# 数据结构与算法

1. 数据结构

分类：

1. 线性结构

* 最常用的数据结构，数据与元素之间存在一对一的线性关系；
* 分为两种不同的存储结构，即顺序存储结构（数组）和链式存储结构（链表）。顺序存储的线性表称为顺序表，顺序表中的存储元素（地址）是连续的。链式存储的线性表称为链表，链表中的存储元素不一定是连续的，元素节点中存放数据元素以及相邻元素的地址信息；
* 线性结构常见的有：数组、队列、链表和栈。

1. 非线性结构

* 非线性结构包括：多维数组、广义表、树结构、图结构。

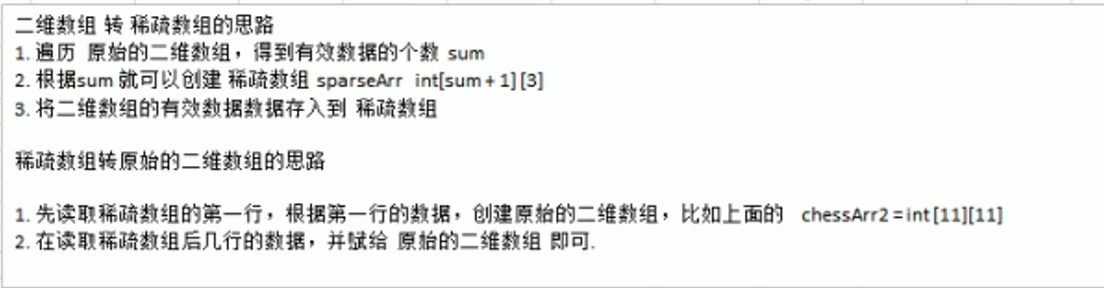
1. 数组（array）：
2. 稀疏数组（sparsearray）：



* 记录数组有几行几列，有多少个不同的值；
* 把具有不同值的元素的行列及值记录到在一个小规模的数组中，从而缩小程序的规模。

**案列：**如玩五子棋游戏需要保存上一局游戏结果时，传统方式使用的是二维数组来保存的，这种方式会保存很多无用重复的数据（无棋子的坐标），导致占用大量的空间。这个时候可以使用稀疏数组来替换，如上图就是将有6行7列8个不同值的数组转换为稀疏数组后的样子。

