第十四章 数据结构与算法

14.1 概述

数据结构是计算机存储、组织数据的方式。算法是解决问题的步骤。

14.1.1 时间复杂度

时间复杂度用大O表示。

14.2 线性表

14.2.1 顺序表

顺序表的插入操作：

```
def insert(self, pos, item):
    if pos < 0 or pos > self.length:
        raise IndexError
    for i in range(self.length, pos, -1):
        self.data[i] = self.data[i-1]
    self.data[pos] = item
    self.length += 1
```

14.2.2 链表

链表节点定义：

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
```

14.3 栈与队列

栈是后进先出（LIFO）的结构。

操作	说明
push	入栈
pop	出栈
peek	查看栈顶

14.4 树

14.4.1 二叉树

二叉树的遍历：

```
def inorder(root):
    if root:
        inorder(root.left)
        print(root.data)
        inorder(root.right)
```

14.4.2 堆

堆是一种特殊的完全二叉树。

14.5 图

图的表示方法：

- 邻接矩阵
- 邻接表

14.6 排序

14.6.1 冒泡排序

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

14.6.2 快速排序

快速排序的平均时间复杂度为O(n log n)。

14.7 查找

14.7.1 二分查找

```
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1
```

14.7.2 哈希表

哈希表通过哈希函数将键映射到数组下标。

14.8 小结

本章介绍了线性表、栈、队列、树、图、排序和查找。