

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра ІІІ(ІСТ)

Звіт

з лабораторної роботи № 9 з дисципліни

«Теорія алгоритмів»

„Динамічне програмування ”

Виконали

ІА-31 Дук М.Д, Клим'юк В.Л, Сакун Д.С, Самелюк А.С

Перевірив

Степанов А.С

Київ 2024

Завдання

Застосування динамічного програмування для задачі про рюкзак є одним з яскравих прикладів потужності цього підходу. Дана задача формулюється наступним чином:

Дано n різних предметів, про які відомі їх розмір, або вага, w_i та вартість v_i . Є рюкзак, в який необхідно покласти ці предмети. Для рюкзака відома його місткість (сумарний розмір, або вага, предметів, що можуть бути розміщені у рюкзаку) — W . Необхідно відібрати таку множину предметів S серед усіх заданих предметів, що (1) їх сумарна розмірність не перевищує місткість рюкзака W та (2) сумарна вартість предметів в множині S є максимально можливою серед усіх інших множин.

Для розв'язку задачі про рюкзак методом динамічного програмування необхідно спочатку

сформулювати розв'язок початкової задачі через розв'язання задач меншої розмірності. Для цього позначимо через S — максимальну вартість предметів, які можна розмістити у рюкзаку. Припустимо, що останній предмет № n належить S , тоді $S - \{v_n\}$ — оптимальний розв'язок для перших $n - 1$ предметів і місткості рюкзака $(W - w_n)$.

Тепер можна сформулювати рекурсивне правило для отримання розв'язку задачі. Позначимо через $V(i, x)$ найбільшу вартість, таку що (1) розглянуті тільки перші i предметів та (2) загальний розмір предметів не перевищує x . Тоді:

$$V(i, x) = \max \{ V(i - 1, x), V(i - 1, x - w_i) + v_i \} \quad (*)$$

Примітка: у випадку коли $w_i > x$, $V(i, x) = V(i - 1, x)$.

Тож, алгоритм динамічного програмування повинен перебрати підзадачі всіх можливих розмірностей, які визначаються двома змінними:

- i — кількість предметів (1, ..., n)
- x — місткість рюкзака (1, ..., W)

Швидкість такого алгоритму становитиме $O(nW)$. Таким чином його не можна в повній мірі віднести до поліноміальних алгоритмів і через це він має характеристики псевдополіноміального.

Зауваження до програмної реалізації алгоритму

Як зазначено вище, час роботи запропонованого алгоритму динамічного програмування для задачі про рюкзак залежить від кількості предметів n та об'єму рюкзака W . Це слід враховувати під час програмної реалізації алгоритму. Адже за умови великих об'ємів рюкзака та порівняно невеликих об'ємів предметів рекурсивний перехід, який визначається формулою (*), буде займати надто багато часу. Тому під час реалізації даного алгоритму слід уникати виконання зайвої роботи.

Формат вхідних/вихідних даних

Розроблена програма повинна зчитувати вхідні дані з файлу заданого формату та записувати дані у файл заданого формату. У вхідному файлі зберігається інформація про об'єм рюкзака, кількість предметів та характеристики всіх предметів (вага w_i та вартість v_i).

Вхідний файл представляє собою текстовий файл. Перший рядок файлу містить два числа: W (загальний об'єм рюкзака) та n (кількість предметів). Далі йде n

рядків, кожен з яких містить пару чисел: вартість v_i та вага w_i предмету i . Вихідний файл є текстовим, в якому записана оптимальна сумарна вартість предметів, що були розміщені у рюкзаку. Нижче наведені приклади вхідного та вихідного файлу для п'яти предметів.

