Java的基本数据类型

Byte Int short long float double char Boolean

算术符

Java流程控制:顺序，分支，循环

/\*\*

\* 记录操作执行总时间.

\*

\* @param <T> the generic type

\* @param clazz the clazz

\* @param methodName the method name

\*/

public <T> void recordTime(Class<T> clazz, String methodName) {

long start = System.currentTimeMillis();

System.out.println("start: " + start);

Method[] declaredMethods = clazz.getDeclaredMethods();

for (Method method : declaredMethods) {

String name = method.getName();

if (name.equals(methodName)) {

try {

method.invoke(clazz.newInstance());

} catch (Exception e) {

e.printStackTrace();

}

}

}

long end = System.currentTimeMillis();

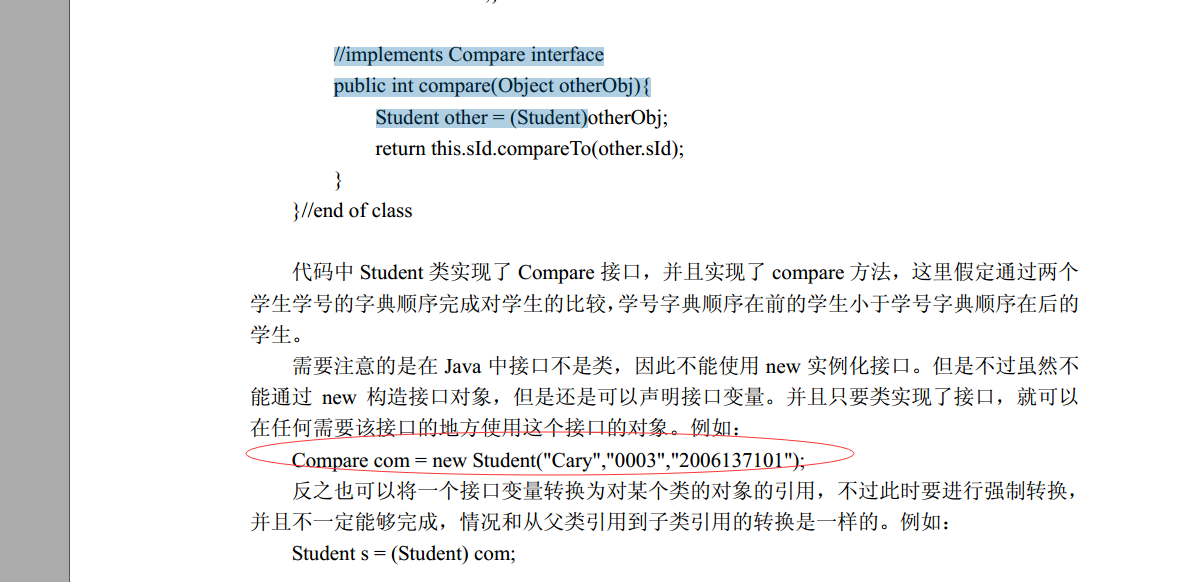
System.out.println("end: " + end);

System.out.println("duration: " + (end - start) + " ms");

}

This关键字：调用其他的构造方法，对隐式参数的引用

Super关键字：调用父类构造方法和父类方法



这种场景怎么使用？

Exception：处理异常的两种方法，try 。。。catch。。。； throws 或者throw 抛出异常

操作符new 的作用：为对象开辟足够的内存空间

数据结构：

数据

数据元素：数据的基本单元，数据集合的个体

数据对象：

数据结构： 逻辑机构 ；存储结构

逻辑结构：集合，线性结构，树形结构，图状结构

存储结构：数据元素本身的存储，以及数据元素之间的额关系表示

二元组

图形

线性结构 linearity：除了第一个元素外，每个数据元素有却只有一个直接前驱元素，除了最后一个元素外，每个数据元素有且只有一个直接后续元素，这种数据元素之间一对一关系

树形结构tree：除了第一个元素外，每个数据元素有且只有一个直接前驱元素，但可以有对个直接后续元素，这种数据元素之间1对N的关系

图状结构graph：这种数据元素之间M对N的关系

数据元素之间的关系：数据映像，非数据映像 –》

顺序存储结构和链式存储结构

数据类型：一组性质相同的数据元素的集合和加在这个集合上的操作

作用：隐藏计算机硬件及其特性和差别，使硬件对于用户来说是透明的，即用户不需要知道数据类型是怎么实现的而可以使用它；用户可以使用数据类型使用的操作，而方便的解决问题

