ДИСКРЕТНА МАТЕМАТИКА

ЛАБОРАТОРНІ РОБОТИ

МОДУЛЬ 1

12 балів

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Практичне засвоєння дискретної математики є ключовим етапом навчання майбутніх фахівців у галузі інформаційних технологій. Теоретичні основи, які розглядаються в межах даного посібника, закладають фундамент для розуміння основних понять і методів. Лабораторний практикум, дозволить поглибити отримані знання, закріпити їх через практичні завдання та набути навичок застосування інструментів програмування для розв'язання задач дискретної математики.

Лабораторні роботи пропонується виконувати з використанням мови програмування Python та інтерактивного середовища Jupyter Notebook. Такий підхід обрано через його зручність, інтерактивність і широкі можливості для реалізації математичних концепцій та алгоритмів.

Мова програмування Python є популярним інструментом у сфері комп'ютерних наук і програмної інженерії завдяки її зрозумілому синтаксису та потужному функціоналу. У контексті дискретної математики Python є ефективним засобом для виконання складних обчислень, візуалізації даних та автоматизації процесів.

Python надає багатий набір бібліотек для математичних обчислень і роботи з множинами, графами, послідовностями тощо. Використання цієї мови у практикумі допоможе не лише виконувати лабораторні завдання, а й підготуватися до застосування програмування в більш складних наукових і прикладних задачах.

Jupyter Notebook є інтерактивним середовищем для виконання програмного коду, яке поєднує текстові пояснення, результати виконання та графічну візуалізацію в одному документі. Ця особливість робить Jupyter Notebook ідеальним інструментом для навчання та досліджень.

У рамках лабораторного практикуму Jupyter Notebook використовується як основне середовище для розробки та тестування програмного коду. Воно дозволяє:

- інтерактивно виконувати окремі частини коду;
- переглядати результати обчислень у реальному часі;
- створювати чітко структуровані документи з теоретичними поясненнями та практичними розв'язками.

Однак студенти можуть обирати й інші мови програмування чи середовища для виконання пропонованих завдань. При цьому реалізація завдань має відповідати поставленим цілям і забезпечувати коректне виконання усіх математичних обчислень.

Звіт про виконання лабораторної роботи необхідно оформлювати на аркушах формату А4 (210×297 мм), дотримуючись таких розмірів берегів: лівий – 30 мм, верхній і нижній – 20 мм, правий – 10 мм. Шрифт документа має бути Times New Roman розміром 14 пт, з міжрядковим інтервалом 1,5.

Абзацний відступ – 1,25 см, текст вирівняний по ширині. Нумерація сторінок починається з першої сторінки (титульна сторінка не нумерується, але вважається першою).

Усі графіки, діаграми та схеми повинні бути підписані, та розміщені після їхнього згадування у тексті. Наприклад, "Рисунок 1 – Графік функції f(x)". Перед рисунком та після назви рисунку необхідно залишити порожній рядок.

Цифровий матеріал, який подається у вигляді таблиці, підписується, наприклад, «Таблиця 1 – Результати тестування програми» з абзацу, а другий рядок вирівнюється по лівому краю. Перед назвою таблиці та після таблиці необхідно залишити порожній рядок.

Звіт з виконання лабораторної роботи повинен містити такі основні складові:

1. Титульний аркуш.
2. Мета роботи.
3. Варіант та завдання для виконання.
4. Скріншот ручного виконання завдання.
5. Блок-схема алгоритму програми.
6. Опис програми.
7. Результати тестування програми.
8. Висновки.
9. Додатки (інструкція користувача, лістинг програми, тощо)

Лабораторна робота №1

Тема: Множини. Основні поняття та операції над множинами.

Мета роботи: Набути навиків реалізації основних операцій над множинами мовою програмування Python.

Методичні рекомендації

Множини – це одна з базових структур даних у Python, яка призначена для зберігання лише унікальних елементів. Вони забезпечують високу швидкодію при виконанні операцій, таких як об'єднання, перетин, різниця та симетрична різниця. Крім того, множини особливо зручні для швидкої перевірки наявності елемента в колекції.

Множини в Python створюються за допомогою конструктора `set()` або використанням фігурних дужок `{}` (табл. Л1.1).

Таблиця Л1.1 – Основні операції над множинами

Операція	Синтаксис
Створення множини *A*	`A = {1, 2, 3}` `B = set{1, 2, 3}`
Додавання елемента до множини *A*	`A.add(4)`
Видалення елемента з множини *A*	`A.remove(2)`
Об’єднання множин	`C = A | B` `C = A.union(B)`
Перетин множин	`C = A & B` `C = A.intersection(B)`
Різниця множин	`C = A - B` `C = A.difference(B)`
Симетрична різниця	`A ^ B` `A.symmetric_difference(B)`
Перевірка чи є *A* підмножиною *B*	`A <= B` `A.issubset(B)`
Перевірка чи є *B* є надмножиною *A*	`B >= A` `B.issuperset(A)`

Елементи множини повинні бути незмінними (тобто типами, які можна хешувати, наприклад числа, строки, кортежі), однак сама множина є змінною, тобто її можна доповнювати чи змінювати.

