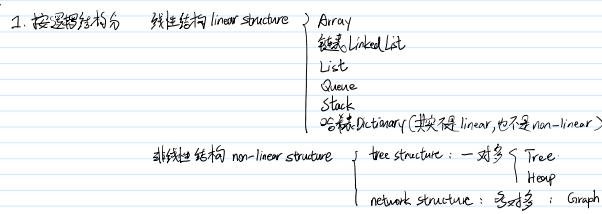


data structure 分类



2. 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 Chapt
	查找 Find	$O(n)$ ✓
	扩容 Extend	$O(1)$ ✓

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	$O(1)$ ✗ 和在位 index 操作相关。
Traverse	$O(n)$ ✓
排序列表	$O(n)$ ✗
排序列表	$O(n)$ ✗ 默认实现是快速排序。

b. List Implementation 相关 operation 的

讲了由 Array 做底层的实现。几个方法的调用还得看。

Ch5

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	$O(1)$ ✓

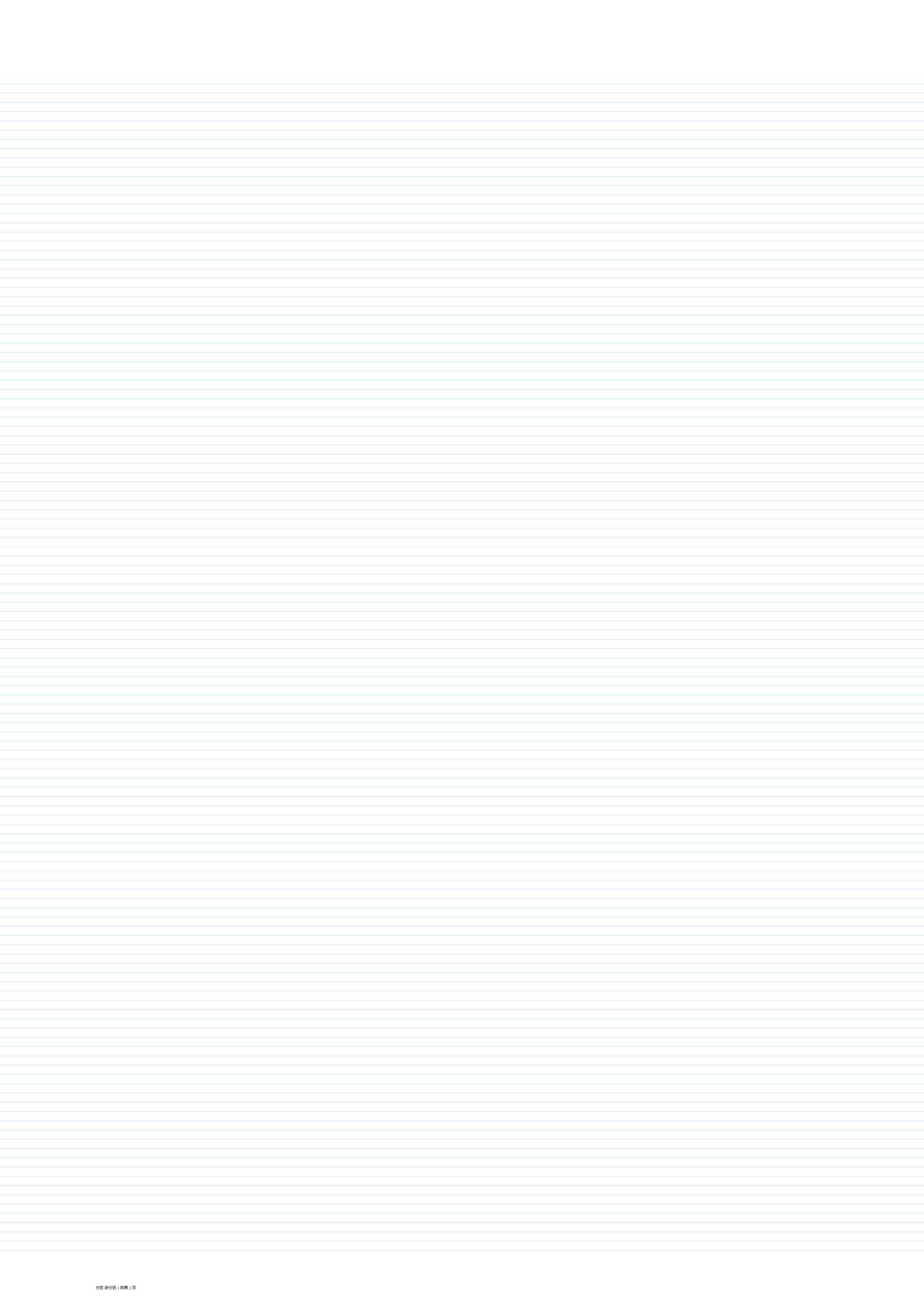
C. Stack Implementation

i. 基于 Linked List 的实现：→ 复习写代码。注意头结点和根顶的关系。

ii. 基于 Array 的实现：→ 复习写代码。注意 Array 尾部和根顶的关系。

iii. 两种不同实现下的各种操作的 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	$O(1)$	✓
Peek	$O(1)$	✓

