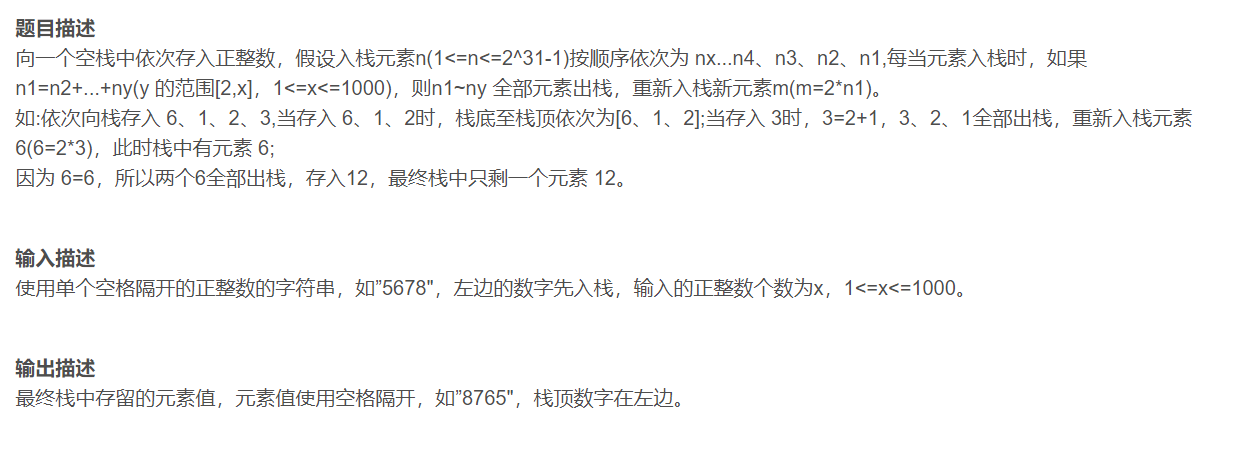
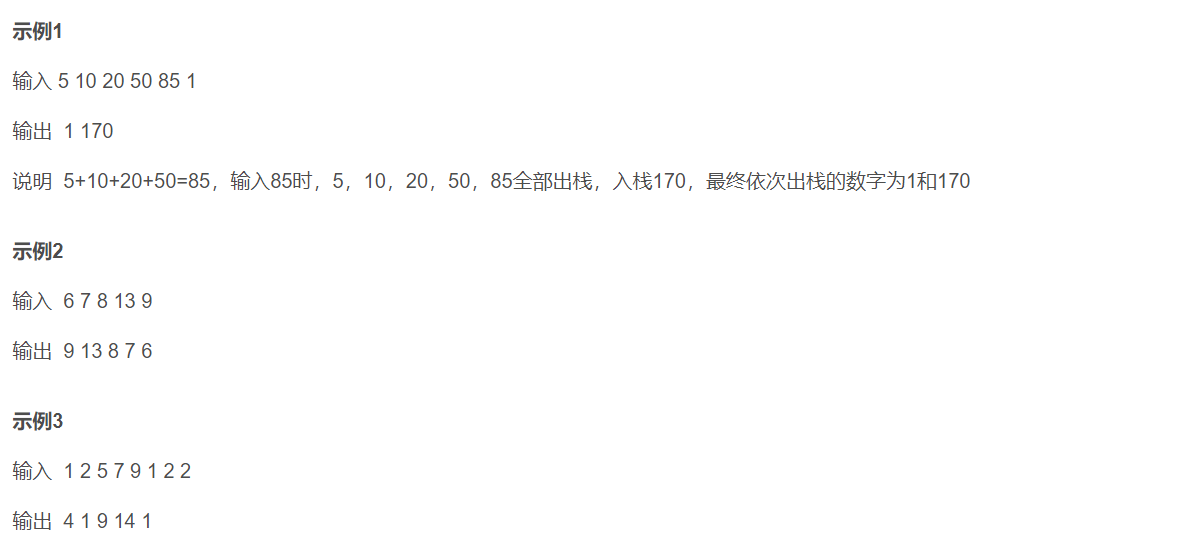
# **E卷-堆栈中的剩余数字[200分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**







import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// 定义栈，容量比可能的最大输入数据要大

long []st = new long[100010];

// 读取输入字符串，并以空格分隔转化为字符串数组

String []s = sc.nextLine().split(" ");

// 将字符串数组转换为长整型数组

long []arr = new long[s.length];

for (int i = 0; i < s.length; i++) {

arr[i] = Long.parseLong(s[i]);

}

int top = -1; // 初始化栈顶指针

// 遍历每个输入的元素

for (long x : arr) {

while (true) {

st[++top] = x; // 元素入栈

long sum = 0;

int index = -1;

// 从栈顶开始向下检查是否有连续元素的和等于当前元素

for (int j = top - 1; j >= 0; j--) {

sum += st[j];

if (sum == x) {

index = j; // 找到满足条件的起始位置

break;

}

}

// 如果找不到，跳出循环

if (index == -1) break;

// 重置栈顶指针到找到的位置之前

top = index - 1;

// 将当前元素替换为新元素，继续检查

x \*= 2;

}

}

// 输出最终栈中的元素，从栈顶开始输出

for (int i = top; i >= 0; i--) {

if (i != top) {

System.out.print(" ");

}

System.out.print(st[i]);

}

}

}



def main():

import sys

# 读取标准输入

input = sys.stdin.read

# 将输入按空格分隔，并转换为列表

data = input().strip().split()

# 把字符串列表转换为整数列表

arr = list(map(int, data))