抽象数据类型ADT(数据对象，关系集，操作集)：由一组数据模型和该数据模型的一组操作

算法：指令的集合，为解决特定问题的一系列操作。有着明确的计算过程，一一组数据进行输入，并产生一组数据集合作为输出

五个特性：输入，输出，可行性，有穷性，确定性

可行性：算法的每条指令都是可以实现的，均可在有限的时间内完成

有穷性：算法的指令是有限，而且每条指令都可在有限的时间内完成

确定性：对于每一条合法的输入，其输出都是唯一的

确定性算法；随机算法

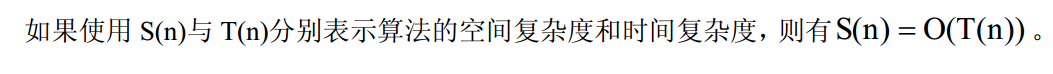
算法的时间复杂度（渐进时间复杂度）：如选择排序的时间复杂度为Ο(n^2)

Ο:运行时间的上限，即时间复杂度的上限

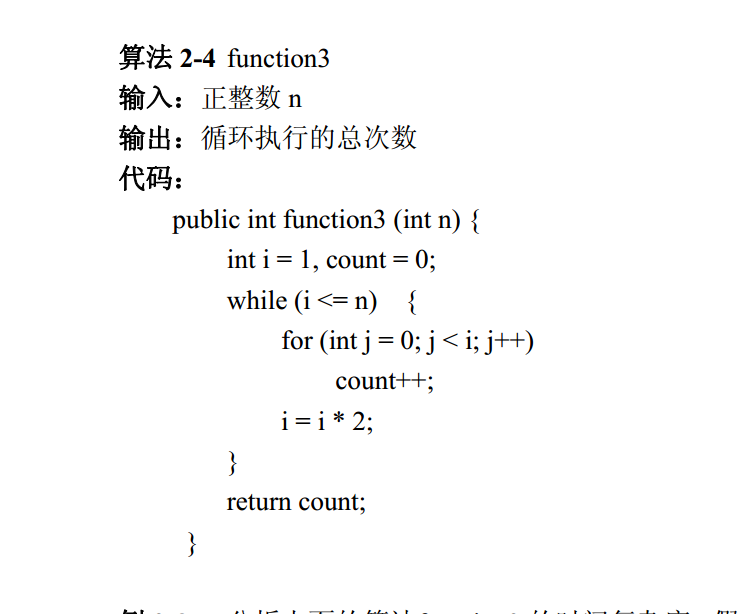
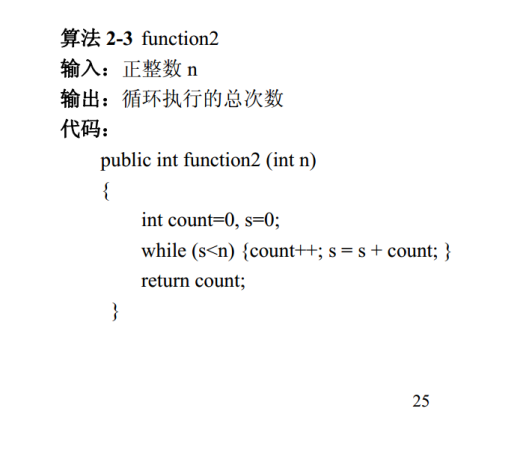
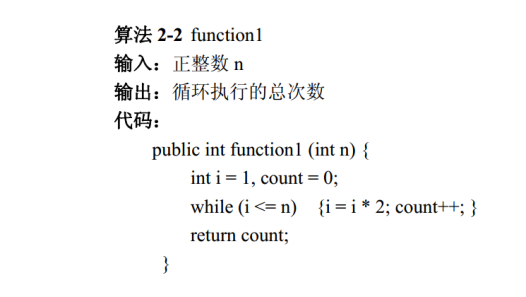
Ω：在时间复杂度常数因子的范围内给出时间复杂度的下界Ω(n^2)

