原题

四点考完，七点坐德旺敲的。

1. 已知$f(n)=6n^2, g(n)=n^2$ 要求用Θ的定义证明$f(n)=\Theta(g(n))$
2. 用递归求(1,2,..n)的全排列
3. 用动态规划求解最大公共子序列（要求原理、伪代码）
4.
    - 描述归并排序的算法
    - 给出m个已经按从小到大排好顺序的数组，求前k个最小的
5. 用最优队列求解最短路径
    - 求单源最短路径Dijkstra的算法
    - 求第二最短路径算法
6. 给定一个网络G=(V,E)，令f为G的流，G_f为G关于f的剩余网络，令f'为G_f的流

    证明：
    - f+f'仍为G的流
    - $|f+f'|=|f|+|f'|$
7. 用回溯法求解最大装载问题，给出最大重量W=12，$w=<8,6,2,3>$
    - 写出约束函数C(i)的伪代码
    - 写出限界函数B(i)的伪代码
    - 画出解空间树，标出每个结点的约束函数值和限界函数值
8. NP、NPC的概念
    - NP的英文全称
    - NPC的定义
    - 什么是验证算法
    - 如何证明一个问题为NPC

重点在于书本上的概念、定理证明、经典算法、经典题目分析

复习的时候整理了前7节的考点，大家可以参考对比一下

1. 算法的特点

- **有穷性**
- **可行性**
- **确定性**
- **输入**
- **输出**

2.算法正确性证明（循环不变量证明）

用循环不变量证明**算法的正确性**。对于一个给定的循环不变量，我们必须遵循三个原则。

- **初始步**：在循环的第一次迭代之前，循环不变量为真。
- **归纳步**：假设在循环的k次迭代之前循环不变量为真，那么在k+1次迭代之前循环不变量同样为真。
- **终止步**：当循环结束时，不变量能够提供我们有用的属性，用于帮助我们证实算法是正确的。

3. 时间复杂度

- Θ

  存在c_1，c_2，n_0，对于任意给定的$n \geq n_0$，满足

  $$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$

  则记$f(n) = \Theta(g(n))$，表示两个函数同阶

  在数学中极限的概念为

  $$\lim_{n->\infty} \frac{f(n)}{g(n)} = c \quad (0 < c < \infty)$$

- O

  存在c_1, n_0，对于任意给定的$n \geq n_0$，满足

  $$0 \leq f(n) \leq c_1 g(n)$$

  则记$f(n) = O(g(n))$，称g(n)为f(n)的渐进上界

  在数学中极限的概念为

  $$\lim_{n->\infty} \frac{f(n)}{g(n)} = 0$$

- Ω
  - 存在c_1, n_0，对于任意给定的$n \geq n_0$，满足

    $$0 \leq c_1 g(n) \leq f(n)$$

    则记$f(n) = \Omega(g(n))$，称g(n)为f(n)的渐进下界

    在数学中极限的概念为

    $$\lim_{n->\infty} \frac{f(n)}{g(n)} = \infty$$

- 渐进符号的性质

- $f(n)=\Theta(g(n))$当且仅当$f(n)=O(g(n))$且$f(n)=\Omega(g(n))$
- $f_1(n)=O(g_1(n))$，$f_2(n)=O(g_2(n))$，则$f_1+f_2=O(max\{g_1,g_2\})$
- 传递性

$$f(n)=\Theta(g(n)),g(n)=\Theta(h(n))$$
$$f(n)=\Theta(h(n))$$

- 可加性

$$f(n)=\Theta(h(n)),g(n)=\Theta(h(n))$$
$$f(n)+g(n)=\Theta(h(n))$$

- 自反性

$$f(n)=\Theta(f(n))$$

- 对称性(仅对Θ成立)

