**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра прикладних інформаційних систем**

ЗВІТ

Про виконання лабораторної роботи №7

з дисципліни: «Системний аналіз та теорія прийняття рішень»

*Студента 3 курсу*

*групи ПП-31*

*спеціальності 122 «Комп’ютерні науки»*

*ОП «Прикладне програмування»*

*Вовка Олександра*

*Викладач:*

*Білий Р.О.*

1. Київ – 2023

**Тема:** Методи голосування

**Мета роботи:** вивчення методів голосування і дослідження їх властивостей; опанування методикою обробки профілів колективного голосування, знаходження переможця та відновлення колективного ранжування (навчальний посібник Присяжнюк) – стор.30, 20 варіантів

**Завдання**

**Варіант №4**

**Завдання до виконання**

**Задача голосування порядку денного**

(ранжування визначає значущість проекту та пріоритетність у фінансуванні) Список проектів-кандидатів:

A. відновлення смертної кари

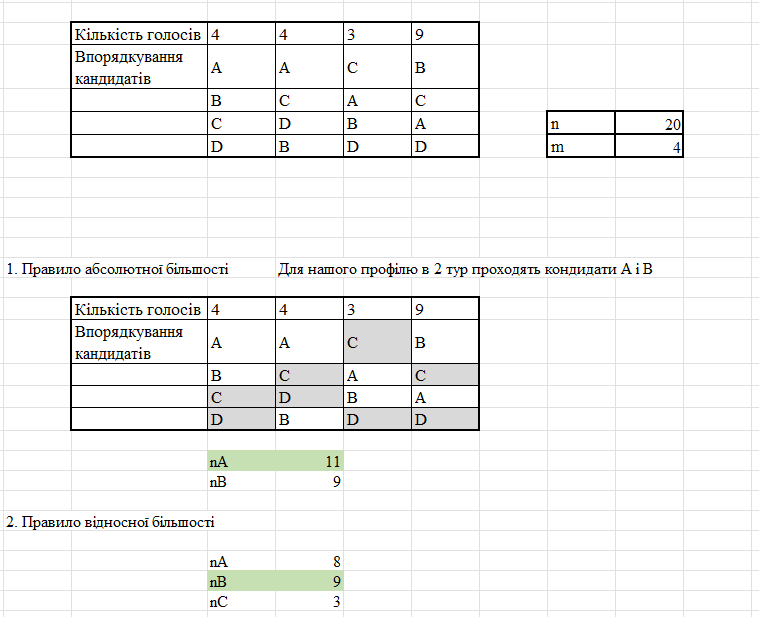
B. легалізація наркотиків

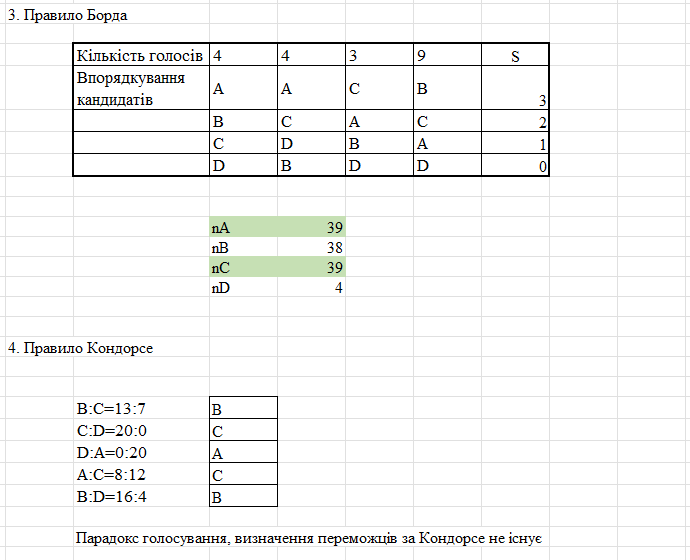
C. заборона алкогольної реклами на ТВ

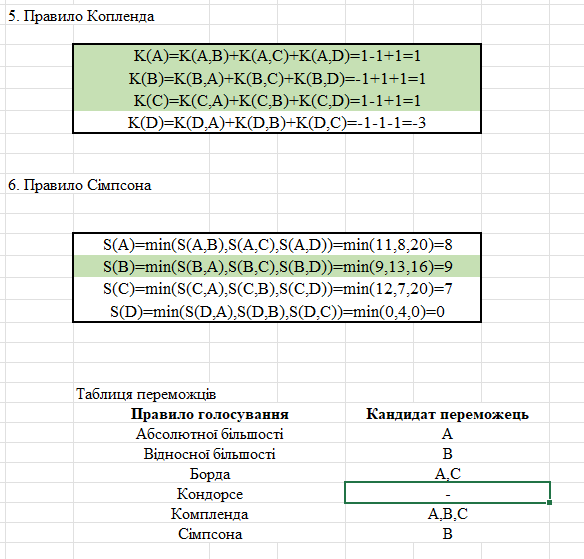
D. реєстрація гомосексуальних шлюбів

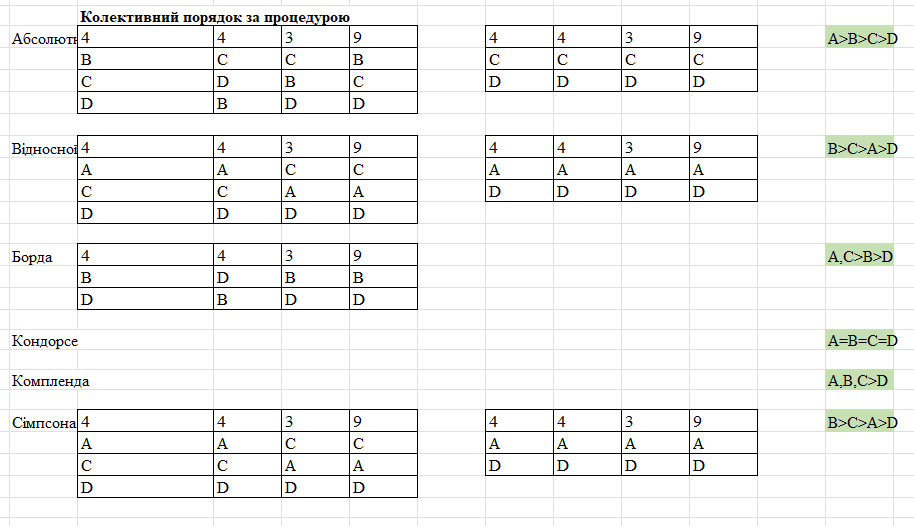
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кількість голосів | 4 | 4 | 3 | 9 |
| Впорядкування  кандидатів | A  B  C  D | A  C  D  B | C  A  B  D | B  C  A  D |

**Розв’язок в Excel**









Розв’язок в Python.

import numpy as np  
from sklearn.preprocessing import LabelBinarizer  
  
def relative\_majority(e, votes):  
 return (e[0] \* votes).sum(axis=1)  
  
  
def encoded(matrix, label\_binarizer):  
 encoded = label\_binarizer.fit\_transform(matrix.flatten())  
 encoded = encoded.reshape(matrix.shape + (4,))  
 encoded = encoded.transpose((0, 2, 1))  
 return encoded  
  
  
def relative(labels, e, votes):  
 relative\_majority\_score = relative\_majority(e, votes)  
 relative\_majority\_result = labels[relative\_majority\_score.argmax()]  
