Chapter 1 -2

1、算法的基本概念、性质

\* 能够回答算法的五个重要特性

\* 能够回答算法时间复杂度的含义

算法的定义：一个定义良好的可计算过程，取I 个或一组值作为输入，并产生出一个或一组值作为输出。即，算法就是一系列的计算步骤，用来将输入数据转换成输出结果。

算法的五个重要特性：

1 输入输出，算法有0个或多个取自指定集合的输入值，有1个多个与之联系的输出值。

2 确定性，算法的每一个指令步骤都是明确的。

3 有限性，对算法的每次输入，算法都必须在有限步骤（即有限时间内结束）。

4 正确性 对每一次输入，算法应产生正确的输出值

5 通用性 算法的执行过程可应用于所有同类求解问题，而不是仅适用于特殊的输入。

问题，规定了输入输出之间的关系，可以用通用语言来描述。有输入，有输出，有关系。

问题实例，某一个问题的实例包含了求解该问题所需的输入。

程序并不都满足算法的特征。

算法的时间复杂度的含义：算法的时间复杂度是一个关于代表算法输入值的字符串的长度的函数，他定性的描述了算法的运行时间。

2、了解插入排序算法的思想、性质，熟悉插入排序的排序过程

\* 插入排序的算法时间复杂度，对给定的具体算例正确排序

插入排序的问题描述。

思想、性质：对少量元素排序有效算法，类似于排序一手扑克牌，原址排序（再原数组中重排，只有常熟个数字存储在数组外）。

伪代码。

时间复杂度，最好排好序的n，最坏逆序的西塔n平方。

具体例子。

循环不变式：在每次for循环开始前，子数组由原来在其中的元素组成，但已经排好序。

证明循环不变式，证明三个式子：初始化&保持&终止。

分治算法：

最差的时间比插入排序法少得多。

运行时间可以用递归方程表示。

最坏情况西塔nlgn。

3、伪代码

\* 给出算法伪代码能够分析出算法的时间复杂度，能够用渐近记号表示

伪代码：ci代价 & n次数。

推出循环时，循环计数器保持其值。

最坏情况运行时间&平均情况运行时间分析。

Chapter 3

1. 了解时间复杂度渐近上界、渐近下界、渐近紧确界的含义

算法实际运行时间f（n），分析所得的限界函数为g（n）。

前者与机器和语言有关，后者无关。

渐进上界，算法最坏情况下时间复杂度，隐含给出了任意输入下运行时间的上界。阶最小的g（n）紧确上界。（松散的、紧确的）

渐进下界类似。

渐进紧确界，表示算法在最好最坏情况下的计算时间就一个常数因子范围内而言是相通的。

西塔1表示算法的执行时间为固定量，与问题规模无关。

2、\* 能够准确给出渐进函数记号：O、Ω、Θ的定义，并能够用它们表示算法复杂度分析结果

五个定义。

3、了解限界函数的基本性质：\*传递性、自反性、对称性、转置对称性

传递性五个，自反性三个，对称性一个，转置对称性2个。

函数渐进性比较和实数比较的类比。

4、\* 熟练了解计算和估算时间复杂度的一些定理，尤其是对数级、多项式级、指数级函数之间的量级关系，以在实际分析中加以应用

n取值大时冕指数时间算法和多项式时间算法在计算时间上非常悬殊。

N大时，现有计算机系统运行O（nlogn）复杂度高的算法困难。

指数时间算法只在n非常小时实用。

想要在顺序处理机上扩大处理问题的规模，有效途径是降低算法的计算复杂度，而不是仅仅依靠提高计算机速度。

Chapter 4

1、\* 掌握分治策略的基本思想，以及它遵循的三个步骤

分治法基本思想，问题规模较大而无法直接求解时，将原始问题分解为几个规模较小，但类似于原始问题的子问题，然后递归求解这些子问题，最后合并子问题的解得到原始问题的解。