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, 队列操作。

iii. 两种不同实现下的各操作 complexity 和思考。

D. Queue 应用: 2个。

CH 6 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. 哈希表的改良方法: 2个

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

2) 开放寻址:

- 线性探测 (包含擦除) 实现, → 复习写代码。对于其其余方法, e.g. FindBucket(), 可以尝试写写。从 chart 裁剪 b) 和 c) 教程上没有代码。

CH 7 Tree d. Tree 相关 Def: 共 18 个, 有总结

b. Tree 相关操作:

	Complexity
初始化	$O(1) / O(n)$
插入	$O(1)$
删除	$O(1)$

BFS 和 DFS 在用 Linked List 和 Array 实现下的复杂度相同。

Complexity

LevelOrder	BinarySearchTree 查找
PreOrder	BinarySearchTree 插入
InOrder	BinarySearchTree 擦除
PostOrder	BinarySearchTree 中间值查找

C. Tree 的相关实现

1) 层遍历实现 → 复习写代码。这属于 BFS, 借助 Queue 实现。这有点相当于前链表实现 Tree。

2) DFS 遍历实现 → 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 偏左二叉树 → 需要层序遍历中显式写出空节点, 这样层序遍历就可以表示偏左二叉树了。只有一行代码。

5) 用 Array 表示 完全二叉树 → 特别合适。复习写代码。

6) BinarySearchTree 的操作 → 复习写代码。有难度。

D. BinarySearchTree 的应用 → 3 个

CH 8 Heap a. Heap 相关 Def: 共 6 个, 有总结

b. 相关操作

Complexity

push (完全了 Heapify)
pop (完全了 Heapify)
peek
size
isEmpty

Complexity

借助入堆 操作实现进堆操作
通过遍历化实现进堆操作
Top-k 问题之 堆序法
Top-k 问题之 排序法
Top-k 问题之 堆法

C. 相关实现

1) 用数组实现堆 Push, SiftUp
↓ Pop, SiftDown → 复习写代码。

2) 建堆操作: 通过倒序遍历堆化以实现进堆操作。→ 复习写代码。有难度。

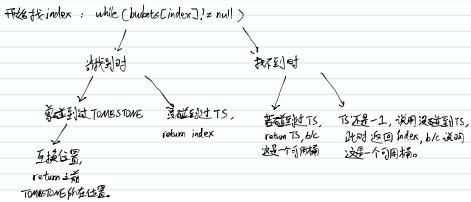
3) Top-k 问题之 堆法

D. 堆的常见应用: 3 个。

- 遍历法 → 无代码。有难度。
- 排序法 → 无代码。有难度。
- 堆法 → 复习写代码。有难度。

)理解。

FindBucket (int key) 中间如何判断 TOMBSTONE 已被检测到过, 和需要更新 lastTombstone 值的代码, 没有体现在图中。



D. 堆的常见应用：3个。

- a) 遍历选择法 → 无代码，有复杂度。
- b) 排序法 → 无代码，有复杂度。
- c) 堆法 → 复习写代码，有复杂度。

CH9 Graph a. Graph 相关Def：共一个，有总结。

b. 相关操作

邻接矩阵的	增 删 改 查	Complexity 基于邻接矩阵的图的实现	Complexity 基于adjList的图的实现	Linked List 实现		Complexity 基于adjList的图的实现
				添加边	删除边	
			添加边	$O(1)$	✓	添加边
			删除边	$O(n)$	✓	删除边
			添加顶点	$O(n)$	✓	添加顶点
			删除顶点	$O(n^2)$	✓	删除顶点
			初始化	$O(n^2)$	✓	初始化

c. 相关实现 1) 图的表示

a) 邻接矩阵 → 无代码。

b) 邻接表 → 无代码。

2) 图的基本操作

a) 基于 adjMat 的实现 → 复习写代码。有复杂度。

b) 基于 adjList 的图的实现

不能混淆 < 1. 教程中这部分复杂度是基于邻接矩阵实现的，无代码。
2. 教程中这部分代码是基于 adjList 的，有复杂度。

D. 图的常见应用：3个。

c) 图的遍历

1. BFS 实现 → 是用邻接矩阵实现的图的 BFS 实现。→ 复习写代码，有复杂度。

2. DFS 实现 → 是用邻接矩阵实现的图的 DFS 实现。→ 复习写代码，有复杂度。

实现 adjList 的图的实现:

	Complexity	Complexity
添加边	<code>以 adjList 实现的图的 BFS</code>	
删除边		<code>以 adjList 实现的图的 DFS</code>
添加顶点		
删除顶点		
初始化		