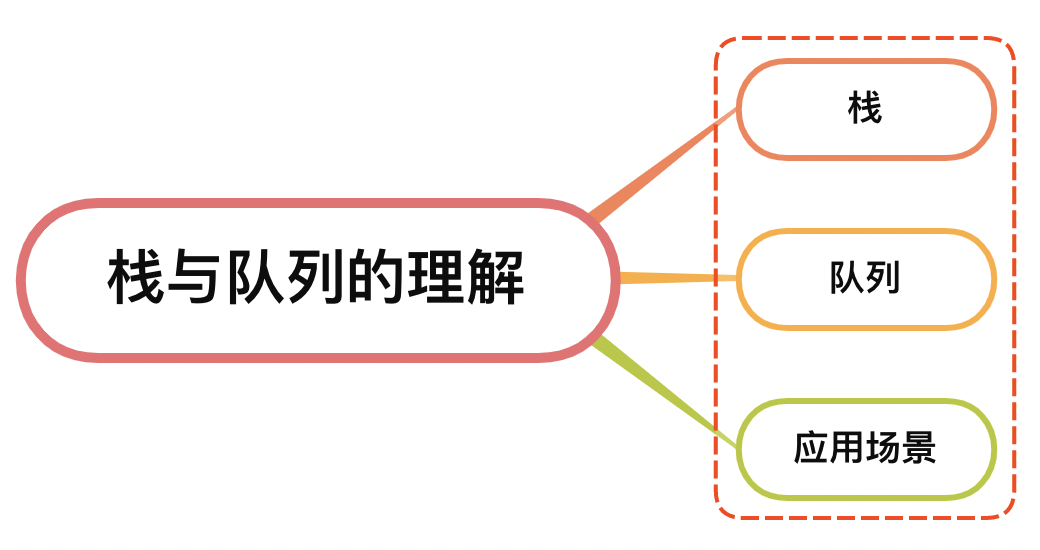
# 面试官：说说你对栈、队列的理解？应用场景？



## 一、栈

栈（stack）又名堆栈，它是一种运算受限的线性表，限定仅在表尾进行插入和删除操作的线性表

表尾这一端被称为栈顶，相反地另一端被称为栈底，向栈顶插入元素被称为进栈、入栈、压栈，从栈顶删除元素又称作出栈

所以其按照先进后出的原则存储数据，先进入的数据被压入栈底，最后的数据在栈顶，需要读数据的时候从栈顶开始弹出数据，具有记忆作用

关于栈的简单实现，如下：

class Stack {  
 constructor() {  
 this.items = [];  
 }  
  
 /\*\*  
 \* 添加一个（或几个）新元素到栈顶  
 \* @param {\*} element 新元素  
 \*/  
 push(element) {  
 this.items.push(element)  
 }  
  
 /\*\*  
 \* 移除栈顶的元素，同时返回被移除的元素  
 \*/  
 pop() {  
 return this.items.pop()  
 }  
  
 /\*\*  
 \* 返回栈顶的元素，不对栈做任何修改（这个方法不会移除栈顶的元素，仅仅返回它）  
 \*/  
 peek() {  
 return this.items[this.items.length - 1]  
 }  
  
 /\*\*  
 \* 如果栈里没有任何元素就返回true,否则返回false  
 \*/  
 isEmpty() {  
 return this.items.length === 0  
 }  
  
 /\*\*  
 \* 移除栈里的所有元素  
 \*/  
 clear() {  
 this.items = []  
 }  
  
 /\*\*  
 \* 返回栈里的元素个数。这个方法和数组的length属性很类似  
 \*/  
 size() {  
 return this.items.length  
 }  
}

关于栈的操作主要的方法如下：

* push：入栈操作
* pop：出栈操作

## 二、队列

跟栈十分相似，队列是一种特殊的线性表，特殊之处在于它只允许在表的前端（front）进行删除操作，而在表的后端（rear）进行插入操作

进行插入操作的端称为队尾，进行删除操作的端称为队头，当队列中没有元素时，称为空队列

在队列中插入一个队列元素称为入队，从队列中删除一个队列元素称为出队。因为队列只允许在一端插入，在另一端删除，所以只有最早进入队列的元素才能最先从队列中删除，故队列又称为先进先出

简单实现一个队列的方式，如下：

class Queue {  
 constructor() {  
 this.list = []  
 this.frontIndex = 0  
 this.tailIndex = 0  
 }  
 enqueue(item) {  
 this.list[this.tailIndex++] = item  
 }  
 unqueue() {  
 const item = this.list[this.frontIndex]  
 this.frontIndex++   
 return item  
 }  
}

上述这种入队和出队操作中，头尾指针只增加不减小，致使被删元素的空间永远无法重新利用

当队列中实际的元素个数远远小于向量空间的规模时，也可能由于尾指针已超越向量空间的上界而不能做入队操作，出该现象称为"假溢"

在实际使用队列时，为了使队列空间能重复使用，往往对队列的使用方法稍加改进：

无论插入或删除，一旦rear指针增1或front指针增1 时超出了所分配的队列空间，就让它指向这片连续空间的起始位置，这种队列也就是循环队列

下面实现一个循环队列，如下：

class Queue {  
 constructor(size) {  
 this.size = size; // 长度需要限制, 来达到空间的利用, 代表空间的长度  
 this.list = [];  
 this.font = 0; // 指向首元素  
 this.rear = 0; // 指向准备插入元素的位置  
 }  
 enQueue() {  
 if (this.isFull() == true) {  
 return false  
 }  
 this.rear = this.rear % this.k;  
 this.\_data[this.rear++] = value;  
 return true  
 }  
 deQueue() {  
 if(this.isEmpty()){  
 return false;  
 }  
 this.font++;  
 this.font = this.font % this.k;  
 return true;  
 }  
 isEmpty() {  
 return this.font == this.rear - 1;  
 }  
 isFull() {  
 return this.rear % this.k == this.font;  
 }  
}

上述通过求余的形式代表首尾指针增1 时超出了所分配的队列空间

## 三、应用场景

### 栈

借助栈的先进后出的特性，可以简单实现一个逆序数处的功能，首先把所有元素依次入栈，然后把所有元素出栈并输出

包括编译器的在对输入的语法进行分析的时候，例如"()"、"{}"、"[]"这些成对出现的符号，借助栈的特性，凡是遇到括号的前半部分，即把这个元素入栈，凡是遇到括号的后半部分就比对栈顶元素是否该元素相匹配，如果匹配，则前半部分出栈，否则就是匹配出错

包括函数调用和递归的时候，每调用一个函数，底层都会进行入栈操作，出栈则返回函数的返回值

生活中的例子，可以把乒乓球盒比喻成一个堆栈，球一个一个放进去（入栈），最先放进去的要等其后面的全部拿出来后才能出来（出栈），这种就是典型的先进后出模型

### 队列

当我们需要按照一定的顺序来处理数据，而该数据的数据量在不断地变化的时候，则需要队列来帮助解题

队列的使用广泛应用在广度优先搜索种，例如层次遍历一个二叉树的节点值（后续将到）

生活中的例子，排队买票，排在队头的永远先处理，后面的必须等到前面的全部处理完毕再进行处理，这也是典型的先进先出模型

## 参考文献

* https://baike.baidu.com/item/%E6%A0%88/12808149
* https://baike.baidu.com/item/%E9%98%9F%E5%88%97/14580481