

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

Звіт

з лабораторної роботи №9 з дисципліни
«Алгоритми та структури даних 2. Структури даних»

«Динамічне програмування»
Варіант 3

Виконав студент ІП-15, Борисик Владислав Тарасович
(шифр, прізвище, ім'я, по батькові)

Перевірів Соколовський Владислав Володимирович
(прізвище, ім'я, по батькові)

Мета лабораторної роботи

Мета роботи – вивчити основні підходи реалізації динамічного алгоритму на прикладі задачі про рюкзак.

Завдання

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.

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 рюкзак.

Псевдокод алгоритму

```
Функція find_max_items_price(backpack: Backpack, items: list[Item]):
    backpack_capacity := backpack.get_capacity()
    items_amount := len(items)

    # таблиця із значеннями розв'язків
    # рядки - ціна об'єктів
    # стовпці - межі ваги об'єктів
    table := generate_empty_table(backpack_capacity, items_amount)

    # ітеруємось по стовпцям
    Для i від 1 до items_amount + 1 з кроком 1:
        # ітеруємось по елементам
        Для j від 1 до backpack_capacity + 1 з кроком 1:
            # перевіряємо що б вага об'єкта i була меншою за загальну вагу допустиму
            для комірки j
                item_weight := items[i - 1].get_weight()

                Якщо item_weight <= j:
                    # перший елемент (обчислюється за формулою)
                    value1 := items[i - 1].get_price() + table[i - 1][j - items[i -
1].get_weight()]]
                    # загальний розмір предметів (обчислюється за формулою)
                    value2 := table[i - 1][j]

                    # обираємо максимальний з двох варіантів
                    table[i][j] := max(value1, value2)
                # якщо вага об'єкта i перевищує допустиму вагу j.
                Інакше:
                    table[i][j] := table[i - 1][j]

            Все якщо

        Все повторити

    Все повторити

    # повертаємо останню клітинку таблиці
    # (вона має найбільшу сумарну вартість предметів, які не перевищують розмірність
рюкзака)
    Повернути table[items_amount][backpack_capacity]
```

Програмна реалізація алгоритму на мові Python

```
def find_max_items_price(backpack: Backpack, items: list[Item]):
    backpack_capacity = backpack.get_capacity()
    items_amount = len(items)

    # таблиця із значеннями розв'язків
    # рядки - ціна об'єктів
    # стовпці - межі ваги об'єктів
    table = generate_empty_table(backpack_capacity, items_amount)

    # ітеруємось по стовпцям
    for i in range(1, items_amount + 1):
        # ітеруємось по елементам
        for j in range(1, backpack_capacity + 1):
            # перевіряємо що б вага об'єкта i була меншою за загальну вагу допустиму
            # для комірки j
            item_weight = items[i - 1].get_weight()

            if item_weight <= j:
                # перший елемент (обчислюється за формулою)
                value1 = items[i - 1].get_price() + table[i - 1][j - items[i - 1].get_weight()]
                # загальний розмір предметів (обчислюється за формулою)
                value2 = table[i - 1][j]

                # обираємо максимальний з двох варіантів
                table[i][j] = max(value1, value2)
            # якщо вага об'єкта i перевищує допустиму вагу j.
            else:
                table[i][j] = table[i - 1][j]

    # повертаємо останню клітинку таблиці
    # (вона має найбільшу сумарну вартість предметів, які не перевищують розмірність
    # рюкзака)
    return table[items_amount][backpack_capacity]
```

Вихідний код

main.py:

```
from functions import *

file_content = read_input_file("input.txt")

backpack = parse_backpack(file_content)
items = parse_items(file_content)

max_items_price = find_max_items_price(backpack, items)

write_output_file("output.txt", max_items_price)
```

classes.py:

```
class Backpack:
    def __init__(self, capacity):
        self.__capacity = capacity

    def get_capacity(self):
        return self.__capacity

class Item:
    def __init__(self, price, weight):
        self.__price = price
        self.__weight = weight

    def get_price(self):
        return self.__price

    def get_weight(self):
        return self.__weight
```

