# РЕФЕРАТ

Пояснювальна записка до курсової роботи: 39с., 13 рисунків, 7таблиць, 6 джерел.

**Предмет дослідження**: розв’язання задачі про суму підмножини.

**Мета роботи**: дослідження шляхів розв’язання задачі про суму підмножини основі методу динамічного програмування та методу повного перебору.

Виконано порівняльний аналіз цих методів розв’язання.

Виконана програмна реалізація на мові програмування Python .

ДИНАМІЧНЕ ПРОГРАМУВАННЯ, АЛГОРИТМ ПОВНОГО ПЕРЕБОРУ, МЕТОД ГРУБОЇ СИЛИ, МНОЖИНА, ПІДМНОЖИНА.

ЗМІСТ

[РЕФЕРАТ 7](#_Toc453578520)

[ВСТУП 10](#_Toc453578521)

[1 ПОСТАНОВКА ЗАДАЧІ 11](#_Toc453578522)

[1.1 Математична постановка задачі 11](#_Toc453578523)

[1.2 Змістовна постановка задачі 12](#_Toc453578524)

[2 ОПИС МЕТОДУ РОЗВЯЗАННЯ ЗАДАЧІ 14](#_Toc453578525)

[2.1 Огляд методів розв’язання задачі 14](#_Toc453578526)

[2.1.1 Метод повного перебору 14](#_Toc453578527)

[2.1.2 Метод динамічного програмування 14](#_Toc453578528)

[2.2 Теоретичні положення методів пошуку підмножин в множині 15](#_Toc453578529)

[2.2.1 Метод повного перебору 15](#_Toc453578530)

[2.2.2 Метод динамічного програмування 16](#_Toc453578531)

[2.3 Схеми алгоритмів 17](#_Toc453578532)

[2.3.1 Метод повного перебору 17](#_Toc453578533)

[2.3.2 Метод динамічного програмування 17](#_Toc453578534)

[2.4 Оцінка складності алгоритму 18](#_Toc453578535)

[2.4.1 Метод повного перебору 18](#_Toc453578536)

[2.4.2 Метод динамічного програмування 18](#_Toc453578537)

[2.5 Результати експериментів 19](#_Toc453578538)

[3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 22](#_Toc453578539)

[3.1 Вхідна дані 22](#_Toc453578540)

[3.2 Вихідні дані 23](#_Toc453578541)

[3.3 Опис архітектури програмного продукту 24](#_Toc453578542)

[3.4 Інструкція користувача 25](#_Toc453578543)

[4 ПРИКЛАДИ РОЗВ’ЯЗАННЯ ЗАДАЧ 29](#_Toc453578544)

[4.1 Метод повного перебору 29](#_Toc453578545)

[4.1.1 Розв’язання задачі вручну 29](#_Toc453578546)

[4.1.2 Розв’язання задачі за допомогою розробленого програмного продукту 30](#_Toc453578547)

[4.2 Метод динамічного програмування 30](#_Toc453578548)

[4.2.1 Розв’язання вручну 30](#_Toc453578549)

[4.2.2 Розв’язання задачі за допомогою розробленого програмного продукту 31](#_Toc453578550)

[ВИСНОВКИ 32](#_Toc453578551)

[ПЕРЕЛІК ПОСИЛАНЬ 33](#_Toc453578552)

[Додаток А 34](#_Toc453578553)

# ВСТУП

В нашому житті потрібно завжди потрібно приймати якісь рішення. В будь-яких ситуаціях в побутових, професійних нам це необхідно. Зазвичай ми обираємо, якийсь варіант розвитку подій навіть не задумуючись.

Рішенням повинний бути найкращий варіант за певних умов. Щоб обрати найкращий нам потрібно обрати, якісь показники ефективності. Показники повинні бути, кількісними, щоб їх можна було порівняти і на основі цього порівняння прийняти рішення.

Ми досить часто стикаємось в житті з задачами розміщення предметів з різною вагою в замкнутому просторі, або з необхідністю обрання підмножини з певним результатом з системи втрати/надбання. І саме такі повсякденні задачі можна вирішити за допомогою обрання, якогось варіанту на основі кількісних показників.

В даній курсовій роботі пропонується аналіз способів отримання варіантів , які є підмножиною певної множини.

# ПОСТАНОВКА ЗАДАЧІ

## Математична постановка задачі

Вхідними даними до задачі є множина різноманітних чисел з певним значенням . Значення позначається Ci , i = 1 … n, де n – кількість чисел в множині, а і – поточне число. Потрібно визначити, чи є така підмножина даної множини, сума якої буде рівна певному значенню і якщо є то знайти її. При цьому значення конкретного числа може бути як більше заданої значення суми, так і менше.

Нехай хj – належність конкретного числа до шуканої підмножини. Дана умова наведена в формулі (1.1).

|  |  |  |
| --- | --- | --- |
|  |  | (1.1) |

Нехай, S – сума , яку нам потрібно знайти.

Тоді цільова функція – мінімізувати різницю між сумою підмножини та шуканою сумою. Вона наведена у формулі (1.2).

|  |  |  |
| --- | --- | --- |
|  |  | (1.2) |

Х належить до булевих значень, формула (1.3):

|  |  |  |
| --- | --- | --- |
|  |  | (1.3) |
| Встановимо обмеження на цілочисельність, формули (1.4, 1.5) : | | |
|  |  | (1.4) |
|  |  | (1.5) |

## Змістовна постановка задачі

**Задача 1. Опис предметного середовища:** Важкоатлет готується до змагань і йому потрібно залишатися в своїй ваговій категорії. Для цього йому потрібно отримувати в день 3900 калорій.

Він їх отримує з їжею , а за допомогою вправ спалює .

Дані про отримані та спалені калорії наведено в таблиці 1.1.