Python підтримує такі основні операції над множинами (табл. Л1.1):

1. Об'єднання множин (`|` або `union`) створює нову множину, яка містить усі унікальні елементи з обох множин.
2. Перетин множин (`&` або `intersection`) повертає елементи, спільні для двох множин.
3. Різниця множин (`-` або `difference`) містить лише ті елементи, які є в одній множині, але відсутні в іншій.
4. Симетрична різниця множин (`^` або `symmetric_difference`) створює множину, яка включає всі елементи, що є унікальними для кожної множини.

Python також підтримує функції для перевірки підмножин та надмножин, такі як `issubset` і `issuperset`. Зокрема,

- Оператор <= означає "підмножина або рівна" (аналог `issubset`).
- Оператор < означає "строго підмножина" (тобто підмножина, але не рівна).
- Оператор >= означає "надмножина або рівна" (аналог `issuperset`).
- Оператор > означає "строго надмножина" (надмножина, але не рівна).

Операції над множинами дозволяють виконувати багато практичних завдань, таких як порівняння даних; фільтрація унікальних елементів; знаходження спільних або відмінних характеристик у наборах даних, тощо.

Завдання до лабораторної роботи

Завдання 1. Задайте універсальну множину *U*, яка містить 10 довільних елементів. Для парних варіантів елементами множини *U* є літери алфавіту, а для непарних – натуральні числа. Використовуючи універсальний набір елементів *U* створити довільно множини *A, B, C* та *D*, за умови що $|A| = n$, $|B| = m$, $|C| = (n - k)$, $|D| = 5$ (n – кількість літер імені студента, m – номер групи, k – номер підгрупи). Для заданих множин *A, B, C* та *D* виконайте наведені у табл. 1.1 (індивідуальні практичні задання №1) операції та перевірте результат ручним розрахунком.

Таблиця 1.1 – Варіанти завдань

1. $(A \cup B) \cap (D \setminus C)$;	31. $(\bar{A} \cup \bar{B}) \cup (\bar{C} \cap D)$;
2. $(A \setminus B \setminus \bar{C}) \cup D$;	32. $\big((A \cap \bar{C}) \cap D\big) \cup (B \cup \bar{A})$;
3. $(\overline{A \cup D}) \setminus (B \setminus C)$;	33. $\Big(\big(\overline{A \cap \bar{D}}\big) \cup (B \cup C)\Big) \cap \bar{A}$;
4. $(A \setminus D) \cup (B \cap C)$;	34. $(\overline{\bar{B} \cup C}) \cap (\overline{\bar{A} \cup D}) \cap \bar{A}$;
5. $(A \setminus C)\ \Delta\ (B \cup D)$;	35. $\big((A \setminus B) \cup D\big) \setminus (C \cup D)$;
6. $(A \cup C) \cup (\bar{B} \setminus C)$;	36. $\big((A \cap \bar{C}) \cap D\big) \cup \bar{B}$;

Продовження табл. 1.1

7. $\big((A \cup B) \cup C\big) \cap (B \cup \bar{A})$;	37. $(A \cap B) \cup \big(\overline{\bar{C} \cap D}\big)$;
8. $\big((A \cap B) \cup C\big) \cup D$;	38. $\big((A \backslash B) \cup (\overline{B \cap D})\big) \backslash C$;
9. $(\bar{A} \cup B) \cup (B \cup D)$;	39. $(\bar{A} \cup B)\, \Delta\, (B \cap C)$;
10. $(A \cup \bar{B}) \cap (C \backslash D)$;	40. $(A \cup \bar{D}) \cap (\overline{B \cup C})$;
11. $(A \cap B) \backslash (\overline{C \cup D})$;	41. $\overline{(A \cup B) \cup (\overline{B \cap C})} \cup (\bar{A} \cup D)$;
12. $\big((A \backslash B) \cup (\overline{C \cap A})\big) \cup (D \cup \bar{B})$;	42. $(A \cap B) \cup (\bar{D} \backslash C)$;
13. $\big((A \backslash C) \backslash (B \cap D)\big) \cup A$;	43. $(A \cup B) \cup (D \backslash \bar{C})$;
14. $(\bar{B} \cup C) \cup (D \backslash A)$;	44. $\big((\overline{B \cap D}) \backslash (C \cup D)\big) \cup \bar{A}$;
15. $(A \cap D) \cap (B \cap \bar{C})$;	45. $(\overline{A \cup D}) \cap (B \cup C)$;
16. $\big(B \cap (\overline{D \cap C})\big) \cup A$;	46. $\big(\overline{\bar{A} \cup B}\big) \cup (\bar{C} \cap \bar{D})$;
17. $(A \cap B) \cup (C \cap \bar{D})$;	47. $\big(A \cup (B \cup C)\big) \cap \bar{D}$;
18. $(A \cap B) \cap \big(A \backslash (\overline{C \cap D})\big)$;	48. $\big((A \cap B) \cup (\overline{C \cap D})\big) \cap \bar{A}$;
19. $\big((B \cup D) \cap A\big) \cup \bar{C}$;	49. $\big((A \cup D) \cup C\big) \cap B$;
20. $\big((A \cup C) \backslash (B \cap D)\big) \cup (\overline{A \cap D})$;	50. $(B \cup C) \cup \big(C \cap (A \backslash \bar{D})\big)$;
21. $(\overline{A \cap B}) \cup \big(\bar{C} \cap \bar{D}\big)$;	51. $\overline{(B \cap C) \cup \bar{D}} \cap (A \cap D)$;
22. $\big((A \backslash \bar{C}) \cap (B \cap D)\big) \cup D$;	52. $(A \cap B) \backslash \big(C \cup (D \cup \bar{A})\big)$;
23. $(A \backslash D) \cup (\bar{B} \cup C)$;	53. $(\overline{A \cup B}) \cup (A \cap D) \cup \big(\overline{\bar{A} \cup \bar{C}}\big)$;
24. $\big((A \cup B) \cap (C \cup D)\big) \cup (\bar{A} \cap B)$;	54. $\big(C \cup (\bar{A} \cup B)\big) \backslash (A \cap D)$;
25. $\big((A \cap B) \cup C\big) \cup (\overline{D \cap A})$;	55. $\big(\overline{\bar{C} \cap A}\big) \cap (B \cup D)$;
26. $\big((A \backslash B) \cup C\big) \cap \bar{D}$;	56. $\Big(A \cup \big(\overline{\bar{B} \cup A}\big) \cup (\bar{C} \cap D)\Big)$;
27. $A \cap \Big(D \cup \big(\bar{\bar{B}} \cap C\big)\Big)$;	57. $(A \backslash D) \cap \big(B \cap (C \cup \bar{A})\big)$;
28. $\big((\overline{A \cup B}) \cup (C \cap D)\big) \cup (A \cap B)$;	58. $\big((B \cup D)\, \Delta\, A\big) \cup \bar{C}$;
27. $(A \cup B) \cup \big(\overline{C \cap \bar{D}}\big)$;	59. $(A \cup D)\, \Delta\, (\bar{B} \backslash C)$;
28. $(A\, \Delta\, B) \cup \big(\overline{C \backslash \bar{D}}\big)$;	60. $(D \cup C) \cap (\mathrm{A}\, \Delta\, \mathrm{B}) \cup \bar{D}$.