funtions.py:

```
from classes import *

def read_input_file(file_name: str):
    file = open(file_name)

    text = file.read()

    file.close()
    return text

def parse_backpack(text: str):
    first_line = text.split('\n')[0]
    backpack_capacity = int(first_line.split(' ')[0])

    backpack = Backpack(backpack_capacity)

    return backpack

def parse_items(text: str):
    first_line = text.split('\n')[0]
    amount = int(first_line.split(' ')[1])
    items_content = text.split('\n')[1:]

    items = []

    for i in range(amount):
        price = int(items_content[i].split(' ')[0])
        weight = int(items_content[i].split(' ')[1])

        item = Item(price, weight)

        items.append(item)

    return items

def generate_items_weights(items: list[Item]):
    weights = [item.get_weight() for item in items]

    return weights

def generate_items_prices(items: list[Item]):
    prices = [item.get_price() for item in items]

    return prices

def generate_empty_table(backpack_capacity: int, items_amount: int):
```

```

table = []

for i in range(items_amount + 1):
    row = []

    for j in range(backpack_capacity + 1):
        row.append(0)

    table.append(row)

return table

def find_max_items_price(backpack: Backpack, items: list[Item]):
    backpack_capacity = backpack.get_capacity()
    items_amount = len(items)

    # таблиця із значеннями розв'язків
    # рядки - ціна об'єктів
    # стовпці - межі ваги об'єктів
    table = generate_empty_table(backpack_capacity, items_amount)

    # ітеруємось по стовпцям
    for i in range(1, items_amount + 1):
        # ітеруємось по елементам
        for j in range(1, backpack_capacity + 1):
            # перевіряємо що б вага об'єкта i була меншою за загальну вагу допустиму
            для комірки j
            item_weight = items[i - 1].get_weight()

            if item_weight <= j:
                # перший елемент (обчислюється за формулою)
                value1 = items[i - 1].get_price() + table[i - 1][j - items[i -
1].get_weight()]]
                # загальний розмір предметів (обчислюється за формулою)
                value2 = table[i - 1][j]

                # обираємо максимальний з двох варіантів
                table[i][j] = max(value1, value2)
            # якщо вага об'єкта i перевищує допустиму вагу j.
            else:
                table[i][j] = table[i - 1][j]

    # повертаємо останню клітинку таблиці
    # (вона має найбільшу сумарну вартість предметів, які не перевищують розмірність
    рюкзака)
    return table[items_amount][backpack_capacity]

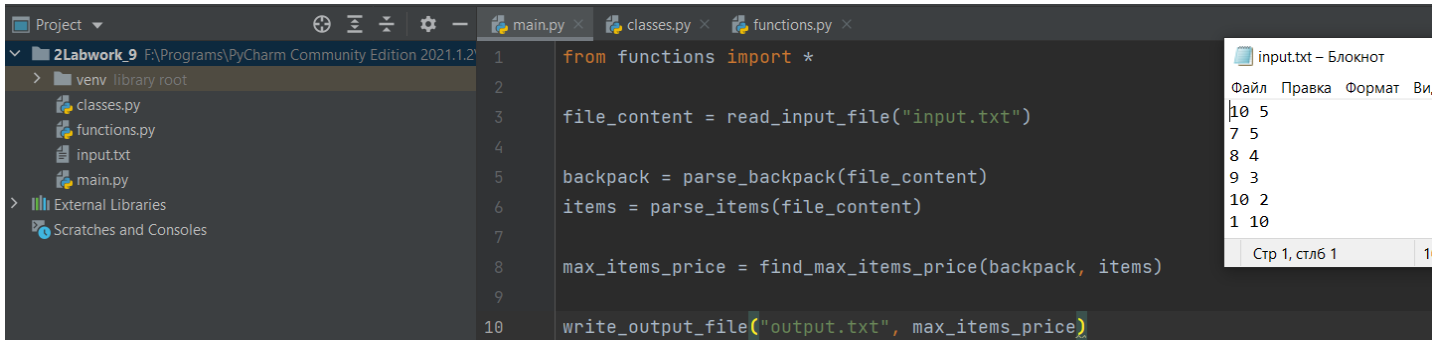
def write_output_file(file_name: str, max_items_price: int):
    text = str(max_items_price)

    file = open(file_name, 'wt')
    file.write(text)

```


Приклади роботи

Для тестування, оберемо файл, що вказаний в прикладі до задачі. Таким чином, ми зможемо порівняти відповіді і зробити висновок про правильність роботи програми.



