Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Комп’ютерний практикум №3

на тему «Відношення переваги при глобальній порівнюваності критеріїв»

З дисципліни «Теорія прийняття рішень»

7 варіант

Виконала:

студентка гр. ІС-91

Ковальська А.А.

Перевірила:

Жураковська О.С.

# Постановка задачі

Задано множину з 20 альтернатив, які оцінені за множиною критеріїв K = {ki}, i = 1,…,12.

8 8 10 1 3 2 1 9 4 6 5 9

8 6 9 1 3 2 1 6 4 1 5 6

6 4 4 1 3 2 1 4 4 1 5 5

5 4 4 1 2 2 1 2 4 1 4 5

8 8 10 9 8 6 2 9 4 10 5 9

8 7 7 8 6 5 2 4 2 5 2 9

2 4 6 4 2 5 2 3 1 5 2 9

2 4 6 4 2 4 2 3 1 4 2 8

6 10 6 6 7 5 3 10 6 5 10 9

10 10 6 6 10 8 3 10 6 6 10 9

2 1 6 4 2 2 2 3 1 4 2 8

5 2 10 4 3 2 8 6 4 4 10 8

5 2 6 4 1 2 1 6 4 2 7 3

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

9 6 9 4 9 6 8 8 4 10 7 9

9 6 9 4 9 6 8 9 4 10 7 9

8 3 2 1 6 6 8 3 4 10 7 9

8 1 2 1 6 4 8 3 1 2 3 4

8 10 6 9 7 5 8 10 6 7 10 9

10 10 6 9 7 10 8 10 8 7 10 9

Відношення строгого порядку на мн-ні критеріїв

(впорядкування за спаданням важливості):

k9>k11>k4>k1>k5>k6>k12>k3>k2>k8>k10>k7

Відношення квазіпорядку на мн-ні критеріїв

(класи впорядковані за зростанням важливості):

{k2,k4,k6,k9} < {k7,k8,k11} < {k1,k3,k5,k10,k12}

Необхідно за інформацією про оцінки альтернатив за критеріями к1-к12 та інформацією про порівнюваність критеріїв побудувати на множині альтернатив відношення переваги та визначити оптимальні альтернативи, якщо:

1) інформація про порівнюваність критеріїв несуттєва (відн. Парето);

2) критерії рівноважливі (мажоритарне в.);

3) на множині критеріїв задане віднош. строгого порядку V1 (лексикографічне в.);

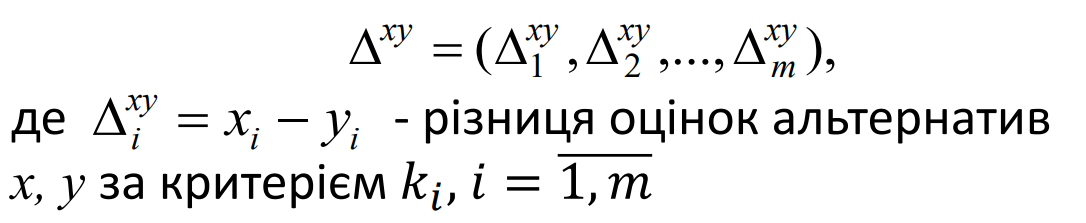
4) на множині критеріїв задане відношення квазіпорядку V2 (відн. Березовського);

5) для випадку рівноважливих критеріїв побудувати на множині альтернатив відношення Подиновського.

# Теоретичні відомості

Для вирішення відношень нам необхідно визначити матрицю векторів знаків різниць оцінок.

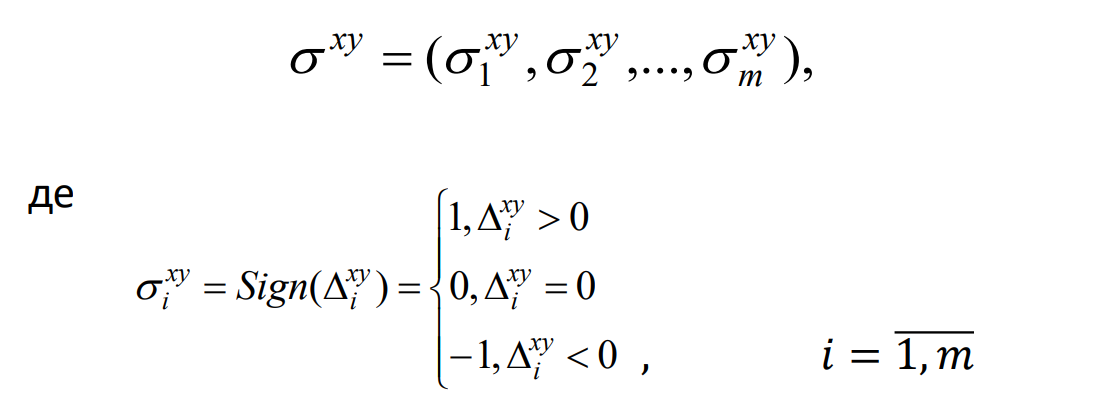
Для цього нам необхідно для кожної пари альтернатив х, у із заданої множини знайти вектор різниць оцінок за формулою:



Фрагмент коду:

def get\_delta(x, y):  
 result = []  
 for i in range(len(x)):  
 result.append(x[i] - y[i])  
 return result

Далі знаходимо саме вектор різниць оцінок за наступним правилом:



Фрагмент коду, де це знаходиться:

def get\_sigma(delta):  
 result = []  
 for i in range(len(delta)):  
 if delta[i] > 0:  
 result.append(1)  
 elif delta[i] < 0:  
 result.append(-1)  
 else:  
 result.append(0)  
 return result

# Опис етапів вирішення задачі

1. Відношення Парето

За умовою інформація про порівнюваність критеріїв несуттєва.

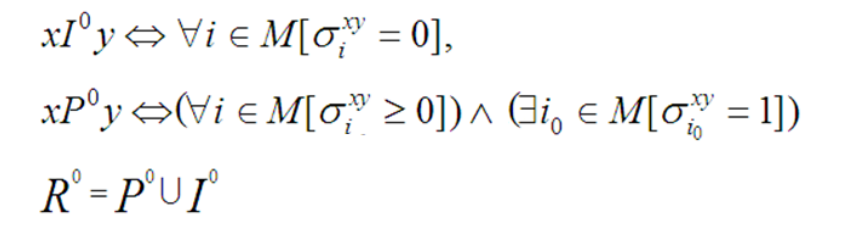
Також за умовою нам дано множину альтернатив Ω та множину критеріїв К, за якими оцінюються критерії.

Спочатку необхідно знайти матрицю векторів знаків різниць оцінок:

def get\_sigma\_table(array):  
 sigma\_table = []  
 for i in range(len(array)):  
 sigma\_table.append([])  
 for j in range(len(array)):  
 delta = get\_delta(array[i], array[j])  
 sigma = get\_sigma(delta)  
 sigma\_table[i].append(sigma)  
  
 print("Вектор різниці оцінок")  
 for i in range(len(array)):  
 for j in range(len(array)):  
 print(f"{sigma\_table[i][j]} ", end='')  
 print()  
  
 return sigma\_table

У цьому методі ми використовуємо додаткові функції: знаходження різниці оцінок та вектор знаків різниць оцінок, які описані у теоретичних відомостях.

Далі знаходимо саме рішення у відношенні Парето, тобто перевіряємо за таблицею чи виконується наступна умова:



Наприклад, за результатами моєї матриці:

σ11 =(0,0,0,0,0,0,0,0,0,0,0,0), то R0 11= 1.

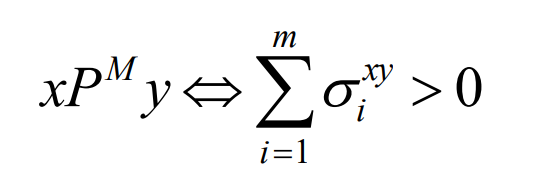
σ21 =(-1,-1, -1,-1, -1,-1, -1,-1, -1,-1,0,0), то R021= 0.