Завдання 2. За допомогою мови програмування Python перевірте виконання індивідуальних практичних завдань №1.

Завдання 3. Використовуючи діаграми Венна (кола Ейлера), розв'яжіть задачу про порівняння трьох мобільних додатків.

Опитано n респондентів щодо використання трьох мобільних додатків: A – для навчання мов, B – для фізичних вправ, C – для фінансового планування. За результатами анкетування встановлено, що: n_1 респондентів використовують додаток A, n_2 респондентів використовують додаток B та n_3 респондентів використовують додаток C. Крім того, відомо що n_{12} респондентів використовують одночасно додатки A та B, n_{13} респондентів – додатки A та C, n_{23} респондентів – B та C, n_{123} респондентів використовують одночасно всі три додатки. Яка загальна кількість із опитаних респондентів використовують принаймні один з цих додатків? Скільки респондентів використовують лише один конкретний додаток? Скільки респондентів не користуються вказаними додатками? Результати опитування наведено у табл. Л1.2

Таблиця Л1.2 – Варіанти завдань

Варіант	n_1	n_2	n_3	n_{12}	n_{13}	n_{23}	n_{123}	n
1.	10	15	20	4	3	5	2	50
2.	12	18	22	5	4	6	3	50
3.	15	20	25	6	5	7	4	50
4.	8	12	16	3	2	4	1	40
5.	20	25	30	7	6	8	5	70
6.	18	22	28	6	5	7	4	70
7.	14	19	23	5	4	6	3	70
8.	9	14	18	4	3	5	2	50
9.	16	21	26	6	5	7	4	50
10.	10	13	17	3	2	4	1	40
11.	11	16	20	4	3	5	2	40
12.	13	17	21	5	4	6	3	40
13.	14	18	22	5	4	6	3	50
14.	19	23	27	7	6	8	5	100
15.	22	27	32	8	7	9	6	100
16.	17	21	25	6	5	7	4	55
17.	15	19	24	5	4	6	3	50
18.	20	26	31	8	7	9	6	80
19.	11	15	19	4	3	5	2	35
20.	18	23	28	7	6	8	5	60
21.	21	26	30	8	7	9	6	60
22.	12	17	22	5	4	6	3	50
23.	19	24	29	7	6	8	5	60
24.	13	18	23	5	4	6	3	42
25.	10	16	21	4	3	5	2	37
26.	14	19	24	6	5	7	4	50

27.	16	22	27	7	6	8	5	50
28.	17	23	28	7	6	8	5	60
29.	22	27	33	9	8	10	7	100
30.	16	21	26	6	5	7	4	50

Порядок виконання лабораторної роботи

Завдання 1.

