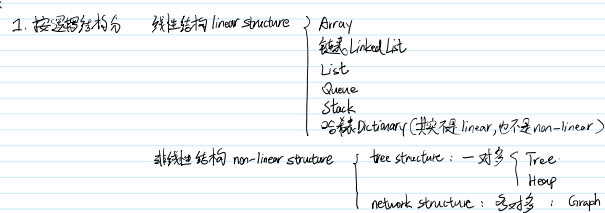


Data structure 续



2. 按存储结构分: 连续空间存储结构 contiguous storage structure: 基于 Array 实现的。也叫静态 data structure。
非连续 non-contiguous: 基于 Linked List 实现的。也叫动态 data structure。

3. CH4

Array	a. 相关操作	complexity
	初始化	$O(1)$ ✗
	随机访问	$O(1)$ ✓
	插入	$O(n)$ ✓
	删除	$O(n)$ ✓
	遍历 Traverse	$O(n)$ ✓ 来自于 ChalkPT
	查找 Find	$O(n)$ ✓
	扩容 Extend	$O(n)$ ✓

b. 典型应用: 5个

Linked List 教程上相关 Def: 6个, 有总结

a. 相关操作

		complexity
$O(1)$	初始化	$O(1)$ ✓ 初始化空链表 $O(1)$ 。
$O(1)$	Insert	$O(1)$ ✓
$O(1)$	Remove	$O(1)$ ✓
$O(n)$	访问 Access	$O(n)$ ✓
$O(n)$	查找 Find	$O(n)$ ✓

b. 链表典型应用, 共8个

List List 相关 Def: 2个, 有总结

List 的实现:

- 由 Array
- 由 Linked List

a. 相关操作

		complexity
	初始化	$O(1)/O(n)$ ✓
	Access	$O(1)$ ✓
	Insert	$O(1)$ ✗
	Remove	$O(1)$ ✗
	Traverse	$O(n)$ ✓
	拼接链表	$O(n)$ ✗
	排序链表	$O(n)$ ✗ 默认实现是快速排序。

> 和在低 index 操作相关。

b. List Implementation

讲了由 Array 做底层的实现。相关 operation 的几个方法的调用还等着。

CH5

stack a. stack 相关 Def: 6个, 有总结

b. 相关操作

		complexity
	初始化	$O(1)/O(n)$ ✓
	Push	$O(1)$ ✓
	Pop	$O(1)$ ✓
	Peek	$O(1)$ ✓

c. Stack Implementation

- 基于 Linked List 的实现: → 复习写代码。注意头结点和栈顶的关系。
- 基于 Array 的实现: → 复习写代码。注意 Array 尾部和栈顶的关系。
- 两种不同实现下的各操作的 complexity 对比和思考。

d. stack 的应用: 3个。

Queue

A. Queue 相关 Def: 5个。

B. 相关操作

complexity

初始化	$O(1)/O(n)$	✓
Push	$O(1)$	✓
Pop	$O(1)$	✓
Peak	$O(1)$	✓

C. Queue Implementation

i. 基于 Linked List 实现: → 复习写代码。注意头节点和队首, 尾节点的队尾的关系。

ii. 基于 Array 实现: → 复习写代码。注意 front, rear, 取余操作。

iii. 两种不同实现下的各操作 complexity 和思考。

D. Queue 应用: 2个。

CH6 哈希表

A. 哈希表相关 Def: 14个, 有总结

B. 相关操作

Complexity

初始化	$O(1)/O(n)$	✓
查询	$O(1)$	✓
添加	$O(1)$	✓
删除	$O(1)$	✓
遍历	$O(n)$	✓

C. 哈希表实现相关

i. 仅用一个数组做哈希表的简单实现。→ 复习写代码

ii. 哈希表的改进方法: 2个

1) 链式地址 → 复习写代码。使用了列表来代替 Linked List, 以简化代码。

2) 开放寻址。

a) 线性探测 (包括删除) 实现 → 复习写代码。对于其中删除法, e.g. FindBucket(), 可以尝试写写 flow chart 帮助

b) 和 c) 教程上没写代码了。

CH7 Tree

a. Tree 相关 Def: 共 10 个, 有总结

b. Tree 相关操作:

Complexity

初始化	$O(n)/O(n^2)$	✗
插入	$O(1)$	✗
删除	$O(1)$	✗

Complexity

Level Order
PreOrder
InOrder
PostOrder

Complexity

Binary Search Tree 查找

C. Tree 的相关实现

这两点都相当于用链表实现 Tree。

1) 层序遍历实现 → 复习写代码。这属于 BFS, 借助 Queue 实现。

2) DFS 遍历实现 → 共三种。复习写代码。借助递归实现。

3) 用 Array 表示完美二叉树 → 无代码。复习映射公式。

4) 用 Array 表示任意二叉树 → 需要层序遍历中显式写出所有空节点。复习写代码。

理解。

Find bucket (int key) 中间如何判断 TOMBSTONE 没有被碰到, 和需要更新 bucket Tombstone 值的代码, 及有流程图。

寻找 index : while (bucket[index] != null)

找到时

找不到时

碰到 TOMBSTONE

没碰到 TS,

碰到 TS,

碰到 TS, b/c

return TS, b/c

这是一个桶

这是一个桶

直接位置,

return index

TOMBSTONE 的位置。

return index

return TS, b/c

这是一个桶

这是一个桶

这是一个桶