# 初始化栈，容量设置的非常大以适应潜在的最大输入

st = [0] \* 100010

# 栈顶指针初始化为 -1，表示栈是空的

top = -1

# 遍历每个从输入中取得的元素

for x in arr:

while True:

# 将当前元素x压入栈中

top += 1

st[top] = x

sum\_ = 0 # 初始化和

index = -1

# 从栈顶向下检查是否存在连续元素之和等于当前元素

for j in range(top - 1, -1, -1):

sum\_ += st[j]

if sum\_ == x:

index = j # 记录开始的位置

break

# 如果没找到符合条件的片段，退出循环

if index == -1:

break

# 修改栈顶指针到找到的位置之前

top = index - 1

# 将当前元素替换为新元素，并继续检查

x \*= 2

# 利用列表推导式收集栈中元素，以从栈顶开始输出

result = [str(st[i]) for i in range(top, -1, -1)]

# 打印最终栈中的元素，栈顶元素在最前

print(" ".join(result))

# 如果脚本是直接运行的，则调用main函数

if \_\_name\_\_ == "\_\_main\_\_":

main()



#include <iostream>

#include <sstream>

#include <vector>

using namespace std;

int main() {

string line;

getline(cin, line); // 读取输入字符串

istringstream iss(line); // 将输入字符串转换为输入流

vector<long long> arr;

long long num;

// 读取输入流中的所有数字到数组中

while (iss >> num) {

arr.push\_back(num);

}

vector<long long> stack(100010);

int top = -1;

// 遍历每个输入的元素

for (long long x : arr) {

while (true) {

stack[++top] = x; // 元素入栈

long long sum = 0;

int index = -1;

// 从栈顶开始向下检查是否有连续元素的和等于当前元素

for (int j = top - 1; j >= 0; --j) {

sum += stack[j];

if (sum == x) {

index = j; // 找到满足条件的起始位置

break;

}

}

// 如果找不到，跳出循环

if (index == -1) break;

// 重置栈顶指针到找到的位置之前

top = index - 1;

// 将当前元素替换为新元素，继续检查

x \*= 2;

}

}

// 输出最终栈中的元素，从栈顶开始输出

for (int i = top; i >= 0; --i) {

if (i != top) cout << " ";

cout << stack[i];

}

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// 定义栈的大小

#define MAX\_SIZE 100010

int main() {

char line[MAX\_SIZE];

// 读取输入字符串

fgets(line, MAX\_SIZE, stdin);

long long arr[MAX\_SIZE];

long long stack[MAX\_SIZE];

int arr\_size = 0;

char\* token = strtok(line, " ");

while (token != NULL) {

arr[arr\_size++] = atoll(token);

token = strtok(NULL, " ");

}

int top = -1;

// 遍历每个输入的元素

for (int i = 0; i < arr\_size; ++i) {

long long x = arr[i];

while (1) {

stack[++top] = x; // 元素入栈

long long sum = 0;

int index = -1;

// 从栈顶开始向下检查是否有连续元素的和等于当前元素

for (int j = top - 1; j >= 0; --j) {

sum += stack[j];

if (sum == x) {

index = j; // 找到满足条件的起始位置

break;

}

}

// 如果找不到，跳出循环

if (index == -1) break;

// 重置栈顶指针到找到的位置之前

top = index - 1;

// 将当前元素替换为新元素，继续检查

x \*= 2;

}

}

// 输出最终栈中的元素，从栈顶开始输出

for (int i = top; i >= 0; --i) {

if (i != top) printf(" ");

printf("%lld", stack[i]);

}

return 0;

}



const readline = require("readline");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

rl.on("line", (line) => {

const data = line.trim().split(" ").map(Number);

const stack = new Array(100010);

let top = -1;

// 遍历每个输入的元素

for (let x of data) {

while (true) {

stack[++top] = x; // 元素入栈

let sum = 0;

let index = -1;

// 从栈顶开始向下检查是否有连续元素的和等于当前元素

for (let j = top - 1; j >= 0; j--) {

sum += stack[j];

if (sum === x) {

index = j; // 找到满足条件的起始位置

break;

}

}

// 如果找不到，跳出循环

if (index === -1) break;

// 重置栈顶指针到找到的位置之前

top = index - 1;

// 将当前元素替换为新元素，继续检查

x \*= 2;

}

}

// 输出最终栈中的元素，从栈顶开始输出

const result = [];

for (let i = top; i >= 0; i--) {

result.push(stack[i]);

}

console.log(result.join(" "));

rl.close();

});



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

func main() {

reader := bufio.NewReader(os.Stdin)

line, \_ := reader.ReadString('\n')

line = strings.TrimSpace(line)

parts := strings.Split(line, " ")

arr := make([]int64, len(parts))

for i, part := range parts {

num, \_ := strconv.ParseInt(part, 10, 64)

arr[i] = num

}

stack := make([]int64, 100010)

top := -1

// 遍历每个输入的元素

for \_, x := range arr {

for {

top++

stack[top] = x // 元素入栈

sum, index := int64(0), -1

// 从栈顶开始向下检查是否有连续元素的和等于当前元素

for j := top - 1; j >= 0; j-- {

sum += stack[j]

if sum == x {

index = j // 找到满足条件的起始位置

break

}

}

// 如果找不到，跳出循环

if index == -1 {

break

}

// 重置栈顶指针到找到的位置之前

top = index - 1

// 将当前元素替换为新元素，继续检查

x \*= 2

}

}

// 输出最终栈中的元素，从栈顶开始输出

result := make([]string, 0, top+1)

for i := top; i >= 0; i-- {

result = append(result, strconv.FormatInt(stack[i], 10))

}

fmt.Println(strings.Join(result, " "))

}