遵循的三个步骤：

1）分解：将原始问题分解为若干个规模较小相互独立形似与愿问题一样的子问题，

2）解决：递归解决子问题，若足够小，否则递归

3）合并：将子问题的结果合并为愿问题的解

需要进一步分解递归合并求解时 递归情况， 不需要进一步求解 基本情况。。

2、\* 递归式的化简：要清楚为什么化简递归式，熟练掌握代换法、递归树法、\* 主方法（主方法的三种情况使用条件）

递归式求解的结果：得到形式简单的渐进限界函数表示（三个）。

预处理：假定自变量正整数&忽略边界条件&向上下取整

代入法：

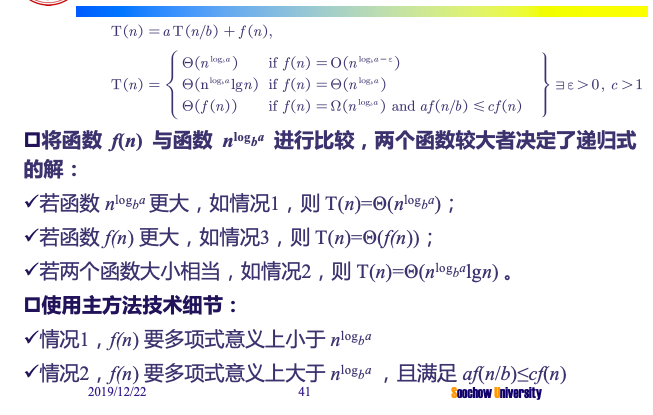
猜测接的形式 & 数学归纳法找出常数。一般证明边界条件符合归纳证明的基本情况，可能出现归纳证明基本情况不能满足的问题——扩展边界条件。

注意事项：有时候猜测正确，数学归纳出现问题，原因在于归纳假设不够强，可以去掉低阶项。通过对更小的值做更强的假设，就可以证明对某个给定值的更强的结论。

注意要证明归纳假设严格一致的形式

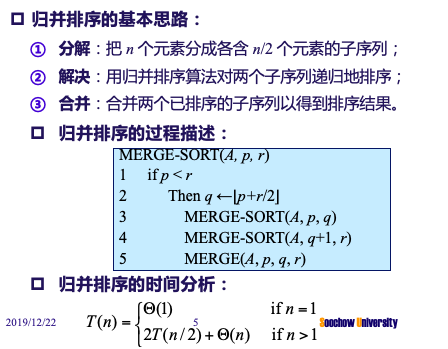
注意改变变量的方法。

递归树法：

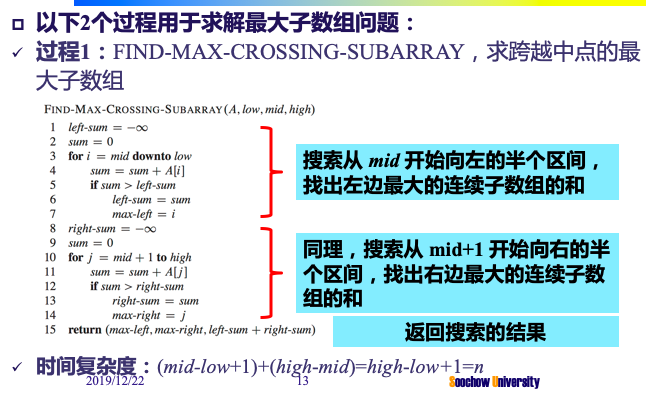
主方法：

3、了解利用分治策略求解相关问题和算法

实例：归并排序：



最大子数组问题：





strassen矩阵乘法：

朴素矩阵乘计算时间为西塔n3方。

基于分治策略：

strassen矩阵乘法：令递归树稍微不难么旺盛，只进行七次递归而不是八次。

Chapter 5

1、了解对相关概念：概率分析、均匀随机排列、随机算法、平均情况运行时间、期望运行时间

2、了解指示器变量的定义和作用

概率分析：使用关于输入分布的知识或对其做的假设，然后分析算法，计算出一个平均情况运行时间。

概率分析实际上是将所有可能的输入产生的运行时间做平均。

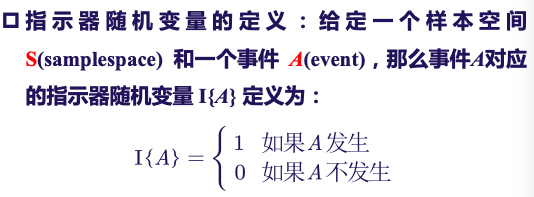
随机算法：如果一个算法的行为不仅由输入决定，而且也由一个随机数生成器产生的数值决定，则称这个算法是随机化的。

