# **КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені**

# **ТАРАСА ШЕВЧЕНКА**



## **ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №7**

# **з курсу**

**«Системний аналіз та теорія прийняття рішень»**

*Студента 3 курсу*

*групи ПП-31 спеціальності 122 «Комп'ютерні науки» ОП «Прикладне програмування»*

Момотюка Михайла Тарасовича

*Викладач:*

Плескач В.Л.

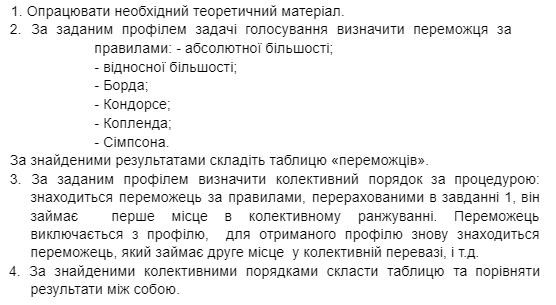
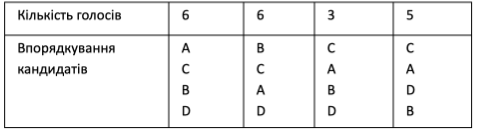
Білий Р.О.

## **Київ – 2023**

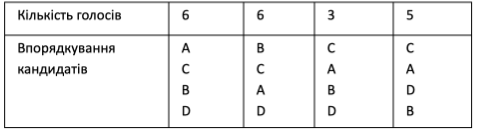
**Тема роботи:** Методи голосування.

**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

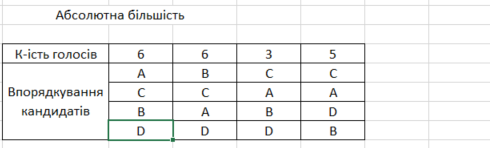
**Завдання:**

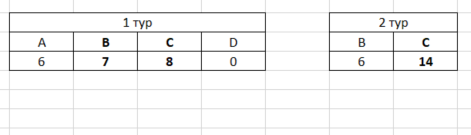
  


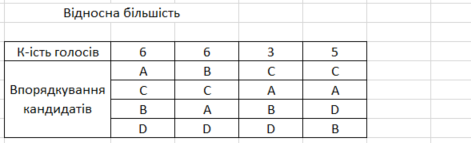
**Хід роботи**

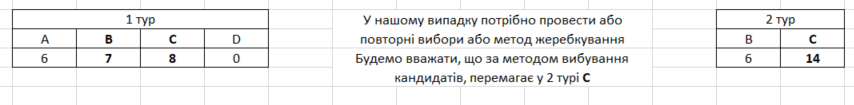


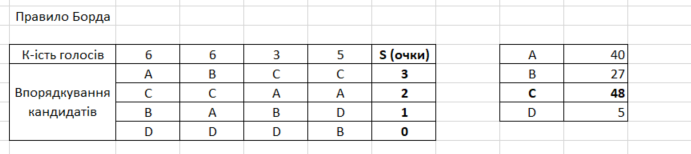
Проводимо розрахунки в Excel:

