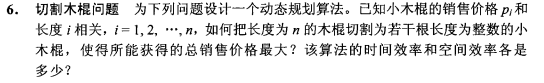
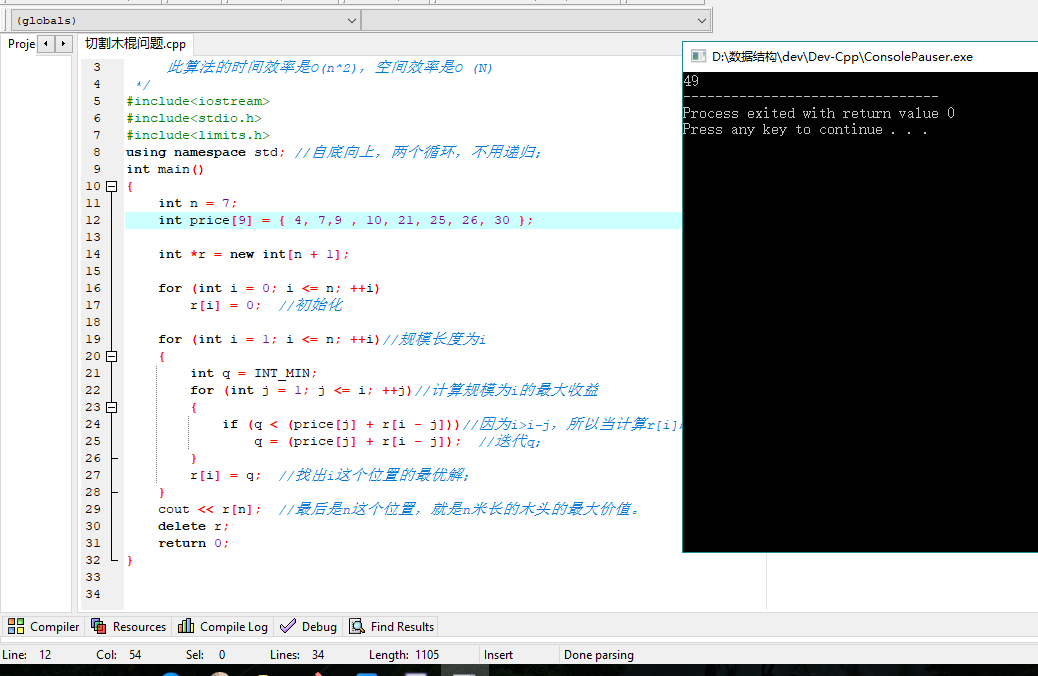
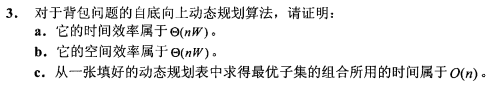


1. 完全使用插入排序时所花时间为143ms，完全使用快速排序时所花的时间为5.89ms，使用k值并且取最接近logn的13时算法最优，所用时间为2.35ms，因此快速排序相对于插入排序减少了95.8%的时间。
2. 快速排序于插入排序混合使用，在算法中插入一个参数k，当快速排序划分的数组元素个数小于k时，执行直接插入排序，当个数大于k时，执行快速排序。设元素个数为n，则k的最优值选为小于logn的最大整数。







a、可以将背包问题的求解过程看作是进行一系列的决策过程，即决定哪些物品应该放入背包，哪些物品不放入背包。如果一个问题的最优解包含了物品n，即xn=1，那么其余x1，x2，...，x(n-1)一定构成子问题1，2，...，n-1在容量W-wn时的最优解。如果这个最优解不包含物品n，即xn=0，那么其余x1，x2，...，x(n-1)一定构成子问题1，2，...，n-1在容量W时的最优解。设c[i,w]表示背包容量为w时，i个物品导致的最优解的总价值，得到下式。显然要求c[n,w]。

C[i,w]= 0 i=0或w=0

C[i-1,w] wi>w

Max{c[i-1,w-wi] +vi，c[i-1,w] i>0且wi<=w

b、设计用二维数组对物品信息进行记录： C[i][j] 用来记录如果当前还有ii个物品，背包容量还剩jj的情况下，当前背包所能得到的最大价值。

很容易发现条件即：

C[0][j]=C[i][0]=0

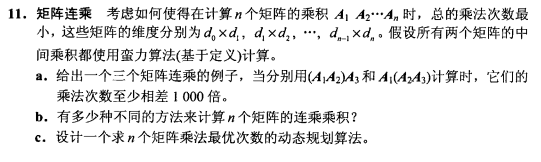
递归定义应该为：

C[i][j]={C[i−1][j], j<wi

max{C[i−1][j],C[i−1][j−wi]+vi }, j≥wi

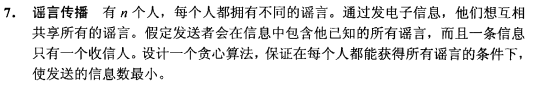
    可以这样理解，每个物品我可以选择是否加入到背包中，首先判断，当前物品是否重量已经大于背包所能容纳的重量；如果能容纳该物体，则进行判断加入该物品C[i−1][j−wi]+vi得到的价值更高，还是不加入该物品C[i−i][j]所能得到的物品的总价值更高。只需要遍历进行，所以只需要考虑循环中的复杂度即可。

T(n)=O(n)+O(W)+O(nW)=O(nW)



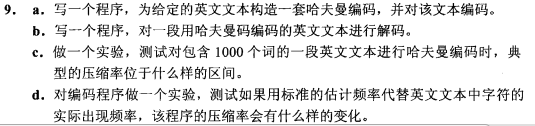
1. 计算三个矩阵连乘{A1，A2，A3}；维数分别为10\*100 , 100\*5 , 5\*50, 按此顺序计算需要的次数((A1\*A2)A3):10X100X5+10X5X50=7500次，按此顺序计算需要的次数(A1(A2\*A3)):100\*5\*50+10\*100\*50=75000次数乘，是前一种计算方式的10倍。
2. 动态规划、strassen矩阵乘法、大整数乘法
3. 以四个矩阵相乘为例，我们定义一个二维数组，数组行列长度为矩阵个数，数组的元素所在第i行，第j列，这个元素的数值就表示从第i个矩阵到第J矩阵所用的最少数乘次数。我们选取半三角。

我们先定义对角线上的元素为0，因为他们表示本身到本身的数乘次数，按照四条箭头的方向依次确定所在元素数值，这样下一条箭头利用上一条箭头的最优解能够顺利的求解出元素数值，以A1——A4为例，我们只需利用循环，从A1开始，一直到A4结束（不包括），我们在A1——A4之间中断，在A2,A3出中断，在A2出中断，A1——A2+A3——A4+（A1——A2合并的数组乘以A3——A4合并的数组相乘所需要的数乘次数），这样我们就可以利用上一条箭头为我们求出的最优解（A1——A2）和（A3——A4）。



将这n个人标记为1, 2, …, n，按照1发信给2, 2发信给3, 3发信给4，…，n-1发信给n的方式发送谣言，该贪心算法基于每次发信都使得当前收信人掌握的谣言更多，最后由n将所有谣言发送给其他n-1个人。

发送信息总数为2n-2，这是最小的发信息数。因为每增加一个人，至少需要增加两次发送信息，当n=2是，发送信息数为2，归纳法可证明2n-2为最小发信息数。



1. 