学号：**ZY2206117** 姓名：**黄海浪** 作业：**Assignment\_2**

**题目1:**

1. **符号说明**

需求量：表示第月的产品需求量。

状态变量：表示第月的产品库存量。

决策变量：表示第月产品生产数量。

状态转移方程：

允许决策集合：

成本：表示第月的产生成本。

最优值函数：表示第月到第月采取最优策略产生的成本最小值。

1. **递推关系式**

其中：

且

1. **计算步骤**
2. 当时,由题，且，

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 0 | 4 | 0 | 0 | 0 | 7 | 7 |
| 1 | 3 | 0 | 0 | 0 | 6 | 6 |
| 2 | 2 | 0 | 0 | 0 | 5 | 5 |
| 3 | 1 | 0 | 0 | 0 | 4 | 4 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 |

1. 当时,由题,

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

因此，时的最优决策如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 0 | 6 | 4 | 0 | 0 | 11 | 11 |
| 1 | 5 | 4 | 0 | 0 | 10 | 10 |
| 2 | 0 | 0 | 4 | 7 | 0 | 7 |
| 3 | 0 | 1 | 3 | 6 | 0.5 | 6.5 |
| 4 | 0 | 2 | 2 | 5 | 1 | 6 |
| 5 | 0 | 3 | 1 | 4 | 1.5 | 5.5 |
| 6 | 0 | 4 | 0 | 0 | 2 | 2 |

1. 当时,由题,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 0 | 3 | 0 | 6 | 11 | 6 | 17 |
|  | 4 | 1 | 5 | 10 | 7.5 | 17.5 |
|  | **5** | **2** | **0** | **7** | **9** | **16** |
|  | 6 | 3 | 0 | 6.5 | 10.5 | 17 |
| 1 | 2 | 0 | 6 | 11 | 5 | 16 |
|  | 3 | 1 | 5 | 10 | 6.5 | 16.5 |
|  | **4** | **2** | **0** | **7** | **8** | **15** |
|  | 5 | 3 | 0 | 6.5 | 9.5 | 16 |
|  | 6 | 4 | 0 | 6 | 11 | 17 |
| 2 | 1 | 0 | 6 | 11 | 4 | 15 |
|  | 2 | 1 | 5 | 10 | 5.5 | 15.5 |
|  | **3** | **2** | **0** | **7** | **7** | **14** |
|  | 4 | 3 | 0 | 6.5 | 8.5 | 15 |
|  | 5 | 4 | 0 | 6 | 10 | 16 |
|  | 6 | 5 | 0 | 5.5 | 11.5 | 17 |
| 3 | **0** | **0** | **6** | **11** | **0** | **11** |
|  | 1 | 1 | 5 | 10 | 4.5 | 14.5 |
|  | 2 | 2 | 0 | 7 | 6 | 13 |
|  | 3 | 3 | 0 | 6.5 | 7.5 | 14 |
|  | 4 | 4 | 0 | 6 | 9 | 15 |
|  | 5 | 5 | 0 | 5.5 | 10.5 | 16 |
|  | 6 | 6 | 0 | 2 | 12 | 14 |
| 4 | **0** | **1** | **5** | **10** | **0.5** | **10.5** |
|  | 1 | 2 | 0 | 7 | 5 | 12 |
|  | 2 | 3 | 0 | 6.5 | 6.5 | 13 |
|  | 3 | 4 | 0 | 6 | 8 | 14 |
|  | 4 | 5 | 0 | 5.5 | 9.5 | 15 |
|  | 5 | 6 | 0 | 2 | 11 | 13 |
| 5 | **0** | **2** | **0** | **7** | **1** | **8** |
|  | 1 | 3 | 0 | 6.5 | 5.5 | 12 |
|  | 2 | 4 | 0 | 6 | 7 | 13 |
|  | 3 | 5 | 0 | 5.5 | 8.5 | 14 |
|  | 4 | 6 | 0 | 2 | 10 | 12 |
| 6 | **0** | **3** | **0** | **6.5** | **1.5** | **8** |
|  | 1 | 4 | 0 | 6 | 6 | 12 |
|  | 2 | 5 | 0 | 5.5 | 7.5 | 13 |
|  | 3 | 6 | 0 | 2 | 9 | 11 |

