数据结构与算法

作业格式模板

张露平　20xx00xx

November 13, 2025

问题一　指数计算的递归函数实现

请实现一个递归函数 `power(base, exponent)`，计算 $base^{exponent}$ 的值。要求支持负指数的情况。例如，`power(2, 3)` 应返回 8，而 `power(2, -3)` 应返回 0.125。

解：

求解思路：

1. 定义基本情况和递归情况：当指数为 0 时，任何数的 0 次方均为 1。
2. 递归情况：
    - 如果指数为正数，则递归调用 `power(base, exponent - 1)` 并将结果乘以 `base`。
    - 如果指数为负数，则递归调用 `power(base, exponent + 1)` 并将结果除以 `base`。

Listing 1 为该递归函数的实现。

问题二　基本概念理解

在算法分析中，我们为什么通常更关注“最差情况时间复杂度”(Worst-Case Complexity) 而不是“最优情况时间复杂度” (Best-Case Complexity)？请结合课程中对插入排序的分析，阐述你的理由。

```python
def power(base, exponent):
    # 基本情况
    if exponent == 0:
        return 1.0
    # 递归情况
    elif exponent > 0:
        return base * power(base, exponent - 1)
    else:
        return power(base, exponent + 1) / base # exponent < 0
```

Listing 1: 递归实现指数计算的power 函数示例

解:

我们更关注最差情况时间复杂度，主要原因如下:

- 提供性能保证：最差情况复杂度为算法的运行时间提供了一个上界。这意味着在任何输入情况下，算法的运行时间都不会超过这个界限，这对于评估系统的稳定性和可靠性至关重要。
- 实际应用中的不可预测性：在许多应用中，我们无法预知或控制输入数据的具体状态。例如，插入排序在处理一个几乎有序的列表时非常快（最优情况），但在处理一个完全逆序的列表时则非常慢（最差情况）。依赖最优情况进行设计是有风险的。
- 平均情况分析的复杂性：平均情况的分析通常比最差情况复杂得多，它需要对所有可能的输入进行概率假设和统计分析。而最差情况分析则更为直接和简洁。

以插入排序为例:

- 最优情况：当输入数组已经是有序的时，插入排序的时间复杂度为 $O(n)$，因为每个元素只需与前一个元素比较一次。
- 最差情况：当输入数组是逆序排列时，插入排序的时间复杂度为 $O(n^2)$，因为每个元素都需要与前面所有元素进行比较和移动。