**二维数组与稀疏数组的转换思路：**

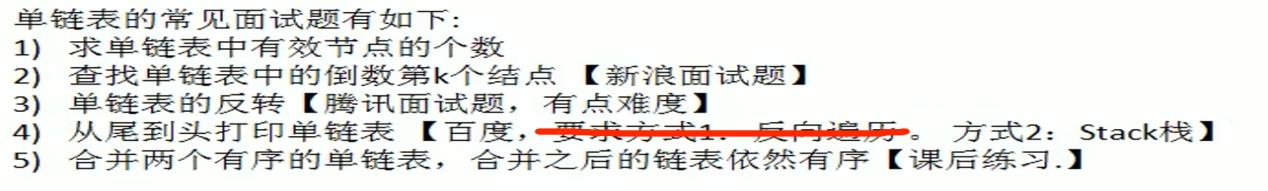


1. 链表（linked list）：

* 节点对象 + 相邻节点指针；
* 不一定是连续存储的；
* 分为带头节点的链表和不带头节点的链表。

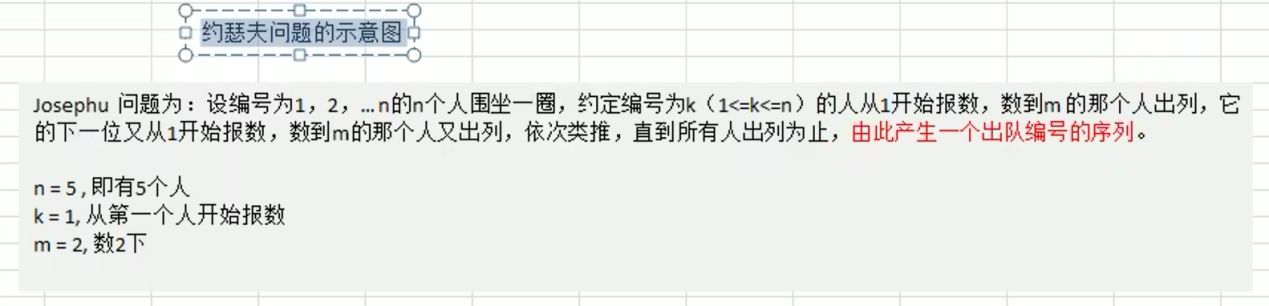
1. 单向链表

* 查找的方向只能是一个方向；
* 不能自我删除，需要靠前一个节点辅助删除。



1. 单向环形链表

* 未节点的下一个节点指向首节点。