期望运行时间：随机算法的输入次序最终由随机数发生器决定，故我们将随机算法的运行时间称为期望运行时间。

平均情况运行时间：

概率分布在算法的输入上，讨论平均情况运行时间；算法本身作出随机选择，讨论算法的期望运行时间。

指示器随机变量的定义：



作用：建立概率和期望之间的联系，用于实现概率和期望之间的转换。

注：一个事件对应的指示器随机变量的期望值等于该事件发生的概率。

Chapter 6-9

1、了解堆的性质，\* 最大堆和最小堆；熟悉最大堆的一些操作的过程：\*维护最大堆，\*建堆，以及堆排序算法，对给定的具体算例会正确操作

3、了解快速排序的基本思想；\* 熟悉快速排序的排序过程；\* 理解快速排序的最坏时间复杂度和平均时间复杂度；\* 对给定的具体算例会正确排序

4、理解计数排序的基本思想，\* 能够熟练写出计数排序的伪代码；\* 对给定的具体算例，能够进行正确排序

5、理解桶排序的基本思想，\* 理解桶排序的线性时间复杂度，\* 能够熟练写出桶排序的伪代码；\* 对给定的具体算例，能够进行正确排序

6、了解中位数、顺序统计量的概念

Chapter 10-14

数据结构：

基本的数据结构？四类。各自数据元素之间的关系、结构特点。

动态集合？元素（关键字、卫星数据、指针、动态集合以其关键字来自于某个全序集合为前提）。基本操作。

实现动态集合的数据结构？

1、\* 熟悉栈和队列的特点，了解栈和队列的操作过程

栈：后进先出。PUSH、POP、EMPTY。S.top。

队列：先进先出。ENQUEUE、DEQUEUE。Q.head、Q.tail。环序。上溢、下溢。

2、\* 熟悉链表的特点，\*理解双向链表和单向链表的操作的不同

特点：线性顺序、指针决定顺序、形式多样。

搜索（最坏运行时间 西塔n）、插入（头插法 最坏运行时间O1）、删除（指针O1or关键字西塔n）。

x.prev x.next x.key || 头：x.prev=NIL 尾：x.next=NIL

哨兵：哑对象，简化边界条件。L.nil 表头表尾间。

有哨兵的链表操作：不能降低渐进时间界，可以降低常数因子。

无显示指针数据类型实现链式数据结构？两种方法。

对象多数组表示：

对象单数组表示：

对象的分配与释放？

垃圾收集器负责确定那些对象未使用。m-n个对象那个是自由的。自由表（保存自由对象的单链表-只使用next数组，头孢存在全局变量free中）

使用数组实现指针，需要自由表的配合。

ALLOCATE-OBJECT（） FREE-OBJECT（）

有根树的表示？二叉树？分支无限制的有根树（节点无限制时该方法无效：不知道预先分配多少属性。左孩子右兄弟表示法。）？

3、理解散列表的特点，\*理解和掌握散列函数的定义和作用，熟悉散列冲突的概念以及解决散列冲突的两种方法：链接法和 \*开放寻址法；利用开放寻址法解决散列冲突中，对于具体给定的实例，能够熟练的运用三种探查法来计算探查序列：线性探查，二次探查，以及\*双重探查