$$\text{有且仅有}g(n)=\Theta(f(n))\text{成立时，}f(n)=\Theta(g(n))\text{成立}$$

- $$n!\geq 2^n\geq n^2\geq nlgn\geq n\geq lgn$$

4. 算法分析

4.1 概率分析

目的：分析算法的平均时间复杂度

大假设：每个实例都有同样的机会作为算法的输入而被计算

exp1 雇佣问题：

考虑一个更实际的问题：假设你要聘用一个秘书。你对目前聘用的秘书不满意，你一直在试图寻找更合适的人选。因此，对当前的求职者面试之后，你决定，如果求职者的能力比当前秘书强，则解雇现在的秘书，并聘用该求职者。当然，面试的费用很低，而聘用的费用是昂贵的。你愿意支付这种策略所需要的费用，但是想预计整个费用是多大。

```
HireAssistant(n)
    best<-0
    for i<-1 to n do
        面试第i个人
        if 第i个人强于当前秘书:
            best<-i
            hire candidate i
```

其中best=0表示原来没有秘书，best表示当前最好的秘书。

设面试费用为$C_{interview}$，聘用费用为C_{hire}，m表示雇佣操作执行的次数

无论是否聘用，都需要面试。故算法总费用为$O(nC_{interview}+mC_{hire})$

- 最好情况

  面试的第一个秘书就是最好的秘书

  总费用为$O(nC_{interview}+C_{hire})$

- 最坏情况

面试的秘书一个比一个好，即每次都要聘用新秘书

总费用为$O(nC_{interview}+nC_{hire})$

- 平均情况

  面试第i个秘书，聘用他的概率为$P(i)$

  总费用为$O(nC_{interview}+P(1)C_{hire}+P(2)C_{hire}+...P(n)C_{hire})$

  关键为P(i)是什么：对于第i个求职者，他是前i个人里最好的那个的概率为$1/i$，故她被聘用的概率也为$1/i$

  由此可得，总费用为：

$$O(nC_{interview}+\sum_{i=1}^{n}1/iC_{hire})=O(nC_{interview}+lgnC_{hire})$$

exp2 电梯问题：

电梯里有12个人，每个人随机的选择10楼中的任一楼层出去。

问，电梯平均需要停下的次数是多少？

设在第i层停下的概率为$P(i)$

$$P(i)=1-\overline{P(i)}=1-(1-1/10)^{12}=1-0.9^{12}$$

总的停下的次数为

$$\sum_{i=1}^{10}P(i)=10*(1-0.9^{12})\approx 7.176$$

4.2 分摊分析

如果整个运算序列的总费用是小的，那么可以推断出每个运算的分摊费用也是小的。即使其中某个运算的费用很大，其他运算的低成本可以通过平均或分摊来抵消该运算的高成本，使得每个运算的平均费用保持较小。(类似生活费的使用)

如果每个操作是不一样的，但是这些操作的总体是固定的，那么我们就可以计算这些操作的总体，然后取平均

4.2.1. 合计方法

将n个操作一起考虑，考虑总体平均的最坏情况，而不是只考虑单个操作最坏的情况

在最坏情况下，由n个运算构成的总体的运行时间和为T(n)；每个运算的分摊费用可定义为T(n)/n。

注意这个分摊费用的计算方法对每个运算都是成立的，即使当序列中存在几种类型的运算时也一样。

exp1. 出入栈

现在对一个初始为空的栈，执行一个由n个Push、Pop和MultiPop运算构成的序列所花费的时间进行分析。栈运算MultiPop(S,k),它弹出栈S顶部的k个元素，但如果栈中元素的个数小于k,则此运算将把栈清空。

- 普通方法

  MultiPop最多可执行n次，由于栈的大小为n,单次运算的时间复杂度最多为$O(n)$。

  最坏情形的总费用为$O(n^2)$，每个运算的平均费用为$O(n^2)/n=O(n)$

- 合计方法

  弹出的数量不可大于压入栈的数量。故MultiPop中的Pop数量加上Pop的数量总和a最多等于Push的次数b。有

  $$a \le b 且 a + b = n$$
  $$求 max\{a + b\}$$

  因此整个运算的费用最多为$O(n)$，每个运算的平均/分摊费用为$O(n)/n = O(1)$

exp2. 计数器

用数组$A[0, .., k-1]$存储一个k位的二进制数，低位在A[0]，高位在A[k-1]。初始状态A=0，问做n次加一运算Increment的最坏时间复杂度是多少？