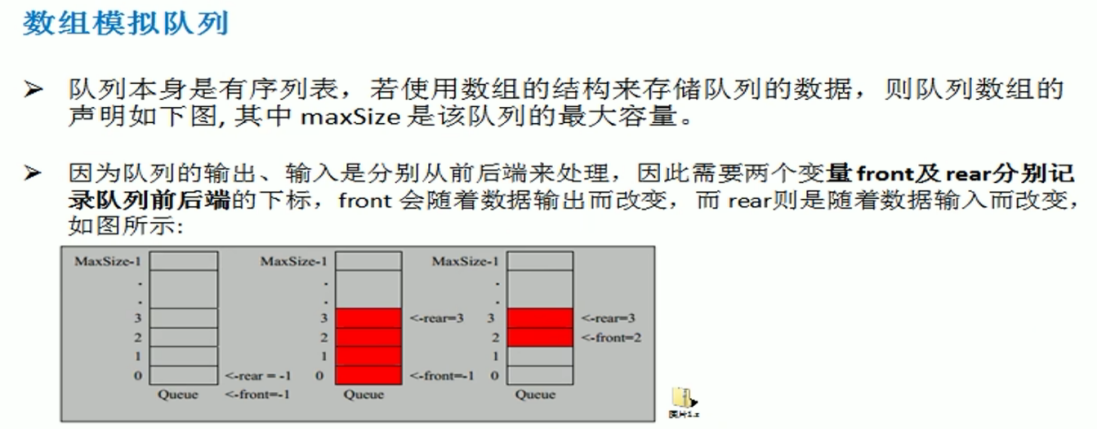
1. 双向链表

* 可以双向查找；
* 可以自我删除。

1. 队列（queue）：

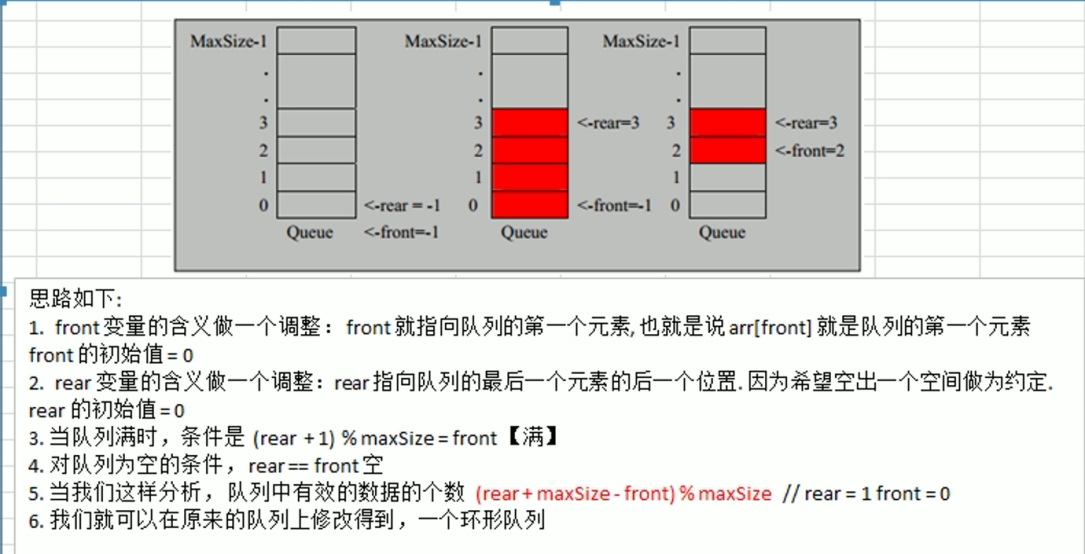
队列是一个有序列表，可以用数组或链表实现，遵循先进先出的原则。

* 数组实现



存在问题：

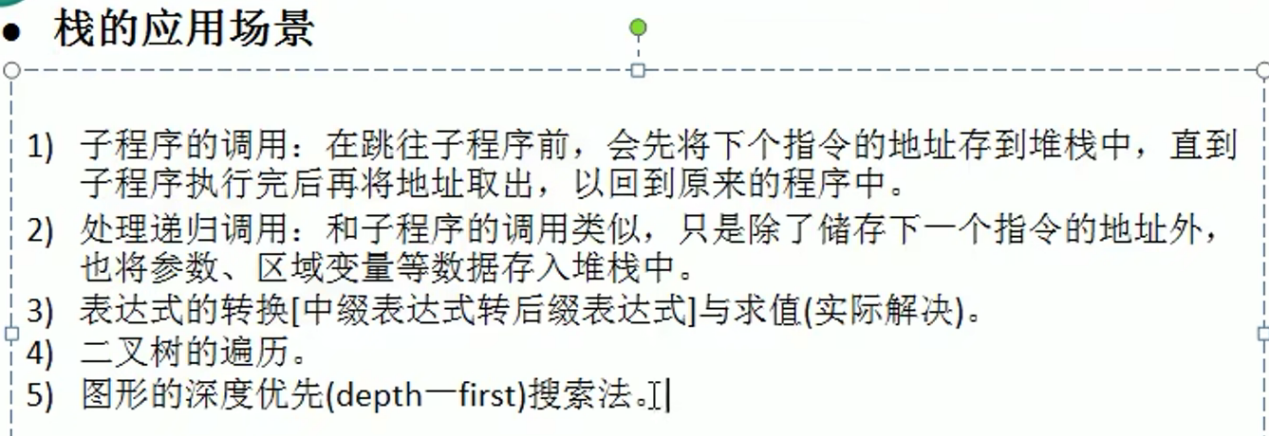
* 数组只能被使用一次，无法复用；
* 使用算法（取模的方式），将数组改成环形队列。