直接寻址表：关键字全域小&每个元素关键字不同。槽k指向集合中关键字为k元素。字典操作简单3基本操作O1时间。缺点：全域大申请失败&元素稀疏浪费空间。

散列表：实现字典操作有效数据结构。克服直接寻址技术缺点，保存快速字典操作优势。

散列函数：关键字映射。问题：多映射一，即冲突。

散列函数定义：../Desktop/屏幕快照%202019-12-19%20下午7.15.35.png

作用：将范围较大的关键字映射到一个范围较小的集合中。

问题：遇到冲突如何解决&找出一个函数h(k)尽量减少冲突。

方法：链表法 & 开放寻址法。

链表法：把同时散列到同一槽中的元素链表形式串联，槽中保存指向链表指针。插入O1，查找最坏与表长正比，删除双链O1、单链同查找（找前驱）。

装载因子？

平均下，全部字典操作O1时间完成。

好的散列函数：近似满足简单均匀散列。两种方法？

除法散列法：h（k）=k mod m。m为不太接近2的整次幂的素数。

乘法散列法？优点：对m的选择不是特别关键，为2的整次幂即可。

开放寻址法：

所有元素存放于散列表，每个表项或包含动态集合的一个元素，或包含NIL。

查找时，系统检查所有表项，直到找到所需元素或查明其不在表中。

散列表可能会被填满，以致于不能插入任何新的元素，装载因子不会超过1。

注意装载因子不应该设置太大，否则很容易发生冲突，最好设置不要超过0.5，即元素数量要小于散列表容量的一半。

好处：不用指针，计算要存取的槽序列。因此不用指针节省的空间使得可以用同样的空间存储更多的槽，潜在地减少冲突，提高检索速度。

缺点：该方法的删除操作，如果删除了某个关键字后，无法检索到以后的关键字了，如果用一个特定的值代替，查找时间就不依赖于装载因子了。

均匀散列的假设：每个关键字的探查序列等可能的为m的阶乘中排列中的一种。

---为了使用开放寻址法插入一个元素，需要连续检查散列表（探查），直到找到一个空槽放置待插入元素，检查顺序依赖于待插入关键字。

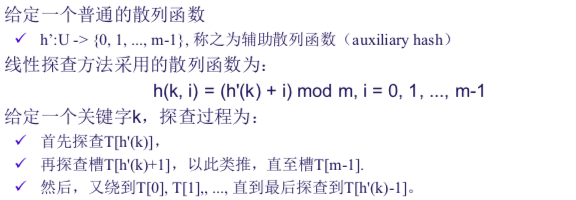
---为了确定要探查哪些槽，扩充散列函数，包含探查号作为第二个输入参数。

---对每个关键字，使用开放寻址法的探查序列。

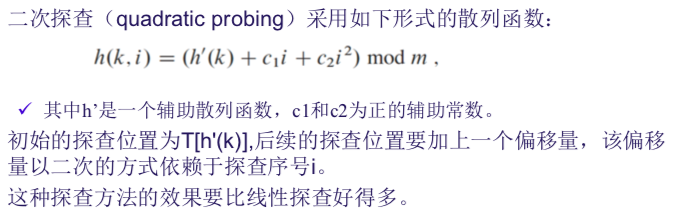
---查找关键字的算法的探查序列与将k插入时的算法一样。因此查找过程中遇到一个空槽，查找算法非成功的停止（因为如果k在表中，k应该在此处）。

探查方法：线性、二次、双重散列。

线性：初始探查位置决定整个序列。只有m种不同探查序列。问题：一次群集（随着被占用槽的增加，平均查找时间不断增加；连续被占用的槽会变得越来越长）。



二次：初始探查位置相同，探查序列相同。初始探查位置决定序列。M种。问题：二次群集（轻度群集）。



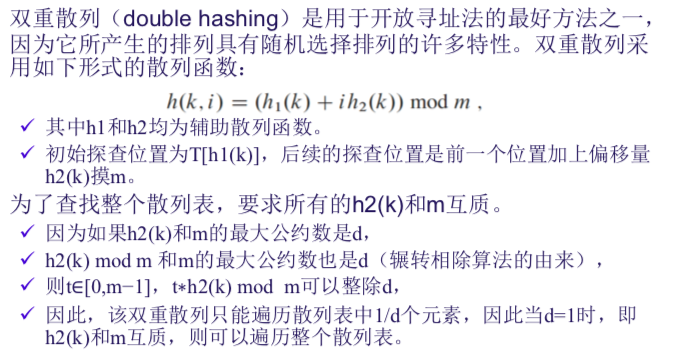
双重散列（双重探查）：开放寻址法最好方法之一：其产生的序列具有随机选择排序的许多特性。

