#### 栈

栈是一种遵从**后进先出（LIFO）**原则的**有序集合**。新添加或待删除的元素都保存在栈的同一端，称为**栈顶**；另一端就叫**栈底**。在栈里，新元素都靠近栈顶，旧元素都接近栈底。

##### 创建一个基于JavaScript数组的栈

###### 声明一个**StackArray** 类来表示栈

class StackArray {

constructor() {

this.items = [];

}

}

###### 向栈中添加元素（push方法）

push(element) {

this.items.push(element);

}

###### 从栈中移除元素（pop方法）

pop() {

if (this.isEmpty()) {

return undefined;

}

return this.items.pop();

}

###### 查看栈顶元素（peek方法）

peek() {

if (this.isEmpty()) {

return undefined;

}

return this.items[this.items.length - 1];

}

###### 检查栈是否为空（isEmpty方法）

isEmpty() {

return this.items.length === 0;

}

###### 清空栈元素（clear方法）

clear() {

this.items = [];

}

###### 获取栈的长度（size方法）

size() {

return this.items.length;

}

##### 创建一个基于JavaScript对象的栈

###### 声明一个**Stack** 类来表示栈

class Stack {

constructor() {

this.count = 0;

this.items = {};

}

}

###### 向栈中添加元素（push方法）

push(element) {

this.items[this.count] = element;

this.count++;

}

###### 从栈中移除元素（pop方法）

pop() {

if (this.isEmpty()) {

return undefined;

}

this.count--;

const result = this.items[this.count];

delete this.items[this.count];

return result;

}

###### 查看栈顶元素（peek方法）

peek() {

if (this.isEmpty()) {

return undefined;

}

return this.items[this.count - 1];

}

###### 检查栈是否为空（isEmpty方法）

isEmpty() {

return this.count === 0;

}

###### 清空栈元素（clear方法）

clear() {

this.items = {};

this.count = 0;

}

###### 获取栈的长度（size方法）

size() {

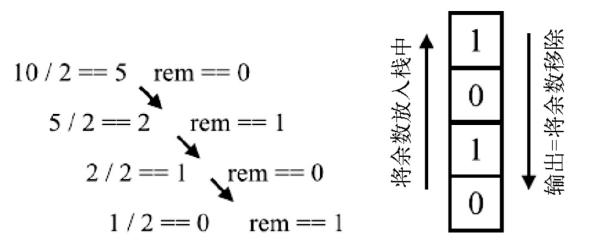
return this.count;

}

##### 用栈解决问题

###### 二进制转为十进制

思路：



方法：

function decimalToBinary(num){

const remStack = new Stack();

let rem, bryStr = '';

while(num > 0){

rem = Math.floor(num % 2);

remStack.push(rem);

num = Math.floor(num / 2);

}

while(!remStack.isEmpty()){

bryStr += remStack.pop();

}

return bryStr;

}

###### 十进制转换成基数为2~36的任意进制

function baseConverter(decNum, base){

const remStack = new Stack();

const digits = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ';

let rem, baseStr = '';

if(!(base >= 2 && base <= 36)){

return '';

}

while(decNum > 0){

rem = Math.floor(decNum % base);

remStack.push(rem);

decNum = Math.floor(decNum / base);

}

while(!remStack.isEmpty()){

baseStr += digits[remStack.pop()];

}

return baseStr;

}