Фрагмент коду:

R0 = []  
for i in range(a\_count):  
 R0.append([])  
 for j in range(a\_count):  
 is\_added = False  
 for index in range(len(sigma\_table[i][j])):  
 if sigma\_table[i][j][index] < 0:  
 R0[i].append(0)  
 is\_added = True  
 break  
 if not is\_added:  
 R0[i].append(1)

1. Мажоритарне відношення

У мажоритарному відношенні спочатку також знаходимо матрицю векторів знаків різниць оцінок, як й у відношенні Парето, але тут треба обрати такі елементи, що задовольняють наступну умову:



Код знаходження сум векторів у матриці векторів знаків різниць оцінок:

vector\_sum\_table = []  
for i in range(a\_count):  
 vector\_sum\_table.append([])  
 for j in range(a\_count):  
 vector\_sum = 0  
 for index in range(len(sigma\_table[i][j])):  
 vector\_sum = vector\_sum + sigma\_table[i][j][index]  
 vector\_sum\_table[i].append(vector\_sum)

Далі знаходимо рішення для мажоритарного відношення, тобто перевіряємо умову чи належить пара відношенню:

RM = []  
for i in range(a\_count):  
 RM.append([])  
 for j in range(a\_count):  
 if vector\_sum\_table[i][j] > 0:  
 RM[i].append(1)  
 else:  
 RM[i].append(0)

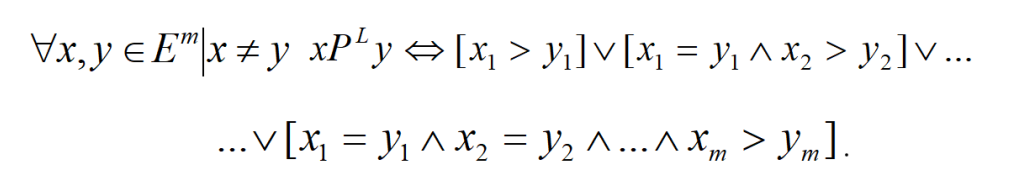
1. Лексографічне відношення

Відношення <V, K>, задане на множині критеріїв, є слабкозв’язним строгим порядком, тобто критерії можна впорядкувати за важливістю.

За умовою множина критеріїв К впорядкована за таким спаданням важливості:

k9>k11>k4>k1>k5>k6>k12>k3>k2>k8>k10>k7.

Тут для знаходження рішення після отримання матриці векторів знаків оцінок, треба обрати такі елементи, які задовольняють наступну умову:



1. Відношення Березовського

Відношення <V,K> , задане на множині критеріїв, є квазіпорядком, тобто критерії розбиті на класи рівноважливих критеріїв , а класи впорядковані за зростанням їх важливості.

В цьому випадку для кожної групи рівноважливих критеріїв необхідно побудувати систему відношень Парето P0j, I0j, N0j, використовуючи співвідношення:



Фрагмент коду:

p\_b = []  
for i in range(a\_count):  
 p\_b.append([])  
 for j in range(a\_count):  
 if p\_system[i][j] == 1 and (p\_prev[i][j] == 1 or n\_prev[i][j] == 1 or i\_prev[i][j] == 1):  
 p\_b[i].append(1)  
 elif i\_system[i][j] == 1 and p\_prev[i][j] == 1:  
 p\_b[i].append(1)  
 else:  
 p\_b[i].append(0)



Фрагмент коду:

i\_b = []  
for i in range(a\_count):  
 i\_b.append([])  
 for j in range(a\_count):  
 if i\_system[i][j] == 1 and i\_prev[i][j] == 1:  
 i\_b[i].append(1)  
 else:  
 i\_b[i].append(0)



Фрагмент коду:

n\_b = []  
for i in range(a\_count):  
 n\_b.append([])  
 for j in range(a\_count):  
 if not (p\_b[i][j] == 1 or p\_b[j][i] == 1 or i\_b[i][j] == 1):  
 n\_b[i].append(1)  
 else:  
 n\_b[i].append(0)

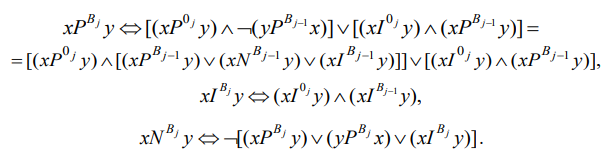
Побудова відношення Березовського виконується ітераційно за l ітерацій.

Крок 1.



if l == 0:  
 i\_prev = i\_system  
 n\_prev = n\_system  
 p\_prev = p\_system  
 continue

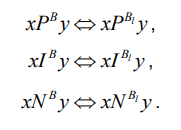
Крок j (j = 2, ..,l). Для всіх пар альтернатив будуємо систему відношень PВj, IВj, NВj, використовуючи співвідношення:



Фрагмент коду, де дане співвідношення задається:

def get\_p\_system(array):  
 result = []  
 for i in range(len(array)):  
 result.append([])  
 for j in range(len(array)):  
 sigma = array[i][j]  
 is\_greater = False  
 is\_lower = False  
 for index in range(len(sigma)):  
 if sigma[index] == 1:  
 is\_greater = True  
 elif sigma[index] == -1:  
 is\_lower = True  
 if is\_greater and not is\_lower:  
 result[i].append(1)  
 else:  
 result[i].append(0)  
 return result  
  
def get\_i\_system(array):  
 result = []  
 for i in range(len(array)):  
 result.append([])  
 for j in range(len(array)):  
 sigma = array[i][j]  
 if all(v == 0 for v in sigma):  
 result[i].append(1)  
 else:  
 result[i].append(0)  
 return result  
  
def get\_n\_system(array):  
 result = []  
 for i in range(len(array)):  
 result.append([])  
 for j in range(len(array)):  
 sigma = array[i][j]  
 is\_greater = False  
 is\_lower = False  
 for index in range(len(sigma)):  
 if sigma[index] == 1:  
 is\_greater = True  
 elif sigma[index] == -1:  
 is\_lower = True  
 if is\_greater and is\_lower:  
 result[i].append(1)  
 else:  
 result[i].append(0)  
 return result

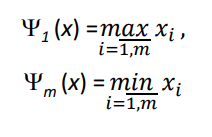
В результаті відношення, отримане на останній ітерації, є відношенням Березовського:



1. Відношення Подинського

За умовою критерії рівноважливі.

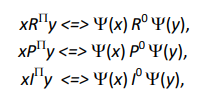
Тому нехай Ψ(x) - вектор-функція, що розташовує усі компоненти вектора x∈Em за спаданням значень, тобто

.

Код, який будує дану вектор-функцію:

def get\_omega(array):  
 result = []  
 for i in range(a\_count):  
 result.append(sorted(array[i], reverse=True))  
 return result

Так як всі критерії рівноважливі за умовою, то виконується наступне:



Де Ψ(x) – вектор-функція, що розташовує усі компоненти вектора x∈E m за спаданням значень,

R0 – відношення Парето,

P0 , I0 – асиметрична та симетрична частини відповідно відношення R0.

Код:

def solve\_Podinovski(array):  
 print("Алгоритм Подиновського")  
 omega = get\_omega(array)  
 print("Вектор-функція")  
 print\_array(omega)  
  
 solve\_Pareto(omega)