**Таблиця 1.1 – Отримані/спалені калорії:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Отримані калорії** | **100** | **220** | **330** | **600** | **335** | **500** | **125** | **605** | **235** | **665** | **223** | **660** | **443** | **120** |
| **Спалені калорії** | **400** | **242** | **523** | **442** | **111** | **454** | **543** | **123** | **553** | **332** | **123** | **121** | **524** | **353** |

**Математична модель:**

Нехай хj – належність конкретного значення до шуканої підмножини. Дана умова наведена в формулі (1.1).

|  |  |  |
| --- | --- | --- |
|  |  | (1.6) |

Нехай S – шукана кількість калорії.

S = 3900

Цільова функція – мінімізувати різницю між сумою підмножини . Вона наведена у формулі (1.7).

|  |  |  |
| --- | --- | --- |
|  |  | (1.7) |

Встановимо обмеження:

Х належить до булевих значень, формула (1.8):

|  |  |  |
| --- | --- | --- |
|  |  | (1.8) |
| Обмеження на цілочисельність, формули (1.9, 1.10) : | | |
|  |  | (1.9) |
|  |  | (1.10) |
|  |  |  |

**Задача 2. Опис предметного середовища:** Фірма займається доставкою товару. Кожен автомобіль, згідно технічного паспорту має обмеження на максимально допустиму завантаженість 2000 т. Потрібно завантажити автомобіль знаючи його допустиму завантаженість та вагу товарів.

Дані вагу товарів наведено в таблиці 1.2.

**Таблиця 1.2 – Ваги товарів:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ товару** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** |
| **Вага** | **40** | **300** | **200** | **222** | **444** | **231** | **412** | **567** | **839** | **123** | **224** | **554** | **290** | **490** |

**Математична модель: Математична модель відповідає моделі минулої задачі , окрім обмеження на невід’ємність:**

|  |  |  |
| --- | --- | --- |
|  |  | (1.11) |

**Задача 3. Опис предметного середовища:** Підприємство надає кошти – 10000грн робітникам на різноманітні курси, які робітники обирають . Потрібно знайти варіанти коли після оплати всіх курсів залишається сума в 3000 грн.

Дані про вартість курсів наведено в таблиці 1.3.

**Таблиця 1.3 – Вартість курсів:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№**  **Курсу** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** |
| **Вартість** | **401** | **411** | **542** | **522** | **1112** | **800** | **900** | **940** | **930** | **930** | **231** | **590** | **220** | **423** |

**Математична модель: Математична модель відповідає моделі минулої задачі , окрім обмеження на невід’ємність:**

|  |  |  |
| --- | --- | --- |
|  |  | (1.11) |

# ОПИС МЕТОДУ РОЗВЯЗАННЯ ЗАДАЧІ

## Огляд методів розв’язання задачі

До методів пошуку підмножин в множині належать : метод повного перебору та метод динамічного програмування.

## Метод повного перебору

Метод повного перебору знаходить всі можливі підмножини даної множини.

Після закінчення роботи цього алгоритму маємо на виході підмножини, а саме підмножини, які ми шукали за сумою елементів підмножини.

Ідея використання даного методу полягає у наступному: алгоритм перебирає всі можливі підмножини з одним елементом , потім з двома і так доки не будуть всі елементи початкової множини в підмножині, при цьому перевіряється чи сума елементів дорівнює шуканій. Так будуть гарантовано знайдені всі можливі підмножини початкової множини, сума елементів, яких рівна шуканій сумі.

## Метод динамічного програмування

Метод динамічного програмування знаходить єдину підмножину даної множини.

Після закінчення роботи цього алгоритму маємо на виході підмножину, а саме підмножину, яку ми шукали за сумою елементів підмножини.

Ідея використання даного методу полягає у наступному:

* Знаходимо суму від’ємних та невід’ємних елементів.
* Будуємо таблицю з булевими значеннями. Кількість рядків дорівнює кількості елементів множини, значення рядків є значеннями множини відповідно індексу . Значення стовпців лежать від значення суми від’ємних елементів до суми додатних елементів. Заповнюємо дану таблицю значеннями True/False.
* Проходимо таблицю для отримання розв’язку.

Як наслідок цих дій ми отримуємо необхідну нам підмножину.

## Теоретичні положення методів пошуку підмножин в множині

Задача про суму підмножин є важливою задачею в теорії складності алгоритмів та криптографії. Завдання полягає в знаходженні (хоча б однієї) непорожньої підмножини деякого набору чисел, щоб сума чисел цієї підмножини дорівнювала нулю. Наприклад, нехай задано множина {-7, -3, -2, 5, 8}, тоді підмножина {-3, -2, 5} дає в сумі нуль. Дана задача відноситься до NP-повних.

Еквівалентною є задача знаходження підмножини, сума елементів якої дорівнює деякому заданому числу S. Завдання про суму підмножин також можна розглядати, як певний спеціальний випадок завдання про ранці .

Що ж таке підмножина? Якщо кожен елемент однієї множини A є елементом множини B, то кажуть, що перша множина A є підмножиною множини.

## Метод повного перебору

Це найпростіший алгоритм. Він переглядає всі підмножини і для кожної з них перевіряє, чи є сума елементів підмножини прийнятною.

Повний перебір або метод грубої сили (англ. Brute force) - метод вирішення математичних задач. Відноситься до класу методів пошуку рішення виключенням деяких варіантів . Складність повного перебору залежить від кількості всіх можливих рішень задачі. Якщо простір рішень дуже великий, то повний перебір може не дати результатів протягом декількох років або навіть століть.

Будь-яка задача з класу NP може бути вирішена повним перебором. При цьому, навіть якщо обчислення цільової функції від кожного конкретного можливого рішення задачі може бути здійснено за поліноміальний час, в залежності від кількості всіх можливих рішень повний перебір може вимагати експоненціальний час.

Найчастіше метод грубої сили використовується в криптографії. Криптографічні атаки, засновані на методі повного перебору, є найбільш універсальними, але і найдовшими.

