

# 算法与数据结构体系课程

liuyubobobo

# 集合和映射

Set and Map

集合

# 集合

- 回忆我们上一小节实现的二分搜索树
- 不能盛放重复元素
- 非常好的实现“集合”的底层数据结构


# 集合

Set<E>

- `void add(E)`
- `void remove(E)`
- `boolean contains(E)`
- `int getSize()`
- `boolean isEmpty()`

# 集合

Set<E>

- `void add(E)`  不能添加重复元素
  - `void remove(E)`
  - `boolean contains(E)`
  - `int getSize()`
  - `boolean isEmpty()`
- 典型应用：客户统计
  - 典型应用：词汇量统计

实践：二分搜索树为底层的集合实现

# 使用链表实现集合



# 集合

Set<E>

- void add(E)
- void remove(E)
- boolean contains(E)
- int getSize()
- boolean isEmpty()



BSTSet<E>

implement

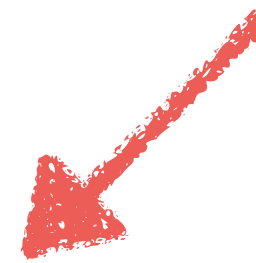


LinkedListSet<E>

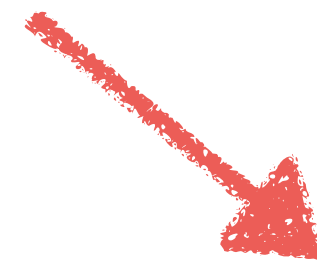
implement

# 集合

- BST 和 LinkedList都属于动态数据结构



```
class Node {  
    E e;  
    Node left;  
    Node right;  
}
```



```
class Node {  
    E e;  
    Node next;  
}
```