 return relative\_majority\_result  
  
  
def absolute(labels, matrix, votes, encoded, label\_binarizer):  
 relative\_majority\_score = (encoded[0] \* votes).sum(axis=1)  
 sorted\_result = np.sort(relative\_majority\_score)[-2:]  
 indices = np.where(np.isin(relative\_majority\_score, sorted\_result))[0]  
 second\_indices = np.argmax(np.isin(matrix, labels[indices]), axis=0)  
 top2\_vector = label\_binarizer.transform(  
 matrix[second\_indices, np.arange(matrix.shape[1])]).T \* votes  
 absolute\_score = top2\_vector.sum(axis=1)  
 absolute\_result = labels[absolute\_score.argmax()]  
 return absolute\_result  
  
  
def borda(encoded, votes, labels):  
 mul\_array = encoded \* votes  
 arr\_score = mul\_array \* np.array([3, 2, 1, 0])[:, np.newaxis, np.newaxis]  
 bord\_score = arr\_score.sum(axis=2).sum(axis=0)  
 bord\_result = labels[bord\_score.argmax()]  
 return bord\_result  
  
  
def condorse(matrix, votes, labels):  
 unique\_chars = np.unique(matrix)  
 sorted\_chars = np.sort(unique\_chars)  
  
 char\_to\_num = {char: i for i, char in enumerate(sorted\_chars)}  
 num\_matrix = np.vectorize(char\_to\_num.get)(matrix)  
 matrix\_cond = np.zeros((4, 4))  
  
 for i, col in enumerate(num\_matrix.T):  
 for x in range(col.shape[0]):  
 for y in range(x + 1, col.shape[0]):  
 matrix\_cond[col[x]][col[y]] += votes[i]  
  
 for i, e1 in enumerate(labels):  
 winner = True  
 for j in range(labels.shape[0]):  
 if i != j:  
 if matrix\_cond[i][j] < matrix\_cond[j][i]:  
 winner = False  
 if winner:  
 matrix\_result = e1  
 return matrix\_result  
  
  
def coplend(matrix, votes, labels):  
 unique\_chars = np.unique(matrix)  
 sorted\_chars = np.sort(unique\_chars)  
  
 char\_to\_num = {char: i for i, char in enumerate(sorted\_chars)}  
 num\_matrix = np.vectorize(char\_to\_num.get)(matrix)  
 matrix\_cond = np.zeros((4, 4))  
  