θ：算法时间复杂度的精确阶

空间复杂度：为了求解实例而求解的步骤所使用的存储空间的数目但并不包括存储输入实例的空间



计算循环次数：

分析最高频度的基本操作：

最好，最坏和平均分析：

均摊分析

线性结构：

线性表：n个类型相同的元素的有序集合

数组是一种具体的数据结构

线性表是一个抽象的数据类型，可以通过数组来实现

线性表是元素之间有一对一线性关系的数据元素的集合，数组是一组数据元素与其下标的一一映射

List接口：

Strategy接口：实现各种不同元素相互之间独立的比较策略

线性表的顺序存储：用一组地址连续的存储单元来存储数据元素

LOC(a0) 表示0号元素的a0的存储地址，通常称为线性表的起始地址

特点：以数据元素在机内存储地址相邻来表示线性表种数据元素间的逻辑关系

线性表的链式存储：避免了连续的单元存储的缺点，执行插入，删除运算时，不在需要移动元素而腾出空间，但需要在每个存储单元中设置指针来表示元素之间的逻辑关系，因而增加了额外的存储空间

单链表：每个存储单元有两个域：数据域和指针域

链表中每个结点的next的引用都相当于一个指针，指向下一个结点，借助这些结点，我们可以从链表的首结点移动到尾结点，这就是单链表（single linked list）

只能通过直接前驱找到后续结点，而无法通过后续结点找到前驱结点

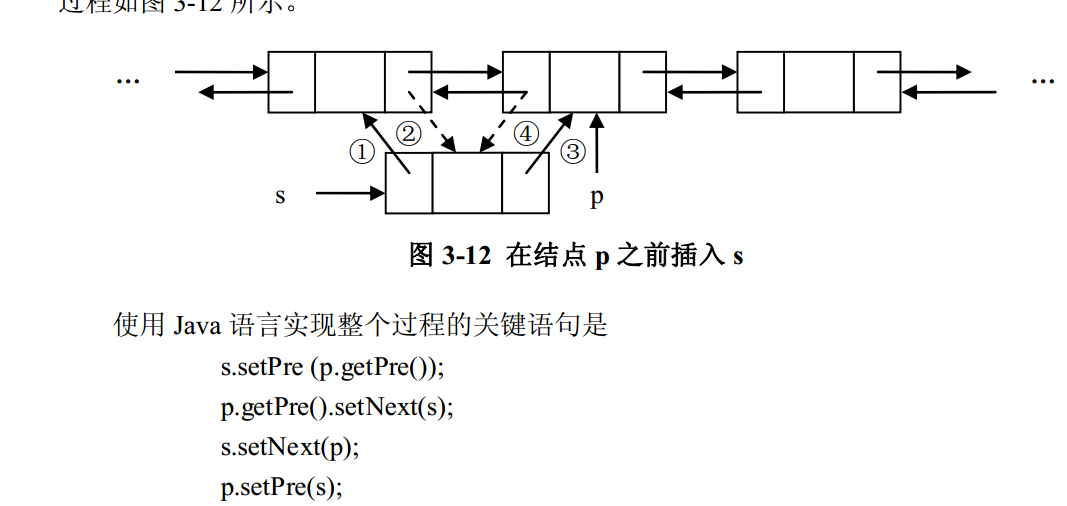
由于单链表中数据元素的插入是在结点的插入完成的，因此单链表数据元素的插入操作要比数组元素的插入操作要快的多（删除也一样）

