数据结构

MLE 算法指北

2025 年 2 月 3 日

1 线性数据结构

线性数据结构的特点是数据元素按照顺序排列，每个元素最多只有一个前驱和一个后继。

1.1 数组（Array）

定义：数组是一种固定大小的线性数据结构，元素存储在连续的内存空间中。

常见操作：

- 访问：$O(1)$
- 插入/删除（固定位置）：$O(n)$
- 搜索：$O(n)$（无序）/ $O(\log n)$（有序）

1.2 链表（Linked List）

定义：链表由一系列节点组成，每个节点包含数据和指向下一个节点的指针。

- 单链表（Singly Linked List）
- 双向链表（Doubly Linked List）
- 循环链表（Circular Linked List）

常见操作：

- 访问：$O(n)$
- 插入/删除：$O(1)$（在头部或尾部）
- 搜索：$O(n)$

1.3 栈（Stack）

定义：栈是一种后进先出（LIFO, Last In First Out）的数据结构。

基本操作：

- 压栈（push）：$O(1)$
- 弹栈（pop）：$O(1)$
- 访问栈顶元素（top/peek）：$O(1)$

1.4 队列（Queue）

定义：队列是一种先进先出（FIFO, First In First Out）的数据结构。

类型：

- 普通队列
- 双端队列（Deque）

常见操作：

- 入队（enqueue）：$O(1)$
- 出队（dequeue）：$O(1)$
- 访问队头元素：$O(1)$

2 非线性数据结构

定义：树是一种分层结构的数据结构，由节点（Node）组成，每个节点可以有多个子节点。

2.1 二叉树（Binary Tree）

2.2 树（Tree）

定义：树是一种分层结构的数据结构，由节点（Node）组成，每个节点可以有多个子节点。**类型**：

- 二叉树（Binary Tree）
- 二叉搜索树（Binary Search Tree, BST）
- 平衡二叉树（AVL 树、红黑树）
- 堆（Heap）

常见操作：

- 遍历（前序、中序、后序、层序）：$O(n)$
- 插入/删除：$O(\log n)$（平衡树）
- 搜索：$O(\log n)$（二叉搜索树）

2.3 图（Graph）

定义：图由顶点（Vertex）和边（Edge）组成，可以是有向图或无向图。**表示方法**：

- 邻接矩阵（Adjacency Matrix）：$O(V^2)$
- 邻接表（Adjacency List）：$O(V+E)$

常见算法：

- 深度优先搜索（DFS）：$O(V+E)$
- 广度优先搜索（BFS）：$O(V+E)$
- Dijkstra 最短路径算法：$O((V+E)\log V)$
- Kruskal / Prim 最小生成树：$O(E\log E)$

2.4 堆（Heap）

定义：堆是一种特殊的树结构，常用于优先队列。**类型**：

- 最小堆（Min-Heap）：父节点的值小于等于子节点
- 最大堆（Max-Heap）：父节点的值大于等于子节点

常见操作：

- 插入：$O(\log n)$
- 删除最小/最大值：$O(\log n)$
- 获取最小/最大值：$O(1)$

2.5 哈希表（Hash Table）

定义：哈希表是一种通过哈希函数将键映射到数组索引的数据结构。**常见哈希冲突解决方法**：

- 拉链法（Chaining）
- 开放寻址法（Open Addressing）

常见操作：

- 插入：$O(1)$（理想情况）
- 搜索：$O(1)$（理想情况）
- 删除：$O(1)$（理想情况）

3 总结

不同数据结构适用于不同类型的问题，选择合适的数据结构可以极大提升算法的效率。以下是常见数据结构的时间复杂度对比表：

数据结构	访问	搜索	插入/删除
数组	$O(1)$	$O(n)$	$O(n)$
链表	$O(n)$	$O(n)$	$O(1)$
栈	$O(n)$	$O(n)$	$O(1)$
队列	$O(n)$	$O(n)$	$O(1)$
二叉搜索树	$O(\log n)$	$O(\log n)$	$O(\log n)$
哈希表	$O(1)$	$O(1)$	$O(1)$