У теорії алгоритмів класом NP (від англ. Non-deterministic polynomial) називають множину задач, вирішення яких можна перевірити на машині Тьюринга за час, що не перевищує поліном від розміру вхідних даних, при наявності деяких додаткових відомостей (так званого сертифікату рішення).

Час роботи даного алгоритму для нашої задачі оцінюється, як O (2NN), оскільки є 2N підмножин, а для перевірки кожної підмножини необхідно скласти не більше N елементів.

## Метод динамічного програмування

Динамічне програмування (планування) – розділ прикладної математики, що використовується при математичному аналізі багатокрокових процесів прийняття рішень. Багатокроковий процес прийняття рішень - діяльність, при якій приймаються послідовні рішення, спрямовані на досягнення певної мети. Багатокрокові процеси прийняття рішень зустрічаються у всіляких ситуаціях: у задачах управління запасами, при приготуванні обіду, при керуванні космічним кораблем. ДП можна розглядати як єдину теорію завдяки ряду ідей і процедур, що використовуються при математичному аналізі таких задач.

Динамічне програмування широко застосовується при оптимальному плануванні управляючих процесів (під управляючими маються на увазі процеси, на хід яких ми можемо впливати). Планування таких процесів може бути інтерпретоване як багатокроковий процес прийняття рішень .

Виведемо рекурентне співвідношення :

Нехай :

S – шукана сума .

Ci - значення числа , i = 1 … n, де n – кількість чисел в множині.

хj – належність конкретного числа до шуканої підмножини. Дана умова наведена в формулі (2.1).

|  |  |  |
| --- | --- | --- |
|  |  | (2.1) |

w – сума всіх елементів.

Нехай d(i,w) максимальна сума ≤ w, підмножини взятої з 1…i елементів.

Заповнимо d(0,w) нулями.

Тоді змінюючи і від 1 до n, розрахуємо на кожному кроці d(i,w) для w від 0 до S за рекурентною формулою(2.2):

|  |  |  |
| --- | --- | --- |
|  |  | (2.2) |

Після виконання в d(n,S) буде знаходитися максимальна сума підмножини, що не перевищує задане значення.

## Схеми алгоритмів

## Метод повного перебору

Вхідними даними для алгоритму є множина чисел Сі де і=1…n та шукана сума S.

|  |  |
| --- | --- |
| **Крок 1.** | ВИЗНАЧИТИ кількість елементів, що повинні бути в підмножині j=i=1…n, СТВОРИТИ множини ANSW, FINISH\_ANSW для знайдених підмножин; |
| **Крок 2.** | Для всіх кожної кількості елементів j ВИКОНАТИ:   |  |  | | --- | --- | | **Крок 2.1** | ЗАПИСАТИ в масив А всі можливі варіанти підмножин з поточною кількістю j, підмножини складаються з чисел Сі . | | **Крок 2.2** | Для всіх елементів(підмножин, що зберігаються в А ) множини А:  ЗНАХОДИМО суму елемента (підмножина) ;  ЯКЩО сума дорівнює S:  ЗАПИСУЄМО елемент в ANSW;  ПРОДОВЖУЄМО Крок 2.2. | |
| **Крок 3.** | Для всіх елементів ANSW:  **Крок 3.1** ЯКЩО в FINISH\_ANSW немає елемента, що дорівнює поточному елементу ANSW:  ЗАПИСАТИ поточний елемент в FINISH\_ANSW |

## Метод динамічного програмування

Вхідними даними для алгоритму є множина чисел Сі де і=1…n та шукана сума S.

|  |  |
| --- | --- |
| **Крок 1.** | ВИЗНАЧИТИ суми негативних N та позитивних P елементів ; |
| **Крок 2.** | ЯКЩО S > P або N < P:  ПЕРЕРВАТИ виконання алгоритму. |
| **Крок 3.** | ЗНАЙТИ кількість рядків , що дорівнює n , та кількість стовпців m = P-N+1. |
| **Крок 4.** | ЗАПОВНИТИ таблицю Q розмірністю n x m значеннями False. |
| **Крок 5.** | ЗАПИСАТИ в таблицю Q значення True, на позицію n=0 a m = C0 . |
| **Крок 6.** | СТВОРИТИ для масив для розв’язку ANSW. |
| **Крок 7.** | Для всіх рядків i = 1…n:  **Крок 7.1** Для всіх стовпців j=N .. P+1:  ЗАПИСАТИ Qij = Q( i-1, j)АБО (Сі ==j) АБО Q( i-1,j-Ci) |
| **Крок 8.** | Для всіх рядків і = n ..0   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Крок 8.1** | Для всіх стовпців j=N .. P+1:   |  |  | | --- | --- | | **Крок 8.1.1** | ВИЗНАЧИМО j = S | | **Крок 8.1.2** | ЯКЩО Q(i , j) == True і Q(i-1, j) = F:  ЗАПИШЕМО Ci в ANSW;  ВИЗНАЧИМО наступні координати :Q(i-1,j-Ci);  ВИКОНАЄМО Крок **8.1.2 .** | |  | ІНАКШЕ ЯКЩО Q(i , j) == True і Q(i-1, j) = True:  ВИЗНАЧИМО наступні координати :Q(i-1,j) ;  ВИКОНАЄМО Крок **8.1.2 .** | | **Крок 8.1.3** | ЯКЩО Q(i , j) == True та і = 0:  ЗАПИШЕМО Ci в ANSW. | | |

## Оцінка складності алгоритму

## Метод повного перебору

Час роботи даного алгоритму для нашої задачі оцінюється, як O (2NN), оскільки є 2N підмножин, а для перевірки кожної підмножини необхідно скласти не більше N елементів.

