

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Российский химико-технологический университет имени Д.И.  
Менделеева»

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №10

Выполнил студент группы КС-36: Золотухин Андрей Александрович

Ссылка на репозиторий: [https://github.com/  
MUCTR-IKT-CPP/  
ZolotukhinAA\\_36\\_ALG](https://github.com/MUCTR-IKT-CPP/ZolotukhinAA_36_ALG)

Принял: Крашенников Роман Сергеевич

Дата сдачи: 12.05.2025

Москва  
2025

# Оглавление

Описание задачи . . . . .	1
Описание метода/модели . . . . .	2
Выполнение задачи . . . . .	2

## Описание задачи

## Описание метода/модели

Идея **динамического программирования** в том, что мы разбиваем задачу на малые подзадачи, спускаясь вниз, далее, находим самую минимальную подзадачу, решаем её, затем, имея решение, мы создаём подзадачи, которые включают в своё решение решённую более меньшую подзадачу, снова получаем их решения, сохраняем их, и движемся далее вверх.

По итогу динамическое программирование требует следующий условий для решений какой-либо задачи:

- перекрывающиеся подзадачи;
- оптимальная подструктура;
- возможность запоминания решения часто встречающихся подзадач.

## Выполнение задачи

Задача реализована на языке *Java*.

```
1 package com.dp;
2
3 import java.util.Random;
4
5 public class Main
6 {
7     public static void main( String[] args )
8     {
9         //Random random = new Random();
10        //int n = random.nextInt(20) + 1;
11        //int[] w = new int[n + 1];
12        //int total = 0;
13
14        //System.out.println("Generated stones: " + n);
15        //System.out.print("Weights of stones: ");
16
17        //for (int i = 1; i <= n; i++) {
18        //w[i] = random.nextInt(10000) + 1;
19        //total += w[i];
20        //System.out.print(w[i] + " ");
21        //}
22
23        //System.out.println("\nTotal weight: " + total);
24        //int minDiff = solveKnapsack(w, total);
25        //System.out.println("Min diff: " + minDiff);
26
27        Scanner sc = new Scanner(System.in);
28        int n = sc.nextInt();
29        int[] w = new int[n + 1];
30        int total = 0;
31
32        for (int i = 1; i <= n; i++) {
33            w[i] = sc.nextInt();
```

```

34     total += w[i];
35 }
36
37 System.out.println(solveKnapsack(w, total));
38 }
39
40 private static int solveKnapsack(int[] w, int total) {
41     int W = total / 2;
42     int[][] dp = new int[2][W + 1];
43
44     for (int j = 0; j <= W; j++) {
45         dp[0][j] = 0;
46     }
47
48     for (int i = 1; i < w.length; i++) {
49         int curr = i % 2;
50         int prev = 1 - curr;
51
52         for (int j = 0; j <= W; j++) {
53             if (w[i] <= j) {
54                 dp[curr][j] = Math.max(dp[prev][j], dp[prev][j - w[i]] + w[i]);
55             } else {
56                 dp[curr][j] = dp[prev][j];
57             }
58         }
59     }
60
61     int maxSum = dp[(w.length - 1) % 2][W];
62     return total - 2 * maxSum;
63 }
64 }

```