# 实践：使用链表实现集合

# 平衡二叉树的复杂度分析

实践： 比较链表实现的集合  
和平衡二叉树实现的集合

# 集合的时间复杂度分析

LinkedListSet

BSTSet

增 add

$O(n)$

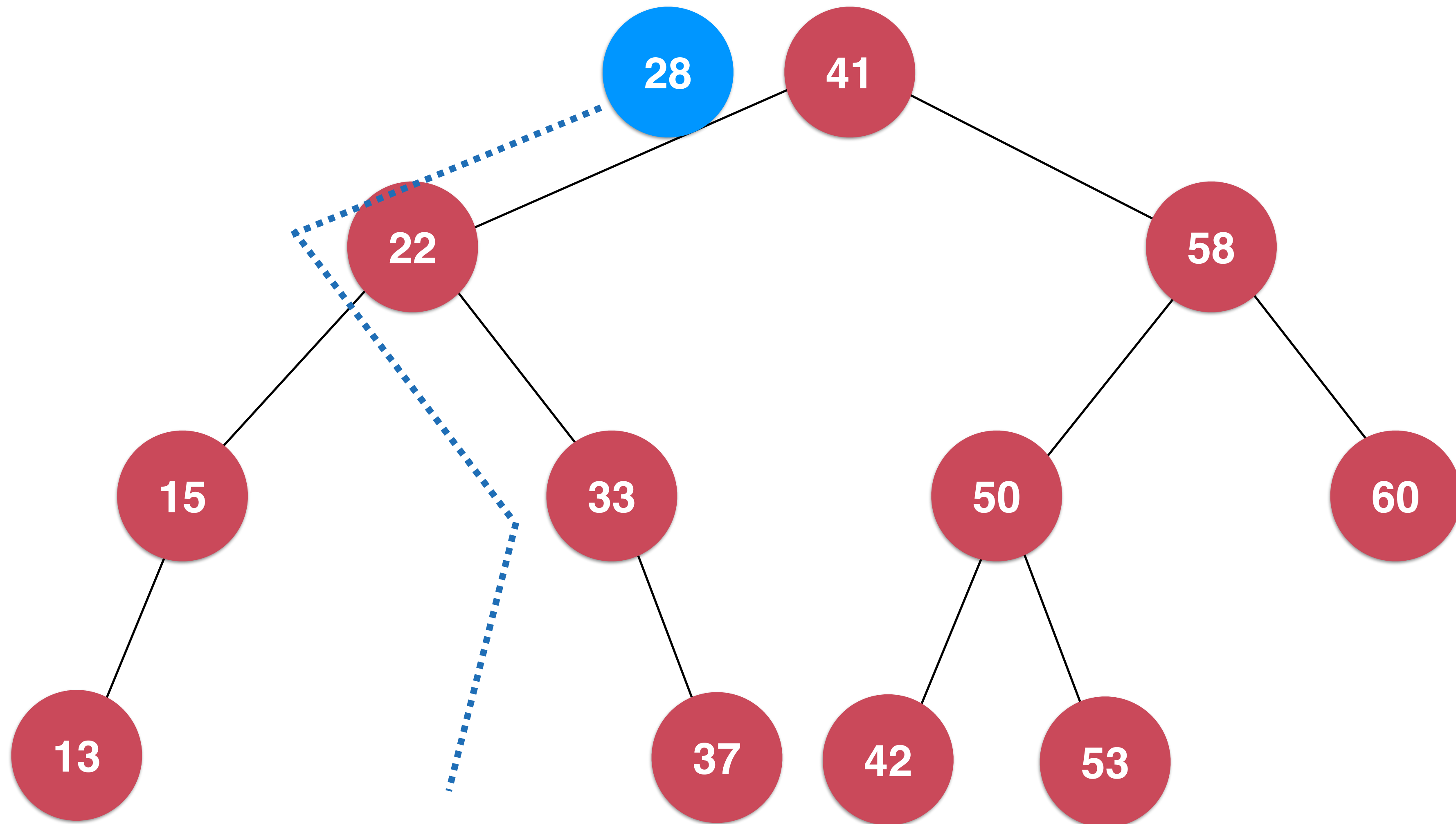
查 contains

$O(n)$

删 remove

$O(n)$

# 二分搜索树的复杂度分析



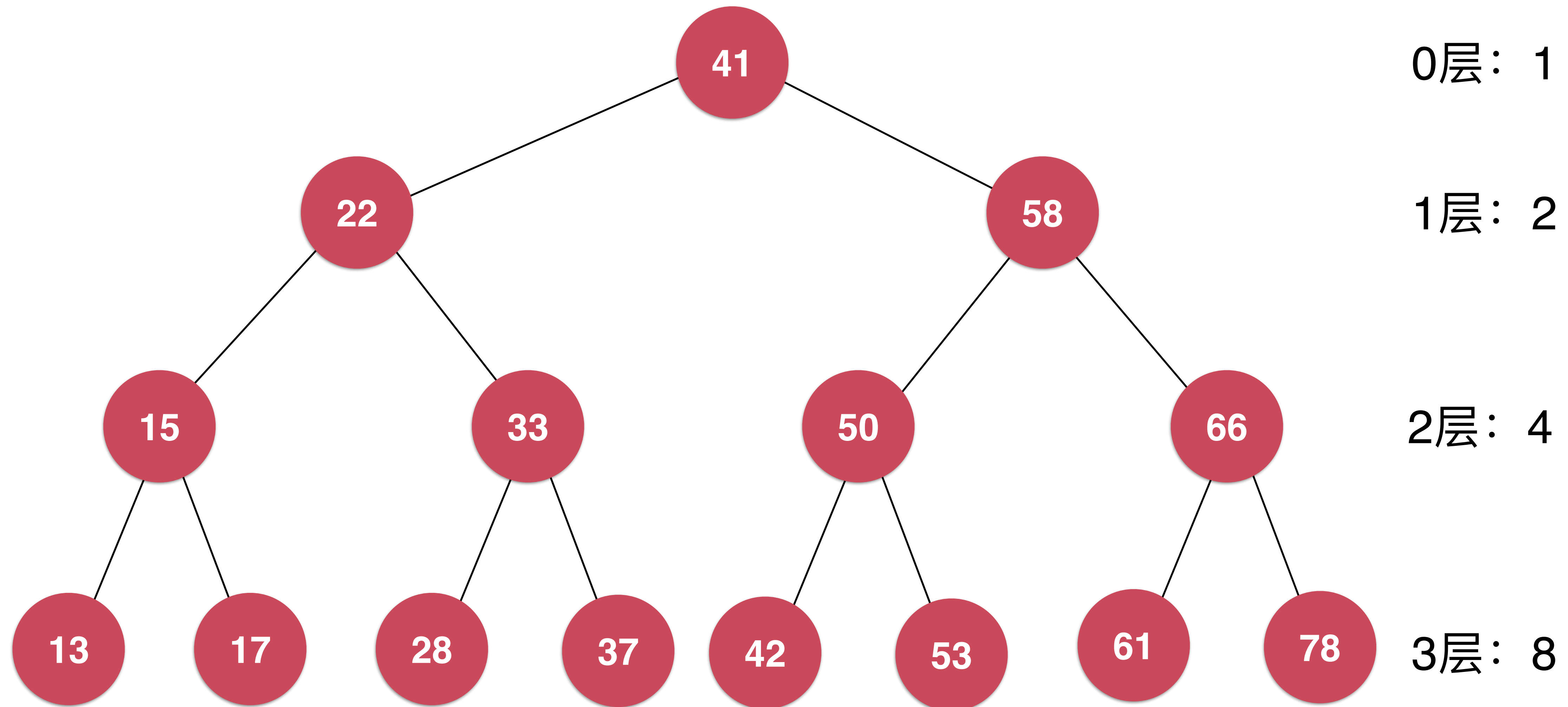
$O(h)$

# 集合的时间复杂度分析

	LinkedListSet	BSTSet
增 add	$O(n)$	$O(h)$
查 contains	$O(n)$	$O(h)$
删 remove	$O(n)$	$O(h)$



# 二分搜索树 Binary Search Tree



# 二分搜索树 Binary Search Tree

0层： 1

1层： 2

2层： 4

3层： 8

4层： 16

...

h-1层：  $2^{(h-1)}$

h层，一共多少个节点？

$$2^0 + 2^1 + 2^2 + 2^3 + 2^4 + \dots + 2^{h-1}$$

$$= \frac{1 \times (1 - 2^h)}{1 - 2} = 2^h - 1$$

# 二分搜索树 Binary Search Tree

h层，一共多少个节点？

$$2^0 + 2^1 + 2^2 + 2^3 + 2^4 + \dots + 2^{h-1}$$

$$= \frac{1 \times (1 - 2^h)}{1 - 2} = 2^h - 1 = n$$

$$h = \log_2(n + 1)$$

$$= O(\log_2 n) = O(\log n)$$

# 集合的时间复杂度分析

	LinkedListSet	BSTSet
增 add	$O(n)$	$O(h)$ $O(\log n)$
查 contains	$O(n)$	$O(h)$ $O(\log n)$
删 remove	$O(n)$	$O(h)$ $O(\log n)$