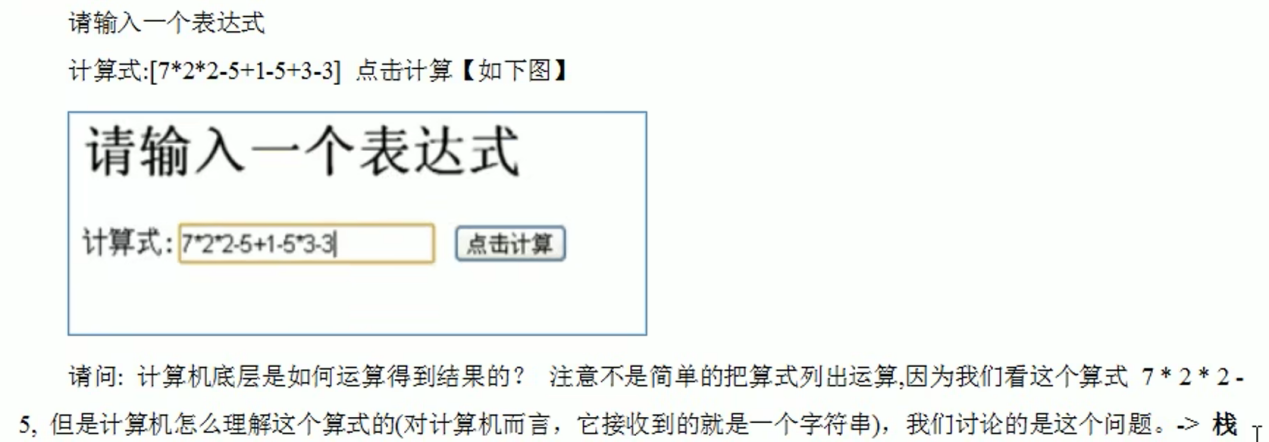
* 链表实现

1. 栈（stack）：

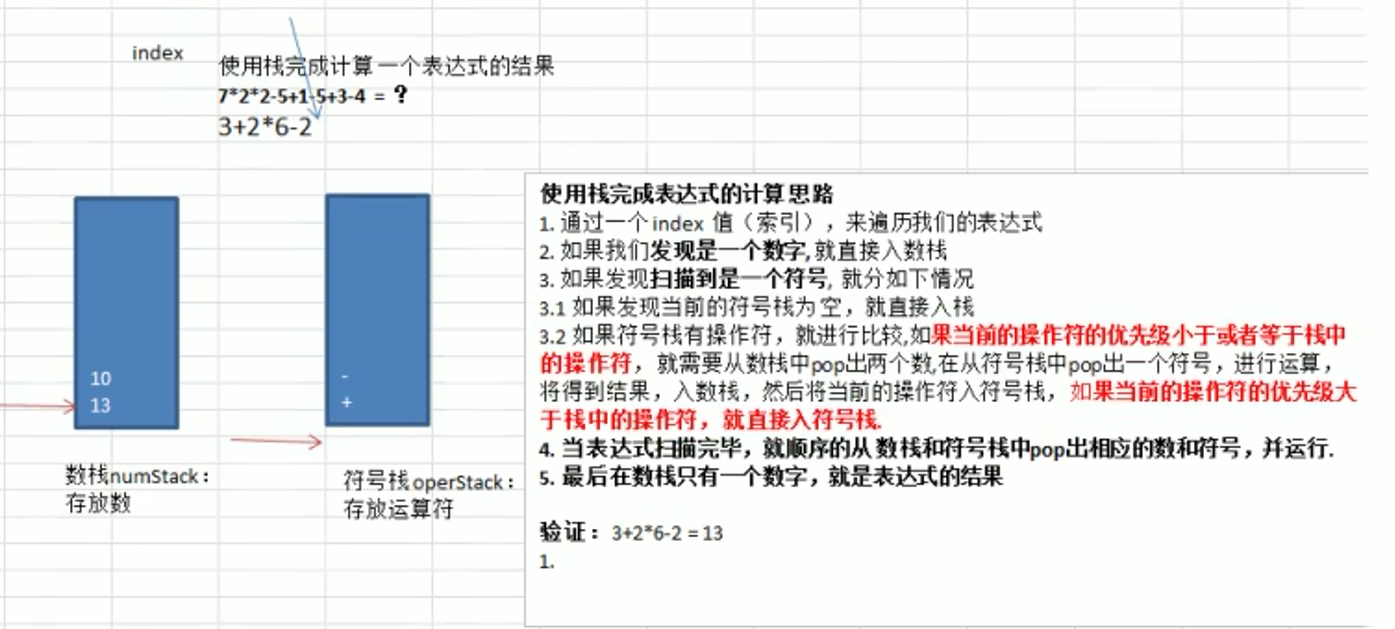
* 先进后出（first in last out）的有序列表；
* 只能在一端操作（栈顶），另一端固定不能操作（栈低）；
* 最先放入的元素在栈低，最后放入的元素在栈顶。而删除正好相反，最先入栈的最后被删除，最后入栈的最先被删除。