# Опис функцій та методів

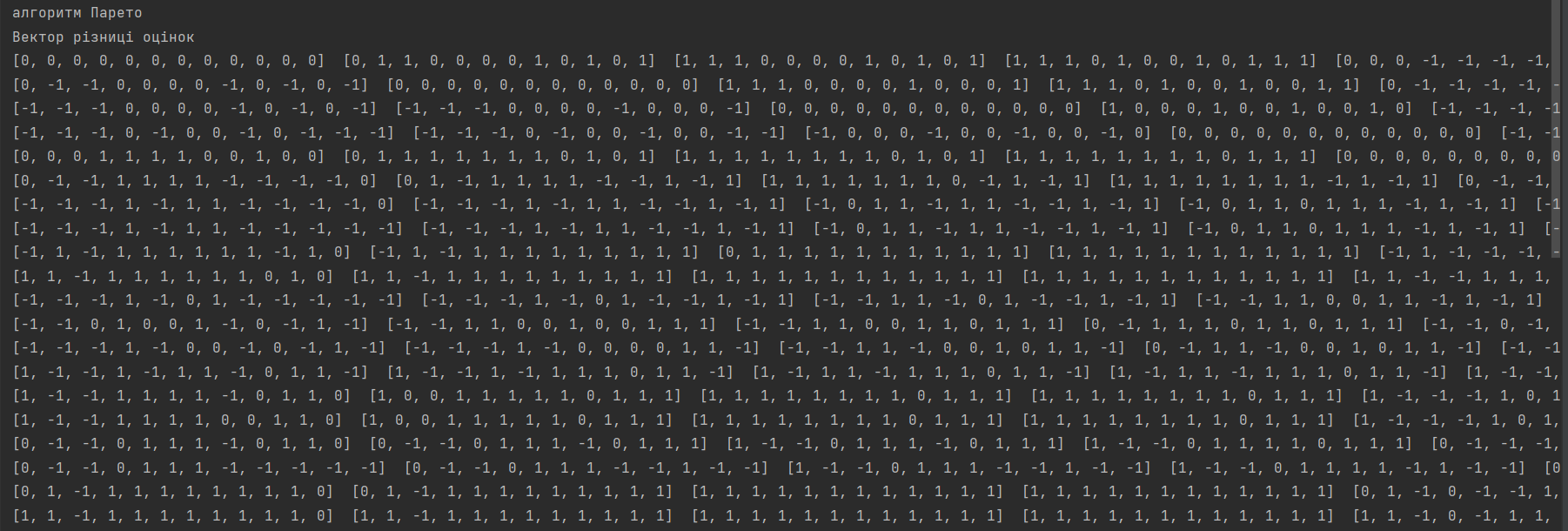
|  |  |  |  |
| --- | --- | --- | --- |
| Функція/метод | Параметри | Опис | Значення, що повертає |
| solve\_Pareto | array - вхідний масив | Алгоритм знаходження відношення Парето на основі матриці векторів знаків різниць оцінок | R0 – відношення Парето |
| solve\_ Magoritar | array - вхідний масив | Алгоритм знаходження мажоритарного відношення на основі матриці векторів знаків різниць оцінок | RM - мажоритарне відношення |
| solve\_ Lecsicograph | array - вхідний масив | Алгоритм знаходження лексикографічного відношення на основі матриці векторів знаків різниць оцінок | RL- лексографічне відношення |
| solve\_ Berizovskogo | array - вхідний масив | Алгоритм знаходження відношення Березовського | p\_b – відношення Березовського |
| get\_p\_system | array - вхідний масив | Знаходження відношення Pi для і-тої ітерації | result - матриця Pi |
| get\_i\_system | array - вхідний масив | Знаходження відношення Ii для і-тої ітерації | result - матриця Ii |
| get\_n\_system | array - вхідний масив | Знаходження відношення Ni для і-тої ітерації | result - матриця Ni |
| solve\_ Podinovskogo | array - вхідний масив | Алгоритм знаходження відношення Подиновського | RP -відношення Подиновського |
| get\_omega | array - вхідний масив | Знаходження матриці омега, де значення оцінок критеріїв у кожній альтернативі впорядковані за спаданням | result - матриця омега |
| print\_array | array - вхідний масив | Виведенння у консоль масиву |  |
| get\_delta | x - перша альтернатива, що складається з k критеріїв  y - друга альтернатива, що складається з k критеріїв | Розрахунок дельта-веткору двох альтернатив | result - дельта-вектор |
| get\_sigma | delta - дельта-вектор | Розрахунок сігма-вектору на базі дельта-вектору | result - cігма-вектор |
| get\_sigma\_table | array - вхідний масив | Отримання матриці векторів знаків різниць оцінок шляхом розрахунку сігма-векторів | sigma\_table - матриці векторів знаків різниць оцінок |

Таблиця 1. Опис функцій та методів

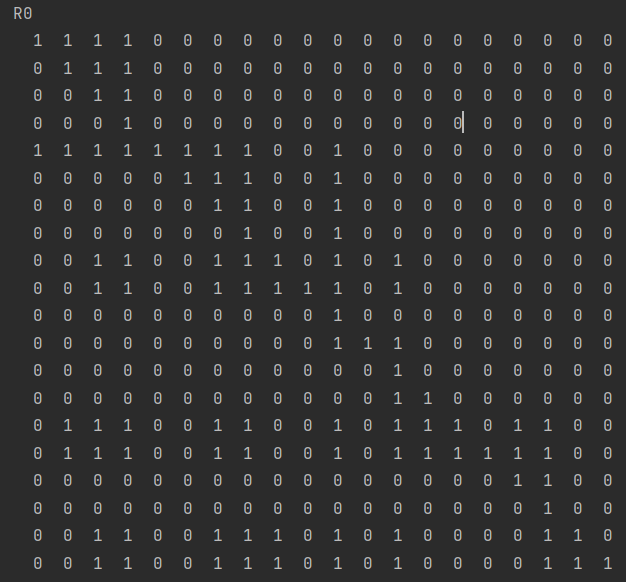
# Розв’язок та проміжні результати

1. Відношення Парето

Знаходимо матрицю векторів знаків різниць оцінок:



Відношення Парето:



Оптимізація за домінуванням:

I≠∅, тому підмножина найкращих альтернатив – найбільший та строго найбільший елемент по R на множині Ω.

Відповідь: X\*R = {}, X\*\*R = {}.

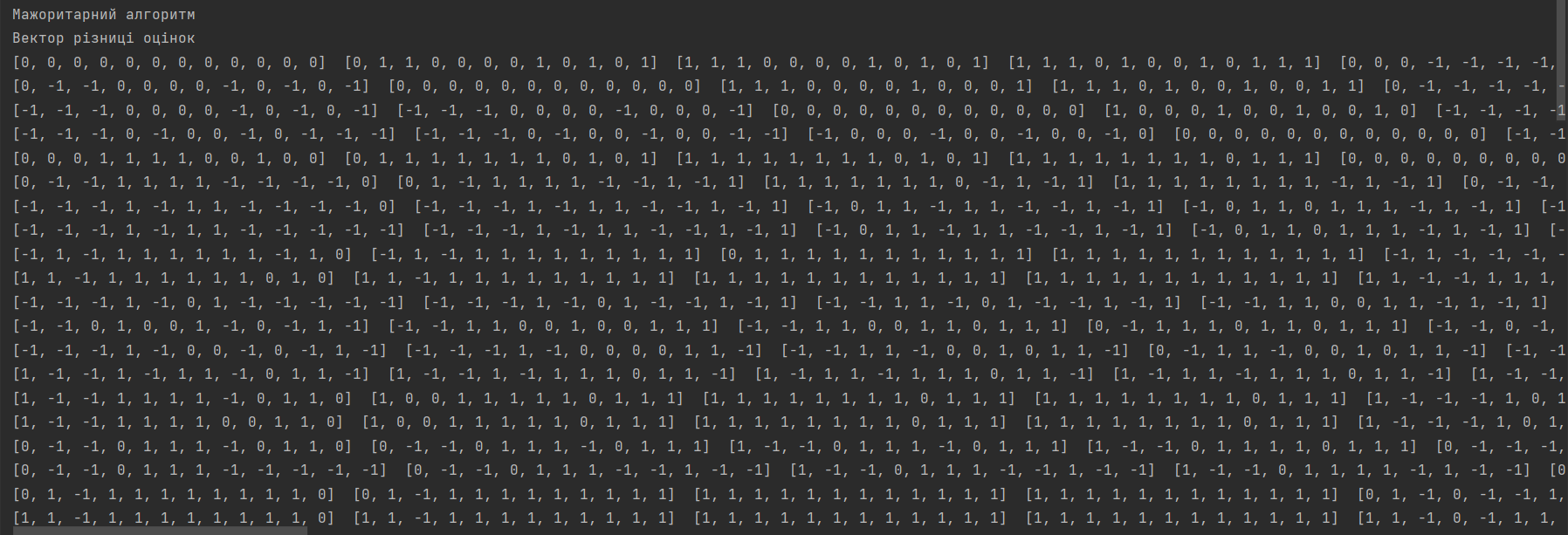
Оптимізація за блокуванням:

I≠∅, тому множина ненайгірших альтернатив – максимальний та строго максимальний елемент по R на множині Ω.

