**一、用动态规划方法手工求解下面的问题**

某工厂调查了解市场情况，估计在今后四个月内，市场对其产品的需求量如下表所示。

|  |  |
| --- | --- |
| 时期（月） | 需要量（产品但闻） |
| 1 | 2 |
| 2 | 3 |
| 3 | 2 |
| 4 | 4 |

已知：对每个月来讲，生产一批产品的固定成本费为 3 (千元)，若不生产，则为零。每 生产单位产品的成本费为 1 （千元)。同时，在任何一个月内，生产能力所允许的最大生产 批量为不超过 6 个单位。

又知每单位产品的库存费用为每月 0.5 （千元），同时要求在第一个月开始之初， 及 在第四个月末，均无产品库存。

问：在满足上述条件下，该厂应如何安排各个时期的生产与库存，使所花的总成本费用 最低？

要求：写出各种变量、状态转移方程、递推关系式、和详细计算步骤。

**解：**

设K表示月度标识，则：

Pk：当月生产量

Nk：当月需求量

Lk：当月初剩余量

Vk：当月成本（生产成本+存储成本）

由题意得：

Lk满足：Lk = Lk-1 + Pk-1 - Nk-1，且L1=L5=0

Pk满足：max{0, Nk - Lk} <= Pk <= min{0, Nk + Nk+1 + ... + N4 - Lk}

Vk满足：Vk = 0.5\*(Lk - Nk) ,Pk=0

= 3 + Pk + 0.5\*(Lk + Pk - Nk) ,Pk!=0

设F(Lk)表示第k月初的剩余量为Lk时到第4月结束时的总成本，则其递推关系满足：

F(Lk) = min{Vk + F(Lk+1)}

= min{0.5\*(Lk - Nk) + F(Lk - Nk)} ,Pk=0

= min{3 + Pk + 0.5\*(Lk + Pk - Nk) + F(Lk + Pk - Nk)} ,Pk!=0

L5=0且F(L5)=0

计算过程如下:

首先计算K=4时，L4=0，1，2，3，4这五种情况的费用

|  |  |  |
| --- | --- | --- |
| Lk | Pk | Vk |
| 0 | 4 | 7 |
| 1 | 3 | 6 |
| 2 | 2 | 5 |
| 3 | 1 | 4 |
| 4 | 0 | 0 |

K=3时，由于前两个月最多一共生产12个产品单位，共消耗掉5个，最多剩余7个，又由于根据Pk所满足的条件得，Lk最多为6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lk | Pk | Vk | Lk+1 | Pk+1 | Vk+1 | V |
| 0 | 2 | 5 | 0 | 4 | 7 | 12 |
| 3 | 6.5 | 1 | 3 | 6 | 12.5 |
| 4 | 8 | 2 | 2 | 5 | 13 |
| 5 | 9.5 | 3 | 1 | 4 | 13.5 |
| 6 | 11 | 4 | 0 | 0 | **11** |
| 1 | 1 | 4 | 0 | 4 | 7 | 11 |
| 2 | 5.5 | 1 | 3 | 6 | 11.5 |
| 3 | 7 | 2 | 2 | 5 | 12 |
| 4 | 8.5 | 3 | 1 | 4 | 12.5 |
| 5 | 10 | 4 | 0 | 0 | **10** |
| 2 | 0 | 0 | 0 | 4 | 7 | **7** |
| 1 | 4.5 | 1 | 3 | 6 | 10.5 |
| 2 | 6 | 2 | 2 | 5 | 11 |
| 3 | 7.5 | 3 | 1 | 4 | 11.5 |
| 4 | 9 | 4 | 0 | 0 | 9 |
| 3 | 0 | 0.5 | 1 | 3 | 6 | **6.5** |
| 1 | 5 | 2 | 2 | 5 | 10 |
| 2 | 6.5 | 3 | 1 | 4 | 10.5 |
| 3 | 8 | 4 | 0 | 0 | 8 |
| 4 | 0 | 1 | 2 | 2 | 5 | **6** |
| 1 | 5.5 | 3 | 1 | 4 | 9.5 |
| 2 | 7 | 4 | 0 | 0 | 7 |
| 5 | 0 | 1.5 | 3 | 1 | 4 | **5.5** |
| 1 | 6 | 4 | 0 | 0 | 6 |
| 6 | 0 | 2 | 4 | 0 | 0 | **2** |

K=2时，本月初最多剩余4个产品单位（6-2）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lk | Pk | Vk | Lk+1 | min{Vk+1} | V |
| 0 | 3 | 6 | 0 | 11 | 17 |
| 4 | 7.5 | 1 | 10 | 17.5 |
| 5 | 9 | 2 | 7 | **16** |
| 6 | 10.5 | 3 | 6.5 | 17 |
| 1 | 2 | 5 | 0 | 11 | 16 |
| 3 | 6.5 | 1 | 10 | 16.5 |
| 4 | 8 | 2 | 7 | **15** |
| 5 | 9.5 | 3 | 6.5 | 16 |
| 6 | 11 | 4 | 6 | 17 |
| 2 | 1 | 4 | 0 | 11 | 15 |
| 2 | 5.5 | 1 | 10 | 15.5 |
| 3 | 7 | 2 | 7 | **14** |
| 4 | 8.5 | 3 | 6.5 | 15 |
| 5 | 10 | 4 | 6 | 16 |
| 6 | 11.5 | 5 | 5.5 | 17 |
| 3 | 0 | 0 | 0 | 11 | **11** |
| 1 | 4.5 | 1 | 10 | 14.5 |
| 2 | 6 | 2 | 7 | 13 |
| 3 | 7.5 | 3 | 6.5 | 14 |
| 4 | 9 | 4 | 6 | 15 |
| 5 | 10.5 | 5 | 5.5 | 16 |
| 6 | 12 | 6 | 2 | 14 |
| 4 | 0 | 0.5 | 1 | 10 | **10.5** |
| 1 | 5 | 2 | 7 | 12 |
| 2 | 6.5 | 3 | 6.5 | 13 |
| 3 | 8 | 4 | 6 | 14 |
| 4 | 9.5 | 5 | 5.5 | 15 |
| 5 | 11 | 6 | 2 | 13 |

