# 算法引论

算法：适合用计算机实现的解决问题的方法；数据结构：便于算法操作的组织数据的方法；

算法是指解决问题的过程与方法：输入、输出、确定性、有限性；

算法的问题对象（操作的数据）：数字、字符串、局针、声音.....

算法涉及到的运算：矩阵运算、加减乘除逻辑运算等.....

高级语言可以抽象表达算法；

算法从自然语言转化为代码需要做的事情：

* 选用数据模型、确定初始状态、结果状态，探索从初始状态到结果状态的关系，怎么从初始状态到结果状态；
* 自顶向下，逐步求精，设计算法顶层运算，在设计底层运算；顶层步骤是定义在数据模型级别上的步骤，是算法的主干，是抽象的，操作的是数据模型内的对象；底层运算是微观运算，是与数据模型定义的运算，具体实现的细节依赖数据模型的内部表示，为了让2层不影响，需要定义接口抽象，这个接口就是抽象数据类型（ADT）；
* 抽象数据类型实际就是定义了一个类型。

# 动态规划

与分治法类似，基本思想也是将问题分解为子问题，通过子问题的解得到问题的解；不同的地方是，动态规划问题分解的子问题往往不是相互独立的，若用分治法解决，则分分解了很多重复的子问题，耗费了指数级的时间，实际上，去除冗余，子问题的数量是多项式级别的，保存已解决的子问题的代码，避免重复计算，需要一个表来记录所有已解决的子问题答案；这就是动态规划，具体的动态规划算法是多种多样的，但是都具有相同的填表形式。动态规划算法适合来解最优化问题，通常可以按照以下步骤设计动态规划算法：

* 找出最优解的性质，并刻画其结构特征；
* 递归的定义最优值；
* 以自底向上的方式计算出最优值；
* 根据计算最优值时得到的信息，构造最优解。

## 3.1 矩阵连乘问题

给定n个矩阵，相邻的矩阵都是可乘的，矩阵乘法满足结合律，所以可以是任意的计算次序；可以用加括号的方式唯一确定一个执行顺序；加括号的计算乘积的递归定义如下：

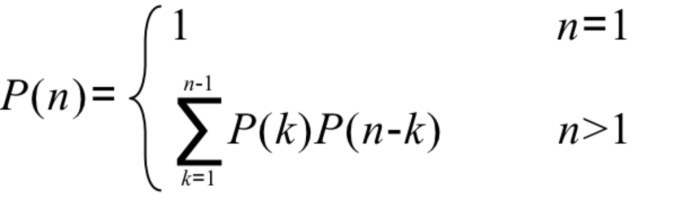
* + 单个矩阵是完全加括号的，是最高优先级；
  + 乘积A是完全加括号的，表示A的乘数矩阵B与C都是完全加括号的即。

矩阵乘积顺序关系到计算量，计算2个矩阵的标准算法如下：



计算量是3次循环的数乘；顺序对计算量影响很大，考虑3个矩阵的相乘，，维数分别是，如果顺序计算，则乘积需要次数乘，如果按照的顺序计算的，需要的数乘次数为；于是有了矩阵乘积的最优计算次序问题，就是要求最少的数乘次数。

穷举搜索法可以解决这个问题，把每种可能都计算出来，选出最优的；但计算量太大，对n个矩阵的乘积，设不同的计算次序种类为，由于可以在第k个与k+1个之间将矩阵连乘分为2个子连乘，，在对2个子矩阵连乘加括号计算次序种类，最终可以得到原连乘矩阵的加括号方式；递推式如下：



解方程，得到这是一个组合数，是指数级别的；下面按照动态规划的算法计算：

* 分析最优解的结构：刻画最优解的结构特征将矩阵连乘简记为，计算的最优次序，设这个计算次序在k与k+1位置处断开，则完全的加括号的形式为，依此计算，则先计算A[1:k]与A[k+1,n]在将计算结果相乘，这个问题的一个关键的特征是，计算A[1:n]的子矩阵的次序也是最优的，矩阵连乘计算次序的最优解包含其子问题的·最优解，这种性质叫做最优子结构问题，问题的最优子结构性质是问题可以被动态规划算法求解的显著特征；
* 建立递归关系：递归的定义最优值，设计算的最少数乘次数为，当i=j时，=Ai,为单一矩阵无需计算，m[i][j]=0；当i<j时，可利用最优子结构计算，设在k处断开，则由于不知道k的位置，k有j-i种可能，k是使这个计算量达到最小的位置从而m[i][j]的递归定义如下：  
  ，这个式子计算出了最小数乘次数与连接断开位置k；
* 计算最优值：根据上面的递归式计算将消耗指数级的时间，事实上，对于1<=I<=J<=n，不同的有序对，对应不同的子问题，不同子问题的个数最多有  
  个，许多子问题被重复计算了，用动态规划算法解决这个问题，依据递归式自底向上的方式计算，计算时，保存已解决的子问题答案，每个子文推只计算一次，后面需要是查询一次，得到多项式的时间的算法，这里看书啊，写不动了；
* 构造最优解：

## 3.2 动态规划算法的基本要素