这里的h2（k）必须对m互素：

若m为2的幂，h2（k）总是产生奇数。

若m为素数，h2（k）总是产生小于m的正整数。

每一对可能的（h1（k），h2（k））都会产生一个不同的探查序列，因此双重散列通常产生比较好的结果。



7、\*理解与掌握二叉搜索树的性质，了解二叉搜索树的一些查询操作和时间复杂度：搜索，返回最大，最小等，\* 掌握二叉搜索树的插入操作，了解二叉搜索树的删除操作；\* 对于给出具体的二叉搜索树实例，能够进行正确操作

性质：左树中key均小于根key，右树中key均大于根key。

中序遍历二叉查找树，可得从小到大序列。中序遍历代码，西塔n时间。

查询：SEARCH（Oh树高h）、MINIMUM（沿left，Oh）、MAXIMUM（沿right，Oh）、SUCCESSOR、PREDECESSEOR。输入根节点指针&待查找关键字，输出指向关键字的节点的指针/NIL。

注：迭代方式重写递归，效率高。

前驱和后继（Oh）：前驱：有左子树，左子树的最右节点/无左子树，往上走直到遇到一个右孩子。后继同理。

插入：插入位置即查找过程中查找不成功的节点位置（Oh）。插入总在某个叶节点位置。

删除：三种情况？（Oh）

随机构建二叉搜索树：向一棵空树随机插入n个关键字得到的树。期望高度Olgn。

。

8、\* 了解平衡树二叉搜索树的概念；\*理解和掌握红黑树的性质，\*了解红黑树的查询和修改操作，以及时间复杂度，包括搜索，插入等；\* 掌握红黑树的旋转操作，\* 包括左旋和右旋；\* 对于给出具体的二叉搜索树实例，能够进行正确操作

二叉搜索树：算法时间：静态操作5&动态操作2，均Oh。

什么是平衡二叉搜索树？

属于二叉搜索树 & 在动态改变中保证其树的高度为O(lgn)。

平衡：没有一个节点过深。

红黑树：可实现平衡二叉搜索树，确保没有一条路径会比其他路径长出两倍，因而时接近平衡。

红黑树确保没有一条从根节点到叶节点的简单路径会比其他路径长出两倍以上，因而红黑树近似平衡。

本章叶节点及其内部节点黑高被忽略。

红黑树特性：

1. 自平衡二叉搜索树，具有二叉搜索树性质
2. 节点颜色非黑即红
3. 根节点颜色为黑
4. 叶结点NIL为黑
5. 若红，则其左右孩子黑
6. 对树中任何节点，所有从其出发到其叶节点的路径中包含相同数目的黑色节点。

外部节点：不包含世纪关键字 & 内部节点：包含实际关键字。

节点形式：p left right key color

黑高度：从节点x出发到其叶节点路径上黑色节点个数。据特性6，bh(x)唯一。据特性5，从节点x到叶节点经历黑>=红。若x为根，bh(x)>=h/2。对于叶结点，黑高为0。对于某个非NIL的结点，才有黑高。

定理13.1 具有n个内部节点的红黑树高度最多为2lg(n+1)。

在一个具有n个关键字节点的红黑树上，搜索、最大、最小、前驱、后继可以在Olgn时间内完成。搜索树的插入、删除操作在含n个关键字的红黑树上，运行花费时间Olgn，可能破坏红黑树性质，不能保持红黑树的近似平衡性，必须改变树中的某些节点的颜色和指针结构。

红黑树的基本操作：添加&删除。变色&旋转方法（目的：保持红黑树的特性）。

为了保持红黑树的性质，红黑树上的修改操作涉及以下三种操作：

1. 插入/删除
2. 新插入结点颜色的设置，周边的结点重新染色
3. 通过旋转修改红黑树节点的指针链接

指针结构的修改：旋转。

旋转操作中，只有指针改变，中序遍历的结果不变。旋转操作时间复杂度O1，在操作过程中，只有指针改变，其他属性不变。

插入：

INSERT+FIXUP

将待插入结点着色为红色，像普通二叉树那样插入新节点

通过重新着色和旋转操作修复违反的红黑树性质。

可能违反的性质：

根节点时黑色的、红色节点的字节点时黑色的。

因为插入节点z为红色，若为根，破坏性质，若其父节点为红，破坏性质。

红黑树的插入：三种情况。

叔结点为红（1）

叔结点为黑（2）

RB-INSERT（T，z）运行时间为O（lgn）。

红黑树的删除：两种情况。

删除红：不需要调整

删除黑：性质违反。

删除根，其红孩子成为根，违背2.

