习题6

202328015926048-丁力

2.工作分配问题。设有 n 件工作需要分配给 n 个人去完成。将工作 i 分配给第 j 个人完成所需要的费用为 c_{ij} 。试设计一个算法, 为每一个人分配一件不同的工作, 并使总费用达到最小。

针对这个工作分配问题，我们可以使用匈牙利算法来设计解决方案。以下是匈牙利算法在这种情况下的具体步骤：

1. **建立成本矩阵**：
   - 创建一个 $n \times n$ 的成本矩阵 C，其中 $C[i][j] = c_{ij}$ 表示将工作 i 分配给第 j 个人的费用。
2. **行减法**：
   - 对于矩阵中的每一行，找到该行的最小值，并从该行的每个元素中减去这个最小值。
3. **列减法**：
   - 对于矩阵中的每一列，找到该列的最小值，并从该列的每个元素中减去这个最小值。
4. **覆盖所有零**：
   - 使用最少数量的水平或垂直线覆盖矩阵中的所有零元素。
5. **调整矩阵**：
   - 如果覆盖所有零的线的数量等于矩阵的阶数（即 n），则转到步骤 6，否则继续。
   - 找到未被覆盖的所有元素中的最小值。
   - 从所有未覆盖的行中减去这个最小值，并加到所有覆盖的列中。
   - 重复步骤 4 和 5。
6. **找到最优分配**：
   - 在调整后的矩阵中，找到零元素作为工作分配。
   - 确保每行和每列只选择一个零（即每个人只被分配一项工作，每项工作只分配给一个人）。
7. **计算最小总费用**：
   - 根据最终的分配结果，回到原始成本矩阵 C，计算总费用。

3. 最佳调度问题。假设有 n 个任务要由 k 个可并行工作的机器来完成。完成任务 i 需要的时间为 t_i 。试设计一个算法找出完成这 n 个任务的最佳调度，使得完成全部任务的结束时间最早。

> 算法步骤：
>
> 1. **初始化状态**： 将全局最小完成时间 `cmg` 设为正无穷，表示目前还没有找到可行的调度解。
> 2. **状态空间树**： 创建一个状态空间树，树的节点表示将任务分配给机器的决策。树的根节点表示一个初始的、无任何任务分配的状态。树的每一层对应一个任务的分配，第 (i) 层代表第 (i) 个任务的分配情况。
> 3. **搜索策略**： 用深度优先搜索策略遍历状态空间树。从根节点开始，递归地为每个任务选择一个机器分配。每次选择分配给工作时间最少的机器可以作为一个简单的启发式策略。
> 4. **剪枝条件**： 在每个节点，计算在当前分配下的完成时间 (cm)。如果 (cm) 超过已知的全局最小完成时间 `cmg`，则停止在当前分支继续搜索，因为这条路径不能产生更好的解。
> 5. **更新全局最优解**： 一旦所有任务都被分配（即在叶子节点），计算当前分配的完成时间。如果这个完成时间小于全局最小完成时间 `cmg`，则更新 `cmg` 并记录当前分配作为新的最优解。
> 6. **回溯**： 如果当前分配不能产生更好的解，则返回上一层（即回溯），撤销最后的任务分配，尝试其他可能的分配。
> 7. **重复步骤**： 重复步骤 4 至 6，直到所有可能的节点都被探索完成或者被剪枝。
> 8. **提取结果**： 达到所有叶子节点后，我们会有一个或多个解。最优解是完成时间最短的解，即 `cmg`。

5.最小重量机器设计问题。设某一机器由 n 个部件组成，每一种部件都可以从 m 个不同的供应商处购得。设 w_{ij} 是从供应商 j 处购得的部件 i 的重量，c_{ij} 是相应的价格。试设计一个算法，给出总价格不超过 c 的最小重量机器设计。

> 首先我们定义状态 `dp[i][j]` 表示考虑前 `i` 个部件，花费不超过 `j` 的情况下，能达到的最小重量。我们需要填满这个状态矩阵来找到最终的解。动态规划的递推方程如下：
>
> ```
> dp[i][j] = min(dp[i - 1][j], dp[i - 1][j - c[i][k]] + w[i][k]) for all k such that 0 <= k < m
> ```
>
> 其中 `c[i][k]` 是第 `i` 个部件从供应商 `k` 购得的成本，`w[i][k]` 是相应的重量。
>
> 算法框架如下：
>
> 1. 初始化动态规划表 `dp`，大小为 `(n+1) x (c+1)`，所有值设为无穷大，表示初始状态下，对于任意非零成本，不可能构建机器。

2. 将 `dp[0][0]` 设为 0，表示不购买任何部件，并且没有成本时的重量为 0。

3. 使用两层循环，外循环遍历每个部件，内循环遍历每个可能的成本值 `j`。

4. 在内循环中，遍历所有供应商 `k`，更新状态 `dp[i][j]`。

5. 最终的解为 `dp[n][c]`。