The screenshot shows the PyCharm IDE interface. The main editor window displays the `main.py` file with the following Python code:

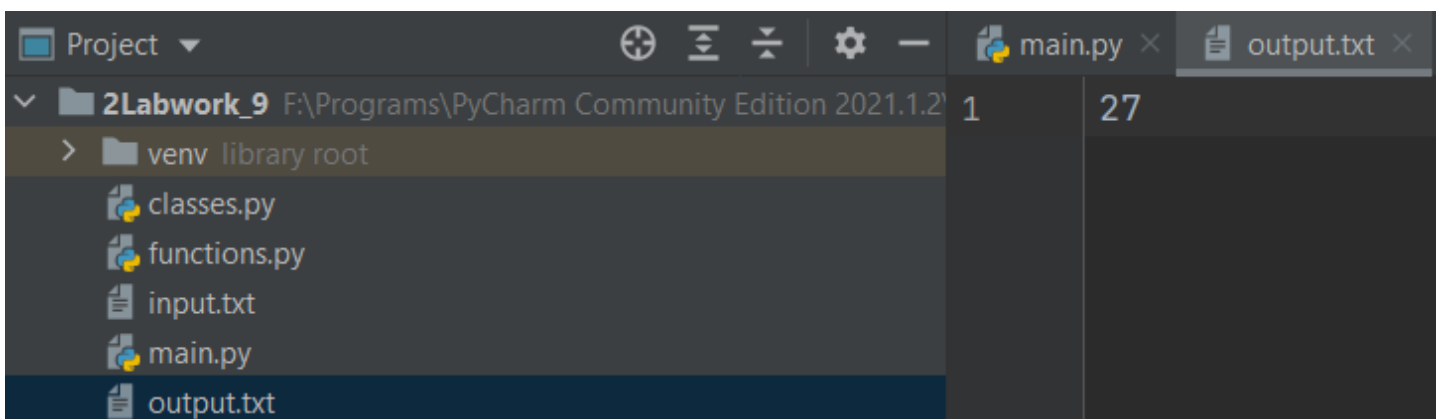
```
1 from functions import *
2
3 file_content = read_input_file("input.txt")
4
5 backpack = parse_backpack(file_content)
6 items = parse_items(file_content)
7
8 max_items_price = find_max_items_price(backpack, items)
9
10 write_output_file("output.txt", max_items_price)
```

On the right side, a Notepad window titled "input.txt – Блокнот" is open, displaying the input data:

```
10 5
7 5
8 4
9 3
10 2
1 10
```

The status bar at the bottom right indicates "Стр 1, стлб 1" (Line 1, Column 1).

Після виконання програми ми отримали ось такий результат:



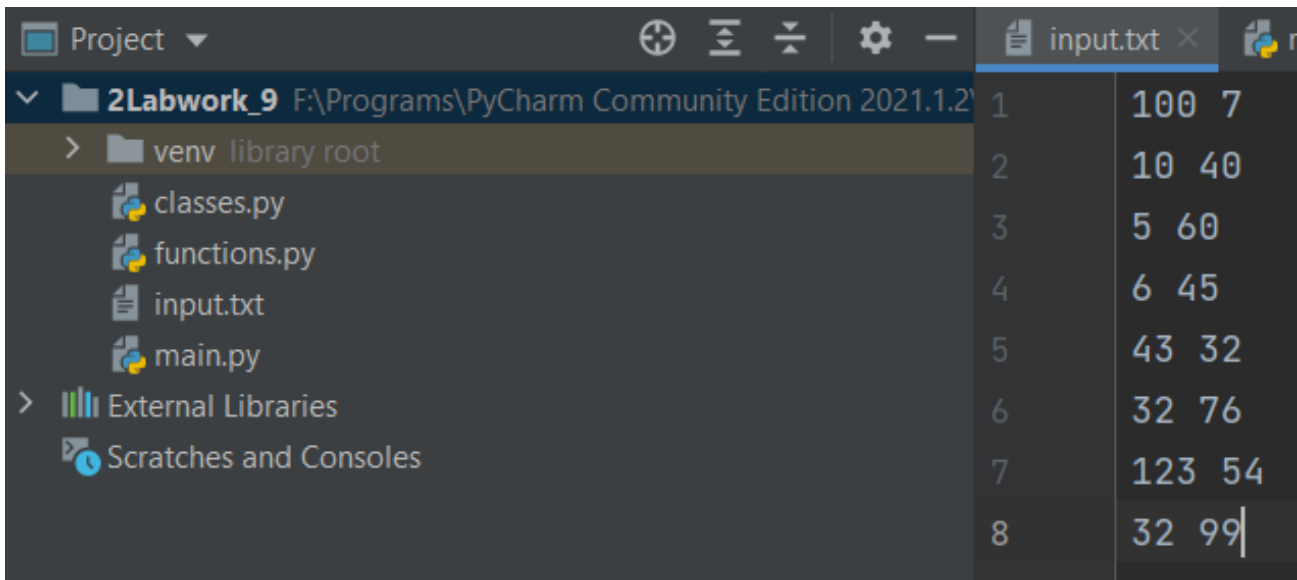
The screenshot shows the PyCharm IDE interface. The main editor window displays the `output.txt` file with the following content:

```
1 27
```