1. У робочому вікні Jupyter.Notebook задаємо універсальну множину та вводимо вхідні параметри для визначення потужностей множин *A, B, C* та *D* (рис. Л1.1). Результат виконання програми наведено на рис. Л1.2.

```
# Задання універсальної множини U
U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

# Задання параметрів множин
name = input("Введіть ваше ім'я: ")
group = int(input("Введіть номер групи: "))
group_h = int(input("Введіть номер підгрупи: "))

# Обчислення розмірів множин
n = len(name)  # Потужність множини A
m = group  # Потужність множини B
k = len(name) - group_h # Потужність множини C

# Виведення результатів
print(f"Потужність множини A: |A| = {n}")
print(f"Потужність множини B: |B| = {m}")
print(f"Потужність множини C: |C| = {m}")
print(f"Потужність множини D: |D| = 5")
```

Рисунок Л1.1

```
Введіть ваше ім'я:  Сергій
Введіть номер групи:  3
Введіть номер підгрупи:  2
Потужність множини A: |A| = 6
Потужність множини B: |B| = 3
Потужність множини C: |C| = 3
Потужність множини D: |D| = 5
```

Рисунок Л1.2

2. Задаємо множини *A, B, C, D* визначеної потужності та обчислюємо згідно заданого варіанту невідому множину *X* (рис. Л1.3). Результат виконання програми наведено на рис. Л1.4.

```
# Задаємо множини, що відповідають умові
A = {1, 2, 3, 6, 7, 9}
B = {3, 8, 9}
C = {1, 9, 10}
D = {2, 4, 6, 7, 8}

# Виведення результатів
print(f"Універсальна множина U: {U}")
print(f"Множина A (|A| = {n}): {A}")
print(f"Множина B (|B| = {m}): {B}")
print(f"Множина C (|C| = {m}): {C}")
print(f"Множина D (|D| = 5): {D}")

# Виконуємо дії над множинами
X = (A | B)&(C - D)
print(f"X = (A ∪ B) ∩ (C \ D) = {X}")
```

Рисунок Л1.3

```
Множина A (|A| = 6): {1, 2, 3, 6, 7, 9}
Множина B (|B| = 3): {8, 9, 3}
Множина C (|C| = 3): {1, 10, 9}
Множина D (|D| = 5): {2, 4, 6, 7, 8}
X = (A ∪ B) ∩ (C \ D) = {1, 9}
```

Рисунок Л1.4

3. Перевіряємо результат виконання програми, виконавши ручний розрахунок множини X.

Завдання 2. Аналогічно виконуємо програмну перевірку виконання індивідуальних практичних завдань №1 для власного варіанту.

Завдання 3. Використаємо діаграми Венна (кола Ейлера) для розв'язку наступної задачі: серед 100 випускників школи, що брали участь у ЗНО, 19 складали біологію, 21 – хімію, 23 – географію, 7 – біологію і хімію, 9 – біологію і географію, 8 – хімію і географію. Скільки учнів не вибрали жодне із вказаних ЗНО, якщо всі вказані ЗНО вибрало 3 учні?

1. У робочому вікні Jupyter.Notebook імпортуємо бібліотеку Matplotlib та інструмент для побудови діаграм Венна (venn3):

```
from matplotlib_venn import venn3
import matplotlib.pyplot as plt
```

2. Вводимо вхідні дані задачі та позначаємо відповідні області на діаграмах Венна для візуалізації взаємозв'язків між множинами (рис. Л1.5). Загальний розподіл учасників ЗНО одержуємо на рис. Л1.6.

```
# Вихідні дані
students_total = 100
biology = 19
chemistry = 21
geography = 23
biology_and_chemistry = 7
biology_and_geography = 9
chemistry_and_geography = 8
all_three = 3

# Побудова діаграми Венна
venn = venn3(
    subsets={
        '100': biology - biology_and_chemistry - biology_and_geography + all_three,
        '010': chemistry - biology_and_chemistry - chemistry_and_geography + all_three,
        '001': geography - biology_and_geography - chemistry_and_geography + all_three,
        '110': biology_and_chemistry - all_three,
        '101': biology_and_geography - all_three,
        '011': chemistry_and_geography - all_three,
        '111': all_three
    },
    set_labels=('Біологія', 'Хімія', 'Географія')
)

plt.title("Розподіл учнів за вибраними предметами ЗНО")
plt.show()
```

Рисунок Л1.5

3. Аналіз діаграм на рис. Л1.6 дозволяє зробити висновок, що лише один предмет ЗНО складали 24 учні (лише біологію – 6 учнів, лише хімію та лише географію – по 9 учнів). Загалом 42 учасники обрали принаймні один предмет, тоді як жодної дисципліни не обрали 58 осіб.

Рисунок Л1.6 – Діаграми Венна

Контрольні питання

1) Як створити множину в Python? Наведіть приклади.
2) Чим відрізняються методи add() і update() при роботі з множинами?
3) Як перевірити, чи є один набір елементів підмножиною або надмножиною іншого?
4) Які функції або методи використовуються для видалення елементів із множини?
5) Що станеться, якщо спробувати додати до множини елемент, який уже існує?
6) Чи можна використовувати списки або інші змінні типи даних як елементи множини?
7) Як знайти кількість елементів у множині? Яка функція для цього використовується?
8) Як реалізувати перевірку наявності елемента в множині?
9) Що таке порожня множина, і як її створити в Python?
10) Як можна використати множини для виявлення унікальних елементів у списку?
11) Які переваги множин у Python порівняно з іншими структурами даних?
12) Чим множини відрізняються від списків і словників у Python?
13) Як множини можна застосовувати для розв'язання реальних завдань?

