Міністерство освіти та науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та

Комп’ютерних технологій

**Звіт**

Про виконання лабораторної роботи №3

“Операції над нечіткими множинами”

Виконав:

Студент групи ФеІ-44

Сапанюк М.І.

Перевірила:

Притула М.

Львів 2022

**Мета:**

Ознайомитися з основними операціями над нечіткими множинами та їх властивостями.

**Хід роботи:**

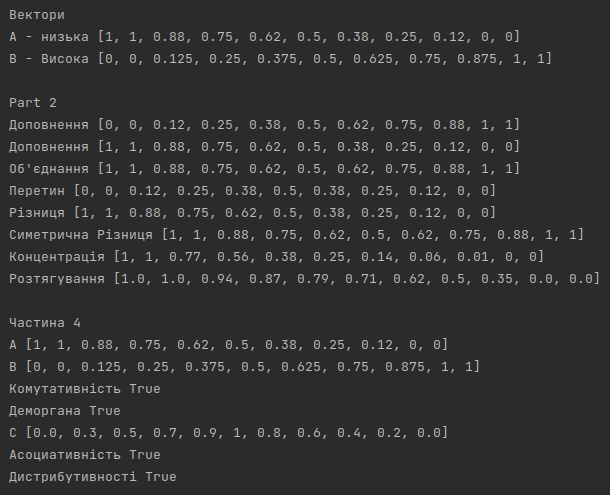
