算法分析第三次作业

算法分析题

**3-1题答案：**

**答：**

输入包含 n个元素的序列 S：S[0],S[1],S[2],⋯,S[n−1]。

输出序列 S的最长单调递增子序列的长度。

1.创建一个长度为n的数组A，将所有元素初始化为 1，表示以每个元素为结尾的最长递增子序列的初始长度为 1。

2. 初始化变量:maxlen=1，追踪最长递增子序列的长度。

3. 外层循环，i从 1到 n−1：内层循环，j从 0到 i−1。

若S[i]>S[j]，则更新

A[i]=max(A[i],A[j]+1)。

更新maxlen=max(maxlen,A[i])。

4. 返回 maxlen作为最长递增子序列的长度，根据 maxLen在 A[i]中的位置找到子序列的位置。

时间复杂度：含两层嵌套循环，故时间复杂度为O(n2)。

**3-4题答案：**

**答：**

定义一个三维的动态规划数组 dp，其中dp[i][j][k] 表示前i个物品，使用容量j，容积k所能得到的最大价值。

如果不选择物品i，则 dp[i][j][k] = dp[i-1][j][k]

如果选择物品 i，则 dp[i][j][k] = dp[i-1][j-w[i]][k-v[i]] + p[i]，其中 p[i] 是物品i的价值。

dp[i][j][k] = max(dp[i][j][k], dp[i-1][j-w[i-1]][k-v[i-1]]

+ p[i-1])

核心三重循环可知，O(ncd)

算法实现题

**3-3题答案：**

**答：**

最⼩得分递推式：

最⼤得分递推式：

cyclic表⽰循环操作，确保⾸尾相连。⽤两层嵌套循环来计算dp\_min、dp\_max。

读取输⼊数据：n（⽯⼦堆数）和⼀个包含n个整数的数组stones（每堆⽯⼦的个数）。

初始化两个⼆维数组 dp\_min 和 dp\_max ，均为⼤⼩为nxn的数组，表⽰最⼩得分和最⼤得分。

dp\_max [i][j]表示将stone中第i堆到第j堆石子合并的最大得分。初始时，dp\_min[i][i]和dp\_max[i][i]都等于0。从长度为2的区间开始，逐渐增加区间长度，对于每个区间[i,j]，分为[i,k]和[k+1,j]，计算两部分合并的得分，更新min和max。(递推式)

min[i][i+n-1]和max[i][i+n-1]就分别是将这个区间内的石子合并的最小得分和最大得分。遍历dp\_min和dp\_max两个数组，得到最小值和最大值。

**3-13题答案：**

**答：**

定义数组dp[i][j]表示将I的前i位划分为j段的最大乘积，num[i][j]表示 I 从第i位到第j位组成的数字。

状态转移方程：dp[i][j]=max(dp[i][j],dp[k][j-1] \* num[k + 1][i])。当j = 1时，dp[i][1] = num[1][i]。最终答案是dp[n][k]。

先计算num数组，再初始化dp数组，最后按状态转移方程计算。i从1到n，j从2到k，m从1到i循环。

计算num数组时间复杂度： O(n2)初始化dp数组时间复杂度为O(n)，状态转移方程计算时间复杂度为O(kn2)，所以算法总复杂度为O(kn2)。

**3-14题答案：**

**答：**

创建⼀个五维数组 dp[a][b][c][d][e],选择a件第1种商品、b 件第2种商品、c件第3种商品、d 件第4种商品、e件第5种商品情况下的最少费⽤。priceA、priceB、priceC、priceD、priceE 分别表⽰不同种商品的单价。对于每⼀种组合I,定义 A[i]、B[i]、C[i]、D[i]、E[i] 表⽰第i种组合下不同种类商品需要的数量，price[i]则表⽰第i种组合的花费费⽤。

初始化dp[a][b][c][d][e]为不考虑组合优惠时的花费:a\*priceA + b\*priceB + c\*priceC + d\*priced + e\*priceE 。

对于每⼀种组合i考虑应⽤组合优惠：更新dp[a][b][c][d][e]为dp[a-A[i]][b-B[i]][c-C[i]][d - D[i]][e-E[i]]+price[i]，如果 a>=A[i]且b>=B[i]且c>= C[i]且d>=D[i]且e>= E[i]。

在s∈S、a∈Ka、b∈Kb、c∈Kc、d∈Kd、e∈Ke情况下，计算所有可能的 dp[a][b][c][d][e]。最终的答案即为 dp[Ka][Kb][Kc][Kd][Ke] 。

时间复杂度：O(S \* K5)，其中 S 表⽰组合数，K 表⽰购买每种商品的最⼤数量。

**3-17题答案：**

**答：**

**数据输入**

从文件input.txt读取数据，第一行是字符串A，第二行是字符串B，第三行是空格与其他字符的距离定值k。

**结果输出**

将字符串A和B的扩展距离输出到文件output.txt 。

**算法描述**

定义dq(i,j)表示A的子串Ai​=a1​a2​⋯ai与B的子串Bj​=b1​b2​⋯bj​的扩展距离， d(a,b)表示字符a与b的距离。

当ai​对应空格：dq(i,j)=dq(i−1,j)+k。

当bj​对应空格：dq(i,j)=dq(i,j−1)+k。

当ai​对应bj​：dq(i,j)=dq(i−1,j−1)+d(ai​,bj​)。

取上述三种情况的最小值：dq(i,j)=min{dq(i−1,j)+k, dq(i,j−1)+k, dq(i−1,j−1)+d(ai​,bj​)}。

**算法分析**

设A的长度为m，B的长度为n，计算每个dq(i,j)的时间复杂度为O(1)，故算法时间复杂度为O(mn)。