Лабораторна робота №2

Тема: Задача про покриття. Основні методи розв’язання задачі про покриття на множинах.

Мета роботи: Набути практичних навичок реалізації основних методів розв’язання задачі про покриття на множинах за допомогою мови програмування Python.

Методичні рекомендації

Задача про покриття є моделлю великої кількості оптимізаційних задач дискретної математики. Теоретичні та практичні підходи що використовуються для її розв’язання детально розглянуто у третій темі даного посібника.

Метод повного перебору дозволяє отримати всі можливі покриття, включаючи надлишкові, для будь-якої задачі про покриття. Однак цей метод не є раціональним, оскільки потребує значних обчислювальних ресурсів.

Програмно реалізувати даний метод можна за допомогою функції `combinations`, яка дозволяє поступово перебрати всі можливі комбінації підмножин. Приклад коду з використанням даної функції наведено на рис. Л2.1.

```python
from itertools import combinations

# Універсальна множина
U = {1, 2, 3, 4, 5, 6, 7, 8}

# Підмножини
A = [
    {1, 2, 4, 6, 5},    # A1
    {1, 2, 3, 4},       # A2
    {2, 4, 5, 6, 7, 8}, # A3
    {1, 2, 4, 7, 8}     # A4
]

covered_sets = []

# Знайдемо всі комбінації підмножин, які покривають універсальну множину
for r in range(1, len(A) + 1):
    for subset_combination in combinations(A, r):
        union_result = set().union(*subset_combination)
        if union_result == U:
            covered_sets.append(subset_combination)

# Форматуємо результат для виводу у вигляді об'єднання підмножин
formatted_results = []
for combination in covered_sets:
    subsets_indices = [f"A{A.index(subset) + 1}" for subset in combination]
    formatted_results.append(" ∪ ".join(subsets_indices))

print("Покриттям є: ")
for result in formatted_results:
    print(result)
```

Рисунок Л2.1 – Метод повного перебору

На початку коду імпортується функція `combinations`, після чого задається універсальна множина U та всі підмножини A_i. Далі за допомогою циклу для кожної комбінації підмножин обчислюється їхнє об'єднання з подальшою перевіркою рівності U. Якщо об'єднання дорівнює U, то ця комбінація додається до списку покриттів. У результаті виконання коду подається список у вигляді об'єднання підмножин які є покриттям універсальної множини U (рис. Л2.2).

```
Покриттям U є:
A2 ∪ A3
A1 ∪ A2 ∪ A3
A1 ∪ A2 ∪ A4
A2 ∪ A3 ∪ A4
A1 ∪ A2 ∪ A3 ∪ A4
```

Рисунок Л2.2 – Результат виконання методу повного перебору

У представленому переліку (рис. Л2.2) наведено всі можливі покриття, включаючи надлишкові, для універсальної множини U.

Програмний код, зображений на рис. Л2.1, можна вдосконалити для визначення всіх безнадлишкових покриттів або побудови одного найкоротшого покриття.

Завдання до лабораторної роботи

Завдання 1. Розробити схему алгоритму та програму побудови всіх безнадлишкових покриттів методом повного та граничного перебору. Розрахувати ціну кожного безнадлишкового покриття.

Завдання 2. Розробити схему алгоритму та програму побудови найкоротшого покриття методом мінімального стовпчика - максимального рядка та методом ядерних рядків.

Порядок виконання лабораторної роботи

1. Для виконання лабораторної роботи необхідно обрати варіант, який відповідає індивідуальному практичному завданню №2.
2. Для обраного варіанту протестувати виконання коду, представленого на рис. Л2.1.
3. Внести зміни у код (рис. Л2.1) для побудови найкоротшого покриття.
4. Внести зміни у код (рис. Л2.1) для побудови безнадлишкових покриттів та розрахунку їхньої ціни.
5. Для парного варіанта розробити алгоритм та програму для побудови безнадлишкових покриттів методом граничного перебору.
6. Для непарного варіанта розробити алгоритм та програму для побудови найкоротшого покриття методом мінімального стовпчика - максимального рядка та методом ядерних рядків.

7. Навести результати тестування програмного коду та порівняти їх із результатами ручного розрахунку для індивідуального практичного завдання №2.

8. Зробити висновки щодо результатів застосування методів побудови покриття, відзначивши їхні ключові особливості та відмінності.

Варіанти завдань

B1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A	1						1			1
B			1		1				1	1
C		1		1					1	2
D		1					1	1		1
E	1		1		1			1		2
F	1		1			1	1			3
G		1		1	1			1	1	3
H	1			1		1				2

B2	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A		1	1				1	1		3
B		1	1	1	1					2
C						1		1	1	2
D	1			1	1	1			1	3
E	1		1	1						2
F	1	1				1				1
G					1		1		1	2
H				1		1				1

B3	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A						1	1		1	2
B		1	1		1	1	1			3
C					1	1		1		1
D	1			1				1	1	3
E	1	1	1	1						2
F			1	1			1			1
G	1	1			1					1
H		1				1				1

B4	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A	1	1				1				1
B		1	1	1	1					2
C						1	1	1		1
D		1	1				1		1	3
E	1		1	1						2
F	1			1	1	1		1		3
G					1			1	1	1
H	1					1				1