# logn和n的差距

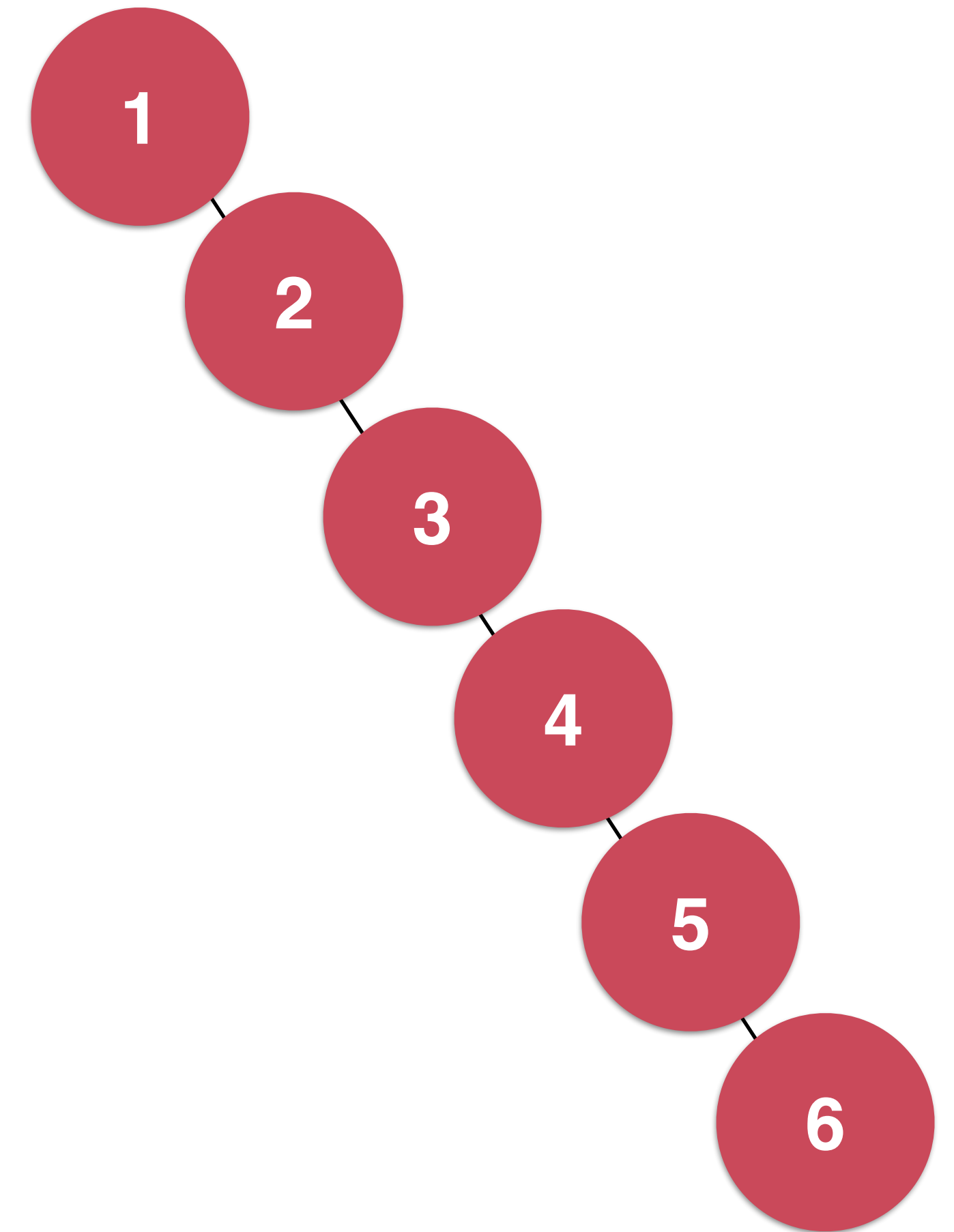
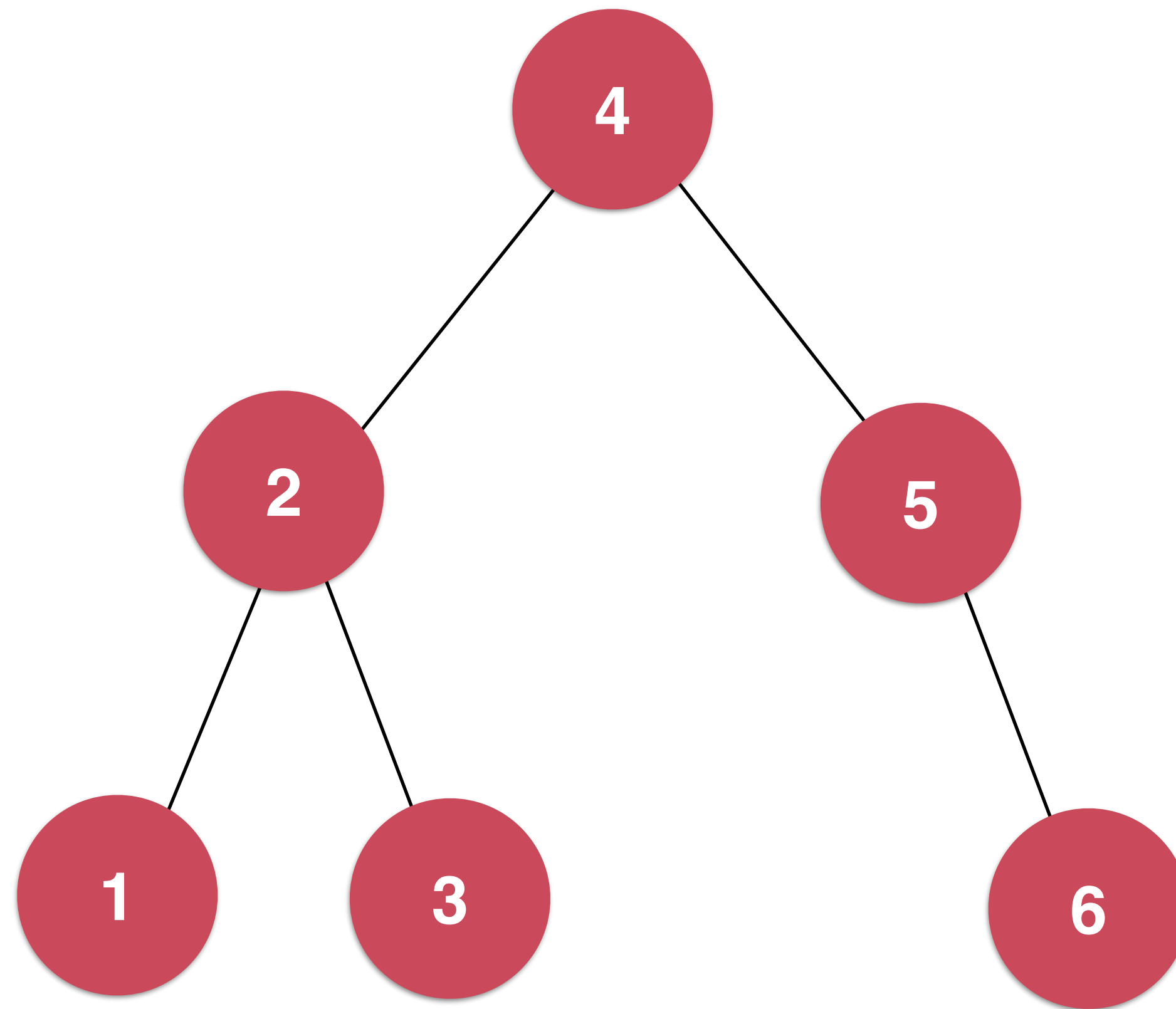
	logn	n	
n = 16	4	16	相差4倍
n = 1024	10	1024	相差100倍
n = 100万	20	100万	相差5万倍