K=1时，月初只有0个产品单位

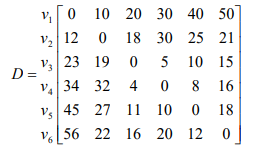
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lk | Pk | Vk | Lk+1 | min{Vk+1} | V |
| 0 | 2 | 5 | 0 | 16 | 21 |
| 3 | 6.5 | 1 | 15 | 21.5 |
| 4 | 8 | 2 | 14 | 22 |
| 5 | 9.5 | 3 | 11 | **20.5** |
| 6 | 11 | 4 | 10.5 | 21.5 |

因此，最终成本最低值为20.5（千元），且每月产量与月末剩余情况如下所示：

|  |  |  |
| --- | --- | --- |
| 月度 | 生产 | 月末剩余 |
| 1 | 5 | 3 |
| 2 | 0 | 0 |
| 3 | 6 | 4 |
| 4 | 0 | 0 |

**二、用动态规划方法编程求解下面的问题：**

某推销员要从城市 v1 出发，访问其它城市 v2，v3，…，v6 各一次且仅一次，最后返回 v1。D 为各城市间的距离矩阵。 问：该推销员应如何选择路线，才能使总的行程最短？

****

要求：写出递推关系式、伪代码和程序相关说明，并分析时间复杂性。（请遵守第一节课提 出的有关 assignment 的要求）

**解：**

思路大致为，若求从城市V1出发，经过且只经过1次其余城市节点，并最终返回V1的最短距离，则可以划分为从城市V1到Vi，然后再从Vi城市经过且只经过1次其余城市节点（初始城市-V1-Vi的集合），不断递归下去，直到剩余城市为空集，则直接返回V1，返回上层调用时，从Vi到Vj中选择结果最小的。

令（i，S）表示当前所在城市为Vi，需经过S集合中的城市各1次，并返回V1。在当前情况下，Vi城市可以选择去往S中任意一个城市Vj，距离为D[i][j]，此时节点位于Vj，因此状态变为（j，S-{Vj}），直到S为空集时，此时（t，S）表示直接从Vt返回V1。

令F(i,S)表示从当前所在城市为Vi出发，需经过S集合中的城市各1次，并返回V1时所经过的最短距离，可以得出**递推关系式**如下：

|  |
| --- |
| F(i,S) = min(D[i][j]+F(j,S-{Vj})) (j在S中遍历) ,S不为空集  = D[i][0] ,S为空集 |

**伪代码及程序说明：**

函数：Tsp(int matrix[][])

输入：邻接矩阵表示的有向图

过程：n表示节点个数，dp[][]用于存储从城市出发去往集合中剩余城市各一次并返回V1城市的最短距离，path[][]用于存储从当前城市前往集合中所有城市的下一个城市节点。这两个矩阵行数为n，列数为2n-1。

输出：从V1出发，经过剩余n-1个城市，且经过1次，并返回V1，返回最短距离，并输出所经过的路径。

|  |
| --- |
| for i:1 to n:  dp[i][0] = matrix[i][0]//当S为空集时，F(i,S)等于D[i][0]  endfor  for j:1 to 2n-1:  for i:1 to n:  dp[i][j] = -1//初始化F(i,S)为-1，默认表示当前情况下无法按要求返回V1  for t in S://循环遍历每一个在当前集合中的城市t  tmp = D[i][t]+dp[i][k]  if dp[i][j] > tmp:  dp[i][j] = min(D[i][t]+dp[i][k])//计算从i到t的距离，再加上从t到k集合中所有城市且一次并返回V1的距离，k为S集合除掉t城市，求最小值  path[i][j]=t//将取得最小值的那种情况的下一个城市t返回给path[i][j]  endif  endfor  endfor  endfor  for i:1 to n:  if D[0][i] + dp[i][S] < res:  res = D[0][i] + dp[i][S]//计算从V1前往下一个不同城市节点，所得到的最短距离  a = i//保存下一个应该前往的节点  endif  endfor  while S not empty:  print(a)//依次输出经过的结点，直到S为空集，便直接返回V1  a = path[a][S]  S = S.remove(a)  endwhile |

**时间复杂性：**

dp数组初始化第一列（操作n次）

dp数组递归求解过程（外层循环2n-1列，内层循环n行，遍历集合中下一个节点常数次操作n）

最外层计算最短路径（操作n次）

输出最短路径（遍历n次）

总体时间复杂度：n+2n-1\*n\*n+n+n，因此为O(2n\*n2)