Вхідний файл	Вихідний файл
10 5 7 5 8 4 9 3 10 2 1 10	27

```
using System;
using System.IO;
using System.Reflection.Emit;

namespace Lab9
{
    class BackpackProblem
    {
        static int Backpack(int[] weights, int[] values, int n, int W)
        {
            int[,] K = new int[n + 1, W + 1];

            for (int i = 0; i <= n; i++)
            {
                for (int w = 0; w <= W; w++)
                {
                    if (i == 0 || w == 0)
                        K[i, w] = 0;
                    else if (weights[i - 1] <= w)
                        K[i, w] = Math.Max(values[i - 1] + K[i - 1, w - weights[i - 1]], K[i - 1, w]);
                    else
                        K[i, w] = K[i - 1, w];
                }
            }

            return K[n, W];
        }

        static void SolveBackpackProblem(string inputFilePath, string outputFilePath)
        {
            string[] lines = File.ReadAllLines(inputFilePath);
            string[] firstLine = lines[0].Split(' ');
            int W = int.Parse(firstLine[0]);
            int n = int.Parse(firstLine[1]);
        }
    }
}
```

```

        int[] values = new int[n];
        int[] weights = new int[n];
        for (int i = 0; i < n; i++)
        {
            string[] data = lines[i + 1].Split(' ');
            values[i] = int.Parse(data[0]);
            weights[i] = int.Parse(data[1]);
        }

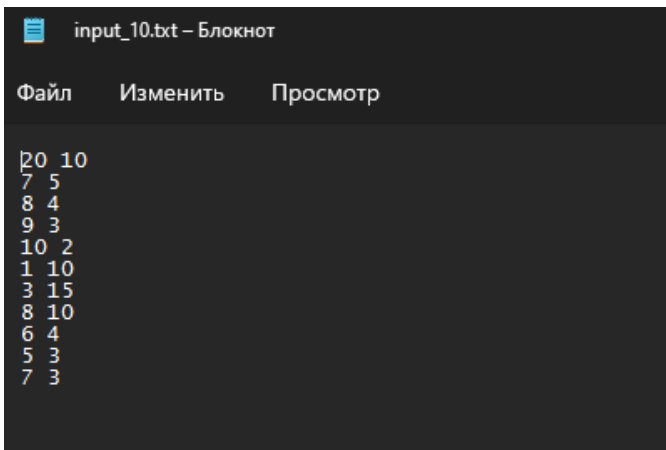
        int result = Backpack(weights, values, n, W);

        File.WriteAllText(outputFilePath, result.ToString());
    }

    public static void Main()
    {
        string inputFilePath = "D://LAB_TA//Lab9//Lab9//test_data//input_5.txt";
        string outputFilePath =
"D://LAB_TA//Lab9//Lab9//test_data//test_output_5.txt";
        SolveBackpackProblem(inputFilePath, outputFilePath);
    }
}

```

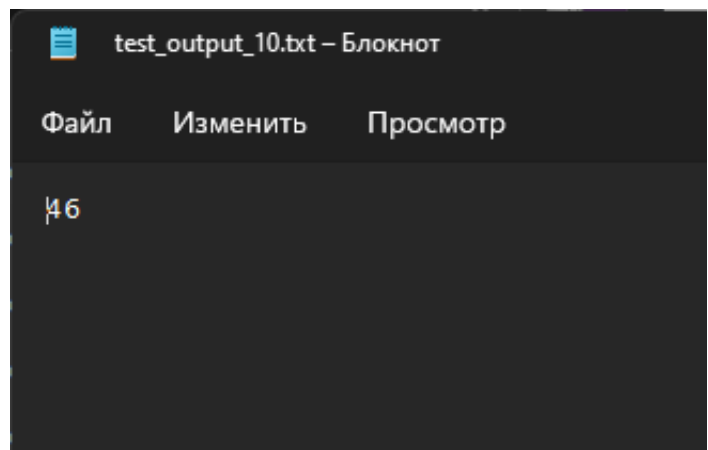
Код 1.1 – Program.cs



```

20 10
7 5
8 4
9 3
10 2
1 10
3 15
8 10
6 4
5 3
7 3

```



```

46

```

Рисунок 1.2 – Приклади вихідних файлів

Висновок: У даній лабораторній роботі було успішно розв’язано задачу визначення послідовності медіан для заданого вхідного масиву, використовуючи структури даних піраміди. Запропонований алгоритм ефективно опрацьовує нові елементи вхідного масиву в режимі реального часу, забезпечуючи обчислення медіан з часовою складністю $O(\log(i))$ на кожному кроці. Таким чином, запропонований алгоритм є оптимальним рішенням для задачі визначення медіан у режимі реального часу і може бути корисним для широкого кола практичних застосувань, де необхідно швидко обробляти потоки даних і динамічно обчислювати їхні медіани.