# 集合的时间复杂度分析

	LinkedListSet	BSTSet	平均
增 add	$O(n)$	$O(h)$	$O(\log n)$
查 contains	$O(n)$	$O(h)$	$O(\log n)$
删 remove	$O(n)$	$O(h)$	$O(\log n)$

# 同样的数据， 可以对应不同的二分搜索树

1, 2, 3, 4, 5, 6



二分搜索树可能退化成链表




# 集合的时间复杂度分析

	LinkedListSet	BSTSet	平均	最差
增 add	$O(n)$	$O(h)$	$O(\log n)$	$O(n)$
查 contains	$O(n)$	$O(h)$	$O(\log n)$	$O(n)$
删 remove	$O(n)$	$O(h)$	$O(\log n)$	$O(n)$

Leetcode上集合相关的问题和更多集合类

# 集合

Set<E>

- `void add(E)`  不能添加重复元素
  - `void remove(E)`
  - `boolean contains(E)`
  - `int getSize()`
  - `boolean isEmpty()`
- 典型应用：客户统计
  - 典型应用：词汇量统计

实践：Leetcode 804

# 有序集合和无序集合

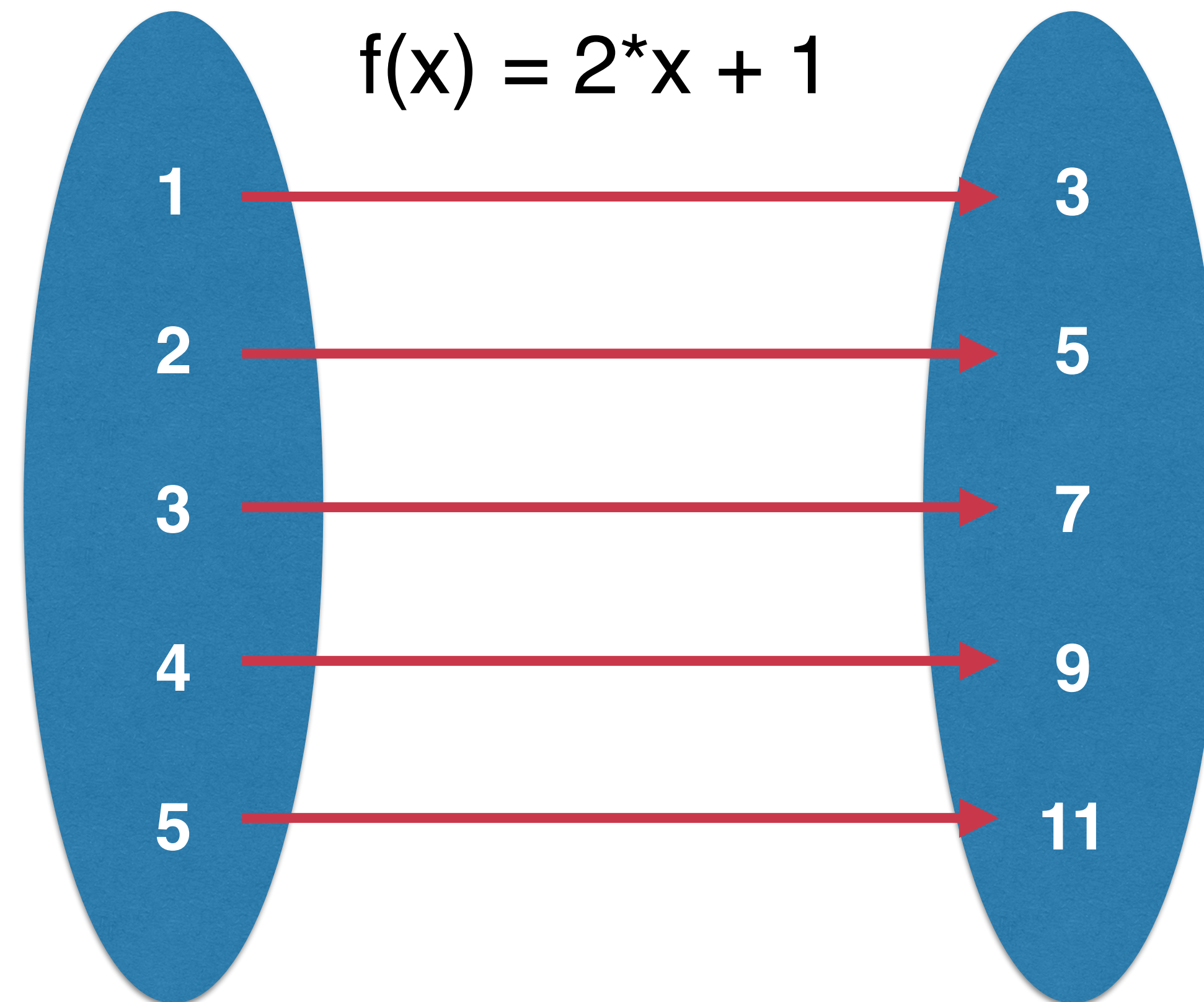
- 有序集合中的元素具有顺序性    ← 基于搜索树的实现
- 无序集合中的元素没有顺序性    ← 基于哈希表的实现