案例：

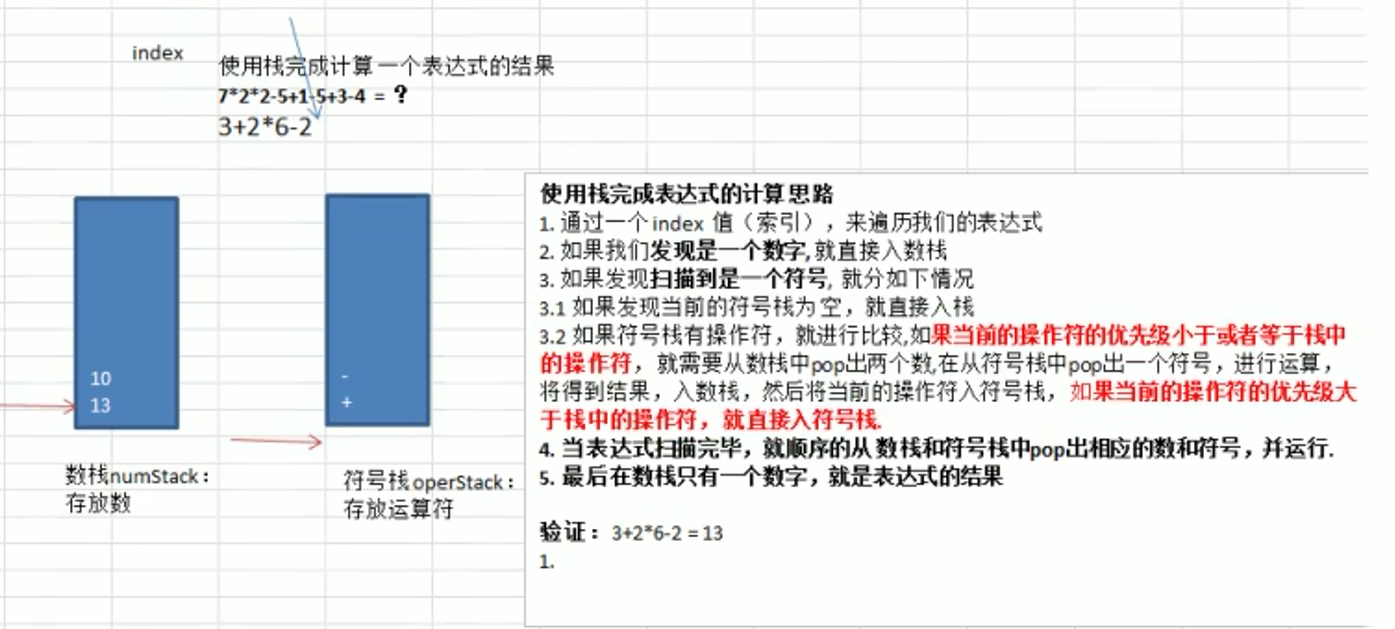


实现思路（中缀表达式）：

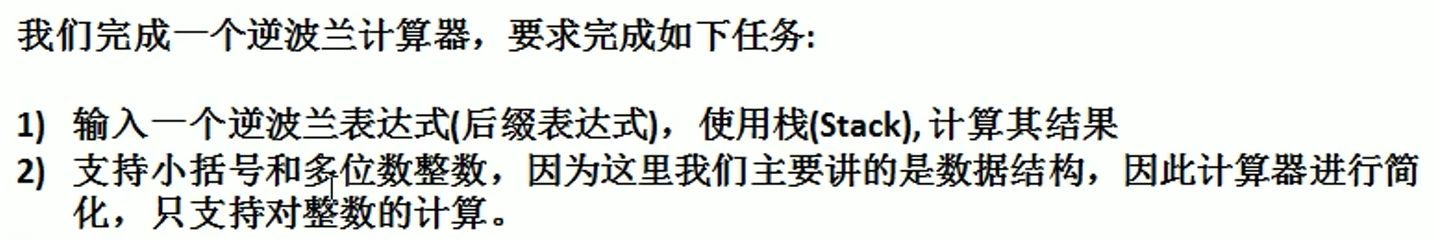


* 前缀表达式（波兰表达式）
* 从右至左扫描表达式；
* 遇到数字时，将数字压入栈中。遇到运算符时，弹出栈顶的两个数，并用运算符做对应的计算，然后将结果入栈。重复上述过程，直到扫描到最左端，最后运算得出的值即为表达式的结果；
* 如表达式：(3 + 4) \* 5 - 6，转为前缀表达式为：- \* + 3 4 5 6。
* 中缀表达式
* 使用我们人所使用的运算规则去运算；
* 对计算机不太好操作，需要判断优先级。因此，在计算结果时，往往会将其转换为其它表达式来运算（一般转为后缀表达式）；
* 如表达式：(3 + 4) \* 5 - 6。