B5	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A		1			1			1		1
B		1				1	1		1	3
C	1		1	1	1			1		3
D			1	1			1			2
E	1		1			1	1			2
F	1			1		1				1
G					1			1	1	1
H			1					1		1

B6	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A	1				1				1	1
B						1	1	1	1	2
C			1			1	1			1
D		1	1					1		2
E		1	1		1		1		1	3
F	1			1		1				3
G		1		1	1					2
H		1						1		1

B7	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A			1	1			1			1
B	1				1		1			2
C		1	1	1					1	2
D		1		1	1		1	1		4
E		1				1		1		2
F	1		1			1			1	3
G					1			1	1	1
H		1		1					1	4

B8	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***a***
A				1	1		1			1
B	1					1			1	2
C	1			1	1			1	1	3
D			1					1	1	1
E		1	1			1	1			3
F	1	1		1		1	1			2
G		1			1			1		2
H				1			1			1

B9	1	2	3	4	5	6	7	8	9	*a*
A		1	1		1			1		3
B	1		1	1				1		2
C	1			1		1	1		1	4
D		1				1			1	1
E			1	1					1	2
F	1						1	1		2
G					1	1	1			1
H		1			1					3

B10	1	2	3	4	5	6	7	8	9	*a*
A		1		1				1		2
B			1			1			1	1
C		1			1		1		1	3
D	1			1			1		1	2
E	1		1		1					1
F	1		1	1		1		1		3
G		1						1		2
H							1	1	1	2

B11	1	2	3	4	5	6	7	8	9	*a*
A		1					1	1		1
B						1		1	1	2
C		1	1		1	1				3
D			1	1					1	2
E		1						1		1
F	1				1		1			1
G	1		1	1		1				2
H	1			1			1	1	1	2

B12	1	2	3	4	5	6	7	8	9	*a*
A	1			1		1	1		1	3
B	1				1		1		1	1
C		1				1	1			2
D		1	1		1			1		3
E	1		1	1	1					2
F		1					1			2
G	1					1		1		2
H			1	1					1	1

B13	1	2	3	4	5	6	7	8	9	*a*
A			1					1		1
B		1		1		1				2
C	1	1		1	1			1		3
D	1			1		1			1	2
E	1				1		1			1
F		1			1				1	2
G		1				1				2
H			1			1	1		1	3

B14	1	2	3	4	5	6	7	8	9	*a*
A		1				1		1		1
B		1		1			1		1	3
C				1	1			1	1	2
D			1			1			1	2
E		1		1						1
F					1		1			1
G	1		1	1						2
H	1		1		1	1		1		3

B15	1	2	3	4	5	6	7	8	9	*a*
A	1			1					1	2
B	1						1		1	1
C	1		1			1		1		3
D					1		1	1		2
E		1		1	1		1		1	4
F		1	1	1						1
G	1				1	1				1
H					1		1			2

B16	1	2	3	4	5	6	7	8	9	*a*
A	1	1						1		1
B		1		1	1		1	1		4
C		1		1		1				2
D					1	1	1			1
E			1				1	1		1
F	1		1			1			1	3
G				1	1				1	2
H		1					1	1		3

B17	1	2	3	4	5	6	7	8	9	*a*
A			1		1				1	1
B		1	1			1	1			3
C		1	1	1					1	2
D	1			1	1			1	1	3
E					1	1		1		1
F	1	1		1		1				2
G	1						1	1		2
H	1								1	1

B18	1	2	3	4	5	6	7	8	9	*a*
A					1		1	1		2
B	1			1		1				1
C	1	1	1							1
D		1			1	1			1	3
E			1		1		1		1	2
F	1		1	1	1			1		4
G				1				1	1	1
H	1		1							1

B19	1	2	3	4	5	6	7	8	9	*a*
A			1		1				1	1
B		1	1	1				1		3
C			1	1			1		1	3
D	1				1	1	1		1	4
E					1	1		1		2
F	1	1					1			1
G	1			1		1				1
H			1						1	1

B20	1	2	3	4	5	6	7	8	9	*a*
A	1	1					1			2
B				1	1				1	2
C					1	1	1	1		2
D		1				1			1	1
E	1		1		1	1				3
F			1	1				1		1
G	1			1			1	1	1	4
H			1				1			1

B21	1	2	3	4	5	6	7	8	9	*a*
A	1			1	1		1		1	4
B	1		1			1	1			2
C	1			1				1		1
D			1				1		1	1
E				1	1	1				2
F		1	1			1		1		3
G	1			1			1	1		3
H		1			1				1	1

B22	1	2	3	4	5	6	7	8	9	*a*
A		1			1	1				2
B	1	1		1						1
C	1	1			1			1	1	4
D			1					1	1	2
E	1					1	1	1		2
F					1		1		1	1
G	1			1						1
H			1	1		1	1			3

B23	1	2	3	4	5	6	7	8	9	*a*
A		1	1	1			1	1		3
B			1	1	1					1
C			1				1		1	2
D	1	1		1						1
E		1				1		1		1
F					1	1	1	1		2
G	1				1	1			1	3
H			1				1			1

