栈和队列
stack & queue

无隅

栈和队列

- 栈：　LIFO(Last in first out)　后进先出
- 队列：　FIFO(First in first out)　先进先出

栈的初始化与基本操作

- Java 类库 : Stack<String> stack = new Stack<>();
- 基本操作：

方法	参数	返回值	时间复杂度
push	Element	Element/void	O(1)
pop	void	Element	O(1)
peek	void	Element	O(1)
isEmpty	void	boolean	O(1)

栈的实现

- 定义属性字段

1. 在数组的基础上实现
2. 属性： elements, size, capacity
3. 构造器实现

- 定义方法

1. push, pop, peek, isEmpty

栈的实现 – 属性及构造器

```java
public class Stack<T> {
    private int size;
    private int capacity;
    private T[] elementData;

    public Stack(int capacity) {
        this.size = 0;
        this.capacity = capacity;
        this.elementData = new T[capacity];
    }
}
```

栈的实现 – 方法、函数

```
public T push(T element) {
    if (size == capacity) {
        resize();
    }
    elementData[size++] = element;
    return element;
}
```

```
public T peek() {
    if (this.size == 0) {
        // throw Exception
        throw new IllegalStateException();
    }
    return elementData[size - 1];
}
```

```
public T pop() {
    if (size == 0) {
        // throw Exception
        throw new EmptyStackException();
    }

    T element = elementData[--this.size];
    return element;
}
```

```
public boolean isEmpty() {
    return this.size == 0;
}
```

什么时候考虑使用栈

- 调用函数
- 递归
- 深度优先搜素 DFS(Depth-first Search)

栈和堆在计算机操作系统上的概念

- 栈区　stack

1．存储 primitive variables function call

2．栈区的读取速度更快

- 堆区　heap

1．堆区存放引用类型变量

2．堆区可以动态地分配内存空间

队列的初始化与基本操作

- Java 类库 : Queue<String> queue = new LinkedList<>();
- 基本操作：

方法	参数	返回值	时间复杂度
offer (add)	Element	boolean	O(1)
poll (remove)	void	Element	O(1)
Peek (element)	void	Element	O(1)

队列的实现

- 定义属性字段

1．在链表的基础上实现

2．属性：　elements, size, capacity

3．构造器实现

- 定义方法

1. offer, poll, peek, isEmpty

队列的实现 – 属性及构造器

什么时候考虑使用队列

- 广度优先搜索 BFS (Breadth-first Search)
- 优先队列　Priority Queue (Heap)
- 多任务调度

栈与队列典型题目

Implement Queue using Stacks

Implement the following operations of a queue using stacks.

- push(x) -- Push element x to the back of queue.
- pop() -- Removes the element from in front of queue.
- peek() -- Get the front element.
- empty() -- Return whether the queue is empty.

Implement Queue using Stack 思路

- Use two Stacks to implement a queue
- Consider the basic operations of these two data structures
- How to minimize the time complexity

Max Stack

- Design a stack that supports push, pop, top, and retrieving the maximum element in constant time.
- push(x) -- Push element x onto stack.
- pop() -- Removes the element on top of the stack.
- top() -- Get the top element.
- getMax() -- Retrieve the maximum element in the stack.

Max Stack 思路

- One stack cannot tell the max in O(1)
- Need another stack to store the max
- time <=> space

Valid Parentheses

- Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.
- The brackets must close in the correct order, "()" and "()[]{}" are all valid but "(]" and "([)]" are not.

Valid Parentheses 思路

- We need to use the stack to store the information: which left parentheses has not been matched yet
- Stack can help store every unused parentheses, no matter when the parentheses appears

Valid Parentheses – follow up

- 返回括号对的个数
- 返回”{}”类型括号对的个数

Decode String

作业