二进制运算的时间复杂度的主要来源是位的翻转

- 普通方法

  当A全部为1时，为单次Increment的最坏情况，此时所有位数都需要翻转。时间复杂度为$O(k)$

  做n次Increment的最坏时间复杂度为$O(k) * n = O(kn)$，每个运算的分摊费用为$O(kn)/n = O(k)$

- 合计方法

  从初始状态做n次Increment，不是每次都会有k次翻转的。

  列出表格可知，最A[0]每次都翻转，A[1]每两次Increment翻转一次，A[2]每4次Increment翻转一次....A[i]每2^i次Increment翻转一次。且$2^i \le n$，$i \le log_2 n$

  $$\sum_{i=0}^{log_2 n} \frac{n}{2^i} \approx 2n$$

  做n次Increment的最坏时间复杂度为$O(n)$，每个运算的分摊费用为$O(n)/n = O(1)$

4.2.2. 记账方法

将不同的运算赋予不同的费用，当其他操作执行时消耗/增加这些操作的存款

在选择每个运算的分摊费用时，必须使总的分摊费用为总的实际费用的上界。令序列中的第i运算的实际费用为c_i，分摊费用为$\hat{c}_i$，则需要保证

$$\sum_{i=1}^{n} c_i \le \sum_{i=1}^{n} \hat{c}_i$$

主要思想：用$\hat{c}_i$代替c_i，证明$\sum_{i=1}^{n} c_i \le \sum_{i=1}^{n} \hat{c}_i$成立，最后得到总体平均时间复杂度$O(\hat{c}_i n)$

4.3. 势能方法(略)

将存款赋予整体数据结果

太难了，略

5、递归：替换方法、递归树估计复杂度、公式法

递归复杂度分析

$$定义T(1)=1；T(0)=0$$

- ## 替换方法

  1. 猜测或通过递归树得出复杂函数的上界
  2. 通过数学归纳法证明上界的正确性

  exp 6.

$$T(n)=T(n-1)+(n-1)$$

  分析其递归复杂度

  - 猜测$T(n)=O(n^2)$
  - 证明$T(n)\le cn^2$
    1. 当n=1时，$T(1)=1\le c*1^2$，当$c\ge 1$时
    2. 假设在n=k-1时成立，$T(k-1)\le c(k-1)^2$

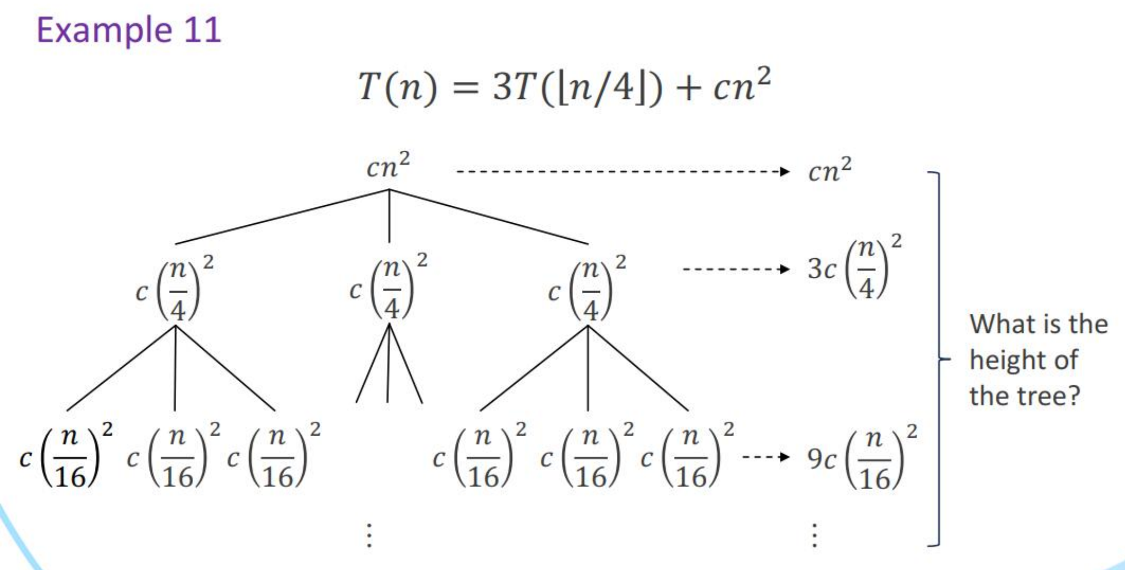
$$\begin{aligned}T(k)&=T(k-1)+(k-1)\\&\le c(k-1)^2+(k-1)\\&=ck^2-2ck+c+k-1\\&\le ck^2-2ck+2c+k-1\\&=ck^2+(k-1)(1-2c)\\&\le ck^2(c\ge\frac{1}{2}时)\end{aligned}$$

