1. DP方法可用来解决具有哪两个特点的问题？
   1. 具有最优子结构
   2. 子问题的重叠性
2. 使用DP方法求解的问题与使用分治法求解的问题有何不同？

动态规划也是一种分治思想，但分只是自顶向下动态规划先自底向上求解最小的子问题，然后把结果存到查找表中，然后在求解大问题时直接从表中查找解，避免重复计算。

1. Levenshtein提出了什么样的字符串间距离的度量方式？给出Levenshtein距离的公式化表示。

一个字符串变换为另一个字符串时的最小编辑操作数（单个字符插入、删除、替换）

1. 说明Levenshtein距离DP算法中的最优子结构和子问题重叠性

最优子结构：从动态规划方程

这是递归关系式，而第（i>0,j>0）项是由上面，左面和左上的元素决定

子问题重叠性：最终的编辑操作与步骤是查表得到的。

1. 给出Levenshtein距离DP算法的伪代码，并描述其复杂度。

for I = 0 to m

E[I,0] = I;

for Ij = 0 to n

E[0,j] = j;

For I = 1 to m

For j = 1 to n

E[I,j] = min{E[i-1,j]+1, E[I,j\_1]+1, E[i-1,j-1]}

Return E[m,n]

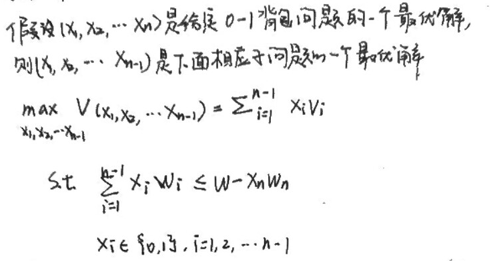
1. 从下页幻灯片左右两栏中各选择1组单词给出计算它们之间距离DP方法的手算表和计算结果。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | S | U | N | N | Y |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| S | 1 | 0 | 1 | 2 | 3 | 4 |
| N | 2 | 1 | 1 | 1 | 2 | 3 |
| O | 3 | 2 | 2 | 2 | 2 | 3 |
| W | 4 | 3 | 3 | 3 | 3 | 3 |
| Y | 5 | 4 | 4 | 4 | 4 | 3 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | D | I | F | F | E | R | E | N | C | E |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| F | 1 | 1 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| R | 2 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 6 | 7 | 8 |
| I | 3 | 3 | 2 | 3 | 4 | 4 | 5 | 5 | 6 | 7 | 8 |
| E | 4 | 4 | 3 | 3 | 4 | 4 | 5 | 5 | 6 | 7 | 7 |
| N | 5 | 5 | 4 | 4 | 4 | 5 | 5 | 6 | 5 | 6 | 7 |
| D | 6 | 5 | 5 | 5 | 5 | 5 | 6 | 6 | 6 | 6 | 7 |

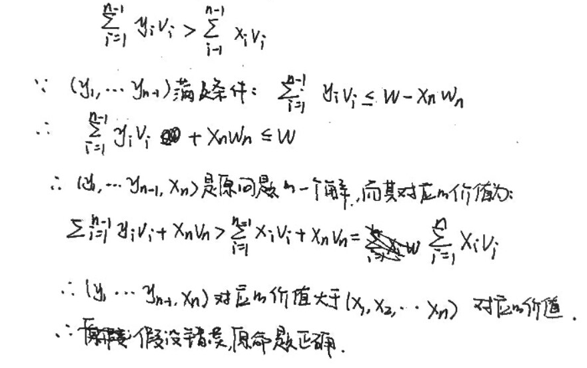
1. 描述0-1背包问题的最优子结构性质，并给出该性质的证明；说明0-1背包问题具有子问题重叠性。

性质：

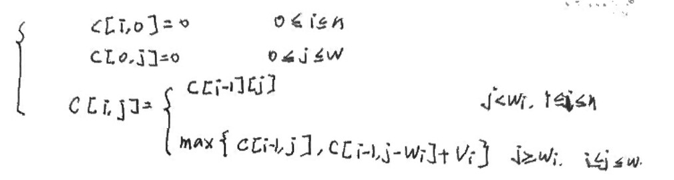


证明：

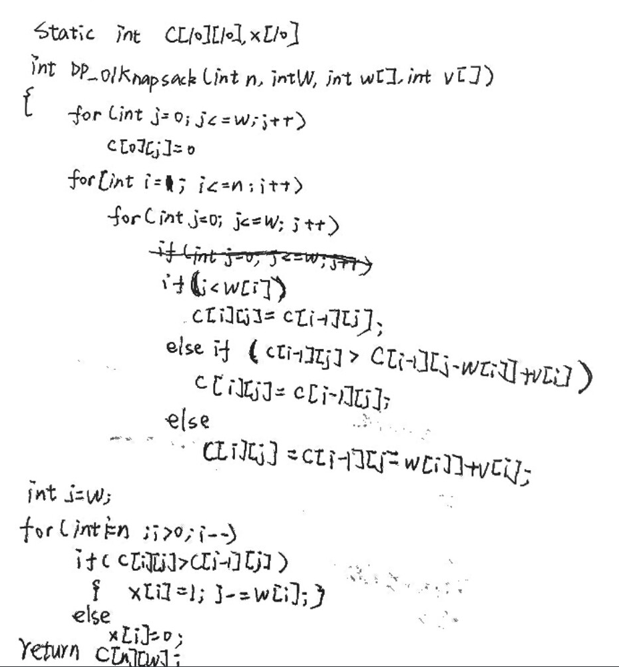
设（x1,x2,……,xn-1）不是上述子问题最优解，而存在(y1,y2,……,yn-1)是上述子问题的一个最优解，则最优向量(y1,y2,……,yn)对应的目标函数值比(x1,x2,……,xn-1)对应的大，即：



1. 给出0-1背包问题的动态规划方程。



1. 给出0-1背包问题DP算法的伪代码，并说明其复杂度，为什么说该复杂度是“伪多项式”的？



1. 针对物品重量为2,3,4,7、价值为1,3,5,9、容量为10的0-1背包问题，给出其DP算法的手算表和计算结果。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 1 | 2 | 3 | 4 |
| 重量 | 2 | 3 | 4 | 7 |
| 价值 | 1 | 3 | 5 | 9 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | J=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| I=0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 |
| 3 | 0 | 0 | 1 | 3 | 5 | 5 | 6 | 8 | 8 | 9 | 9 |
| 4 | 0 | 0 | 1 | 3 | 4 | 9 | 9 | 10 | 12 | 14 | 14 |

∴放2和4

1. 在CAAIS上学习Levenshtein距离DP算法，选长度均不低于7的两个字符串，以20%的交互进行练习，并提交结果。
2. 在CAAIS上学习0-1背包问题的DP算法，选个数不小于5的物品，以20%的交互进行练习，并提交结果