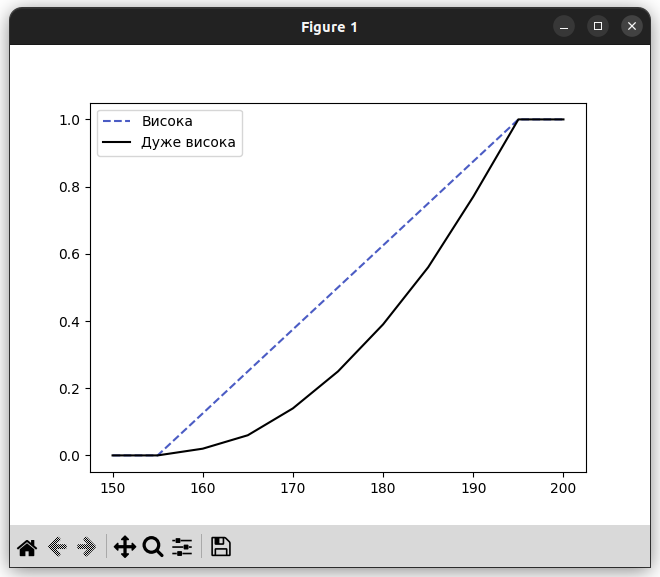
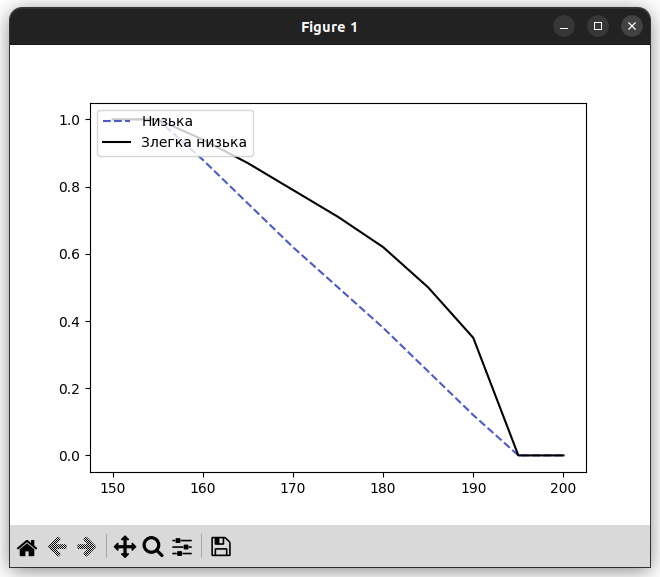
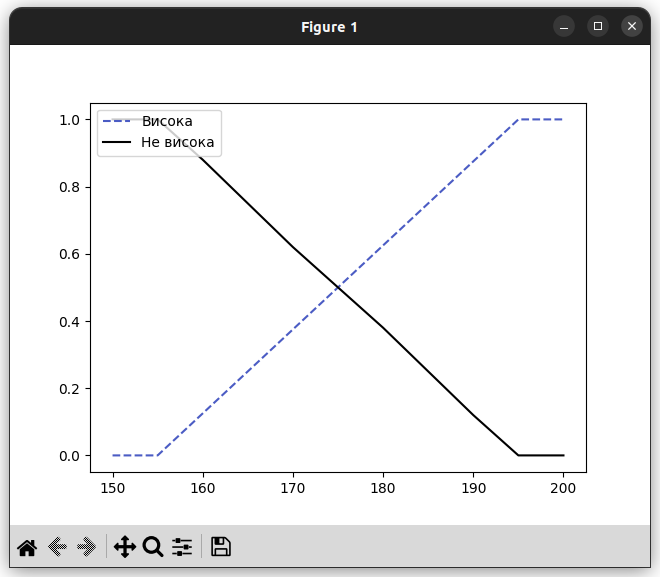
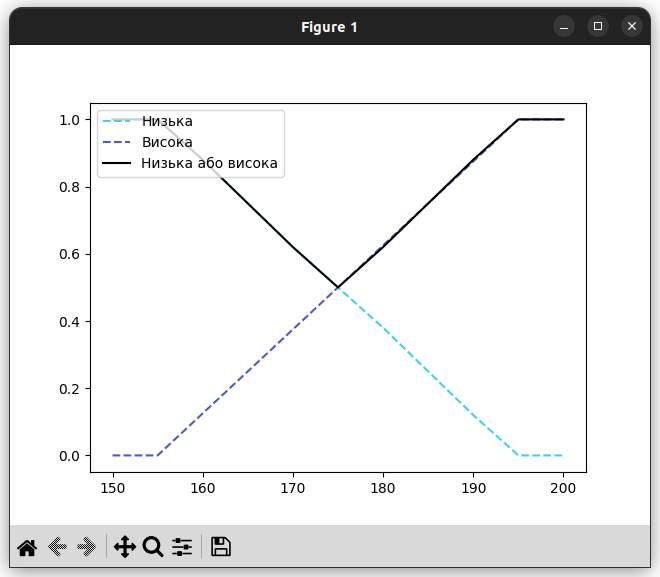
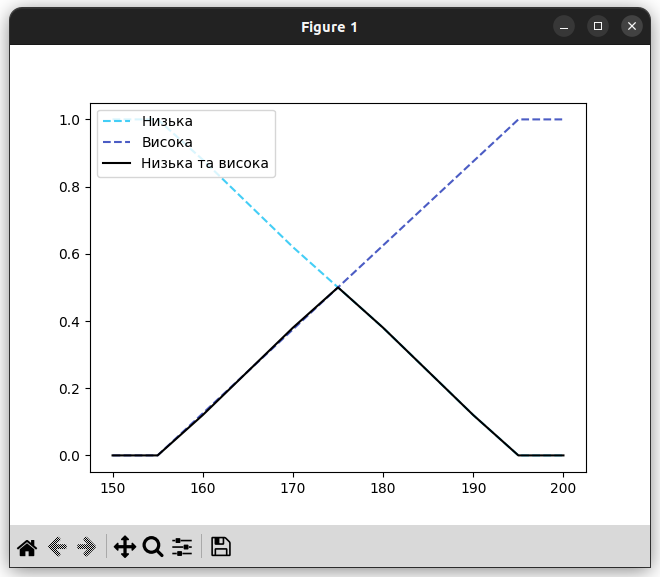
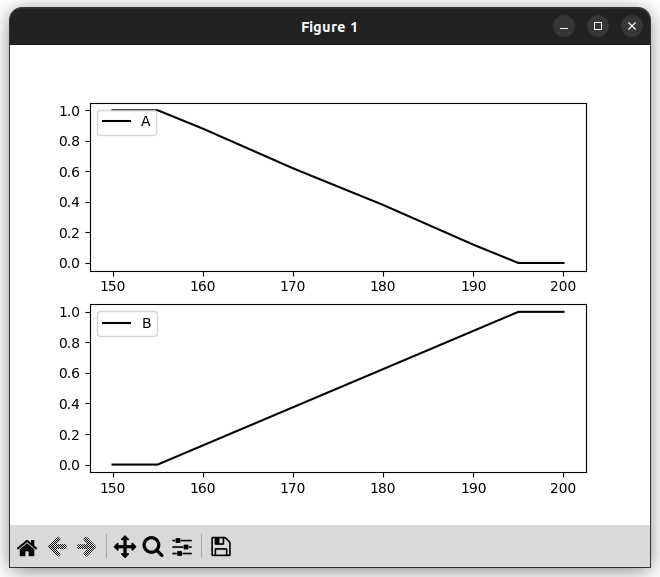
1. Задати функції приналежності нечітких множин А = {низька людина} і B = {висока людина}, заданих на універсумі [150, 200]. Побудувати їх графіки.