  替换法的问题是猜测是正确的，但是数学归纳法推导不出来。一般是因为假设的算法复杂度的上界O不够紧导致的。将**猜测的上界减去一个低阶的项**，这样能够用数学归纳法推导出算法的正确上界

可以使用递归树来对替换方法中的复杂度做一个合理的猜测。递归树这种方式不是一个严格的证明，我们还是要用之前的替代法的方式来证明算法的界

- ## 递归树

  将递归树上所有层上的所有节点的cost加起来就就可以得到一个估计



  设其中有k层，k满足$(\frac{n}{4^k})=1$，$k=log_4n$

总的递归复杂度T(n)为

$$
\begin{aligned}
T(n) &= cn^2+3c(\frac{n}{4})^2+9c(\frac{n}{16})^2+\ldots \\
&= cn^2+\frac{3}{16}cn^2+(\frac{3}{16})^2cn^2+(\frac{3}{16})^3cn^2+\ldots \\
&= \sum_{i=0}^{log_4 n}(\frac{3}{16})^i cn^2 \le \sum_{i=0}^{\infty}(\frac{3}{16})^i cn^2 = cn^2\frac{1}{1-\frac{3}{16}} = O(n^2)
\end{aligned}
$$

- **公式法**

$$T(n) = aT(n/b)+f(n)$$

  - 当$f(n)=\Theta(n^{log_b a})$时

$$T(n)=\Theta(n^{log_b a}lgn)$$

  - 当$f(n)\neq\Theta(n^{log_b a})$，谁大选谁

$$
F = max\{f(n), n^{log_b a}\} \\
T(n) = \Theta(F)
$$

exp13.

$$T(n)=9T(n/3)+n$$

a=9,b=3,f(n)=n，$n^{log_b a}=n^2$

$f(n)=O(n^{log_b a})$，选$n^{log_b a}$

$$T(n)=\Theta(n^2)$$

exp14.

$$T(n)=T(2n/3)+1$$

$a=1$，$b=\frac{3}{2}$，$f(n)=1$，$n^{log_b a}=n^0=1$

$f(n)=\Theta(n^{log_b a})$

$$T(n)=\Theta(n^{log_b a}lgn)=\Theta(lgn)$$

exp15.