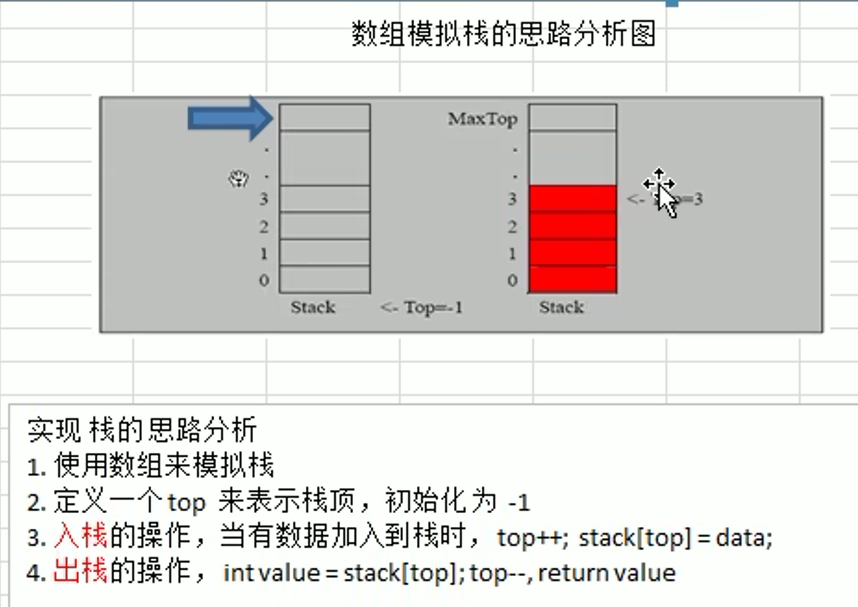
中缀转后缀实现思路：



* 后缀表达式（逆波兰表达式）
* 从左至右扫描表达式；
* 遇到数字时，将数字压入栈中。遇到运算符时，弹出栈顶的两个数，并用运算符做对应的计算，然后将结果入栈。重复上述过程，直到扫描到最右端，最后运算得出的值即为表达式的结果；
* 如表达式：(3 + 4) \* 5 - 6，转为后缀表达式为：3 4 + 5 \* 6 -。



