算法设计与分析 Week 10

第十周作业-Solution

Lecturer: 杨启哲 Last modified: 2024 年 11 月 13 日

1. 对于任意给定的正整数 M, 请给出三个矩阵相乘的例子，它们的一种乘法顺序所需要的数量乘法的次数至少是另一种顺序的 M 倍。

   解答. 考察如下三个矩阵:

   $$A_{1\times M},\ B_{M\times 1},\ C_{1\times M}$$

   则我们有:

   - $A(BC)$ 的乘法次数为 $1\times M\times M+M\times 1\times M=2M^2$
   - $(AB)C$ 的乘法次数为 $1\times M\times 1+1\times 1\times M=2M$

   两种顺序刚好差 M 倍的乘法计算次数。 □

2. 回顾贪心算法中我们讲到的货币兑换问题。有一个货币系统，它有 n 种硬币，面值分别为 $v_1=1, v_2, \ldots, v_n$。现在需要兑换价值为 y 的钱，使得硬币的数量最少。之前提到过这个问题贪心算法不一定能给出最优解，现在请设计一个高效的算法解决这个问题。

   解答. 我们利用动态规划的思想来解决这个问题。记 $C[i][y]$ 为用前 i 种硬币兑换价值为 y 的钱所需要的最少硬币数，则我们有如下递推关系:

   $$C[i][y]=\min\{C[i][y-v_i]+1, C[i-1][y]\}$$

   注意到硬币是可以重复使用的，所以我们并不需要考虑 $C[i][y-v_i]$ 是否已经用过第 i 枚硬币。具体算法如下:

   兑换钱币

   输入: 面值列表 $v_1, \ldots, v_n$，兑换的钱数 y
   输出: 最少的硬币数

   ```
   for i = 1 to n do
       C[i][0] ← 0
   end for
   for j = 1 to y do
       C[0][j] ← ∞
   end for
   for i = 1 to n do
       for j = 1 to y do
   ```

```
        if j < v_i then
            C[i][j] ← C[i − 1][j]
        else
            C[i][j] ← min{C[i][j − v_i] + 1, C[i − 1][j]}
        end if
    end for
end for
return C[n][y]
```

算法需要一个 $n \times y$ 的表格来存储中间结果并对其进行遍历，所以时间复杂度何空间复杂度均为 $O(ny)$。 □

Remark 0.1

这实际上就是一个多重背包问题，事实上我们也可以对其做空间的优化。具体来说，我们可以只用一个长度为 y 的数组来存储中间结果，具体算法如下：

兑换钱币

输入: 面值列表 $v_1, \ldots, v_n$，兑换的钱数 y
输出: 最少的硬币数

```
for j = 1 to y do
    C[j] ← ∞
end for
C[0] ← 0
for i = 1 to n do
    for j = v_i to y do
        C[j] ← min{C[j − v_i] + 1, C[j]}
    end for
end for
return C[y]
```

3. 请对旅行商问题设计一个动态规划算法，这里旅行商问题指的是：给定 n 个城市和一个 $n \times n$ 的距离矩阵 D，其中 $D[i, j]$ 表示从城市 i 到城市 j 的距离，旅行商要从某个城市出发，经过每个城市恰好一次，最后回到出发的城市，问旅行商的最短路线是什么？请给出算法的时间复杂度。

解答. 我们依旧用动态规划的思想来解决这个问题。需要注意的是，为了使经过的城市不重复，我们还需要记忆所有去过的城市。假定 1 是出发城市，我们令 $C[i][S]$ 为从城市 1 出发，经过 S 中的城市恰好一次，最后回到城市 i 的最短路线长度。显然 S 是一个包含 1 的子集。我们有如下递

推关系：

$$C[i][S] = \begin{cases} 0 & \text{if } S = \{1\} \wedge i = 1 \\ \infty \text{if } |S| > 1 \wedge i = 1 & \\ \min_{j \in S, j \neq i}\{C[j][S \setminus \{i\}] + D[j][i]\} & \text{otherwise} \end{cases}$$