## Метод динамічного програмування

Для кожного присвоєння значення Q в правій частині вже відомо, оскільки воно або занесено в таблицю попередніх значень , або оскільки Q (i - 1, j- Ci) = False, якщо j – Ci <A або j – Ci> B. Таким чином, повний час арифметичних операцій становить O (N (B - A)).

Алгоритм не рахується поліноміальних, оскільки B - A не є поліноміальним від розміру задачі, в якості якого виступає число біт в числах. Алгоритм поліноміальний від значень A і B, а вони експоненціально залежать від числа біт.

## Результати експериментів

На випробувані було проведено п'ять експериментів на множинах різної розмірності. Виміри часу виконувалися стандартної бібліотекою “time” мови програмування Python. Їх результати наведені у таблиці 2.1 . Для тестування використовувався комп’ютер з CPU Intel I5 2450M (2.5 Ghz x 2) та RAM 6Gb.

Випробування проводилися на множинах розмірностями:

1. N=10
2. N=20
3. N=30
4. N=45
5. N=70

**Таблиця 2.1 – Середній час виконання**

|  |  |  |
| --- | --- | --- |
| **Кількість елементів множини** | **Повний перебір** | **Динамічне програмування** |
| **Час** | **Час** |
| 10 | 0.00498 | 0:00:00.00500 |
| 20 | 00:00:04.72881 | 0:00:00.13510 |
| 30 | 00:01:36.466453 | 0:00:00.33322 |
| 45 | - | 0:00:00.80954 |
| 70 | - | 0:00:01.92477 |

З таблиці 2.1 бачимо, що через велику кількість операцій всередині методу повного перебору, час його роботи відрізняється від часу роботи динамічного програмування в рази. Це зумовлено великою кількістю можливих підмножин. Також було припинено дослідження для множини розмірністю більше 30 елементів для методу повного перебору, адже навіть 30 елементів перебиралися більше 1 години. На Рисунок 2.1 зображено графік залежності часу виконання від кількості елементів множини для методу Повного Перебору. На Рисунок 2.2 зображено аналогічний графік, але для методу Динамічного програмування . Як видно з результатів алгоритмів, метод динамічного програмування працює в рази швидше проте находить лише одну підмножину.

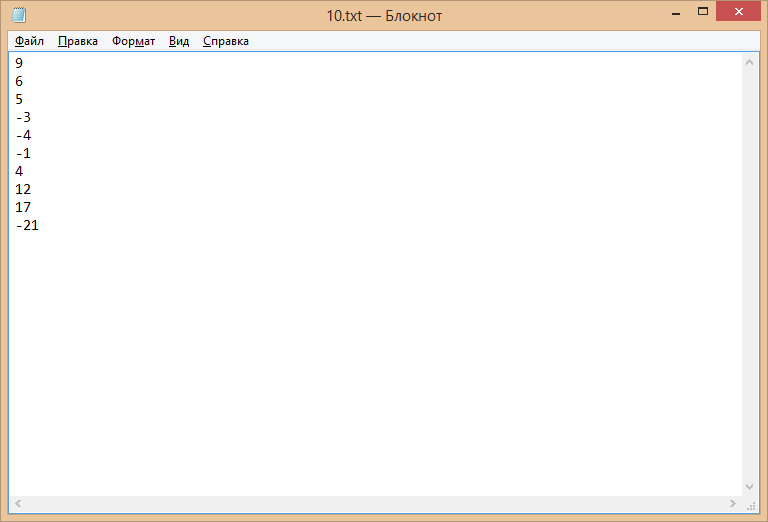
|  |
| --- |
|  |
| Рисунок 2.1 – Графік залежності часу від розміру масиву для методу повного перебору |

|  |
| --- |
|  |
| Рисунок 2.2 – Графік залежності часу від розміру масиву для методу динамічного програмування |

# ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

## Вхідна дані

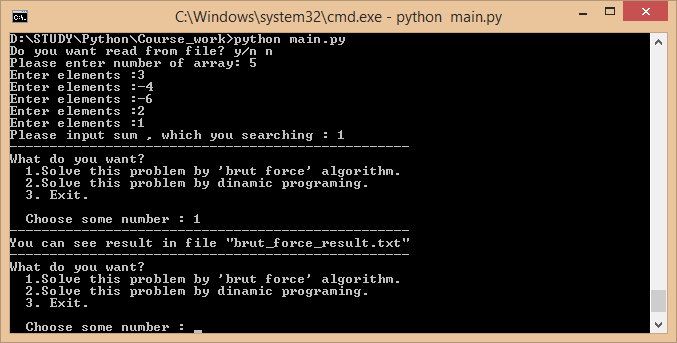
На вхід до програми подається текстовий файл (Рисунок 3.1) наступного зразку:



**Рисунок 3.1 – Зміст вхідного файлу 10.txt**

Кожний рядок містить число, що є елементом множини.

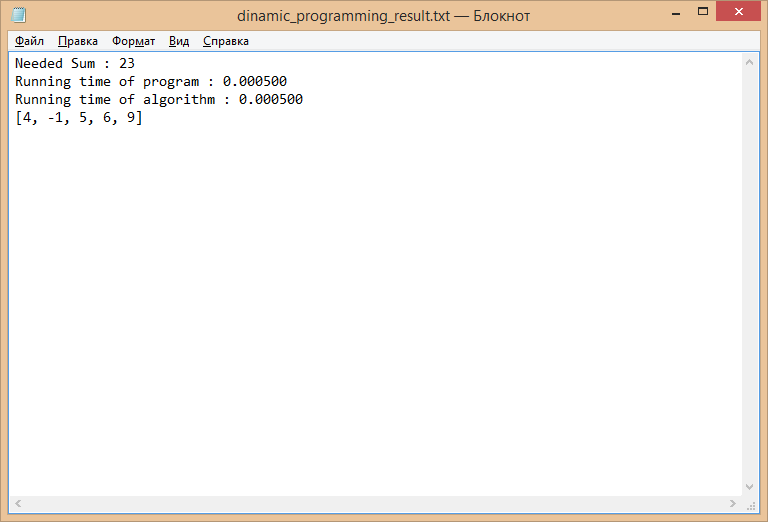
Або користувач має змогу сам ввести числа, що є елементами множини за допомогою консольного інтерфейсу. Результат вводу на Рисунок 3.2.



