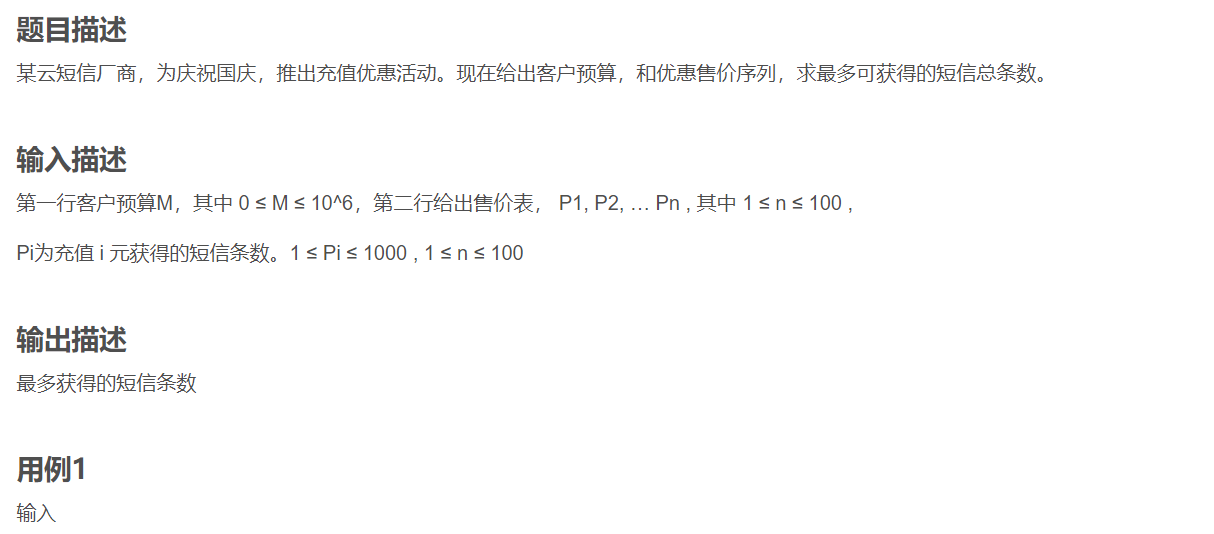
# **E卷-云短信平台优惠活动[200分]（ Java | Python3 | C++ | C语言 | JsNode | Go ）**

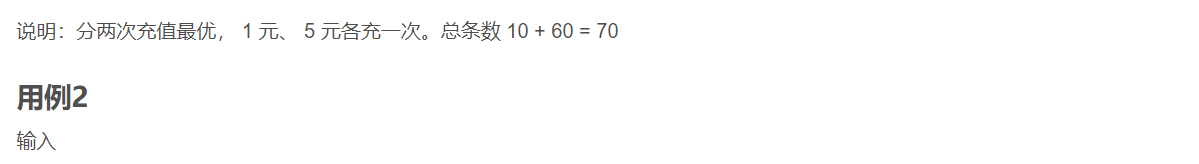


6

10 20 30 40 60



70

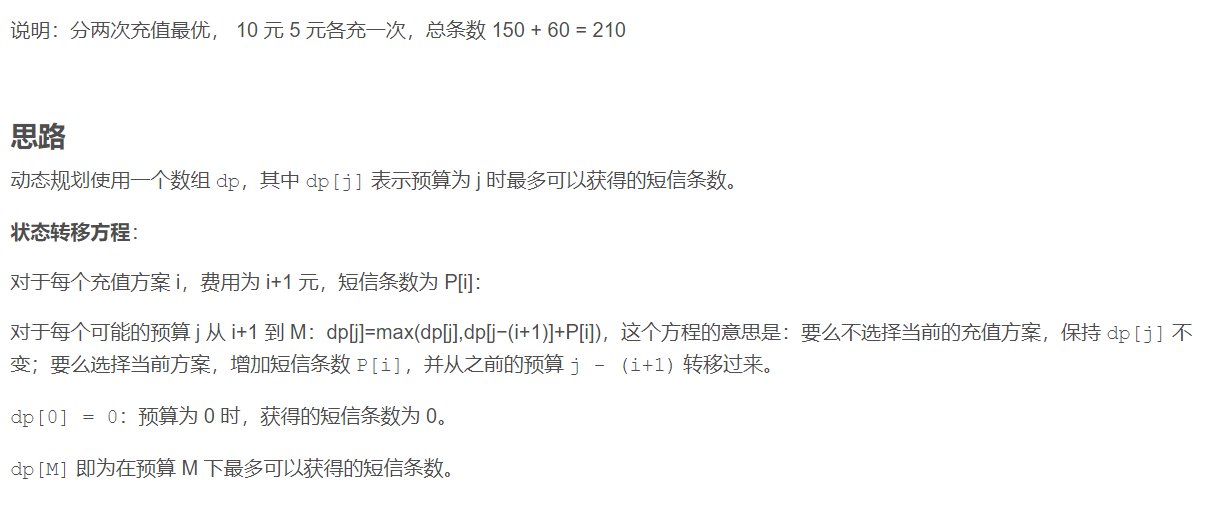


15

10 20 30 40 60 60 70 80 90 150



210





import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int k = scanner.nextInt(); // 读取预算M

scanner.nextLine();