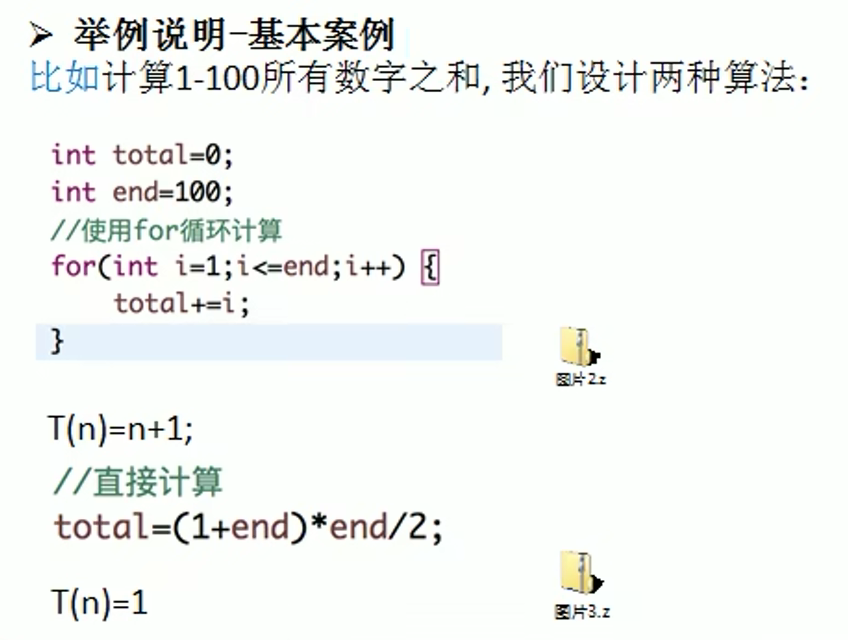
* 数组实现



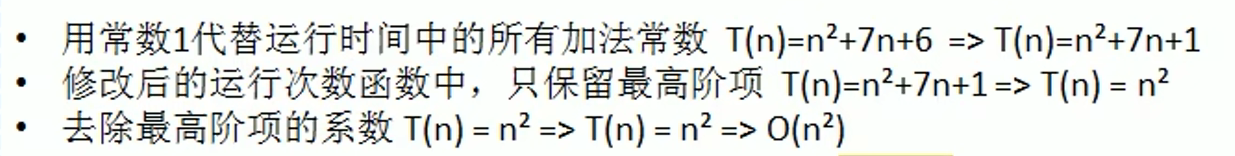
* 链表实现

1. 算法

算法时间复杂度：

* 可以忽略常数项，如：T(n) = n + 2，可以看成T(n) = n，随着n的增大它们无限接近；
* 可以忽略低次项，如：T(n) = 2n^2 + n + 1，可以看成T(n) = 2n^2，随着n的增大它们无限接近。
* 时间频度（T(n)）
* 算法花费的时间与它执行的语句次数成正比，一个算法中的语句执行次数称为语句频度或时间频度。
* 
* 时间复杂度（O(f(n))）
* 一般情况下，算法的时间频度T(n)，若有某个辅助函数f(n)，使得当n趋近于无穷大时T(n) / f(n)的极限值为不等于0的常数，则称f(n)是T(n)的同数量级函数。 记作T(n) = O(f(n))，O(f(n)简称为算法的时间复杂度；
* T(n)不同但算法的时间复杂度可能相同，如：T(n) = n^2 + 7n + 6和T(n) = 3n^2 + 2n + 2，它们的时间复杂度都是O(n^2)；

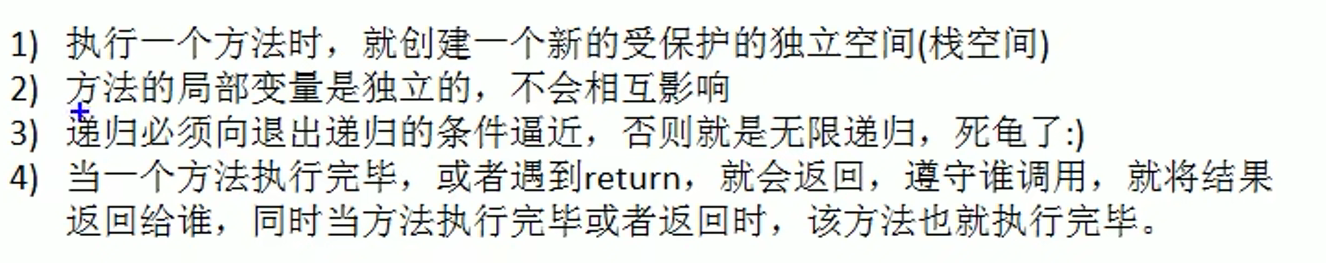
计算时间复杂度的方法：

* 

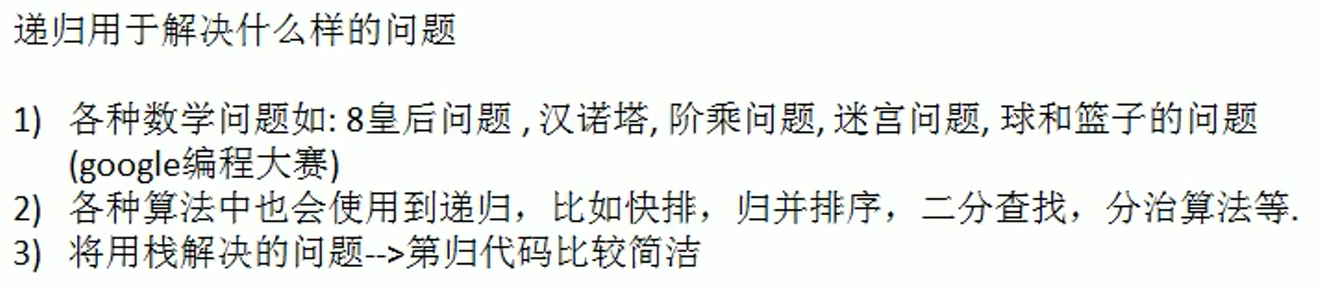
常见的时间复杂度：

1. 递归算法（recursion algorithm）

* 递归的概念
* 方法自己调用自己，每次调用时传入不同的变量；
* 递归算法包含回溯算法（回到起点重新计算）。
* 递归的规则



* 能够解决的问题



案例一：走迷宫、求出最短路径

案例二：八皇后问题

1. 排序算法（sort algorithm）
2. 内部排序

* 指将所有数据都加载到内部存储器（内存）中进行排序。

1. 插入排序
2. 直接插入排序
3. 希尔排序
4. 选择排序
5. 简单选择排序（O(n^2)）

* 排序前指定某个位置的值为最小值，然后遍历找到比指定为最小值还小的值的位置，然后将这两个位置的值进行交换。

1. 堆排序
2. 交换排序
3. 冒泡排序（O(n^2)）
4. 快速排序
5. 归并排序
6. 基数排序（桶排序）
7. 外部排序

* 数据量太大，无法全部加载到内存中，需要借助外部存储进行排序。