单链表无法直接访问直接其前驱结点，必须通过一个结点的引用访问后续结点，要找到某个结点的前驱结点，必须通过首结点出发依次向后查找，需要Q(N)时间

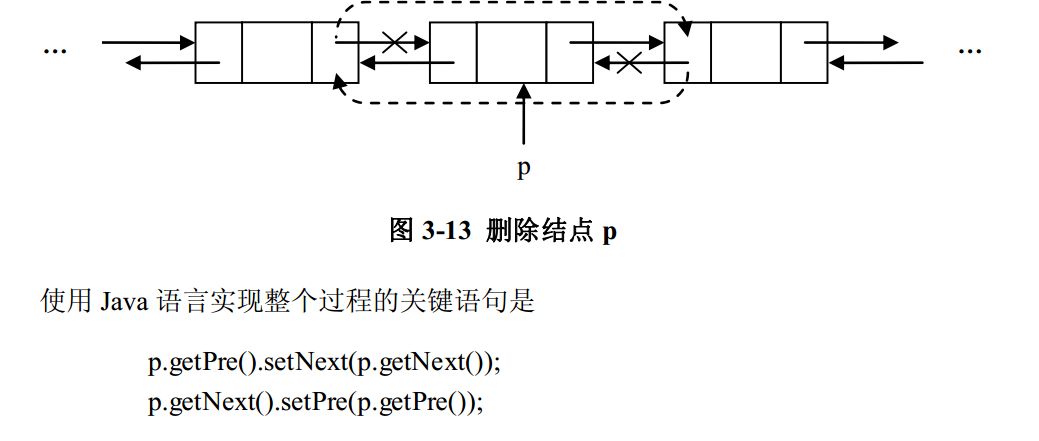
双向链表：

查找时，即可以从首结点开始也可以从尾结点开始

插入结点



删除结点:



线性顺序存储和链式存储的比较：

基于时间的比较：

对于查找操作：基于序号的查找，顺序存储Θ（1）内完成，链式储存需要从头结点开始依次向后查找

基于元素的查找，这类存储，顺序存储和连式存储都需从序号为0的元素开始查找，性能相同弄

基于元素的插入和删除操作：使用数组实现时，使用顺序查找定位，，插入删除，然后是大量元素的移位；链式存储只要在元素定位的基础上，简单的修改几个指针即可完成；

基于序号的插入和删除：顺序存储需要平均移动一半的元素，而链式存储不能直接定位，平均需要比较一半的元素才能定位，所以性能相当

基于空间的比较：

顺序存储，其空间是预先分配的，实现过程中可动态扩展空间，如果线性表的长度变化太大，空间在使用过程中由于存在大量的存储空间，其利用率不高。

链式存储，其存储空间是动态分配的

链式存储需要使用额外的存储空间来表示元素间的逻辑关系，而顺序存储没有，所以当线性表的长度变化不大时，没有利用的空间的比例变小，考虑使用顺序存储

链接表：

一组结点序列以及基于结点进行操作的线性结构的抽象，对链表的抽象

迭代器：

提供一种方法顺序访问聚集对象的各个元素，而不需要暴露对象的内部表示

栈和队列

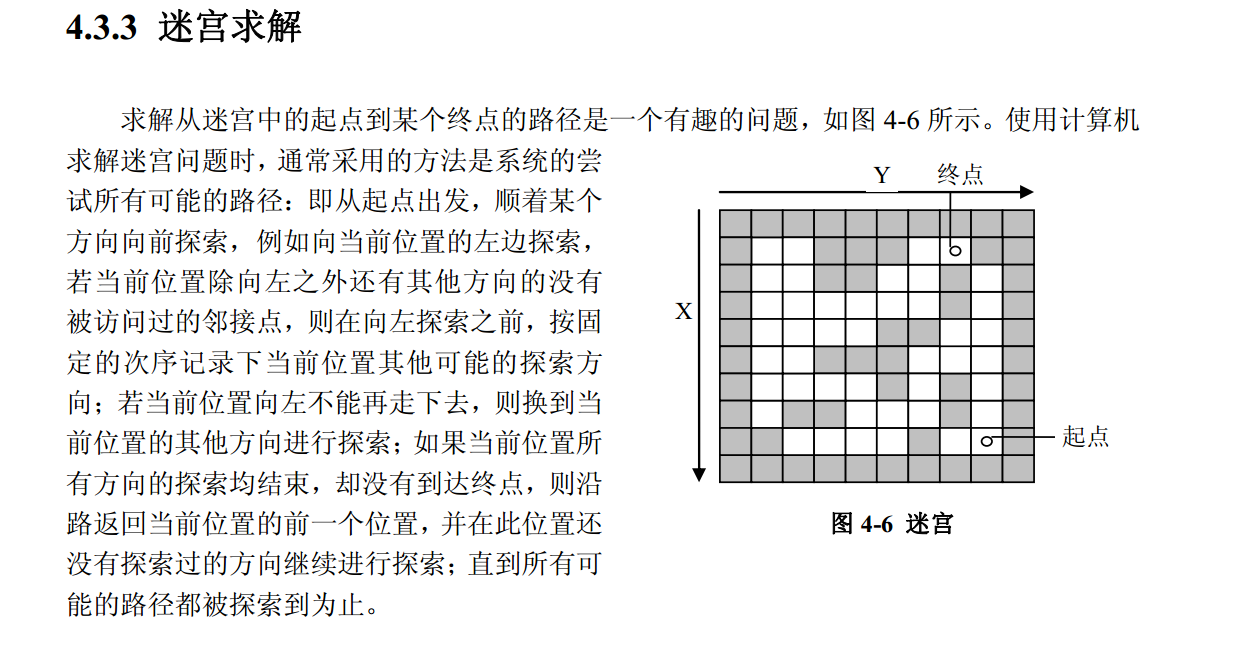
栈（堆栈）：运算受限的线性表，仅允许在表的一端进行插入和删除操作，表的插入，删除的一端为栈顶，另一端为栈底