因此，时的最优决策如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 0 | 5 | 2 | 0 | 7 | 9 | 16 |
| 1 | 4 | 2 | 0 | 7 | 8 | 15 |
| 2 | 3 | 2 | 0 | 7 | 7 | 14 |
| 3 | 0 | 0 | 6 | 11 | 0 | 11 |
| 4 | 0 | 1 | 5 | 10 | 0.5 | 10.5 |
| 5 | 0 | 2 | 0 | 7 | 1 | 8 |
| 6 | 0 | 3 | 0 | 6.5 | 1.5 | 8 |

1. 当时,由题，,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 0 | 2 | 0 | 5 | 16 | 5 | 21 |
| 3 | 1 | 4 | 15 | 6.5 | 21.5 |
| 4 | 2 | 3 | 14 | 8 | 22 |
| **5** | **3** | **0** | **11** | **9.5** | **20.5** |
| 6 | 4 | 0 | 10.5 | 11 | 21.5 |

因此，决策可以达到最低成本，最低成本为20.5。即1月生产5单位，3月生产6单位。

**题目2:**

1. **符号说明**

阶段：表示当前已遍历过个节点，表示从出发，表示回到。

状态变量：表示第阶段的目前所在节点和剩余未遍历的节点。

决策变量：表示第阶段的目前所在节点和下一个遍历的点。

状态转移方程：

允许决策集合：

成本：表示第阶段产生的花费。

最优值函数：从节点经过节点次且仅次返回采取最优策略产生的最小成本。

1. **递推关系式**

且

1. **程序伪代码**

首先构建dp表表示从城市i出发经过城市j到达城市1的最小距离，dp表如下（假设有4个城市）：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 索引 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|  |  |  |  |  |  |  |  |
| 0 |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |

对于第个城市，它的二进制可以表示为，即第位为1。同样，判断某个索引是否包含某个城市可以用判断。

**算法为代码TSP：**

TSP(D,n)

//旅行商问题动态规划求解

//输入：邻接矩阵花费D和节点个数n

//输出：无，但会更新dp表

//init 即各个城市直接到城市1的距离

for 城市i in 所有的城市

dp[i][0] = D[i][0]

end

//dp

for j in 所有的城市组合

for i in 所有的城市

if i 不在j中

for k in j中城市

// 更新i经过j的距离，通过与i-k再经过j-{k}的距离比较

dp[i][j] = min(dp[i][j], d[k][j-{k}]+D[i][k])

end

end

end

end

**算法为代码****getPath：**

getPath()

// 根据dp表的记录获得路径

// 输入：无

// 输出：无（打印路径）

i = 0

while(j中城市不为空) do

打印 城市i

for k in j中所有城市

// 如果最优值通过k城市计算而来

if dp[i][j] == dp[i][j-{k}]+D[i][k]