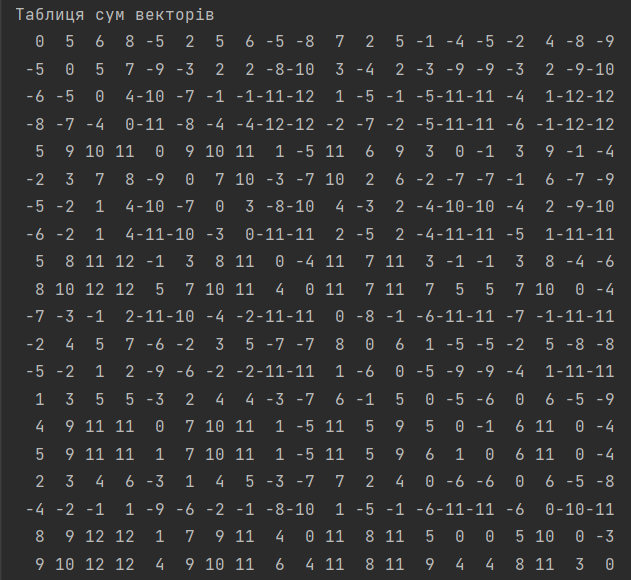
Відповідь: X0R = {5, 10, 12, 16, 20}, X00R = {5, 10, 12, 16, 20}.

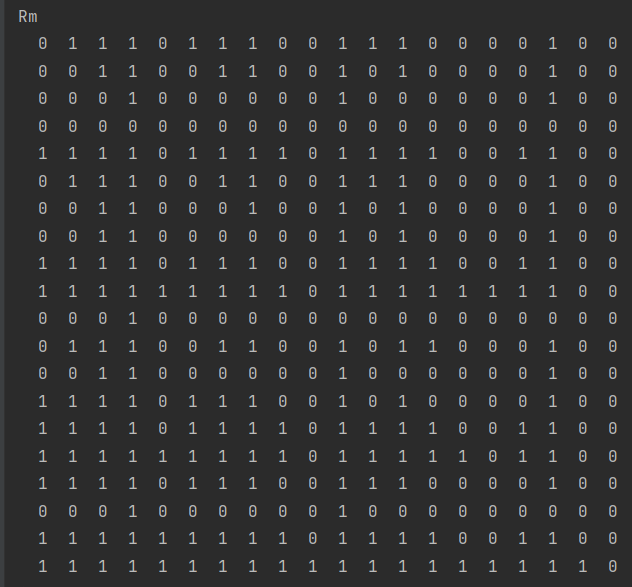
1. Мажоритарне відношення

Знаходимо матрицю векторів знаків різниць оцінок:



Знаходимо суму елементів векторів знаків різниць оцінок:



Мажоритарне відношення:

Оптимізація за домінуванням:

I=∅, тому підмножина найкращих альтернатив – найбільший за домінуванням елемент по Р на множині Ω.

Відповідь: X\*P = {20}.

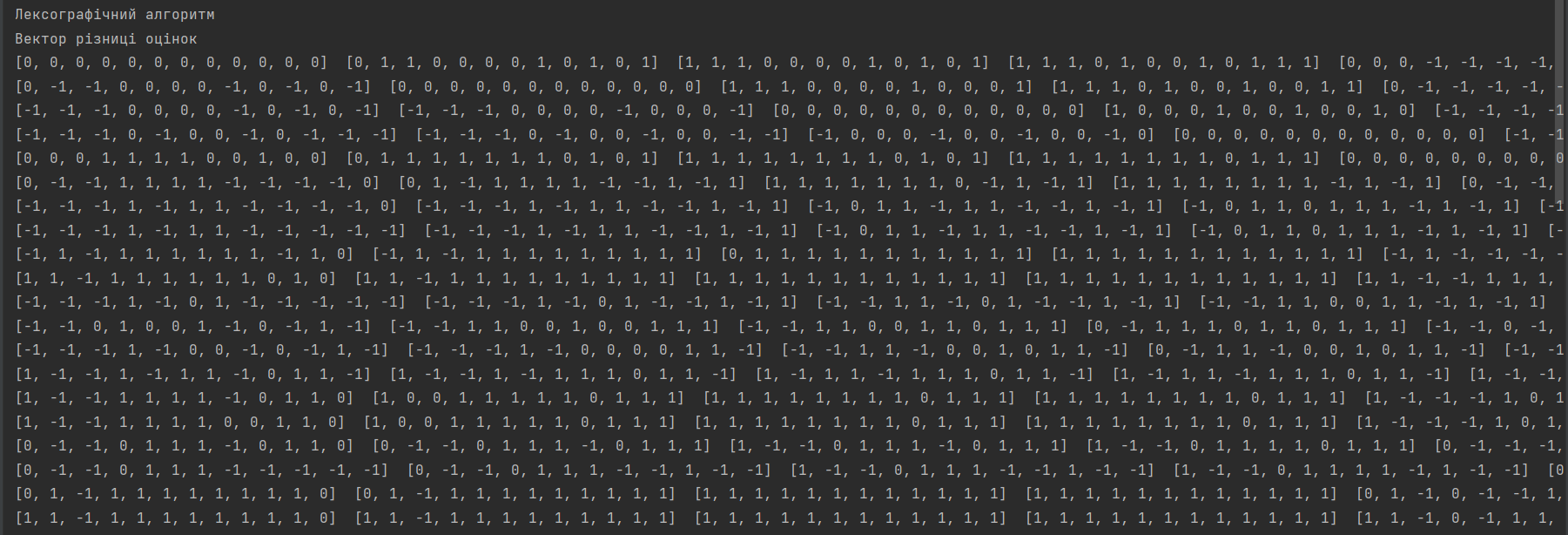
Оптимізація за блокуванням:

I=∅, тому множина ненайгірших альтернатив – максимальний елемент по Р на множині Ω.

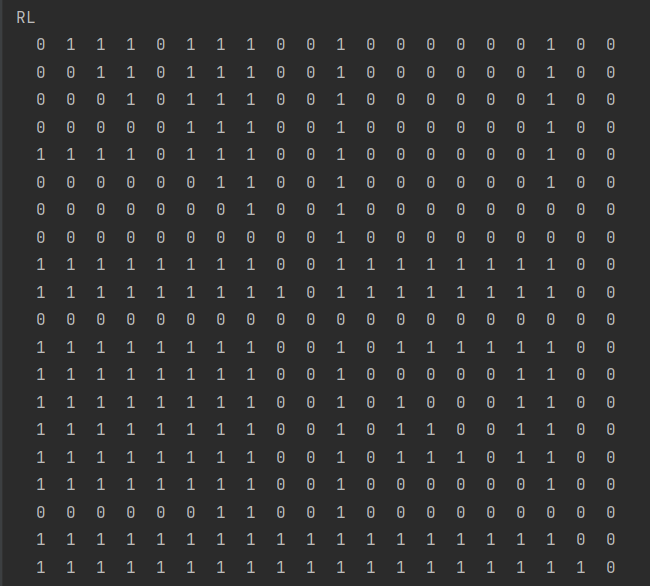
Відповідь: X0P = {20}.

1. Лексографічне відношення

Знаходимо матрицю векторів знаків різниць оцінок:



Лексографічне відношення:



Оптимізація за домінуванням:

I=∅, тому підмножина найкращих альтернатив – найбільший за домінуванням елемент по Р на множині Ω.

Відповідь: X\*P = {20}.

Оптимізація за блокуванням:

I=∅, тому множина ненайгірших альтернатив – максимальний елемент по Р на множині Ω.

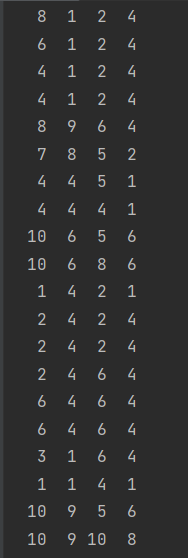
Відповідь: X0P = {20}.