$$T(n)=3T(n/4)+nlgn$$

---

6. 分治：经典排序、大数乘法、矩阵乘法等书上算法

> 常考点
>
> 排序的算法思路必须掌握，概率较大

1. 归并排序

- 证明归并排序的正确性
- 归并排序的递归方程与时间复杂度

$$T(n)=2T(n/2)+\Theta(n)$$

$\Theta(n)$：合并子问题的解的时间

使用公式法可得$T(n)=\Theta(nlgn)$

2. 快速排序

- 证明快速排序的正确性
- 快速排序的递归方程与时间复杂度

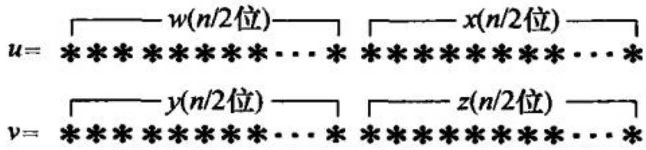
$$T(n)=2T(n/2)+\Theta(n)$$

$\Theta(n)$：合并子问题的解的时间

使用公式法可得$T(n)=\Theta(nlgn)$

3. 大数乘法

假设两个整数u,v分别用n位的二进制表示，每个整数可分解为高位、低位两部分，每部分为n/2位。假设整数u分成w和x，整数v分为y和z两部分



$$u=w*2^{n/2}+x$$
$$v=y*2^{n/2}+z$$
$$uv=wy2^n+(xy+zw)2^{n/2}+xz$$

值得注意的是，乘以2^n表示向左位移n位，这个运算耗时$\Theta(n)$。

- 递归方程与时间复杂度

  要算出T(n)，需要4次两个n/2位整数的乘法，以及三次加法运算，后者耗时$\Theta(n)$，即

$$T(n)=4T(n/2)+\Theta(n)$$

  利用公式法可得$T(n)=\Theta(n^2)$

4. 矩阵乘法

计算两个nxn的矩阵AB的乘积C

可将A,B分为4个n/2xn/2的矩阵

$$\boldsymbol{A}=\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \quad \boldsymbol{B}=\begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \quad \boldsymbol{C}=\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

$$\boldsymbol{C}=\begin{pmatrix} A_{11}B_{11}+A_{12}B_{21} & A_{11}B_{12}+A_{12}B_{22} \\ A_{21}B_{11}+A_{22}B_{21} & A_{21}B_{12}+A_{22}B_{22} \end{pmatrix}$$

- 递归方程与时间复杂度

因此，两个n×n矩阵乘积的计算量是2个n/2×n/2矩阵乘积计算量的8倍，再加上n/2×n/2阶矩阵相加的4倍，后者最多需要$\Theta(n^2)$，因此有

$$T(n) = \begin{cases} \Theta(1) & n=1 \\ 8T(n/2)+\Theta(n^2) & \end{cases}$$

改进

根据Strassen算法，原来需要求解8个子问题，现在只需要求解7个

$$T(n) = \begin{cases} \Theta(1) & n=1 \\ 7T(n/2)+\Theta(n^2) & \end{cases}$$

7. 动态规划：最优子结构的证明、装配线调度、矩阵链乘、最长公共子序列、01背包、最优二叉搜索树

> 常考点，都需要掌握
>
> 较难。

1. 动态规划的算法思想

动态规划算法的计算步骤:

- 找出最优解的结构

  构造出原问题和它的子问题之间的递归方程，例如$F(n)=F(n-1)+F(n-2)$
- 递归定义一个最优解的值
- 将求出的子问题的值保存在一个表(一般为数组)中，以方便数据的保存和读取
- 以递归出口为计算的起点，自底向上将数组填满

