Lesson 03

数据结构
Data Structure

抽象数据类型 与 基本组成元素

抽象数据类型

Abstract Data Type - ADT

是什么：

ADT 规划了 特性 与 操作，类似接口

数据结构

数据结构 使用 基本组成元素 实现出 符合 ADT 的具体代码，类似实现类

基本组成元素：

基本元素	案例
变量	int
数组	int[]
指针	Point, int*

案例

ADT：

特性：

一对数值

需要区分左右

操作：

设置值

取值

给一个值，返回对面的那个值

API：

```
left: Int
right: Int
opposite(value: Int) : Int
```

数据结构 1：

```
Pair - class
+   left: Int
+   right: Int
```

数据结构 2：

```
Pair - class
+   values: Int[2]
```

线性数据结构

List

操作

```
add(index, value)
remove(index)
size()
get(index)
```

特性

保持位置

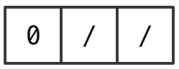
Doubly Linked List

结构：

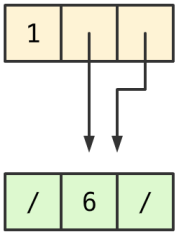
```
LinkedList - class
+   size: Int
+   firstNode: Node
+   lastNode: Node

Node<ElementType> - class
+   value: ElementType
+   prevNode: Node
+   nextNode: Node
```

图解：

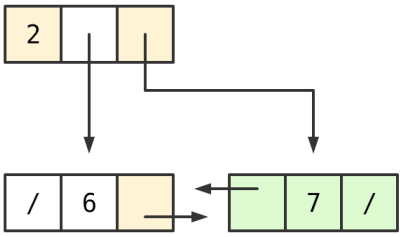
空



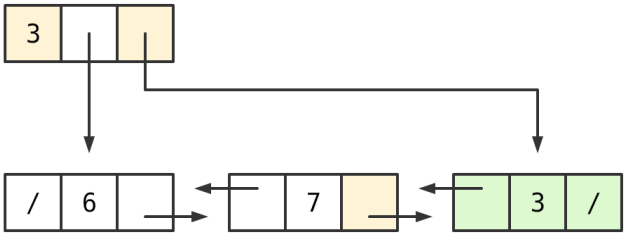
添加



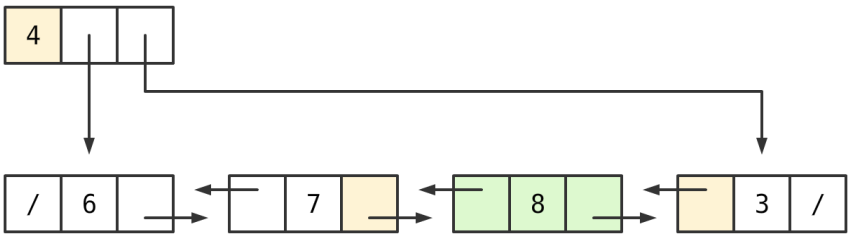
添加



添加



插入



时间复杂度分析：

add(index, value)：

remove(index)：

get(index)：

size()：

提问：

1. 把 8 节点删除掉，会有多少个格的内存被更改
2. 把 3 节点删除掉，会有多少个格的内存被更改

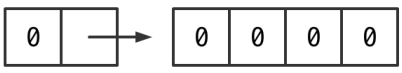
Array List

结构：

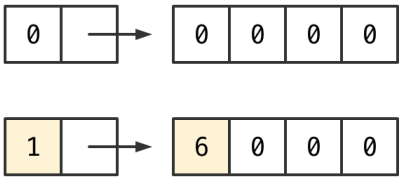
```
ArrayList<ElementType> - class
+   size: Int
+   source: ElementType[]
```

图解：

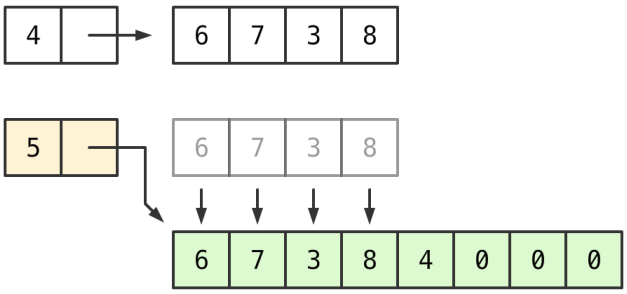
空



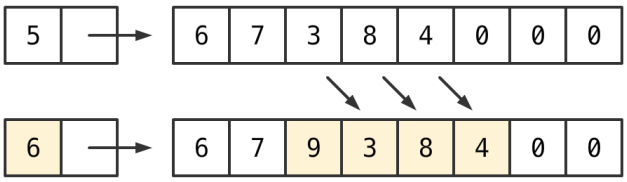
末尾添加



末尾添加



插入



时间复杂度分析：

add(index, value)：

remove(index)：

get(index)：

size()：

语言内置对照表

	List	Link-based List	Array-based List
Java	List	LinkedList	ArrayList
Python	List		
JavaScript	Array		
C++		list	vector
Swift	Array		

Copy 场景

```java
List<Integer> list = ...;
new LinkedList().addAll(list);

int[] arr = ...;
int[] newArr = ...;
Arrays.copy(arr, newArr)
```

注意：

时间复杂度 不可避免的 $O(n)$

合并 场景

```java
"abc" + "1"
```

问题：

时间复杂度 可能会达到 $O(n+m)$

不好情况案例：

```java
public String generateRandom(int length) {
    String str = "";
    for (int i = 0; i < length; i++) {
        str += new Random().nextInt(10);
    }
    Console.println(str);
}
```

时间复杂度为多少

解决方案：

```
StringBuilder
```

Slicing 场景

```java
list.subList(1, 3);
Arrays.copyOfRange(arr, 1, 3);
str.substring(1, 3);
```

注意：

时间复杂度 可能会达到 $O(n)$

不好情况案例：

```java
public List<String> split(String string) {
    List<String> parts = new LinkedList<String>();
    while(true) {
        int index = string.indexOf(" ");
        if (index < 0) {
            break;
        }
        String part = string.substring(0, index);
        parts.add(part);

        string = string.substring(index + 1);
    }
}
```

时间复杂度是多少

解决方案：

```
StringSlice
+   source: String
+   start: Int
+   end: Int
+   charAt(index: Int)
        return source.charAt(start + index)
```