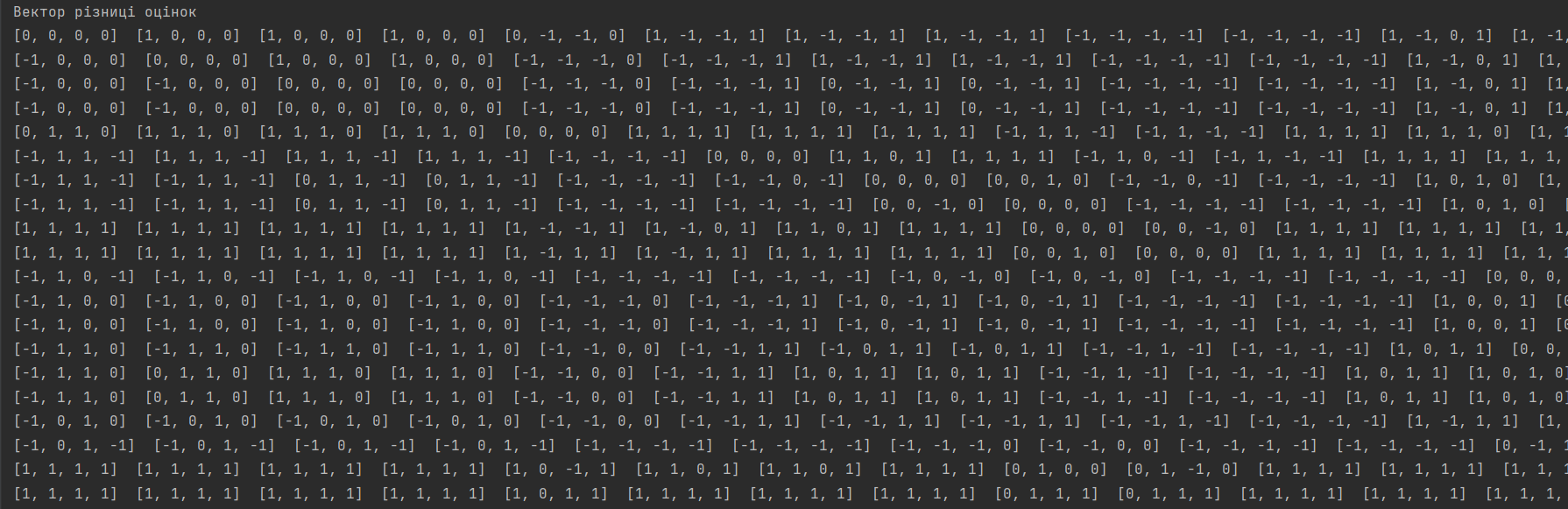
1. Відношення Березовського

Будуємо для кожного класу рівноважливих критеріїв систему відношень Парето P0, I0, N0.

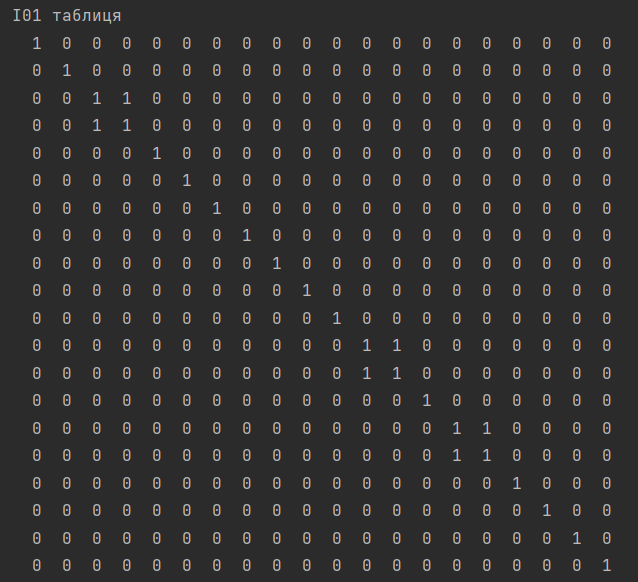
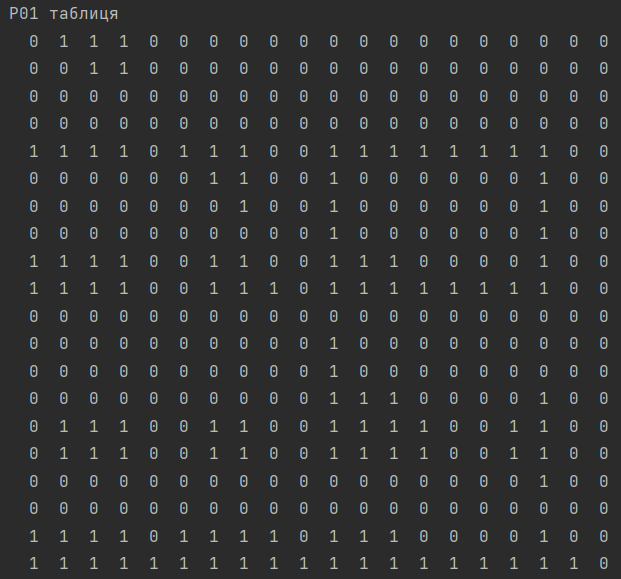
1 крок:

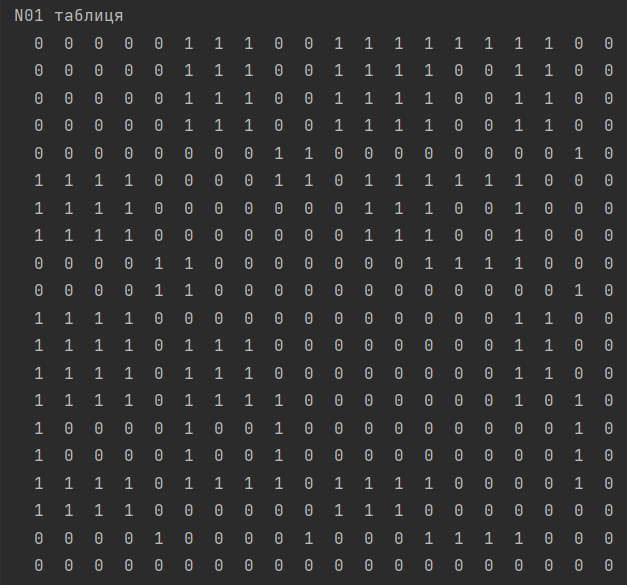
j = 1:



Знаходимо матрицю векторів знаків різниць оцінок:

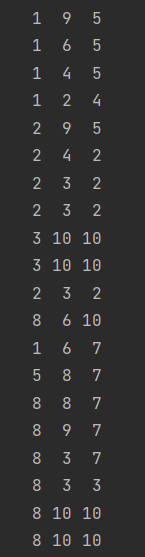
Будуємо систему відношень Парето P01, I01, N01 :

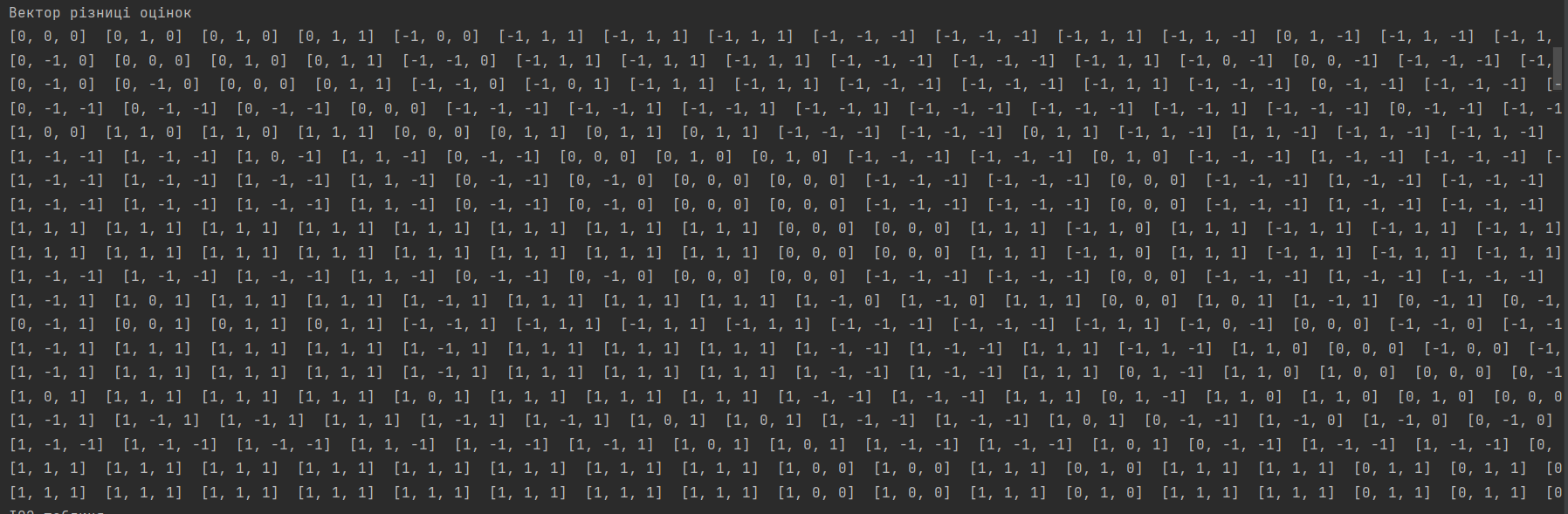


Крок 2:

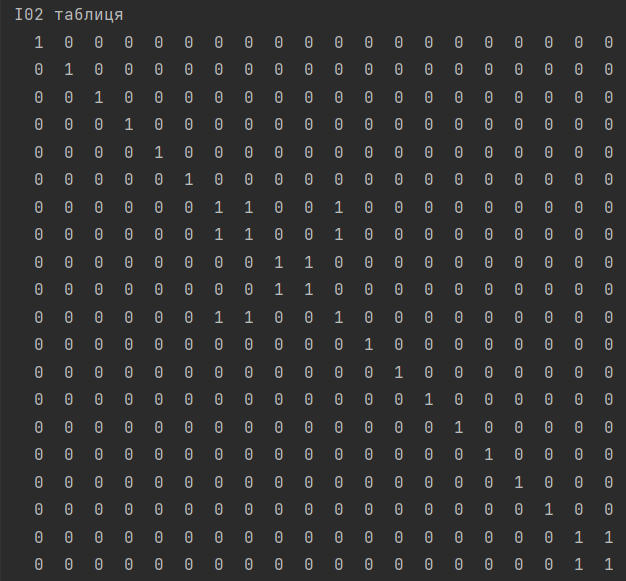
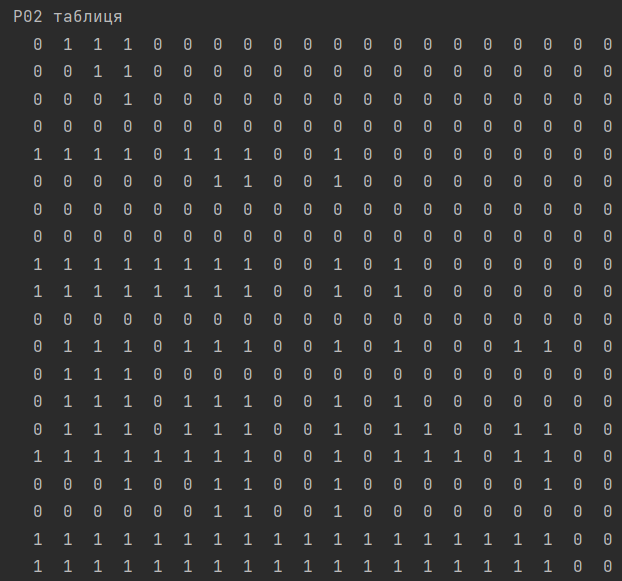
j = 2

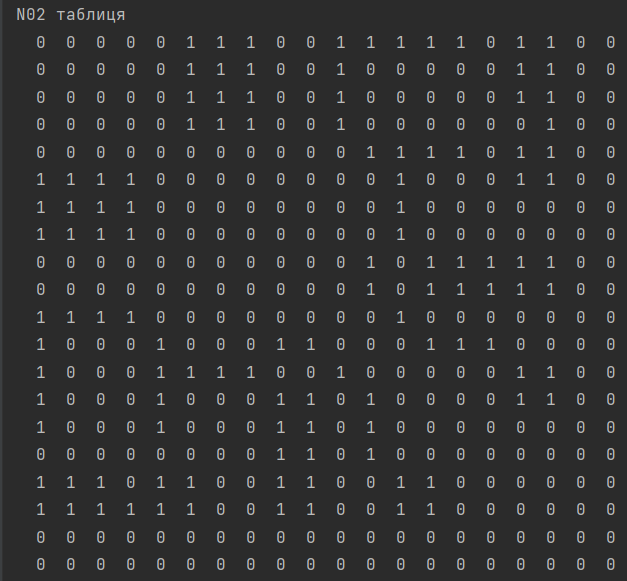


Знаходимо матрицю векторів знаків різниць оцінок:

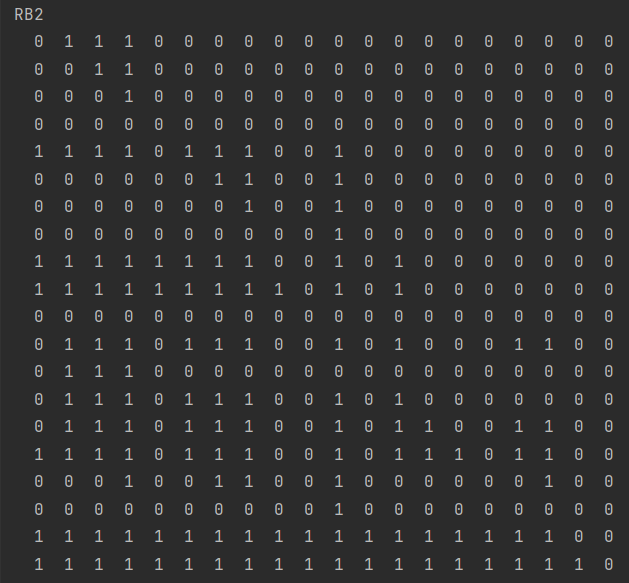
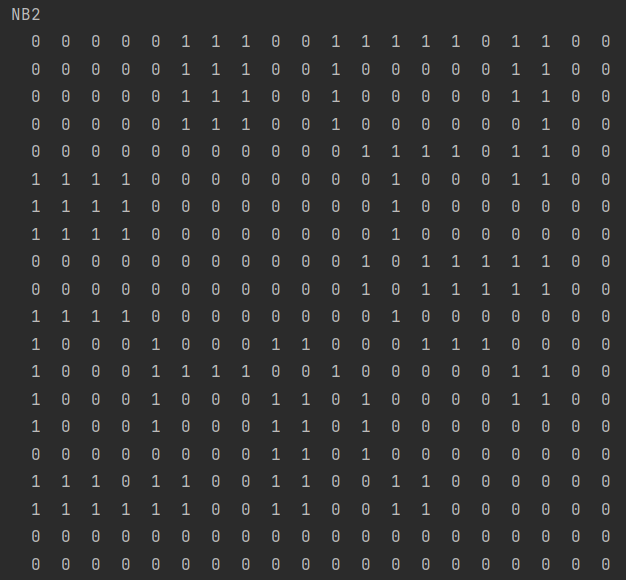


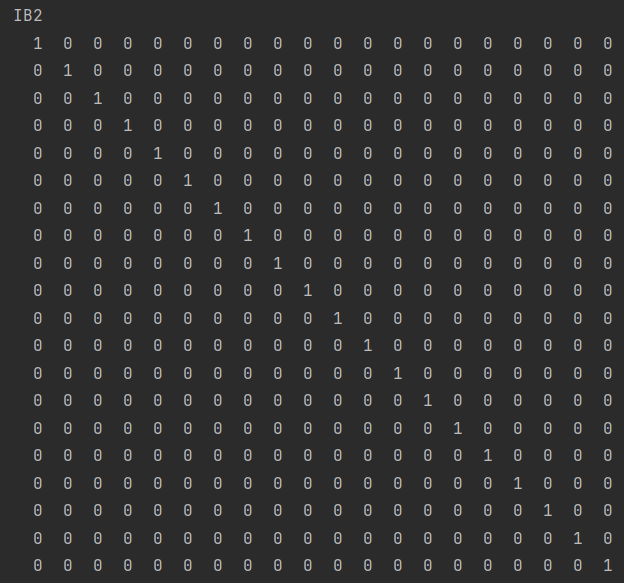
Будуємо систему відношень Парето P02, I02, N02 :

’

Для всіх пар альтернатив будуємо систему відношень PВ2, IВ2, NВ2:

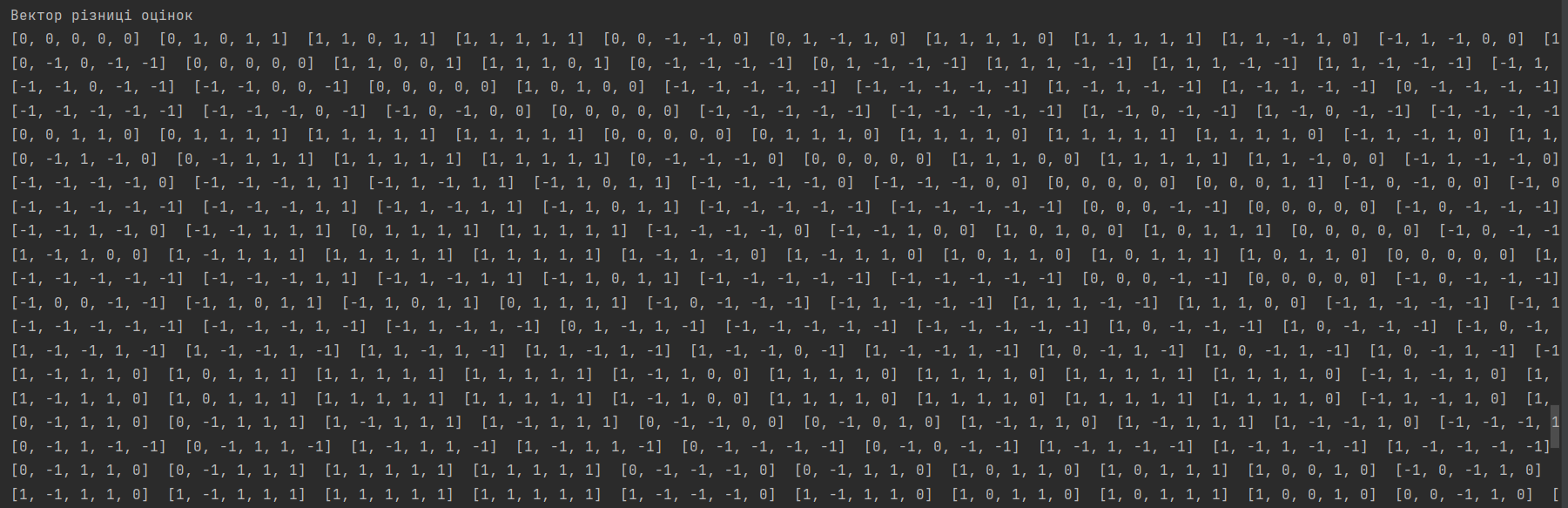
 



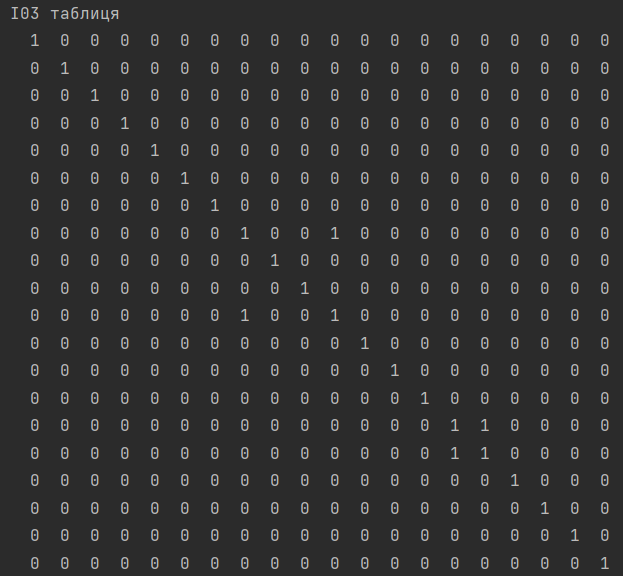
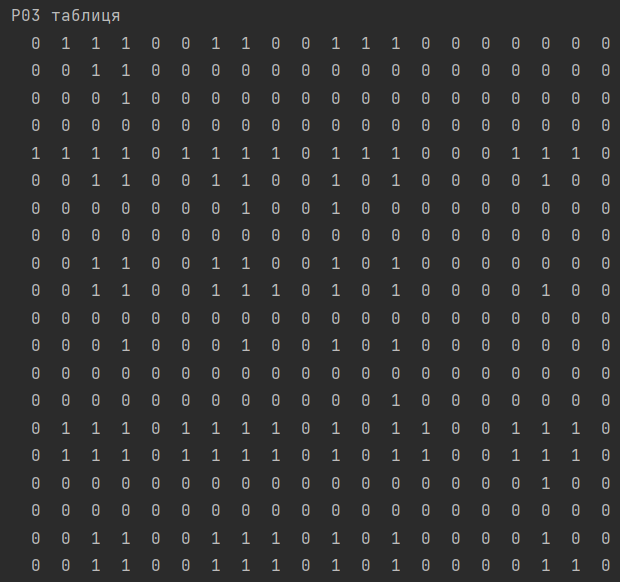
Крок 3:

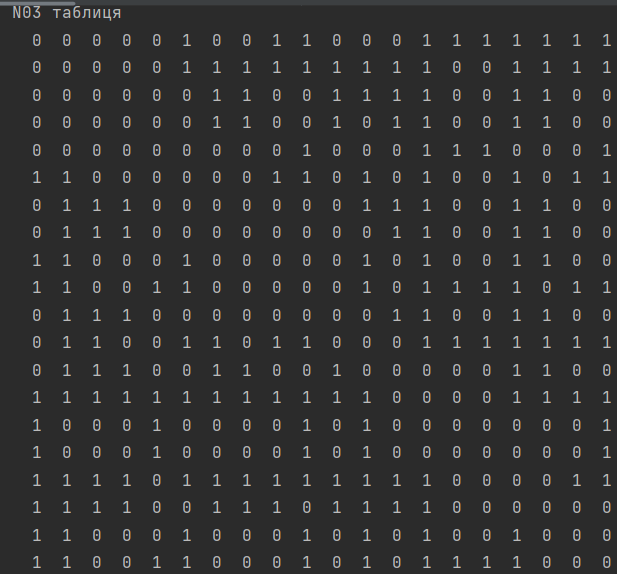


Знаходимо матрицю векторів знаків різниць оцінок:

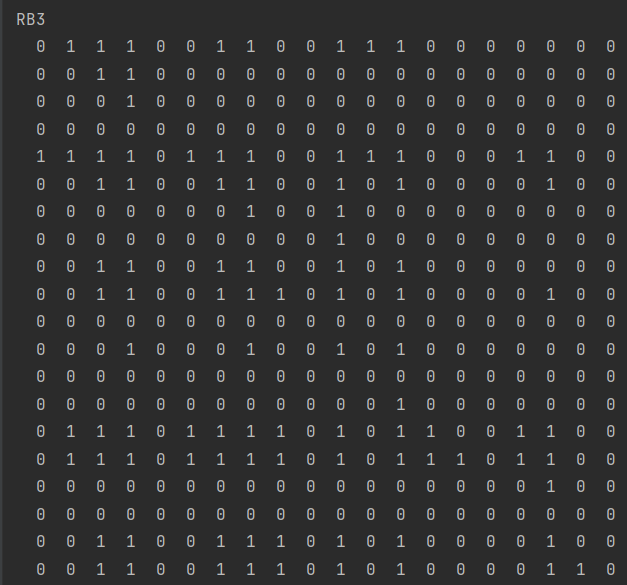
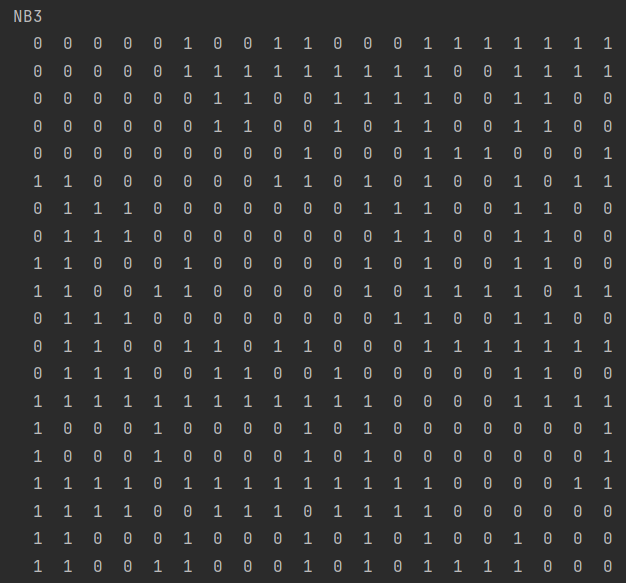