# 多重集合

- 多重集合中的元素可以重复

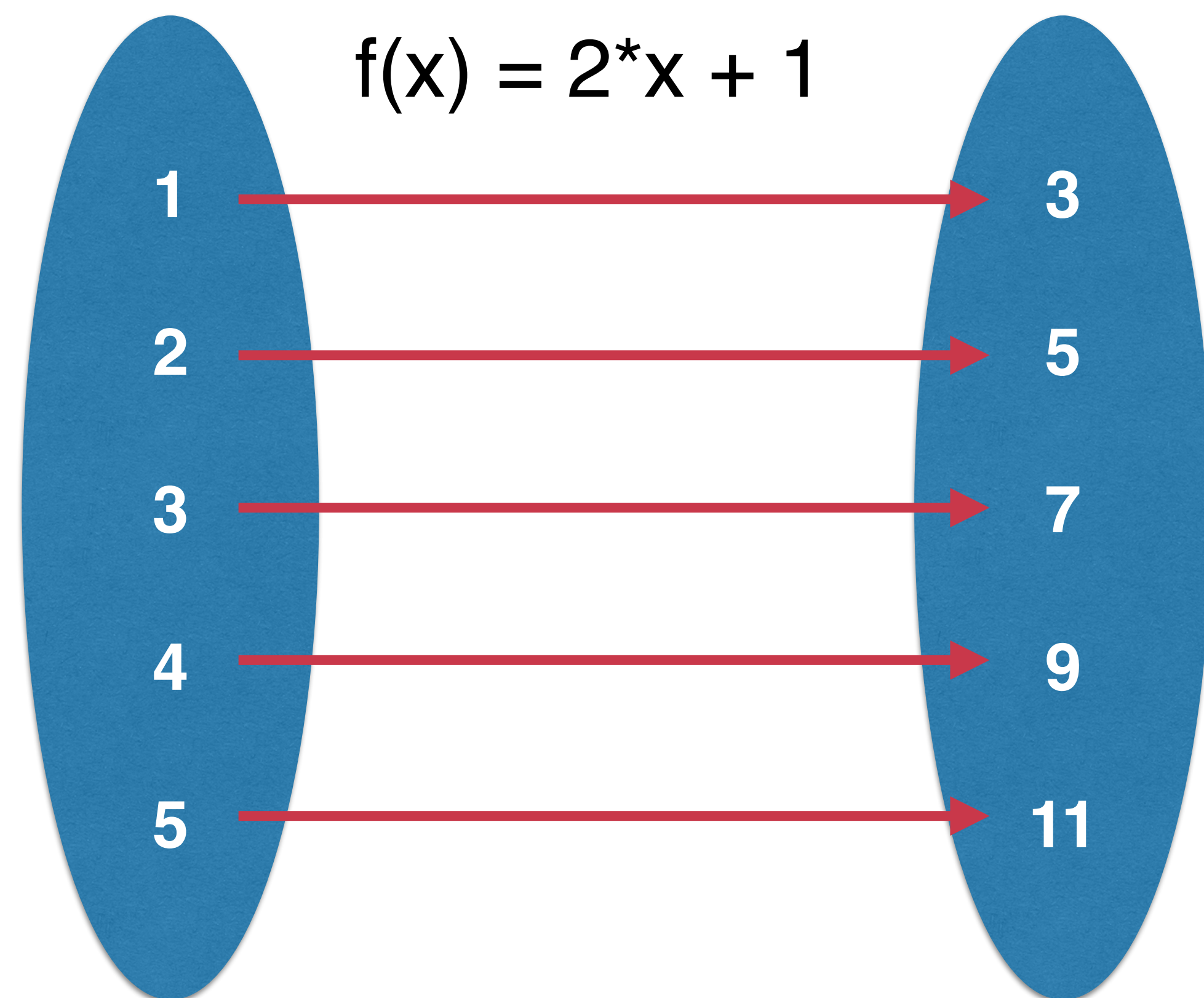
# 映射 Map

# 映射 Map





# 映射 Map



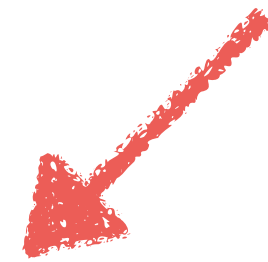
dict	Key	Value
字典	单词	释义
名册	身份证号	人
车辆管理	车牌号	车
数据库	id	信息
词频统计	单词	频率

# 映射 Map

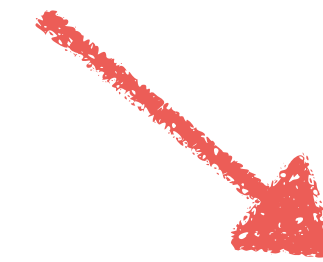
- 存储（键，值）数据对的数据结构（Key, Value）
- 根据键（Key），寻找值（Value）
- 非常容易使用链表或者二分搜索树实现

# 映射 Map

- 非常容易使用链表或者二分搜索树实现



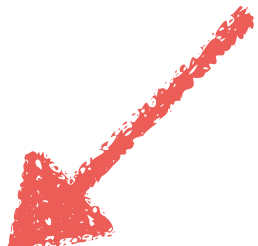
```
class Node {  
    E e;  
    Node left;  
    Node right;  
}
```




```
class Node {  
    E e;  
    Node next;  
}
```

# 映射 Map

- 非常容易使用链表或者二分搜索树实现



```
class Node {  
    K key;  
    V value  
    Node left;  
    Node right;  
}
```



```
class Node {  
    K key;  
    V Value  
    Node next;  
}
```

# 映射 Map

Map<K, V>

- void add(K, V)
- V remove(K)
- boolean contains(K)
- V get(K)
- void set(K, V)
