数据结构与算法动态规划作业

陈宇琪

2020 年 8 月 1 日

摘要

主要内容：动态规划。
说明：动态规划作业将分为两个部分，在之后的三周的时间内完成。

- 第一部分为一些简单编程题目，这些题目中有两题较难，根据自己能力完成。
- 第二部分为理论和编程作业，主要涉及背包问题和算法描述。

希望大家多多编程，准备下周的第一次编程考试！
ddl：2020-05-05

目录

1 作业 Part1 **2**
1.1 编程作业 2

2 作业 Part2 **2**
2.1 简答题 2
2.2 编程作业 2

3 参考答案 **3**
3.1 简答题 3
3.2 编程作业 3
3.3 插入代码 3

1 作业 Part1

1.1 编程作业

图 1: Questions for Part 1

作业要求：请在这 10 道题目中选择 5 道题目完成！

2 作业 Part2

2.1 简答题

1、描述 01 背包问题，完全背包问题、多重背包问题的问题描述和算法分析，重点写出状态转移式子和复杂度分析。

2、对于一个简单的动态规划问题，如果一个状态可以延展出 n 个状态，并且一共有 m 个可能的状态，那么整个算法复杂度是多少？

3、（分数背包问题）有 n 个物品，第 i 个物品的重量与价值分别为 $w[i]$ 与 $v[i]$。背包容量为 V，如何让背包装入的物品具有更大的价值总和（物品可以取一部分）。

（1）给出问题的贪心解法。

（2）给出问题的动态规划解法（设计状态转移方程，类似背包问题）。

（3）分析两种做法的复杂度。

（4）证明动态规划算法的正确性（从贪心最优结构入手）。

4、分析动态规划问题和贪心问题的异同点。（至少从最优子结构这个角度分析）

2.2 编程作业

1、（完整代码）使用贪心算法/动态规划算法实现分数背包问题。

2、完成 EOJ 上题目：

图 2: Questions for Part 2

作业要求：请至少完成第一题！

3 参考答案

3.1 简答题

2、$O(n \times m)$

3、（分数背包问题）有 n 个物品，第 i 个物品的重量与价值分别为 $w[i]$ 与 $v[i]$。背包容量为 V，如何让背包装入的物品具有更大的价值总和（物品可以取一部分）。

（1）给出问题的贪心解法。

按照 $\frac{v[i]}{w[i]}$ 排序之后，优先选择单位体积价值高的物体，如果取走全部将超出总体积，则选择部分后结束程序。

（2）给出问题的动态规划解法（设计状态转移方程，类似背包问题）。

将每个物体分成 $w[i]$ 个物体，每个物体的价值为 $\frac{v[i]}{w[i]}$，这样就是一个多重背包问题。

（3）分析两种做法的复杂度。

贪心的复杂度是：$O(n \times \log(n))$

动态规划的复杂度是：$O(V \times \sum_{i=1}^{n} w[i])$。

如果使用二进制优化多种背包，复杂度是：$O(V \times \sum_{i=1}^{n} \log w[i])$。

（4）证明动态规划算法的正确性（从贪心最优结构入手）。

注意到贪心算法给出的一个可能的最优解中，每个物品取走的体积必定是整数，因为只有最后一件物品可能取走部分体积，前面的物品都是取走完整的体积。又因为所有的物品的体积都是整数，背包的体积也是整数，所以存在一个最优解使得每个物品取走的体积都是整数，所以动态规划算法成立。

4、分析动态规划问题和贪心问题的异同点。（至少从最优子结构这个角度分析）

动态规划不需要排序，而贪心算法一般需要排序之后按照一定策略寻找最优解。

3.2 编程作业

1、（完整代码）使用贪心算法/动态规划算法实现分数背包问题。

用贪心写的需要特别注意排序的时候，大家都是这样写的：

Listing 1: **wrong answer**

```
struct item
{
        int w,v;
        bool operator < (const item &rhs) const
        {
                return 1.0*v/w>1.0*rhs.v/rhs.w;
        }
};
```

但是这样会带来浮点数误差，更好的答案如下：

Listing 2: **correct answer**

```
struct item
{
        int w,v;
        bool operator < (const item &rhs) const
        {
                return 1ll*v*rhs.w>1ll*rhs.v*w;
        }
};
```