队列：运算受限的线性表，仅允许一端进行插入，另一端进行删除操作，插入数据的一端叫对尾，删除数据的一端叫对首

循环数组实现队列，循环队列

判断循环队列队列空还是满：一是少使用一个存储单元，当判断对尾指针下一个就是对首指针所在单元时，停止入队，二，增设一个标志

练习：

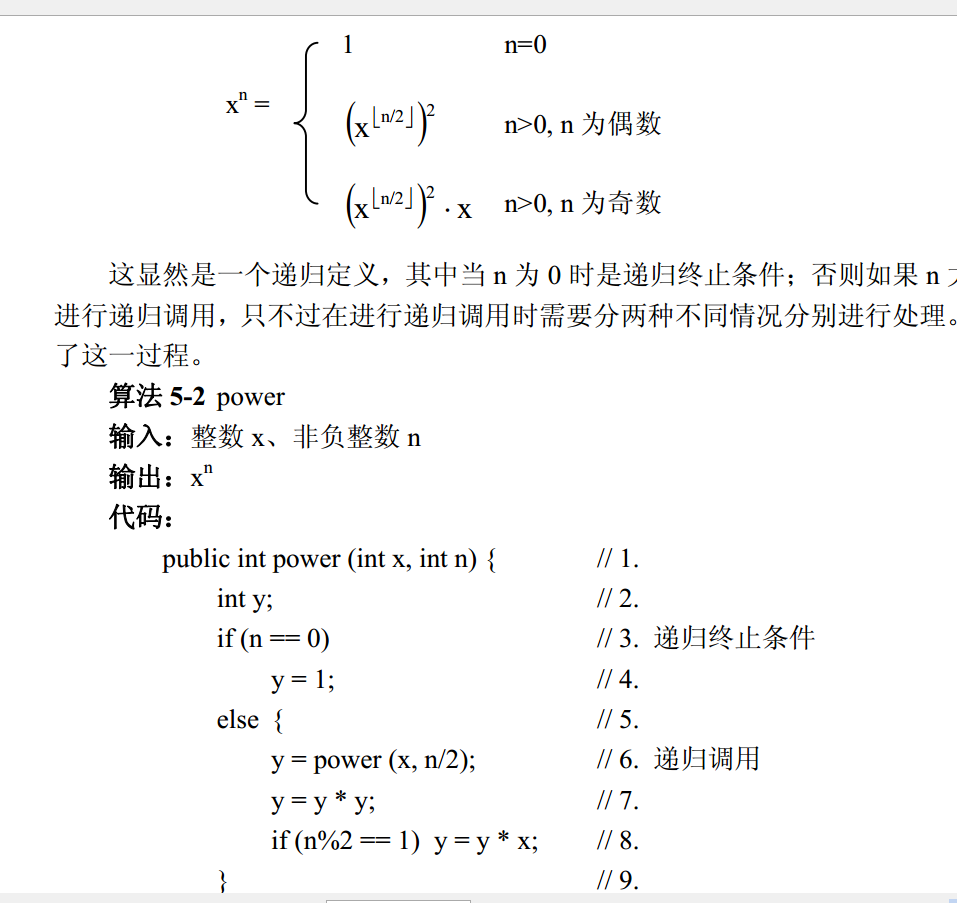


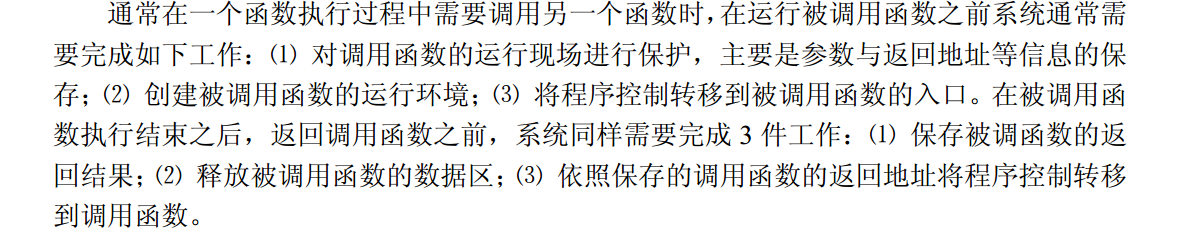
设计算法：