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

int[] a = new int[s.length]; // 读取售价表长度

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

a[i] = Integer.parseInt(s[i]);

}

int[] dp = new int[k + 1]; // 初始化动态规划数组

// 遍历每种充值方案

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

int val = a[i]; // 当前充值方案获得的短信条数

int cost = i + 1; // 当前充值方案的费用

for (int j = cost; j <= k; j++) {

// 更新最大短信条数

dp[j] = Math.max(dp[j], dp[j - cost] + val);

}

}

System.out.println(dp[k]); // 输出最多获得的短信条数

}

}



k = int(input()) # 读取客户预算M

a = list(map(int, input().split())) # 读取售价表，转换为整数列表

dp = [0] \* (k + 1) # 初始化动态规划数组，长度为预算+1，初始值为0

# 遍历每种充值方案

for i in range(len(a)):

val = a[i] # 充值i+1元获得的短信条数

cost = i + 1 # 充值金额

# 从当前预算开始，向上更新到总预算k

for j in range(cost, k + 1):

# 更新最大短信条数

dp[j] = max(dp[j], dp[j - cost] + val)

print(dp[k]) # 输出最多获得的短信条数



#include <iostream>

#include <vector>

#include <algorithm>

int main() {

int k;

std::cin >> k; // 读取预算M

std::vector<int> a;

int n = 0, x;

while (std::cin >> x) {

a.push\_back(x);

n++;

}

std::vector<int> dp(k + 1, 0); // 初始化动态规划数组

// 遍历每种充值方案

for (int i = 0; i < n; i++) {

int val = a[i]; // 当前充值方案获得的短信条数

int cost = i + 1; // 当前充值方案的费用

for (int j = cost; j <= k; j++) {

// 更新最大短信条数

dp[j] = std::max(dp[j], dp[j - cost] + val);

}

}

std::cout << dp[k] << std::endl; // 输出最多获得的短信条数

return 0;

}



#include <stdio.h>

#include <string.h>

int main() {

int k, n = 0, x;

scanf("%d", &k); // 读取预算M

int a[101];

while (scanf("%d", &x) != EOF) {

a[n++] = x;

}

int dp[k + 1];

memset(dp, 0, sizeof(dp)); // 初始化动态规划数组

// 遍历每种充值方案

for (int i = 0; i < n; i++) {

int val = a[i]; // 当前充值方案获得的短信条数

int cost = i + 1; // 当前充值方案的费用

for (int j = cost; j <= k; j++) {

// 更新最大短信条数

if (dp[j] < dp[j - cost] + val) {

dp[j] = dp[j - cost] + val;

}

}

}

printf("%d\n", dp[k]); // 输出最多获得的短信条数

return 0;

}



const readline = require('readline');

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

let inputLines = [];

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

inputLines.push(line);

if (inputLines.length === 2) {

const k = parseInt(inputLines[0]); // 读取预算M

const a = inputLines[1].split(' ').map(Number); // 读取售价表

const dp = Array(k + 1).fill(0); // 初始化动态规划数组

// 遍历每种充值方案

for (let i = 0; i < a.length; i++) {

const val = a[i]; // 当前充值方案获得的短信条数

const cost = i + 1; // 当前充值方案的费用

for (let j = cost; j <= k; j++) {

// 更新最大短信条数

dp[j] = Math.max(dp[j], dp[j - cost] + val);

}

}

console.log(dp[k]); // 输出最多获得的短信条数

rl.close();

}

});



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

func main() {

scanner := bufio.NewScanner(os.Stdin)

// 读取预算M

scanner.Scan()

k, \_ := strconv.Atoi(scanner.Text())

// 读取售价表

scanner.Scan()

input := scanner.Text()

strPrices := strings.Fields(input)

a := make([]int, len(strPrices))

for i, str := range strPrices {

a[i], \_ = strconv.Atoi(str) // 转换每个售价为整数

}

dp := make([]int, k+1) // 初始化动态规划数组

// 遍历每种充值方案

for i := 0; i < len(a); i++ {

val := a[i] // 当前充值方案获得的短信条数

cost := i + 1 // 当前充值方案的费用

for j := cost; j <= k; j++ {

// 更新最大短信条数

if dp[j] < dp[j-cost]+val {

dp[j] = dp[j-cost] + val

}

}

}

fmt.Println(dp[k]) // 输出最多获得的短信条数

}