B24	1	2	3	4	5	6	7	8	9	*a*
A		1			1	1	1			3
B		1						1	1	1
C	1		1				1			2
D			1		1	1			1	2
E	1		1	1				1	1	3
F	1			1		1				1
G				1	1			1		1
H			1				1			1

B25	1	2	3	4	5	6	7	8	9	*a*
A				1			1			1
B	1			1					1	2
C			1			1		1		1
D			1	1	1	1			1	3
E		1		1	1		1			2
F	1	1					1	1		3
G			1				1		1	1
H		1				1	1			1

B26	1	2	3	4	5	6	7	8	9	*a*
A			1	1			1			1
B		1	1		1				1	2
C					1	1		1		2
D	1			1	1				1	3
E							1	1	1	1
F	1	1				1				1
G		1	1			1	1	1		3
H		1			1					1

B27	1	2	3	4	5	6	7	8	9	*a*
A	1			1				1		1
B		1	1			1	1			2
C	1				1				1	1
D			1		1	1				2
E	1			1	1	1	1			3
F		1	1					1	1	3
G			1			1				1
H		1		1			1			1

B28	1	2	3	4	5	6	7	8	9	*a*
A	1				1					1
B				1			1	1	1	3
C		1	1						1	2
D	1	1	1		1	1				3
E			1	1	1			1		2
F	1					1	1			1
G	1			1	1					1
H		1				1		1		2

B29	1	2	3	4	5	6	7	8	9	*a*
A					1		1		1	1
B	1		1			1			1	3
C			1	1		1				2
D	1	1					1			1
E		1	1					1		2
F		1		1	1		1	1		3
G					1	1		1		1
H		1					1			1

B30	1	2	3	4	5	6	7	8	9	*a*
A			1			1			1	2
B	1				1		1	1		2
C		1	1	1						1
D	1			1	1				1	3
E					1	1		1		2
F	1	1					1			1
G		1	1			1	1	1		3
H	1							1		1

Контрольні питання

1) Що таке задача про покриття, та які її основні приклади технічного застосування?

2) Яке покриття називається безнадлишковим, та яка умова використовується для перевірки безнадлишковості?

3) Дайте визначення мінімального та найкоротшого покриття.

4) Розкрийте зміст алгоритму граничного перебору.

5) Який ваш критерій завершення перебору?

6) Проведіть порівняльний аналіз методів повного і граничного переборів.

7) Скільки підмножин аналізується в методі повного перебору?

8) Як в вашому алгоритмі вирішується задача повторів (тобто не будуються множини, які вже були побудовані)?

9) Розкрийте суть методу мінімального стовпчика - максимального рядка

10) В чому зміст методу ядерних рядків?

11) Який рядок називають антиядерним?

Лабораторна робота №3

Тема: Булеві функції. Реалізація функцій алгебри логіки в середовищі програмування.

Мета роботи: Набуття практичних навичок побудови функцій алгебри логіки мовою програмування Python.

Методичні рекомендації

Булеві функції є основою для опису алгоритмів роботи дискретних пристроїв, зокрема цифрових обчислювальних машин та логічних схем. Такі пристрої оперують фізичними сигналами, квантованими на два рівні: 0 та 1, що дозволяє моделювати їхню роботу через набір логічних операцій. Булеві функції широко застосовуються для проектування комбінаційних схем, мультиплексорів, суматорів та інших компонентів цифрових систем.

Операції над булевими функціями у Python можна ефективно виконувати за допомогою бібліотеки NumPy. Ця бібліотека пропонує широкий спектр інструментів (табл. Л3.1), забезпечуючи високу продуктивність і зручність у роботі. Приклад використання основних логічних операцій наведено у програмному коді (рис. Л3.1), а результат його виконання представлено на рис. Л3.2

Таблиця Л3.1 – Основні логічні операції над булевими функціями у Python з використанням бібліотеки NumPy

Операція	**Позначення**	**Опис**	**Синтаксис**
1	2	3	4
Логічне "І" (AND)	$A \wedge B$	Повертає True, якщо обидва елементи є True	`np.logical_and(A,B)`
Логічне "АБО" (OR)	$A \vee B$	Повертає True, якщо хоча б один елемент є True	`np.logical_or(A,B)`
Логічне "НЕ" (NOT)	$\bar{B}$	Повертає протилежне значення	`np.logical_not(B)`
Сума за модулем (XOR)	A⊕B	Повертає True, якщо елементи різні	`np.logical_xor(A,B)` `np.not_equal(A,B)`
Рівнозначність	$A \sim B$	Повертає True, якщо елементи однакові	`np.equal(A,B)`
Імплікація (від A до B)	$A \rightarrow B$	Повертає True, якщо A <= B	`np.greater_equal(B,A)`