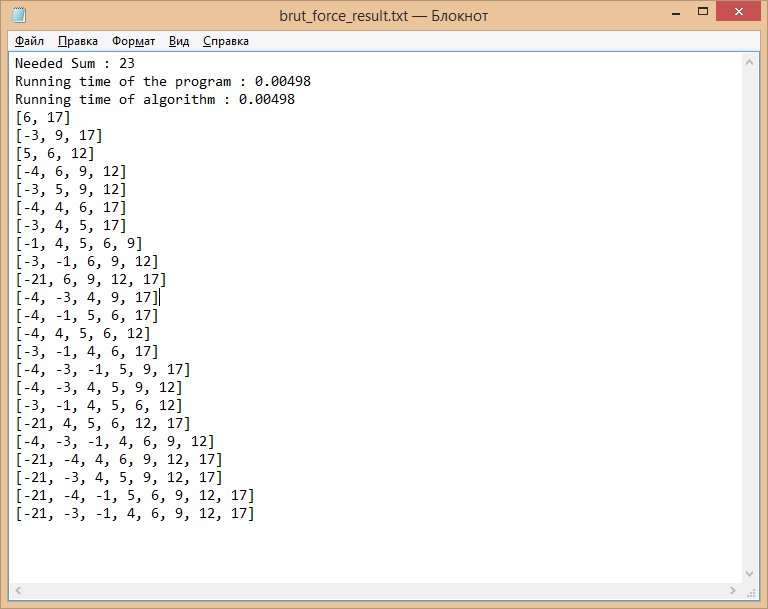
**Рисунок 3.2 – Ввід елементів вручну**

## Вихідні дані

Результатом обчислень є масив що є шуканою підмножиною. Виводиться шукана сума , час роботи програми та алгоритму , шукана підмножина (для методу динамічного програмування) (Рисунок 3.3). Для методу повного перебору виводиться все те ж , проте не одна шукана множина, а всі можливі множини (Рисунок 3.4).



**Риcунок 3.3 – Результат обчислень для DP**



**Риcунок 3.4 – Результат обчислень для BF**

## Опис архітектури програмного продукту

Для розв’язання задачі був створений програмний продукт. Для написання програми було обрано мову програмування python.

Програма включає в себе 2 модулі – по одному для кожного методу:

* dinamic\_programming (метод динамічного програмування);
* brut\_force (метод повного перебору);

Окрім власних методі, в програмі використовується метод time модулю time для знаходження часу роботи програми.

Вхідними даними для кожного алгоритму є список елементів та шукана сума.

У таблиці 3.1 наведений перелік та опис методів, що включає в себе модуль dynamic\_programming :

**Таблиця. 3.1 – Методи модулю** dynamic\_programming

|  |  |
| --- | --- |
| **Назва методу** | **Опис методу** |
| positive\_negative\_sums | Знаходження суми додатних та від’ємних елементів |
| find\_set | Знаходження підмножини за побудованої таблицею |
| subset\_sum | Побудова таблиці для методу динамічного програмування |

У таблиці 3.2 наведений перелік та опис методів, що включає в себе модуль brut\_force :

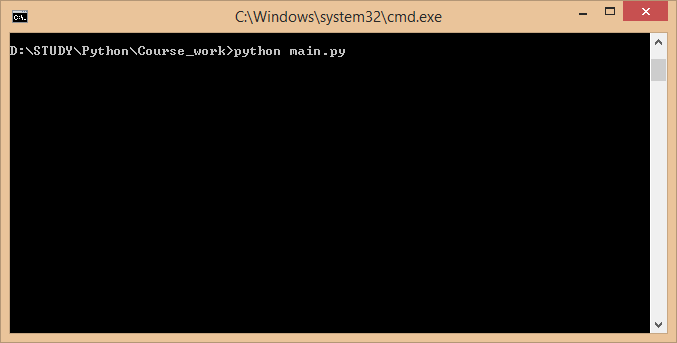
**Таблиця. 3.2 – Методи модулю** brut\_force

|  |  |
| --- | --- |
| **Назва методу** | **Опис методу** |
| combination | Перебір усіх можливих підмножин заданого розміру |
| find\_answer | Пошук підмножин, що дорівнюють шуканій сумі. |

## Інструкція користувача

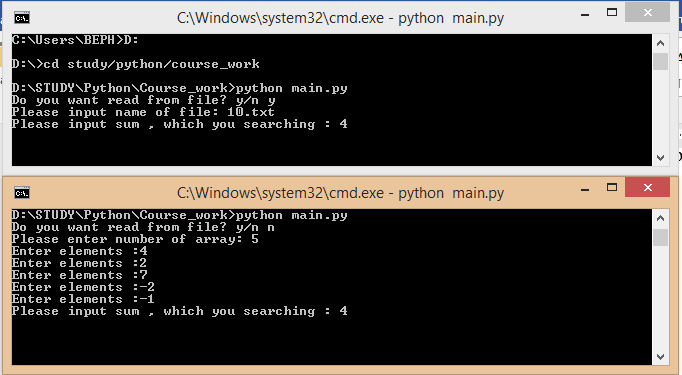
Для початку користування програмним продуктом, необхідно встановити інтерпретатор Python.