The status bar at the bottom right indicates "Стр 1, стлб 1" (Line 1, Column 1).

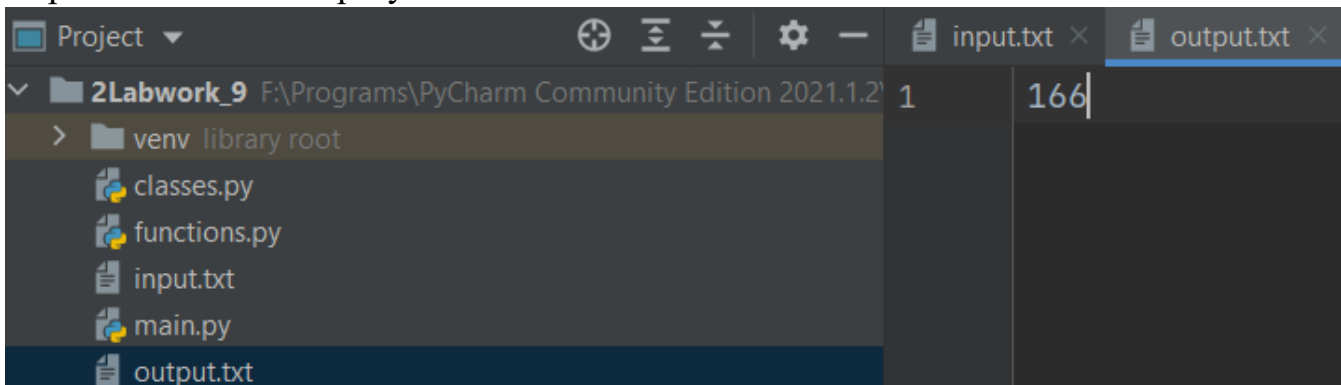
Бачимо, що на цьому прикладі програма спрацювала правильно, так як відповіді збіглись.

Для другого прикладу, візьмемо файл такого формату:



1	100	7
2	10	40
3	5	60
4	6	45
5	43	32
6	32	76
7	123	54
8	32	99

Отримали ось такий результат:



1	166
---	-----

Перевіримо його вручну:

Щоб отримати максимальну сумарну вартість предметів, так щоб вони не перевищували місткість рюкзака, потрібно взяти предмет на сьомому рядку з вагою 54 (його вартість 123) і предмет на п'ятому рядку з вагою 32 (його вартість 43).

Їх сумарна вартість буде дорівнювати 166, а вага 86. Місткість рюкзака - 100. Отже, вага предметів менша за місткість рюкзака

Бачимо, що усі дані збіглись.

На двох прикладах програма коректно знайшла максимальну сумарну вартість предметів, так щоб вони не перевищували місткість рюкзака, отже, вона працює правильно.

Висновок

Під час виконання цієї лабораторної роботи я вивчив основні підходи реалізації динамічного алгоритму на прикладі задачі про рюкзак.

В результаті виконання лабораторної роботи я отримав програму, яка знаходить максимальну сумарну вартість предметів, так щоб вони не перевищували місткість рюкзака за допомогою динамічного алгоритму.