删除x，其孩子红，父红，违背4.

删除不是树唯一节点，删除节点的那一个分支到各叶节点的黑色节点数发生变化，破坏5.

维护删除过程中红黑树的性质：从节点x开始调整，认为他有额外的一层黑色。

RB-DELETE(T,z)运行时间为Olgn。

储存额外信息的方法扩张标准的数据结构

动态顺序统计量：

无序集合：在On时间内确定任何顺序统计量

修改红黑树：Olgn时间内确定任何顺序统计量，计算一个元素的秩（即他在集合线性序中的位置）

顺序统计树：红黑树节点上增加size属性，包含了以x为根的子树的节点数。

查找具有给定秩的元素：OS-SELECT（x，i）。

返回一个指向以x为根的子树中包含第i小的关键字的结点。类似二分查找，最坏情况Olgn。

确定一个元素的秩：OS-RANK（T，x）。

给定舒徐统计树T中结点x的指针，返回对T中序遍历对应的线形序中x的位置。

Chapter 15

1、\*掌握动态规划方法的基本思想，掌握应用动态规划求解相关问题应满足条件，掌握动态规划求解问题遵循的四个步骤

2、\* 掌握动态规划方法和分治策略的相同点和它们之间的不同点，能够准确回答这类问题

3、\* 对给出问题实例能够推导递归求解公式

4、了解最优子结构性、子问题图等概念

5、理解何为 Programming ，能够回答对动态规划带来的改进的理解

6、了解利用动态规划策略求解相关问题和算法，对给出的算例能够正确计算

* 钢条切割 矩阵链乘法
* \* 最长公共子序列，对于给出的算例能够进行正确的求解，基于递归公式通过填表给出详细地计算过程

动态规划用来求解最优化问题；：这一类问题的可行解可能有多个，每个解都有一个值，我们希望寻找具有最优解的值（最小值或最大值）。问题的一个最优解。

最优化问题的数学模型，给定一个函数F（x）目标函数，其中自变量x满足一定条件约束条件，约束条件表示x属于D。求目标函数在约束条件下的最小值或最大值问题就是一般最优问题的数学模型，可以表示为M in F（x）或Max F（x）。

最优化问题分类：线形规划，整数规划，非线性规划，动态规划。

动态规划的思想实质是分治思想和解决冗余。与分治发类似，基本思想也是将待求解问题分解成若干个子问题。但是经分解得到的子问题往往不是互相独立的，不同子问题的数目往往只有多项式量级。在用分治法求解时，有些子问题被重复计算了许多次。

和分治策略的异同：

分治：将问题划分为独立的子问题，子问题之间不包含公共的子子问题，递归地求解子问题，将子问题的解组合成愿问题的解。

动态规划：和分治法类似，要求愿问题具有最优子结构性质，即指问题的最优解包含其子问题的最优解，其基本思想也是将待求解问题分解为若干子问题，然后将子问题的解合并，形成愿问题的解。

与分治法不同，动态规划适用于有子问题重叠的情况，即不同的子问题具有公共的子子问题，动态规划对每个子问题只求解一次，将其解保存在一个表格中，再次遇到，无需重新计算，只要从表格中找到上次计算的结果并加以引用即可，避免不必要的计算工作。

何为Programming？

造表。动态规划和分治法有点类似，都是通过组合子问题的解来求解原问题。

分治法：讲问题划分为互不相交子问题，递归求解子问题，然后将子问题的解组合成原问题的解，如果字问题有重叠，那么递归求解中反复求解公共子问题，算法效率下降。

动态规划：适用于有子问题重叠的情况，即不同的子问题具有公共的子子问题，动态规划对每个子问题只求解一次，将其解保存在一个表格中，再次遇到，无需重新计算，只要从表格中找到上次计算的结果并加以引用即可，避免不必要的计算工作。

动态规划算法的步骤：

1. 找出最优解性质，刻画最优解的结构特征 （划分子问题
2. 递归的定义最优解的值 （给出最优解的递归式
3. 计算最优解的值，通常采用自底向上方法
4. 利用计算出的信息构造最优解