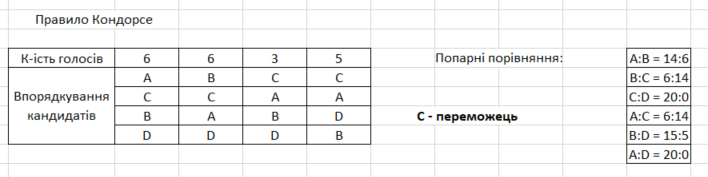


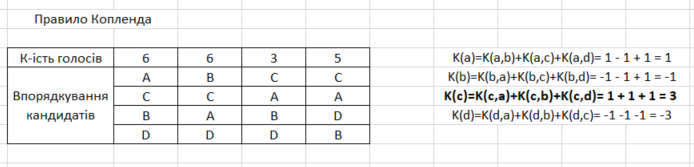


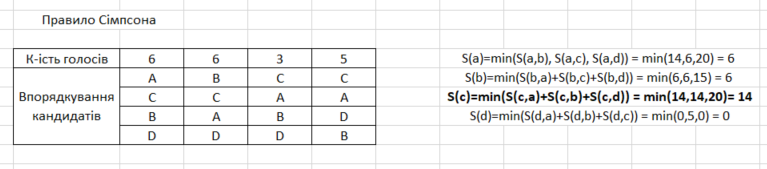










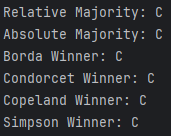




**Лістинг програми:**

import numpy as np  
from sklearn.preprocessing import LabelBinarizer  
  
  
def relative\_majority(e, votes):  
 return np.sum(e[0] \* votes, 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[np.argmax(relative\_majority\_score)]  
 return relative\_majority\_result  
  
  
def absolute(labels, matrix, votes, encoded, label\_binarizer):  
 relative\_majority\_score = np.sum(encoded[0] \* votes, 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[np.argmax(absolute\_score)]  
 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[np.argmax(bord\_score)]  
 return bord\_result  
  
  
def condorcet(matrix, votes):  
 unique\_labels = np.unique(matrix)  
 label\_to\_num = {label: i for i, label in enumerate(unique\_labels)}  
  
 num\_matrix = np.vectorize(label\_to\_num.get)(matrix)  
 matrix\_cond = np.zeros((len(unique\_labels), len(unique\_labels)))  
  
 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]  
  
 return matrix\_cond  
  
  
def find\_winner(matrix\_cond, labels):  
 for i, e1 in enumerate(labels):  
 winner = all(matrix\_cond[i][j] >= matrix\_cond[j][i] for j in range(len(labels)) if i != j)  
 if winner:  
 return e1  
  
  
def coplend(matrix, votes, labels):  
 matrix\_cond = condorcet(matrix, votes)  
 cop\_score = np.zeros(len(labels))  
 for i, e1 in enumerate(labels):  
 for j in range(len(labels)):  
 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[np.argmax(cop\_score)]  
 return cop\_result  
  
  
def simpson(matrix, votes, labels):  
 matrix\_cond = condorcet(matrix, votes)  
 data\_no\_zeros = np.where(matrix\_cond == 0, np.nan, matrix\_cond)  
 simpson\_score = np.nanmin(data\_no\_zeros, axis=1)  
 simpson\_result = labels[np.argmax(simpson\_score)]  
 return simpson\_result  
  
  
def main():  
 label\_binarizer = LabelBinarizer()  
  
 matrix = np.array([['A', 'B', 'C', 'C'],  
 ['C', 'C', 'A', 'A'],  
 ['B', 'A', 'B', 'D'],  
 ['D', 'D', 'D', 'B']])  
 votes = np.array([6, 6, 3, 5])  
  
 enc = encoded(matrix, label\_binarizer)  
  
 labels = np.array(['A', 'B', 'C', 'D'])  
  
 rel = relative(labels, enc, votes)  
 print("Relative Majority:", rel)  
 absol = absolute(labels, matrix, votes, enc, label\_binarizer)  
 print("Absolute Majority:", absol)  
 borda\_res = borda(enc, votes, labels)  
 print("Borda Winner:", borda\_res)  
 condorcet\_res = find\_winner(condorcet(matrix, votes), labels)  
 print("Condorcet Winner:", condorcet\_res)  
 cop\_res = coplend(matrix, votes, labels)  
 print("Copeland Winner:", cop\_res)  
 simpson\_res = simpson(matrix, votes, labels)  
 print("Simpson Winner:", simpson\_res)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Результат:



**Висновок:** У ході лабораторної роботи я вивчив методи голосування і дослідження їх властивостей; опанував методику обробки профілів колективного голосування, знаходження переможця та відновлення колективного ранжування та допомогою відповідних методів абсолютної та відносної більшості, правил Борда, Кондорсе, Копленда і Сімпсона.