其中 S 是一个集合，表示已经去过的城市。根据这个递推关系，我们可以给出如下的算法：

旅行商问题

输入： 距离矩阵 D，城市数 n
输出： 最短路线长度

```
C[1][{1}] ← 0
for s = 2 to n do
    for 所有包含 1 大小为 s 的子集 S ⊆ {1, 2, ..., n} do
        C[1][S] = ∞
        for 所有 j ∈ S, j ≠ 1 do
            C[j][S] ← min_{i∈S, i≠j}{C[i][S \ {j}] + D[i][j]}
        end for
    end for
end for
return min_i{C[i][{1, 2, ..., n}] + D[i][1]}
```

注意到子集一共有 2^{n-1} 个，而每次更新 $C[j][S]$ 需要 $O(n^2)$ 的操作，因此整个算法的时间复杂度为 $O(n^2 2^n)$。 □

4. 为了降低背包问题运行的时间界限 $O(nW)$，我们尝试用这样的思路，我们使用一个大数 K 将容量 W 和每件物品的体积 w_i 都缩小 K 倍。具体来说，令 $W \leftarrow \lceil \frac{W}{K} \rceil$，$w_i \leftarrow \lceil \frac{w_i}{K} \rceil$；再在这个背包问题的新实例上运行我们提出的解决背包问题的算法。

   (1) 现在算法的时间复杂度是多少？

   (2) 这样子的策略其实并不能总得到原实例的最优解，请给出一个反例。

   解答.

   - 回顾一下背包问题的算法：

     背包问题

     输入： n 个物品，重量分别为 $w_1, \ldots, w_n$，价值分别为 $v_1, \ldots, v_n$，背包容量为 W
     输出： 最大价值

     ```
     for j = 0 to W do
         V[j] ← 0
     end for
     for i = 1 to n do
     ```

```
    for j = W to w_i do
        if j ⩾ w_i then
            V[j] ← max{V[j], V[j − w_i] + v_i}
        end if
    end for
end for
return V[W]
```

如果将 $W \leftarrow \lceil \frac{W}{K} \rceil$，$w_i \leftarrow \lceil \frac{w_i}{K} \rceil$，不难发现算法中的循环总共执行了 $\lceil nW/K \rceil$ 次，从而算法的时间复杂度为 $O(nW/K)$。

- 考虑如下的反例：
  - $W = 10$，物品重量为 $w_1 = 9, w_2 = 1$，价值分别为 $v_1 = 1, v_2 = 10$。

  如果 $K = 2$，则我们有：
  - $W = 5$，物品重量为 $w_1 = 5, w_2 = 1$，价值分别为 $v_1 = 1, v_2 = 10$。

  在原始问题中，最优解是两个物品全取，总价值为 11。但在更改后的问题中，最优解是只取第二个物品，总价值为 10。

□

> Remark 0.2
>
> 需要注意的是,K 并不是一个常数，所以计算复杂性的时候不能像省略常数一样省略。

5. 请对如下问题给出一个 $O(nt)$ 时间的算法。

   给定 n 个整数 $a_1, \ldots, a_n$ 和正整数 t，问是否存在一个子集 $S \subseteq \{1, \ldots, n\}$，使得 $\sum_{i \in S} a_i = t$。

   解答. 我们可以考虑如下的子问题：

   令 $M[i][j]$ 表示前 i 个数中是否存在一个子集，使得其和为 j。若有则 $M[i][j] = 1$，否则 $M[i][j] = 0$。

   显然我们有如下的递推关系：

$$M[i][j] = \begin{cases} 1 & \text{if } i = 0 \text{ and } j = 0 \\ 0 & \text{if } i = 0 \text{ and } j \neq 0 \\ M[i-1][j] \vee M[i-1][j-a_i] & \text{o.w.} \end{cases}$$

   根据这个递推关系实现相应算法即可。注意到我们只需要遍历大小为 $n \times t$ 的二维数组一轮，所以时间复杂度为 $O(nt)$。 □