步骤1-3为动态规划算法基本步骤，只需求解最优值情形，步骤4可省略。若要求出问题的一个最优解，必须执行步骤4，步骤3记录的信息是构造最优解的基础。

钢条切割：

长度为n英寸钢条有2的n-1次方种切割方案。

如果一个最优解将总长度为n的钢条切割为k段，每段长度为ij，则有n=i1+i2+。。。+ik。

得到最大收益rn = pi1+pi2+。。。+pik。

对于长度为n的钢条，n大于等于1，设rn是最佳切割的收益，如何获得最优切割？

问题最优解的一个重要性质：最优子结构性。若a=b+c是最优切割收益，则b\c是相应子问题的最优切割收益。

一个自顶向下的递归求解过程：T（n）=2的n次方

钢条切割问题的动态规划求解：

仔细安排求解顺序，对每个字问题求解一次，并将结果保存下来。如果在此需要盖子问题的解，只需查找保存的结果，不必重新计算。动态规划需要付出额外空间保存子问题的解，典型时空权衡。

可获得的改进：动态规划节省了时间，将指数时间转化为多项式时间。

动态规划求解两种方法：

1. 带备忘的自顶向下法。记住已经计算出的结果，通常保存在一个数组或散列表中。
2. 自底向上法：将子问题按规模排序，按从小到大的顺序顺次求解。

自顶向下：递归设计框架

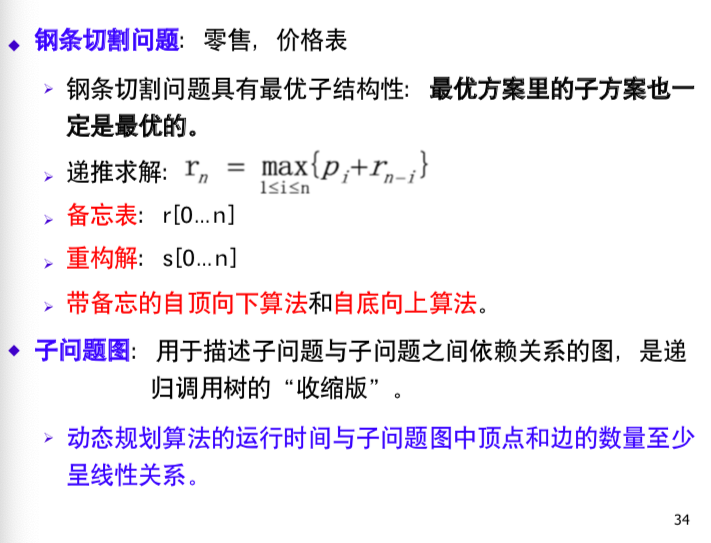
自底向上：迭代设计框架，两者通常具有相同的渐进运行时间西塔n平方。在某些特殊情况下，自顶向下法可能没有递归处理所有可能的字问题。由于自底向下法没有频繁的递归函数调用的开销，所有自底向下法的时间复杂性往往有更小的系数。

字问题图：用于描述子问题和子问题之间的依赖关系。一个有向图，每个顶点唯一的对应一个子问题。若求解子问题x的最优解时需要直接用问题y的最优解，则在子问题图上有一条从x指向y的有向边。子问题图是自顶向下递归调用树的简化版或收缩版。

动态规划算法的运行时间一般情况下与顶点和边的数量至少呈线性关系。

建立基于动态规划策略的计算过程容易吗？不容易，要根据问题的性质构造递推关系式，形成有效的计算过程。要点：子问题的定义rn = max{ pi+rn-i}&备忘表的结构r[0..n]。

钢条切割问题具有最优子结构性：最优方案里的子方案一定也是最优的。



矩阵链乘法：

A p\*r; B r\*q; C p\*q;计算C共需要pqr次标量乘法运算。

矩阵链乘问题相当于在矩阵之间加适当的括号，从而根据组合关系定义出矩阵链乘的计算模式。

问题：不同的加括号方式代表不同计算模式，不同计算模式计算矩阵链乘积的代价是不同的。

求一个完全“括号化方案”，使得计算成绩所需的标量乘法次数最小。

课后习题答案的网址：

https://walkccc.github.io/CLRS/