i = k

j = j-{k}

break

end

end

end

打印 城市i

1. **程序相关说明、结果和分析（更多请参考代码文件：main.cpp）**

运行环境：

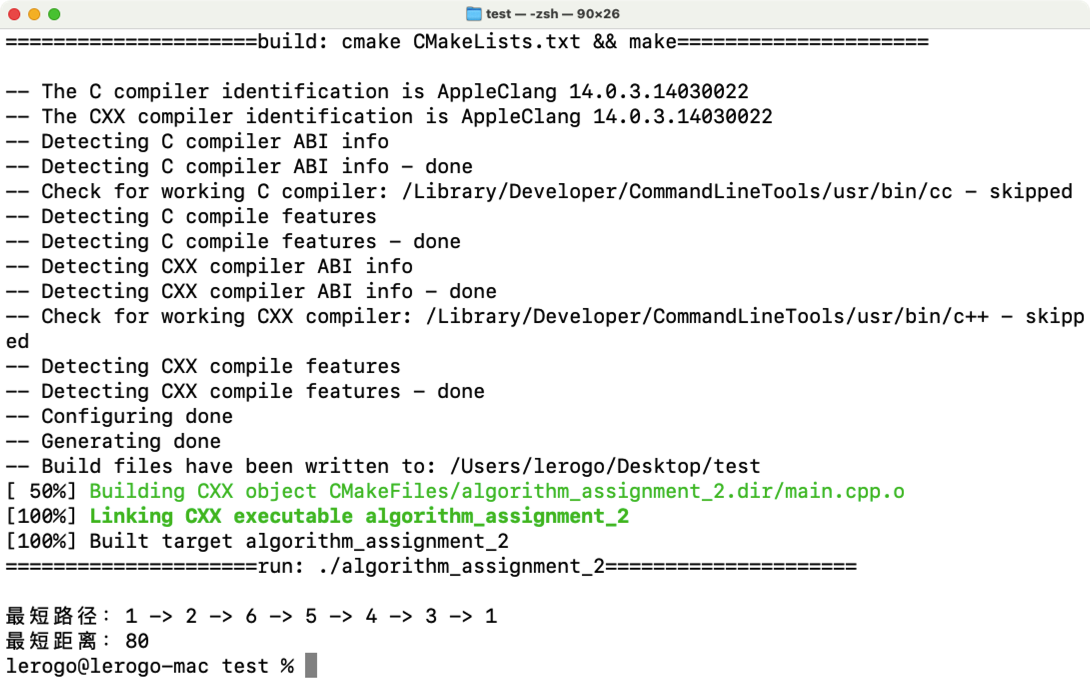
* 操作系统：macOS Ventura13.3.1
* 构建工具：cmake version 3.25.2；GNU Make 3.81
* 构建命令：cmake CMakeLists.txt && make
* 运行命令：./algorithm\_assignment\_2

运行结果：

最短路径：1 -> 2 -> 6 -> 5 -> 4 -> 3 -> 1

最短距离：80

结果截图：



1. **附录-代码**

/\*

\* encoding: utf-8

\* \*/

#include <iostream>

#include <limits.h>

// 6 cities

const int n = 6;

const int m = 1 << (n - 1);

// 6x6 cost matrix

const int D[n][n] = {

{0, 10, 20, 30, 40, 50},

{12, 0, 18, 30, 25, 21},

{23, 19, 0, 5, 10, 15},

{34, 32, 4, 0, 8, 6},

{45, 27, 11, 10, 0, 18},

{56, 22, 16, 20, 12, 0}

};

// 表示从i出发经过j到城市1的距离

int dp[n][m];

void TSP() {

// init 所有城市到城市1的距离

for (int i = 0; i < n; i++) {

dp[i][0] = D[i][0];

}

// dp

for (int j = 1; j < m; j++) {

for (int i = 0; i < n; i++) {

dp[i][j] = INT\_MAX;

// if i is not in j 判断i是否在j中

// j 是用二进制表示的，j的第i位为1表示i在j中

if ((j >> (i - 1)) & 1) {

continue;

}

// 遍历j中的每个城市k，找到最小的dp[i][j]

for (int k = 1; k < n; k++) {

if ((j >> (k - 1)) & 1) {

// j - {k} 表示j中除了k之外的城市

// 由于j中包含k，所以j - {k}中的城市数比j中的城市数少1

// 所以dp[k][j - {k}]已经在上一次循环中计算过了

dp[i][j] = std::min(dp[i][j], dp[k][j - (1 << (k - 1))] + D[i][k]);

}

}

}

}

}

void getPath() {

// j = m - 1 表示所有城市都已经遍历过了

// i = 0 表示从城市1开始

int j = m - 1;

int i = 0;

while (j > 0) {

// 从城市1开始，依次输出城市的编号

std::cout << i + 1 << " -> ";

for (int k = 1; k < n; k++) {

if ((j >> (k - 1)) & 1) {

// 如果dp[i][j] == dp[k][j - {k}] + D[i][k]

// 说明从i出发经过j到城市1的距离等于从k出发经过j - {k}到城市1的距离加上i到k的距离

// 所以i到k是最短路径的一部分

// 所以下一次循环应该从k开始

if (dp[i][j] == dp[k][j - (1 << (k - 1))] + D[i][k]) {

// j -= (1 << (k - 1)) 表示j中去掉k

j -= (1 << (k - 1));

i = k;

break;

}

}

}

}

std::cout << i + 1 << " -> 1" << std::endl;

}

int main() {

TSP();

std::cout << "最短路径：";

getPath();

std::cout << "最短距离：";

std::cout << dp[0][m - 1] << std::endl;

return 0;

}