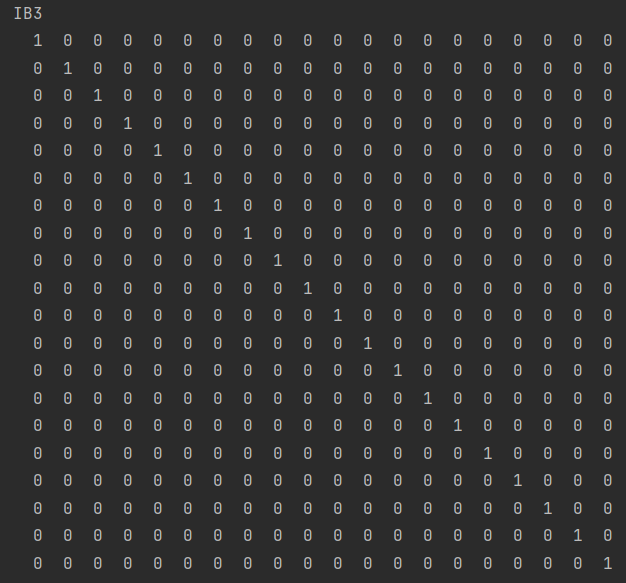
Будуємо систему відношень Парето P03, I03, N03 :

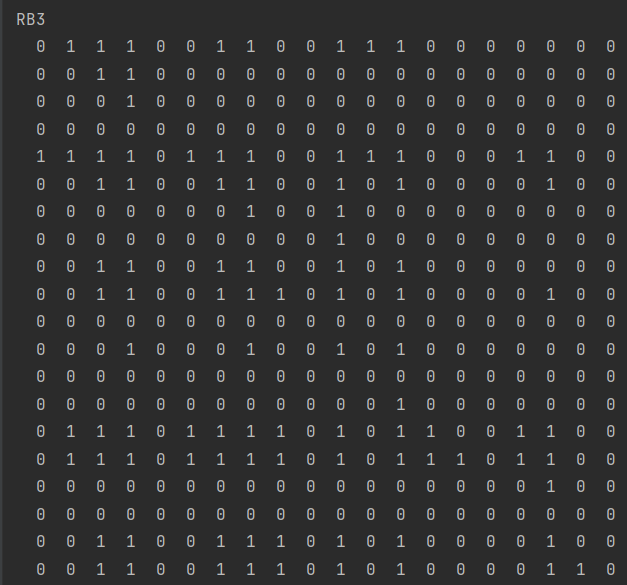


Для всіх пар альтернатив будуємо систему відношень PВ3, IВ3, NВ3:



Отримане в кінці відношення РВ3 і є відношенням Березовського:



Це відношення має властивості асиметричності, антирефлексивності та ациклічності.

Оптимізація за домінуванням:

I=∅, тому підмножина найкращих альтернатив – найбільший за домінуванням елемент по Р на множині Ω.

Відповідь: X\*P = {}.

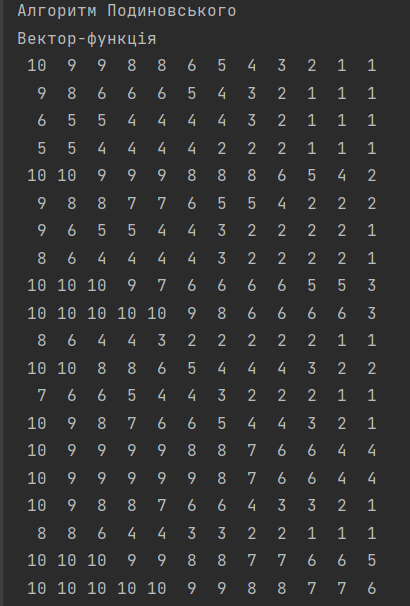
Оптимізація за блокуванням:

I=∅, тому множина ненайгірших альтернатив – максимальний елемент по Р на множині Ω.

Відповідь: X0P = {5, 10, 16, 20}.

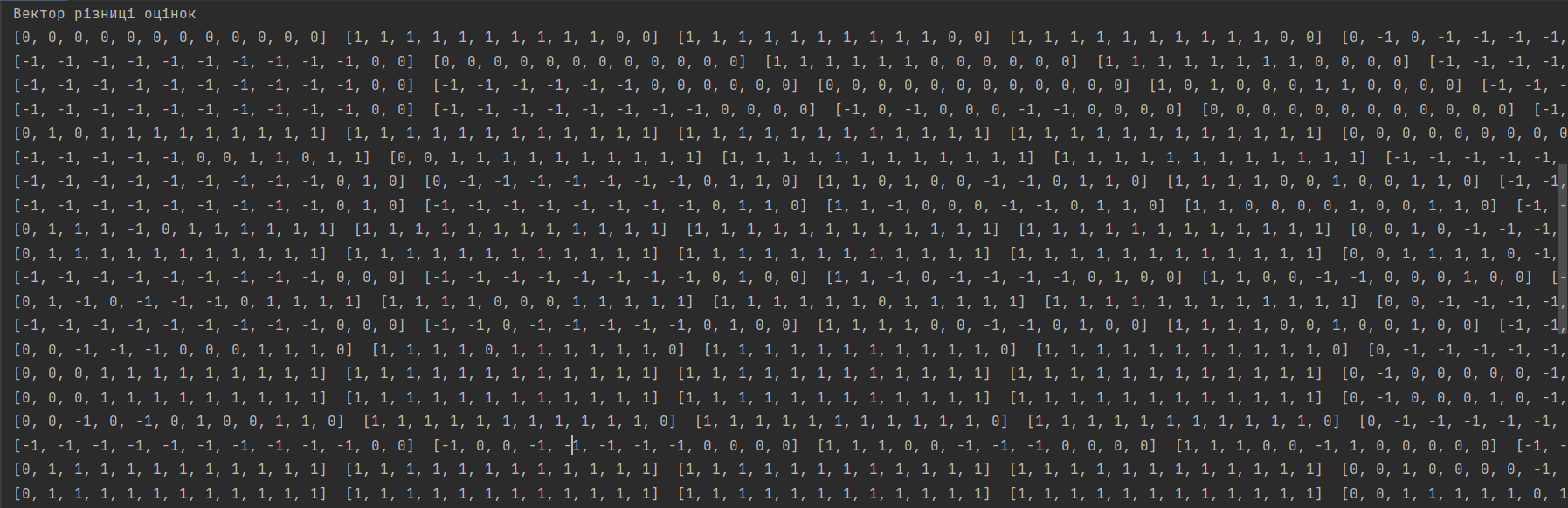
1. Відношення Подиновського

Знаходимо вектор-функцію, де елементи розташовані за спаданням:

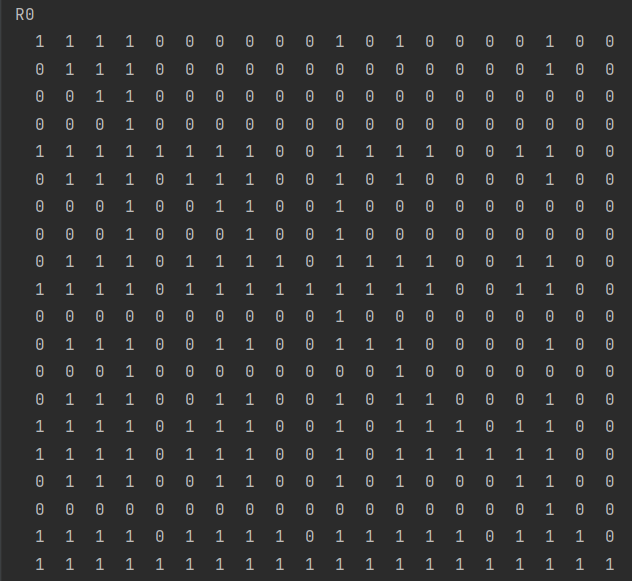


Далі використовуємо алгоритм рішення за алгоритмом Парето.

Знаходимо матрицю векторів знаків різниць оцінок:



Відношення Подиновського:



Оптимізація за домінуванням:

I≠∅, тому підмножина найкращих альтернатив – найбільший та строго найбільший елемент по R на множині Ω.

Відповідь: X\*R = {20}, X\*\*R = {20}.

Оптимізація за блокуванням:

I≠∅, тому множина ненайгірших альтернатив – максимальний та строго максимальний елемент по R на множині Ω.

Відповідь: X0R = {20}, X00R = {20}.