- int getSize()
- boolean isEmpty()

# 实践：基于链表的映射

# 基于二分搜索树的映射

# 实践：基于二分搜索树的映射



# 二分搜索树的复杂度分析

实践： 实验不同Map的性能差距

# 映射的时间复杂度分析

	LinkedListMap	BSTMap	平均	最差
增 add	$O(n)$	$O(h)$	$O(\log n)$	$O(n)$
删 remove	$O(n)$	$O(h)$	$O(\log n)$	$O(n)$
改 set	$O(n)$	$O(h)$	$O(\log n)$	$O(n)$
查 get	$O(n)$	$O(h)$	$O(\log n)$	$O(n)$
查 contains	$O(n)$	$O(h)$	$O(\log n)$	$O(n)$

# 有序映射和无序映射

- 有序映射中的键具有顺序性      ← 基于搜索树的实现
- 无序映射中的键没有顺序性      ← 基于哈希表的实现

# 多重映射

- 多重映射中的键可以重复

# 集合和映射的关系

Set<E>

- void add(E)
- void remove(E)
- boolean contains(E)
- int getSize()
- boolean isEmpty()

Map<K, V>

- void add(K, V)
- V remove(K)
- boolean contains(K)
- V get(K)
- void set(K, V)
- int getSize()
- boolean isEmpty()

# 集合和映射相关的Leetcode问题

实践： Leetcode 349, 350



# Leetcode 上 HashTable标签相关的问题

# 其他

欢迎大家关注我的个人公众号：是不是很酷



# 算法与数据结构体系课程

liuyubobobo