Після встановлення сфередовища для розробки, необхідно запустити програму через консоль, ввівши «python main.py» (Рисунок 3.4) натиснувши *«Open an App»* та вказавши шлях до файлу з лістингом програми*.*



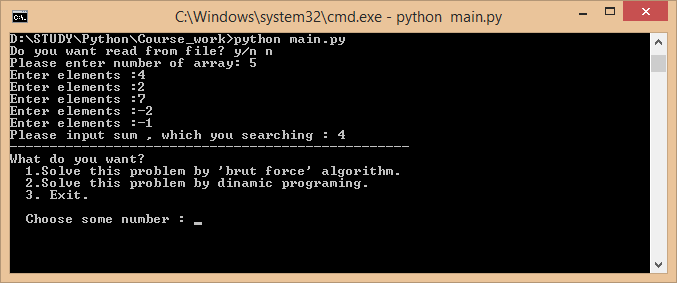
**Рисунок 3.4 – Запуск програми через командну строку**

Після цього програма буде запущена та буде очікувати вибір способу введення даних. Після зчитування з файлу або введення даних вручну потрібно буде ввести необхідну суму. (Рисунок 3.5).



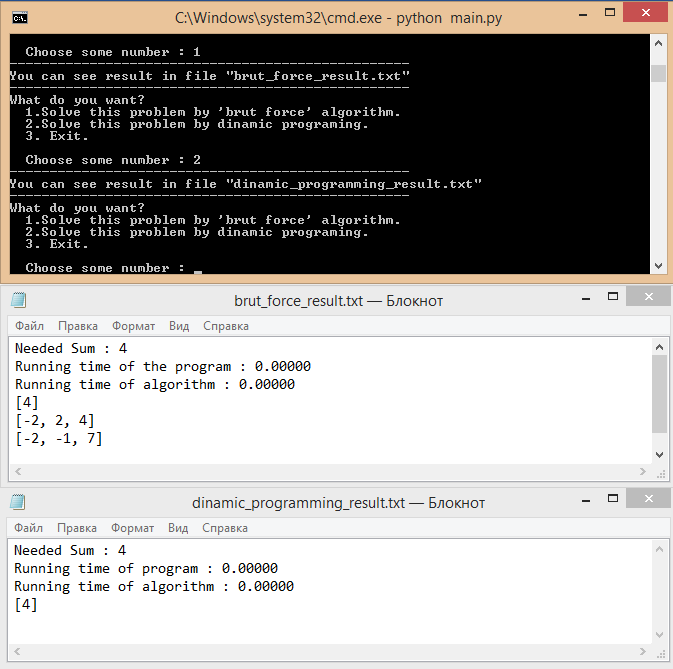
**Рисунок 3.5 – Програма очікує ввід даних**

Після отримання всіх необхідних даних , користувач мусить обрати спосіб розв’язання або вийти з командної строки, ввівши в консоль необхідну цифру (в програмі виводиться, яка цифра, якій дії відповідає) (Рисунок 3.6).



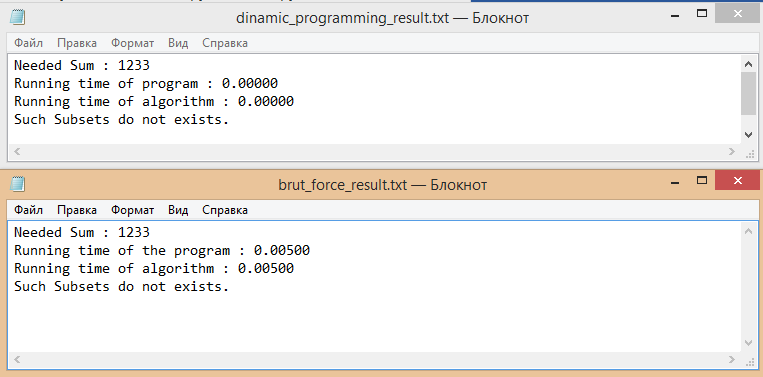
**Рисунок 3.6 – Програма пропонує обрати дію**

Після обрання, яким методом вирішувати програма знаходить розв’язок та інформує в якому файлі можна переглянути, при цьому також файл сам відкривається для перегляду. (Рисунок 3.7).



**Рисунок 3.7 – Програма знайшла необхідні підмножини та відобразила їх**

У випадку, якщо підмножини не знайдено, буде виведено повідомлення про це (Рисунок 3.8).



**Рисунок 3.8 – Програма не знайшла підмножин з такою сумою**

Для роботи з іншим файлом або іншими даними необхідно перезавантажити програму та відкрити інший файл або ввести нові елементи.

# ПРИКЛАДИ РОЗВ’ЯЗАННЯ ЗАДАЧ

## Метод повного перебору

Нехай в нас є множина з 6 елементів: {3,-1,5,4,1,6}.Знайти таку підмножину сума, якої дорівнює 6.

## Розв’язання задачі вручну

У таблиці 4.1 переберемо всі можливі підмножини методом грубої сили:

Таблиця 4.1 – Всі можливі підмножини

|  |  |  |
| --- | --- | --- |
| [3]  [-1]  [5]  [4]  [1]  [6] – розв’язок  [3, -1]  [3, 5]  [3, 4]  [3, 1]  [3, 6]  [-1, 5]  [-1, 4]  [-1, 1]  [-1, 6]  [5, 4]  [5, 1]-розв’язок  [5, 6]  [4, 1]  [4, 6]  [1, 6] | [3, -1, 5]  [3, -1, 4] -розв’язок  [3, -1, 1]  [3, -1, 6]  [3, 5, 4]  [3, 5, 1]  [3, 5, 6]  [3, 4, 1]  [3, 4, 6]  [3, 1, 6]  [-1, 5, 4]  [-1, 5, 1]  [-1, 5, 6]  [-1, 4, 1]  [-1, 4, 6]  [-1, 1, 6] -розв’язок  [5, 4, 1]  [5, 4, 6]  [5, 1, 6]  [4, 1, 6]  [3, -1, 5, 4] | [3, -1, 5, 1]  [3, -1, 5, 6]  [3, -1, 4, 1]  [3, -1, 4, 6]  [3, -1, 1, 6]  [3, 5, 4, 1]  [3, 5, 4, 6]  [3, 5, 1, 6]  [3, 4, 1, 6]  [-1, 5, 4, 1]  [-1, 5, 4, 6]  [-1, 5, 1, 6]  [-1, 4, 1, 6]  [5, 4, 1, 6]  [3, -1, 5, 4, 1]  [3, -1, 5, 4, 6]  [3, -1, 5, 1, 6]  [3, -1, 4, 1, 6]  [3, 5, 4, 1, 6]  [-1, 5, 4, 1, 6]  [3, -1, 5, 4, 1, 6] |

## Розв’язання задачі за допомогою розробленого програмного продукту

На Рисунок 4.1 зображений результат роботи програми пошуку підмножини, сума яких дорівнює шуканій.