 for i, col in enumerate(num\_matrix.T):  
 for x in range(col.shape[0]):  
 for y in range(x + 1, col.shape[0]):  
 matrix\_cond[col[x]][col[y]] += votes[i]  
  
 cop\_score = np.zeros(matrix.shape[0])  
 for i, e1 in enumerate(labels):  
 for j in range(labels.shape[0]):  
 if i != j:  
 if matrix\_cond[i][j] < matrix\_cond[j][i]:  
 cop\_score[i] -= 1  
 elif matrix\_cond[i][j] > matrix\_cond[j][i]:  
 cop\_score[i] += 1  
  
 cop\_result = labels[cop\_score.argmax()]  
 return cop\_result  
  
  
def simpson(matrix, votes, labels):  
 unique\_chars = np.unique(matrix)  
 sorted\_chars = np.sort(unique\_chars)  
  
 char\_to\_num = {char: i for i, char in enumerate(sorted\_chars)}  
 num\_matrix = np.vectorize(char\_to\_num.get)(matrix)  
 matrix\_cond = np.zeros((4, 4))  
  
 for i, col in enumerate(num\_matrix.T):  
 for x in range(col.shape[0]):  
 for y in range(x + 1, col.shape[0]):  
 matrix\_cond[col[x]][col[y]] += votes[i]  
  
 data\_no\_zeros = np.where(matrix\_cond == 0, np.nan, matrix\_cond)  
 simpson\_score = np.nanmin(data\_no\_zeros, axis=1)  
 simpson\_result = labels[simpson\_score.argmax()]  
  
 return simpson\_result  
  
  
def main():  
 label\_binarizer = LabelBinarizer()  
  
 matrix = np.array([['A', 'A', 'C', 'B'],  
 ['B', 'C', 'A', 'C'],  
 ['C', 'D', 'B', 'A'],  
 ['D', 'B', 'D', 'D']])  
 votes = np.array([4, 4, 3, 9])  
  
 e = encoded(matrix, label\_binarizer)  
  
 labels = np.array(['A', 'B', 'C', 'D'])  
  
 a = absolute(labels, matrix, votes, e, label\_binarizer)  
 print("За правилом абсолютної більшості:", a)  
  
 r = relative(labels, e, votes)  
 print("За правилом відносної більшості:", r)  
  
 borda\_res = borda(e, votes, labels)  
 print("За правилом Борда:", borda\_res)  
  
 condorse\_res = condorse(matrix, votes, labels)  
 print("За правилом Кондорсе:", condorse\_res)  
  
 kop\_res = coplend(matrix, votes, labels)  
 print("За правилом Копленда:", kop\_res)  
  
 simpson\_res = simpson(matrix, votes, labels)  
 print("За правилом Сімпсона:", simpson\_res)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Висновок**

В результаті виконання даної лабораторної роботи, я попрацював з методами голосування, навчився розв’язувати задачі голосування порядку денного.

**Контрольні запитання**

1. Які аксіоми визначають апріорі «розумні» «логічні» вимоги до правил голосування?

* Парето-принцип: Якщо всі учасники виборів віддали перевагу варіанту А перед варіантом B, то колектив також повинен віддати перевагу A перед B.
* Транзитивність: Якщо колектив вибрав варіант A перед B і варіант B перед C, то він також повинен вибрати A перед C.

1. Які аксіоми визначають властивості голосування?

* Незалежність від альтернатив: Результат голосування не повинен змінюватися відносно альтернатив, які не беруть участі у порівнянні.
* Незалежність від неважливих альтернатив: Результат голосування має залишатися незмінним при зміні рейтингу альтернатив, які не отримали підтримку.

1. Яке правило голосування відповідає критерію утилітаризму, а яке критерію егалітаризму функції колективної корисності? В чому полягає суть цих критеріїв?

* Утилітаризм: Максимізація загальної корисності. Відповідає принципу більшості.
* Егалітаризм: Максимізація корисності для найменш покращеного члена групи. Відповідає принципу максимального мінімуму.

1. Які є парадокси голосування? Які можуть бути маніпуляції при колективному голосуванні?

* Парадокс вибору альтернатив: Може існувати така ситуація, коли колектив вибрав би іншу альтернативу, якщо б йому було запропоновано вибір із меншею кількістю альтернатив.
* Маніпуляції: Учасники можуть використовувати стратегії, щоб спрямовувати голоси на свою користь, використовуючи недоліки певних систем голосування.

1. Які правила голосування застосовуються в ситуаціях: спортивних змагань, вибору депутатів, вибору президента?

* Спортивні змагання: Зазвичай застосовується система "більшість голосів" або "ранжування", де перемагає альтернатива, яка отримала найбільше голосів або найвищий рейтинг.
* Вибір депутатів: Це може бути здійснено за допомогою систем мажоритарного голосування (переважної кількості голосів) або пропорційного представництва (голоси розподіляються пропорційно підтримці партій).
* Вибір президента: Це може включати в себе використання системи відкритого голосування або системи голосування з виборами у два тури.