Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота № 10

з дисципліни «Проектування та аналіз обчислювальних алгоритмів»

на тему

«Динамічне програмування»

Виконав:

студент гр. ІС-91

Хмелiнiн Андрiй

Викладач:

[ст.вик. Халус О. А.](http://rozklad.kpi.ua/Schedules/ViewSchedule.aspx?v=318f7459-ddf0-47ee-b882-5650e756cf2a)

Київ – 2020

Постановка задачi

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, wi та вартість vi. 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 – {n} — оптимальний розв'язок для перших n – 1 предметів і місткості рюкзака (W – wn).

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 – wi) + vi } (\*)

Примітка: у випадку коли wi > 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 характеристику псевдополіноміального.

Код програми:

def main():

name\_num = "5" #-------------------------------- input / output name number ------------------------------------------

def Knapsack(weight, price, MaxWeight, number):

final = [0] \* (MaxWeight+1) # создаем масив для хранения стоимости предметов в рюкзаке

for i in range(number): # цикл, который итерируеться по всем предметам

for j in range(MaxWeight,weight[i]-1, -1): # цикл, который итерируеться из обьема рюкзака по весу i-того предмета

next\_v=price[i] + final[j-weight[i]] # записываем в final[j] наибольшую стоимость при условии, что расмотренны только первые "i" предметов и общий размер предметов не превышает MaxWeight

if next\_v>final[j]:

final[j]=next\_v # если новая стоимость больше чем предыдущая, записываем ее в массив

return final[MaxWeight] # возвращаем максимально возможную стоимость предметов в рюкзаке

with open("Inputs/input\_"+name\_num+".txt", 'r') as f: # Input -----------------------------------------------------

MaxWeight, number = [int(x) for x in f.readline().split()]

value = []

weight =[]

for i in range(number):

temp = [int(i) for i in f.readline().split()]

value.append(temp[0])

weight.append(temp[1])

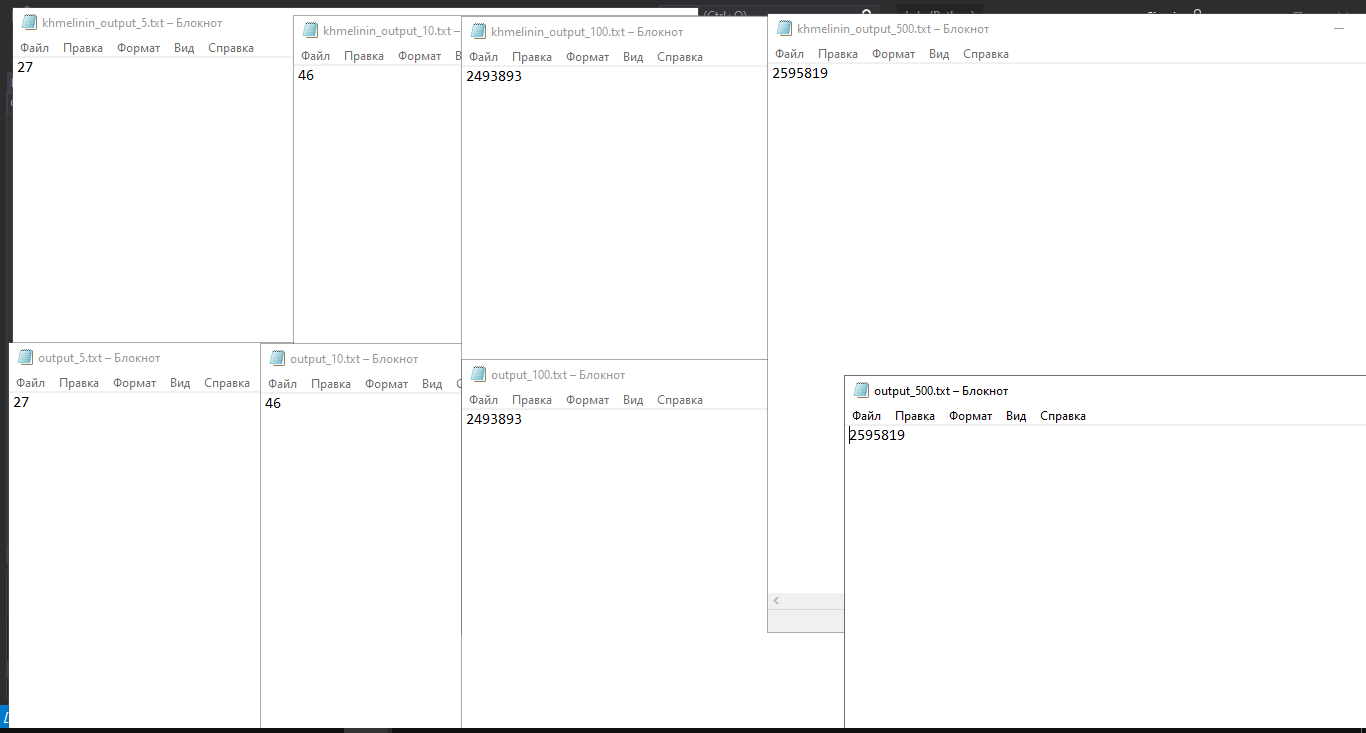
asd = Knapsack(weight, value, MaxWeight, number)

with open("Outputs/khmelinin\_output\_"+name\_num+".txt", 'w') as fi: # Output ---------------------------------------

fi.write(str(asd))

if \_\_name\_\_ == '\_\_main\_\_':

main()



Висновок:

Я вирiшив задачу про рюкзак, яка є одним з яскравих прикладів потужності підходу динамiчного програмування. Як було вже зазначено у завданнi, приклад iз об’ємом рюкзака 500 програма обчислює за 2-3 хвилини.