最优子结构

问题中最优解中所包含的子问题的解为子问题的最优解

2. 装配线调度问题

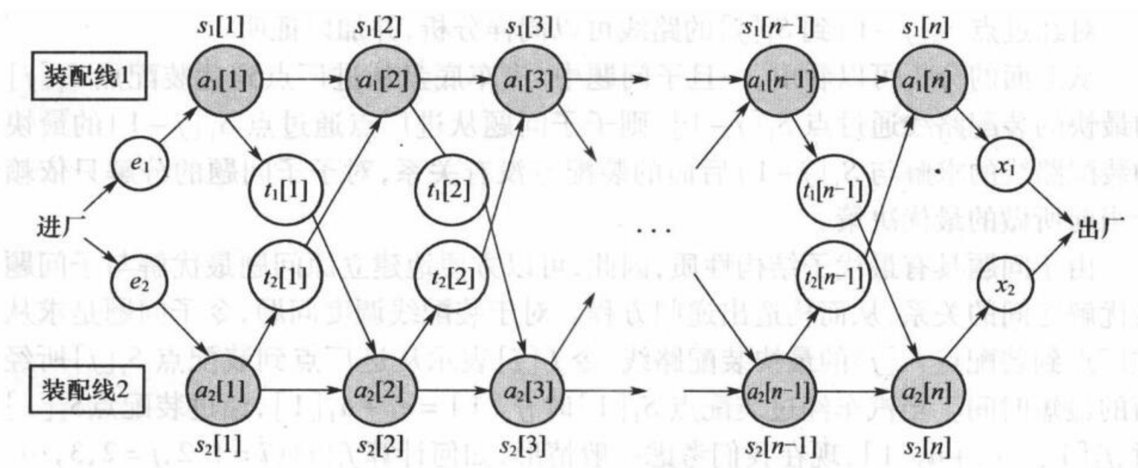


图 6.2

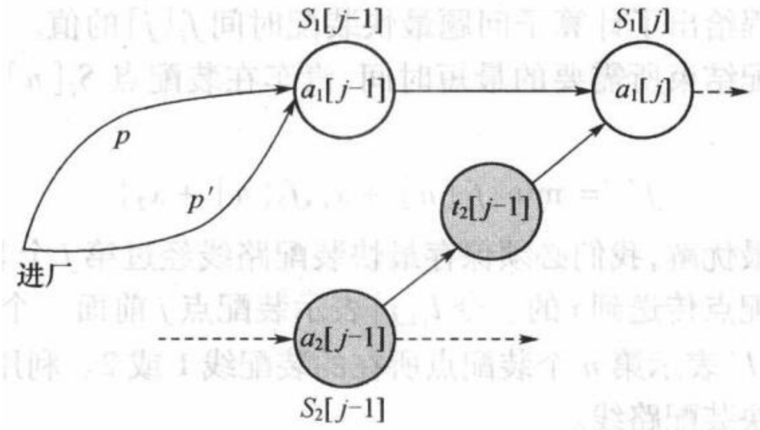
- 最优子结构

考虑子问题，汽车从进厂到装配点$S_1[j]$的最短装配时间问题。有以下三种情况

- 如果j=1，需要计算到达$S_1[1]$需要的时间
- j>1，到达$S_1[j]$有两种走法
  1. 从$S_1[j-1]$直接到$S_1[j]$
  2. 从$S_2[j-1]$经过时间$t_2[j-1]$到$S_1[j]$

定理6.1 若到达$S_1[j]$的最短装配路线是从进厂点经过路线p到$S_1[j-1]$再到$S_1[j]$，则p也一定是从进厂点到$S_1[j-1]$最快的装配路线。(用反证法证明)

对经过$S_2[j-1]$再到$S_1[j]$的情况可同理分析



- 构造原问题最优解与其子问题最优解之间的关系

  递归

$$T_1[j]=min(T_1(j-1)+a_1[j],T_2[j-1]+t_2[j-1]+a_1[j])$$

  递归出口为j=1时，即$T_1[1]$、$T_2[1]$

  时间复杂度为$O(2^n)$

  动态规划

  设$f_i[j]$表示从进厂点到装配点$S_i[j]$所经历的最短时间。将递归出口作为动态规划求解起点

  - 起点

    $f_1[1]=e_1+a_1[1]$，$f_2[1]=e_2+a_2[1]$

  - 动态规划方程

$$f_1[j]=min(f_1[j-1]+a_1[j],f_2[j-1]+t_2[j-1]+a_1[j])$$
$$f_2[j]=min(f_2[j-1]+a_2[j],f_1[j-1]+t_1[j-1]+a_2[j])$$

    采用自底向上的方式求解，从而避免大量重复计算

    时间复杂度为$O(n)$，比指数级的递归有效很多

3. 矩阵链乘

给定一个有n个矩阵的矩阵链A1,A2,...,Ai,...,An,矩阵Ai:(1≤i≤n)的维数为$p_{i-1}\times p_i$，矩阵链乘法问题就是如何对矩阵乘积A1A2...Ai...An加括号，使得它们的标量乘法次数达到最少。

考虑有三个矩阵的矩阵链 A_1,A_2,A_3 的乘积问题。假设它们的维数分别是 10×100, 100×5 和 5×50,则

计算 $((A_1A_2)A_3)$:

$A_1A_2=10\times100\times5=5000$, A_1A_2 的维数为 10×5,因此

$$((A_1A_2)A_3) = 10\times5\times50 = 2500$$

总计 7500 次标量乘法运算。

计算 $(A_1(A_2A_3))$:

$(A_2A_3)=100\times5\times50=25000$, A_2A_3 的维数为 100×50,因此

$$(A_1(A_2A_3)) = 10\times100\times50 = 50000$$

总计 75000 次标量乘法运算。

- 最优子结构

  记$A_{i..j}=A_iA_{i+1}\dots A_j$

  设$A_{i..j}$的一种最优完全括号化方式是把A_k和A_{k+1}用括号隔开，即

  $$A_{i..j}=A_{i..k}A_{k+1..j}$$

  定理6.2

  若$A_{i..j}$的一种最优完全括号化方式是把A_k和A_{k+1}用括号隔开，则其中子问题$A_{i..k}$和$A_{k+1..j}$的加括号方式也一定是最优的（用反证法证）

- 构造原问题最优解与其子问题最优解之间的关系
  - **递归**

    设$P(n)$为n个矩阵有几种加括号的方式

    $$P(n)=\begin{cases}1 & n=1\\ \sum_{k=1}^{n-1}P(n)P(n-k) & n\geq 2\end{cases}$$

    时间复杂度为$O(2^n)$
  - 动态规划

    $$A_{i..j}=A_{i..k}A_{k+1..j}$$

```
设$m[i,j]$为$A_{i..j}$矩阵链相乘的乘法次数，$p_x$为第x个矩阵的列数
$$
m[i,j]=\begin{cases}
0 &i=j\\
\underset{i\leq k<j}{min}(m[i,k]+m[k+1,j]+p_{i-1}p_{k}p_{j}) & i<j
\end{cases}
$$
```

4. 最长公共子序列

公共子序列定义：z的所有元素都在x中，并且是在x中是从左到右（不一定相邻）排列的

- 最优子结构

定理 6.3 给定两个序列 $X_m = \langle x_1, x_2, \cdots, x_m \rangle$ 和 $Y_n = \langle y_1, y_2, \cdots, y_n \rangle$，并令 $Z_k = \langle z_1, z_2, \cdots, z_k \rangle$ 为 X_m 和 Y_n 的某个最长公共子序列，则

(1) 若 $x_m = y_n$，那么 $z_k = x_m = y_n$ 且 Z_{k-1} 就是 X_{m-1} 和 Y_{n-1} 的一个最长公共子序列；

(2) 若 $x_m \neq y_n$，且 $z_k \neq x_m$，那么 Z_k 就是 X_{m-1} 和 Y_n 的一个最长公共子序列；

(3) 若 $x_m \neq y_n$，且 $z_k \neq y_n$，那么 Z_k 就是 X_m 和 Y_{n-1} 的一个最长公共子序列。

- 构造原问题最优解与其子问题最优解之间的关系
  - **递归**

    对X的每个子序列枚举，验证是否是Y的子序列。X的长度为m，Y的长度为n。X有2^m个子序列，要验证每个子序列是否为Y的一个子序列，要花费$\Theta(n)$的时间。

    故算法的时间复杂度为$\Theta(n2^m)$
  - **动态规划**

    设c[i,j]表示X_i和Y_i的最长公共子序列的长度。

    考虑找子问题$X_i =< x_1, x_2, ..x_i >$ 和$Y_j =< y_1, y_2, .., y_j >$的最长公共子序列长度

    递归出口：当i=0或j=0时，$c[i,j] = 0$

    1. 当$x_i = y_j$时

    $$c[i,j] = c[i-1, j-1] + 1$$

    2. 当$x_i \neq x_j$时

    $$c[i,j] = max(c[i,j-1], c[i-1,j])$$

    $$c[i,j] = \begin{cases} 0 & i = 0/j = 0 \\ c[i-1,j-1]+1 & x_i = y_j \\ max(c[i,j-1], c[i-1,j]) & x_i \neq y_j \end{cases}$$