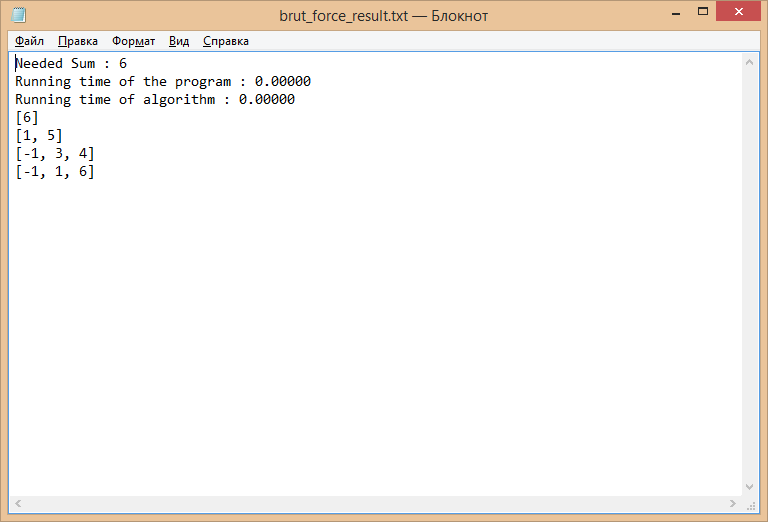


Рисунок 4.1 – Результат роботи програми

Як бачимо, результат розв’язку вручну та за допомогою програми співпали.

## Метод динамічного програмування

Розв’яжемо минулу задачу. Нехай в нас є множина з 6 елементів: {3,-1,5,4,1,6}.Знайти таку підмножину сума, якої дорівнює 6.

## Розв’язання вручну

Для розв’язання вручну побудуємо таблицю та заповнимо значеннями True/False .

У таблиці 4.2 зображено розв’язання задачі методом динамічного програмування.

Таблиця 4.2 – Розв’язання задачі

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | -1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 0 | f | f | f | f | T | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f |
| 1 | T | f | f | T | T | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f |
| 2 | T | f | T | T | T | T | T | f | T | T | f | f | f | f | f | f | f | f | f | f | f |
| 3 | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T |
| 4 | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T |
| 5 | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T |

Розв’язуємо поетапно:

1.Обираємо ст. 6 , ряд. 5;

2.Переходимо на ст. 6 , ряд. 3;

3.Записуємо 4 , переходимо на ст. 2 , ряд. 2 ;

4.Переходимо на ст. 2 , ряд. 1;

5.Записуємо -1, переходимо на ст. 3, ряд. 0;

6.Записуємо 3.

Отже, отриманий розв’язок [4,-1,3].

## Розв’язання задачі за допомогою розробленого програмного продукту

На Рисунок 4.2 зображений результат роботи програми пошуку підмножини, сума яких дорівнює шуканій.

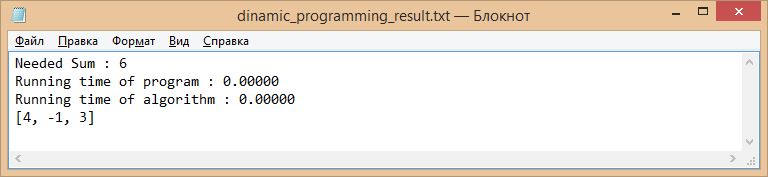


Рисунок 4.2 – Результат роботи програми

Як бачимо, результат розв’язку вручну та за допомогою програми співпали.

# ВИСНОВКИ

Мета та поставлені задачі в курсовій роботі виконані.

Нами були досліджені методи пошуку підмножини в множині, а конкретно методи розв’язання задачі про суму підмножини. Було реалізовано два методи знаходження розв’язку задачі: метод повного перебору (або грубої сили) та метод динамічного програмування.

Обидва алгоритми мають свої переваги та недоліки. Метод повного перебору знаходить всі можливі розв’язки, проте для великих множин він буде працювати години, дні або й тижні. Метод динамічного програмування працює значно швидше, проте знаходить лише один можливий варіант розв’язку задачі.

Також під час виконання курсової роботи кожен з членів команди вдосконалив навик роботи в команді.

Також реалізована неосновна вимога ведення даних вручну та зчитування з файлу.

