一个问题的两半

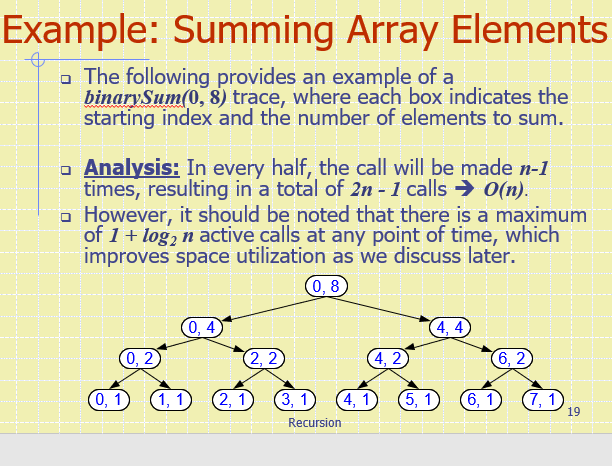
比如说计算一个大小为n的array，

1. 前半部分递归求和
2. 后半部分递归求和
3. 两个值相加



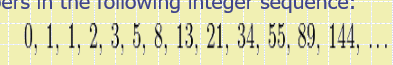
求从i开始的n个数的和 n=1的时候，returnA【I】,不然每次除以2，左边的永远是i开头

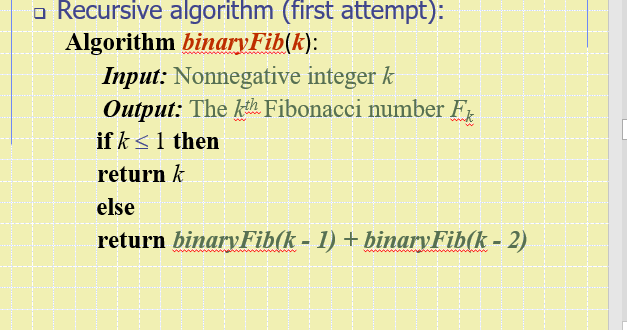
二元递归法每一半会call n-1次，总共2n-1次



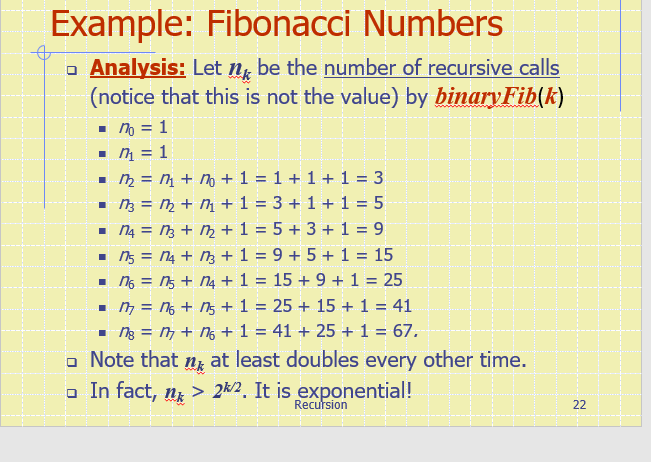
但是space complexity小了一半，虽然也是O（n）

Fibonnaci数





最直白的写法，每次return他前面两个数



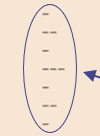
这样会导致重复计算

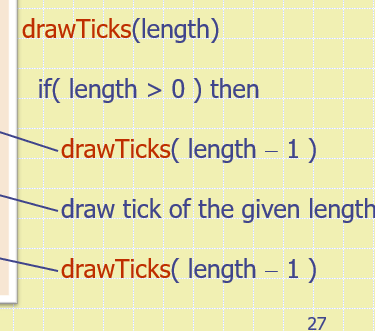
主要问题在于，虽然他看起来用了二元法（二元法不是指的分成一半一半不停分，而是指两个recursion然后相加）

所以这个用binary不好，还是用Linear

LINEAR FIB.JAVA里面有解读

Draw English Ruler的那个

怎么画出这个吊东西

注意if后面那三句话是加在一个花括号里的

所以他总结成了，如果length大于零，它上面会进行length-1的recursive，然后Print自己的长度，然后下面进行length-1的recursive

以3为例子

就是recursive(2)

3

Recursive(2)

Recursive(2)又是

Recursive1

2

Recursive1

总的来说就是

1

2

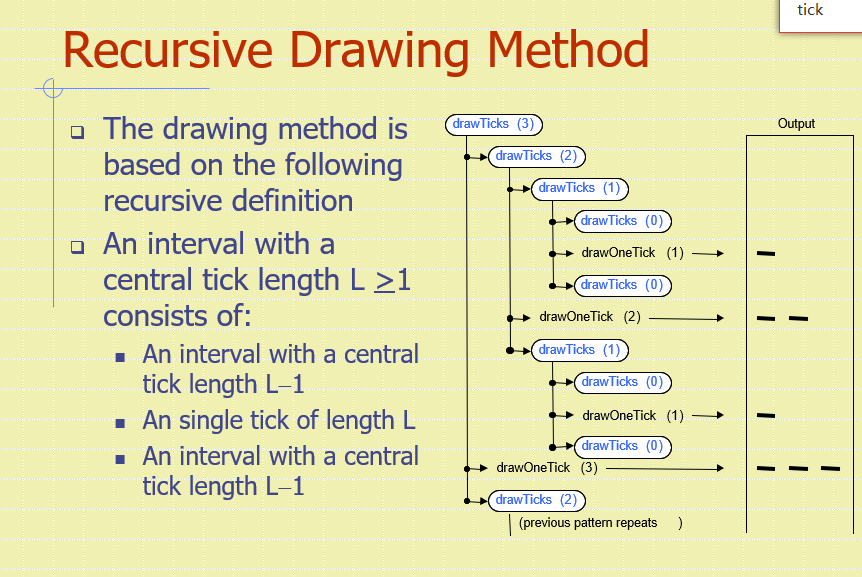
1

3

1

2

1



笔记：

1.DISPLAY(n-1)

s.p(n)

2. s.p.(n)

Display (n-1)

哪个更好，第二个，因为没有S.P(n)在等待