    时间复杂度为$O(m+n)$

5. 01背包

- 最优子结构

定理 6.4 假设子问题(i, w)的最优装载中含有物品 i，则其中子问题$(i-1, w-w_i)$的装载方案也一定是最优的。

- 构造原问题最优解与其子问题最优解之间的关系
  - 动态规划

    V[i,w]为将物品1~物品i装入载重为w的背包中能获得的最大价值，有以下两种情况

    1. 物品i放不进背包

    $$V[i,w] = V[i-1,w]$$

    2. 物品i可以放入背包

    $$V[i,w] = max(V[i-1,w], V[i-1,w-w_i] + v_i)$$

    $$V[i,w] = \begin{cases} V[i-1,w] & w_i > w \\ max(V[i-1,w], V[i-1,w-w_i] + v_i) & w_i \leq w \end{cases}$$

6. 最优二叉搜索树

二叉搜索树：$左孩子值 \le 根结点 \le 右孩子值$

构造一棵二叉树，使得在所有搜索中，访问节点数最小，即求最优二叉搜索树。

设在搜索树上的关键字为序列$K=<k_1,k_2,..,k_n>$，其中$(k_1\le k_2\le..\le k_n)$。设p_i为k_i被搜索到的概率。

设不在二叉搜索树上的关键字为序列$D=<d_0,d_1,..,d_n>$，其中d_0表示所有比k_1小的关键字，d_n表示所有比k_n大的关键字。设q_i为d_i被搜索到的概率

有$\sum_{i=1}^n p_i+\sum_{i=0}^n q_i=1$，其中$\sum_{i=1}^n p_i$表示搜索成功的概率，$\sum_{i=0}^n q_i$表示搜索失败的概率

搜索k_i的代价为从根结点到k_i的访问节点数，即为k_i的深度($d_T(k_i)$)+1。则搜索整棵二叉搜索树的代价为

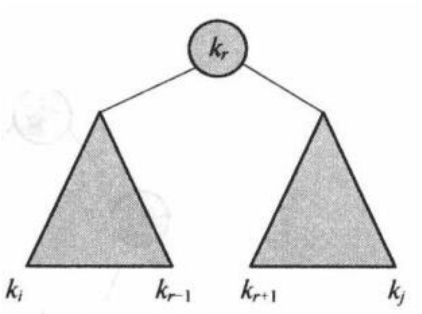
$$E(T)=\sum_{i=1}^n (d_T(k_i)+1)p_i+\sum_{i=0}^n (d_T(d_i)+1)q_i$$

- 最优子结构

  定理 6.5 若一棵最优二叉搜索树 T 有一棵包含关键字 $k_i,\cdots,k_j$ 以及 $d_{i-1},\cdots,d_j$ 的子树 T'，那么这棵子树 T' 是子问题(i,j)的最优二叉搜索树。

- 构造原问题最优解与其子问题最优解之间的关系

  考虑子问题(i,j)，利用关键字$<k_i,k_{i+1},..,k_j>$和$<d_{i-1},d_i,..d_j>$构造最优二叉搜索树。假定从$<k_i,k_{i+1},..,k_j>$中选择一个k_r作为最优二叉搜索树的根，则可分解为构造左右子树的最优二叉搜索树的问题



  - 动态规划

    当j=i-1，树中仅有d_{i-1}，因此$e[i,j]=q_{i-1}$

    当$j\ge i$，选择k_r作为树根。左右子树中的每个结点深度加一，因此搜索期望代价增长为

$$w[i,j]=\sum_{i=1}^n p_i+\sum_{i=0}^n q_i$$

$$e[i,j]=\begin{cases} q_{i-1} & j=i-1 \\ \min\limits_{i\le r\le j}(e[i,r-1]+e[r+1,j]+w[i,j]) & i\le j \end{cases}$$

- 贪心：贪心选择结构证明、任务选择
- 图：Prim、Kruskal、FloydWarshall、Dijkstra（只有这四个

  > 不考各种图的二级结论，只需要掌握经典算法即可

- 网络流

> 不会考太难，掌握基本概念、如何通过增广路径求最大流即可。
>
> 增广那块会考定理推理
>
> 匹配问题、Dinc、二分图匹配都不怎么考

- P 、NP、NPC

  > 必考点
  >
  > 简单的话只出基本概念
  >
  > 难的话要求证NPC

> 一个又想署名又怕被找的小本科生一枚