# ПЕРЕЛІК ПОСИЛАНЬ

1. A Dropbox Challenge And Dynamic Programming [Електронний ресурс] – Режим доступу: http://www.skorks.com/2011/02/algorithms-a-dropbox-challenge-and-dynamic-programming/
2. А. В. Кононов, П. А. Кононова (2014). Приближенные алгоритмы для NP-трудных задач : учеб.-метод. пособие ; Новосиб. гос. ун-т. — Новосибирск: РИЦ НГУ, 2014. — 117 с.
3. Викиконспекты университета ИТМО [Електронний ресурс] – Режим доступу: <http://neerc.ifmo.ru/wiki/index.php?title>=Задача\_о\_рюкзаке#
4. Википедия «Задача о сумме подможеств» [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Задача\_о\_сумме\_подмножеств
5. Документація Python [Електронний ресурс] – Режим доступу https://docs.python.org/2/library/
6. Блог «Спортивне програмування»,стаття «Динамическое програмирование» [Електронний ресурс] – Режим доступу http://awpris.blogspot.com/2012/02/blog-post.html

# Додаток А

**Лістинг програми**

import os

import time

from brut\_force import brut\_force\_algorithm

from dinamic\_programming import dinamic\_algorithm

if \_\_name\_\_ == "\_\_main\_\_":

choice = input("Do you want read from file? y/n ")

if choice.lower() == 'y':

name\_file = input("Please input name of file: ")

file = open (name\_file, 'r')

# for i in file:

# start\_set.append(int(i))

start\_set = [int(i) for i in file]

file.close()

else:

start\_set = []

numb = input("Please enter number of array: ")

for i in range(int(numb)):

el = int(input ("Enter elements :"))

start\_set.append(el)

n = int (input ("Please input sum , which you searching : "))

while True:

print ("-"\*50)

selection = input("""What do you want?

1.Solve this problem by 'brut force' algorithm.

2.Solve this problem by dinamic programing.

3. Exit.

Choose some number : """)

print ("-"\*50)

if selection =='1':

start\_time = time.time()

a = brut\_force\_algorithm().find\_answer(n, start\_set)

f = open ('brut\_force\_result.txt', 'w')

finish\_time = time.time()

f.write('Needed Sum : ' + str(n) + '\n')

f.write('Running time of the program : %.5f' %(finish\_time - start\_time) + '\n')

f.write (a[1]+'\n')

for i in a[0]:

f.write (str(i) + '\n')

if not a[0]:

f.write ('Such Subsets do not exists.')

print ('You can see result in file "brut\_force\_result.txt"')

f.close()

os.system("start "+"brut\_force\_result.txt")

elif selection == '2':

start\_time = time.time()

a = dinamic\_algorithm().subset\_sum(start\_set,n)

f = open ("dinamic\_programming\_result.txt", 'w')

finish\_time = time.time()

f.write('Needed Sum : ' + str(n) + '\n')

f.write ('Running time of program : %.5f' %(finish\_time - start\_time) + '\n')

f.write(a[1]+'\n')

f.write(str(a[0])+'\n')

if not a[1]:

f.write ('Such Subsets do not exists.')

f.close()

print ('You can see result in file "dinamic\_programming\_result.txt"')

os.system("start "+"dinamic\_programming\_result.txt")

elif selection == '3':

break

else:

print ("Unknown Option Selected!")

**Модуль brut\_force**

import time

class brut\_force\_algorithm:

def comb(self, s\_array, r):

s\_array\_tuple = tuple(s\_array)

n = len(s\_array\_tuple)

if r > n:

return

indices = list(range(r))

yield tuple(s\_array\_tuple[i] for i in indices)

while True:

for i in reversed(range(r)):

if indices[i] != i + n - r:

break

else:

return

indices[i] += 1

for j in range(i+1, r):

indices[j] = indices[j-1] + 1

yield tuple(s\_array\_tuple[i] for i in indices)

def find\_answer(self, s\_sum, s\_array ):

start\_time = time.time()

answer\_set = []

new\_answer\_set = []

s\_array\_len = int(len(s\_array))

for i in range(s\_array\_len):

c = self.comb (s\_array, i)

for j in c:

elements\_sum = sum(j)

if (elements\_sum == s\_sum):

qwerty = list(j)

qwerty.sort()

answer\_set.append(qwerty)

continue

for i in answer\_set:

if i not in new\_answer\_set:

new\_answer\_set.append(i)

finish\_time = time.time()

a = ('Running time of algorithm : %.5f' %(finish\_time - start\_time))

return new\_answer\_set,a

**Модуль dynamic\_programming**

import time

class dinamic\_algorithm:

def positive\_negative\_sums(self,seq):

P, N = 0, 0

for c in seq:

if c >= 0:

P += c

else:

N += c

return P, N

def find\_set(self,list\_set,suma,seq):

answer = []

count = len(list\_set)

current = 0

for i in reversed(list\_set):

for j in i:

if(count == len(list\_set)):

if (j[0]==True and j[3]==False and j[2]==suma):

answer.append(j[1])

current = j[2]-j[1]

elif(j[0]==True and j[3]==True and j[2]==suma):

current = j[2]

break

else:

if (j[0]==True and j[3]==False and j[2]==current):

answer.append(j[1])

current = j[2]-j[1]

if (count==1 and j[4]==True):

answer.append(seq[0])

break

elif(j[0]==True and j[3]==True and j[2]==current):

current = j[2]

if count==1:

answer.append(seq[0])

break

count -= 1

return (answer)

# print ('\n',i)

def subset\_sum(self,seq, s):

start\_time = time.time()

P, N = self.positive\_negative\_sums(seq)

if not seq or s < N or s > P:

finish\_time = time.time()

a = ('Running time of algorithm : %.5f' %(finish\_time - start\_time))

b = ("Such Subsets do not exists.")

return b,a

n, m = len(seq), P - N + 1

table = [[False] \* m for x in range(n)]

table[0][seq[0]] = True

list\_set =[]

for i in range(1, n):

b\_table = []

for j in range(N, P+1):

table[i][j] = seq[i] == j or table[i-1][j] or table[i-1][j-seq[i]]

a = (table[i][j], seq[i] , j , table[i-1][j] , table[i-1][j-seq[i]] )

b\_table.append(a)

list\_set.append(b\_table)

if table[n-1][s]==True:

answ = self.find\_set(list\_set,s,seq)

finish\_time = time.time()

a = ('Running time of algorithm : %.5f' %(finish\_time - start\_time))

return answ,a

else:

finish\_time = time.time()

a = ('Running time of algorithm : %.5f' %(finish\_time - start\_time) )

b = ("Such Subsets do not exists.")

return b,a