2. Запрограмувати реалізацію операцій доповнення, перетину, об’єднання, різниці, симетричної різниці, концентрування та розтягування нечітких множин.

3. За допомогою операцій над нечіткими множинами побудувати графіки функцій приналежності, які характеризують висловлювання “низька і висока людина”, “низька або висока людина”, “невисока людина”, “злегка низька людина”, “дуже висока людина”.

4. Перевірити виконання основних властивостей операцій над нечіткими множинами: комутативності, асоціативності, дистрибутивності, інволюції та виконання законів де Моргана.

**Виконання завдання:**



**Висновок:**

Виконавши лабораторну роботу, ознайомився з основними операціями над нечіткими множинами та їх властивостями, та запрограмував їх.

**Додаток:**

from matplotlib import pyplot as plt  
  
  
def addition(array: list[float]) -> list[float]:  
 temp: list[float] = [round(1 - x, 2) for x in array]  
 return temp  
  
  
def union(array1: list[float], array2: list[float]) -> list[float]:  
 temp: list[float] = [round(array1[i], 2) if array1[i] > array2[i] else round(array2[i], 2) for i in  
 range(0, len(array1))]  
 return temp  
  
  
def intersection(array1: list[float], array2: list[float]) -> list[float]:  
 temp: list[float] = [round(array1[i], 2) if array1[i] < array2[i] else round(array2[i], 2) for i in  
 range(0, len(array1))]  
 return temp  
  
  
def diff(array1: list[float], array2: list[float]) -> list[float]:  
 return intersection(array1, addition(array2))  
  
  
def scale2(array1: list[float]) -> list[float]:  
 temp: list[float] = [round(x \*\* 2, 2) for x in array1]  
 return temp  
  
  
def scale05(array1: list[float]) -> list[float]:  
 temp: list[float] = [round(x \*\* 0.5, 2) for x in array1]  
 return temp  
  
  
def aprocZ(x: int, a: int, b: int) -> float:  
 if x <= a:  
 return 1  
 elif a < x < b:  
 return round((b - x) / (b - a), 2)  
 else:  
 return 0  
  
  
def aprocS(x: int, a: int, b: int) -> float:  
 if x <= a:  
 return 0  
 elif a < x < b:  
 return (x - a) / (b - a)  
 else:  
 return 1  
  
  
def part2(array\_mu1: list[float], array\_mu2: list[float]) -> None:  
 print("\nPart 2")  
 print("Доповнення", addition(array\_mu1))  
 print("Доповнення", addition(array\_mu2))  
 print("Об'єднання", union(array\_mu1, array\_mu2))  
 print("Перетин", intersection(array\_mu1, array\_mu2))  
 print("Різниця", diff(array\_mu1, array\_mu2))  
 print("Симетрична Різниця", diff(union(array\_mu1, array\_mu2), intersection(array\_mu1, array\_mu2)))  
 print("Концентрація", scale2(array\_mu1))  
 print("Розтягування", scale05(array\_mu1))  
  