递归：定义自身的同时又出现了对自身的引用，如果一个算法直接或者间接的调用自己，则称为递归算法（递归调用和递归终止条件）

Factorial（阶乘）

幂次方计算：

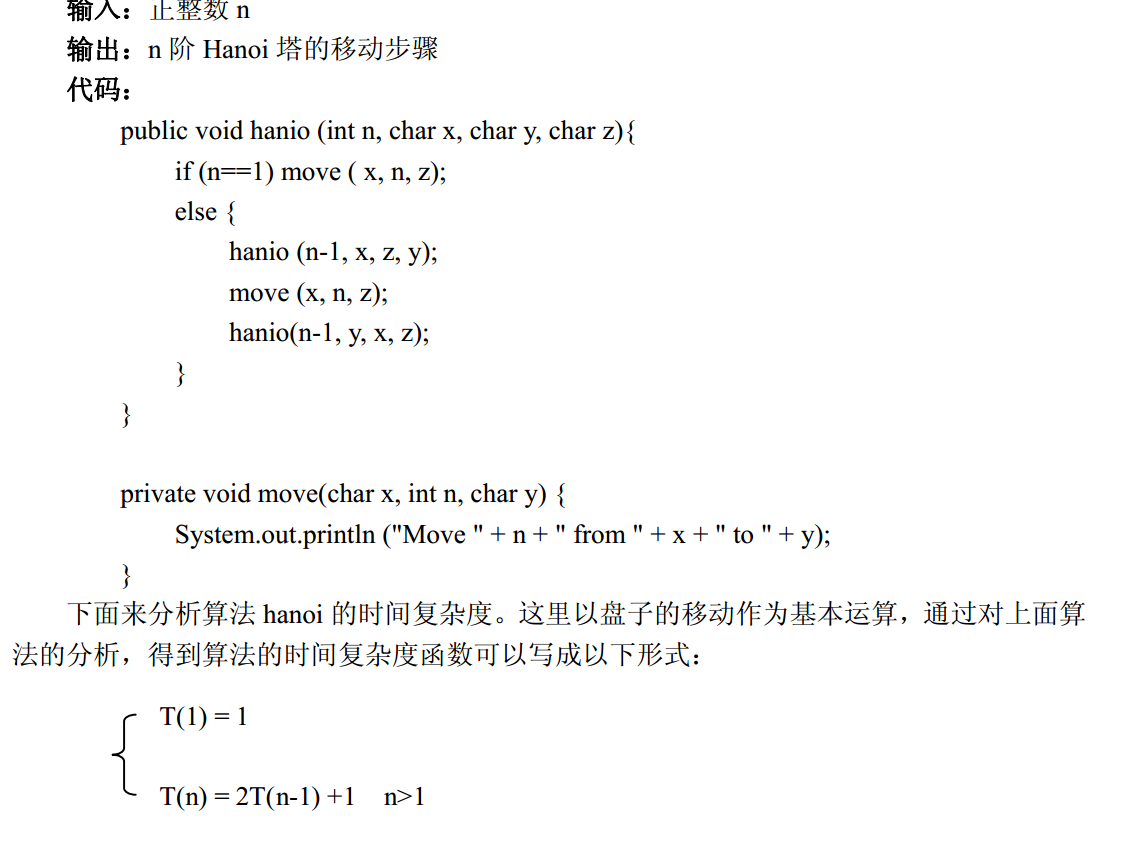




堆栈的使用，递归的调动，其中调用过程中相关的信息的保存使用堆栈来保存

基于归纳的递归：

hanoi n阶汉诺塔：时间复杂度Θ（n2）



递推关系求解：

数学归纳法

迭代法

递归树：