  
def part3(array\_label: list[int], array\_mu1: list[float], array\_mu2:  
list[float]) -> None:  
 plt.plot(array\_label, array\_mu1, color='#44cef6', linestyle="--", label="Низька")  
 plt.plot(array\_label, array\_mu2, color='#4b5cc4', linestyle="--", label="Висока")  
 plt.plot(array\_label, intersection(array\_mu2, array\_mu1), color="#000000", label="Низька та висока")  
 plt.legend(loc=2)  
 plt.show()  
 plt.plot(array\_label, array\_mu1, color='#44cef6', linestyle="--", label="Низька")  
 plt.plot(array\_label, array\_mu2, color='#4b5cc4', linestyle="--", label="Висока")  
 plt.plot(array\_label, union(array\_mu1, array\_mu2), color="#000000", label="Низька або висока")  
 plt.legend(loc=2)  
 plt.show()  
 plt.plot(array\_label, array\_mu2, color='#4b5cc4', linestyle="--", label="Висока")  
 plt.plot(array\_label, addition(array\_mu2), color="#000000", label="Не висока")  
 plt.legend(loc=2)  
 plt.show()  
 plt.plot(array\_label, array\_mu1, color='#4b5cc4', linestyle="--", label="Низька")  
 plt.plot(array\_label, scale05(array\_mu1), color="#000000", label="Злегка низька")  
 plt.legend(loc=2)  
 plt.show()  
 plt.plot(array\_label, array\_mu2, color='#4b5cc4', linestyle="--", label="Висока")  
 plt.plot(array\_label, scale2(array\_mu2), color="#000000", label="Дуже висока")  
 plt.legend(loc=2)  
 plt.show()  
  
  
def commutativity(array\_mu1: list[float], array\_mu2: list[float]) -> bool:  
 temp: list[float] = intersection(array\_mu1, array\_mu2)  
 temp1: list[float] = intersection(array\_mu2, array\_mu1)  
 temp2: list[float] = union(array\_mu2, array\_mu1)  
 temp3: list[float] = union(array\_mu1, array\_mu2)  
 return temp == temp1 and temp2 == temp3  
  
  
def demorgan(array\_mu1: list[float], array\_mu2: list[float]) -> bool:  
 array\_mu1\_adit: list[float] = addition(array\_mu1)  
 array\_mu2\_adit: list[float] = addition(array\_mu2)  
 return addition(intersection(array\_mu1, array\_mu2)) == union(array\_mu1\_adit, array\_mu2\_adit) and addition(  
 union(array\_mu1, array\_mu2)) == intersection(array\_mu1\_adit, array\_mu2\_adit)  
  
  
def associativity(array\_mu1: list[float], array\_mu2: list[float], array\_mu3:  
list[float]) -> bool:  
 return intersection(intersection(array\_mu1, array\_mu2), array\_mu3) == intersection(array\_mu1,  
 intersection(array\_mu2,  
 array\_mu3)) and union(  
 union(array\_mu1, array\_mu2), array\_mu3) == union(array\_mu1, union(array\_mu2, array\_mu3))  
  
  
def distributivity(array\_mu1: list[float], array\_mu2: list[float], array\_mu3:  
list[float]) -> bool:  
 return intersection(array\_mu1, union(array\_mu2, array\_mu3)) == union(intersection(array\_mu1, array\_mu2),  
 intersection(array\_mu1, array\_mu3)) and union(  
 array\_mu1, intersection(array\_mu2, array\_mu3)) == intersection(union(array\_mu1, array\_mu2),  
 union(array\_mu1, array\_mu3))  
  
  
def part4(array\_mu1: list[float], array\_mu2: list[float]) -> None:  
 print("\nЧастина 4")  
 print("A", array\_mu1)  
 print("B", array\_mu2)  
 print("Комутативність", commutativity(array\_mu1, array\_mu2))  
 print("Деморгана", demorgan(array\_mu1, array\_mu2))  
 array\_mu3: list[float] = [0.0, 0.3, 0.5, 0.7, 0.9, 1, 0.8, 0.6, 0.4, 0.2, 0.0]  
 print("C", array\_mu3)  
 print("Асоциативність", associativity(array\_mu1, array\_mu2, array\_mu3))  
 print("Дистрибутивності", distributivity(array\_mu1, array\_mu2, array\_mu3))  
  
  
array\_label: list[int] = [x for x in range(150, 205, 5)]  
array\_mu1: list[float] = [aprocZ(x, 155, 195) for x in array\_label]  
array\_mu2: list[float] = [aprocS(x, 155, 195) for x in array\_label]  
print("Вектори")  
print("A - низька", array\_mu1)  
print("B - Висока", array\_mu2)  
fig, ax = plt.subplots(2)  
ax[0].plot(array\_label, array\_mu1, color='#000000', label='A')  
ax[0].legend(loc=2)  
ax[1].plot(array\_label, array\_mu2, color='#000000', label='B')  
ax[1].legend(loc=2)  
plt.show()  
part2(array\_mu1, array\_mu2)  
part3(array\_label, array\_mu1, array\_mu2)  
part4(array\_mu1, array\_mu2)