Продовження табл. Л3.1

1	2	3	4
Імплікація (від B до A)	$B \to A$	Повертає True, якщо B <= A	`np.greater_equal(A,B)`
Штріх Шефера "NAND" (NOT AND)	$A\|B$	Повертає False, якщо обидва елементи є True	`np.logical_not(np.logical_and(A,B))`
Стрілка Пірса "NOR" (NOT OR)	$A \downarrow B$	Повертає True, якщо обидва елементи є False	`np.logical_not(np.logical_or(A,B))`
Заборона по A	$B\Delta A$	Повертає True, якщо B більше A	`np.greater(B,A)`
Заборона по B	$A\Delta B$	Повертає True, якщо A більше B	`np.greater(A,B)`

```python
import numpy as np

# Задаємо набір значень A та B
B = np.array([0, 1, 0, 1])
A = np.array([0, 0, 1, 1])

result_and = np.logical_and(A, B)       # Логічне І (AND)
result_or = np.logical_or(A, B)         # Логічне АБО (OR)
result_not = np.logical_not(B)          # Логічне НЕ (NOT)
result_equiv = np.equal(A, B)           # Лог. рівнозначність
result_not_equiv = np.not_equal(A, B)   # Лог. нерівнозначність
result_logical_xor = np.logical_xor(A, B)       # Сума за модулем
result_1 = np.greater_equal(B,A)        # Імплікація від A до B
result_4 = np.greater_equal(A,B)        # Імплікація від B до A
result_2 = np.logical_not(np.logical_and(A, B)) # Штріх Шефера
result_3 = np.logical_not(np.logical_or(A, B))  # Стрілка Пірса
result_5 = np.greater(B,A)              # Заборона по A
result_6 = np.greater(A,B)              # Заборона по B

print("A =", A)
print("B =", B)
print("Кон'юнкція AB =", result_and.astype(int))
print("Диз'юнкція A+B =", result_or.astype(int))
print("Логічне не B =", result_not.astype(int))
print("Лог. рівнозначність A~B =", result_equiv.astype(int))
print("Лог. нерівнозначність XOR =", result_not_equiv.astype(int))
print("Сума за модулем 2 XOR =", result_logical_xor.astype(int))
print("Штріх (операція) Шефера", result_2.astype(int))
print("Стрілка Пірса", result_3.astype(int))
print("Імплікація від A до B", result_1.astype(int))
print("Імплікація від B до A", result_4.astype(int))
print("Заборона по A", result_5.astype(int))
print("Заборона по B", result_6.astype(int))
```

Рисунок Л3.1 – Основні логічні операції над булевими функціями у Python з використанням бібліотеки NumPy

```
A = [0 0 1 1]
B = [0 1 0 1]
Кон'юнкція AB = [0 0 0 1]
Диз'юнкція A+B = [0 1 1 1]
Логічне не B = [1 0 1 0]
Лог. рівнозначність A~B = [1 0 0 1]
Лог. нерівнозначність XOR = [0 1 1 0]
Сума за модулем 2 XOR = [0 1 1 0]
Штріх (операція) Шефера [1 1 1 0]
Стрілка Пірса [1 0 0 0]
Імплікація від A до B [1 1 0 1]
Імплікація від B до A [1 0 1 1]
Заборона по A [0 1 0 0]
Заборона по B [0 0 1 0]
```

Рисунок Л3.2 – Результат виконання коду (рис. Л3.1)

Завдання до лабораторної роботи

Завдання 1. Використовуючи бібліотеку NumPy, виконати покроково логічні операції над трьома булевими змінними x_1, x_2, x_3 для знаходження функції f (табл. Л3.2). Побудувати відповідні часові діаграми для кожної з операцій та для кінцевої функції f. Результат знаходження функції f перевірити ручним виконанням.

Завдання 2. Перевірити ручний розв'язок індивідуального практичного завдання №5 (табл. 5.1) щодо знаходження булевої функції чотирьох змінних A, B, C та D за допомогою бібліотеки NumPy.

Таблиця Л3.2. – Варіанти завдань

Номер варіанта	Функція	Номер варіанта	Функція
1.	$f = x_1 \cdot \bar{x}_2 \mid (x_3 + \bar{x}_2)$.	16.	$f = x_1 \cdot (x_3 \oplus \bar{x}_2)$.
2.	$f = x_1 \cdot \bar{x}_2 \to x_2 \cdot \bar{x}_3$.	17.	$f = (x_1 \oplus \bar{x}_2) \to (x_2 \Delta \bar{x}_3)$.
3.	$f = x_1 \to (x_3 + \bar{x}_2)$.	18.	$f = x_1 \Delta (x_3 \oplus \bar{x}_2)$.
4.	$f = (x_1 \oplus \bar{x}_2) \to x_3$.	19.	$f = (x_1 \mid \bar{x}_2) \sim (x_2 \Delta \bar{x}_3)$.
5.	$f = (x_1 \Delta \bar{x}_2) \cdot (x_3 \downarrow x_2)$.	20.	$f = (x_1 \oplus \bar{x}_2) \sim (x_3 \Delta x_2)$.
6.	$f = x_1 \cdot \bar{x}_2 \oplus x_3$.	21.	$f = x_1 \mid \bar{x}_2 \Delta x_3$.
7.	$f = (x_1 \oplus \bar{x}_3) \cdot (x_1 \sim \bar{x}_2)$.	22.	$f = (x_1 \mid \bar{x}_3) \sim (x_1 \Delta \bar{x}_2)$.
8.	$f = (x_1 \mid \bar{x}_2) \to x_3 + \bar{x}_2$.	23.	$f = x_1 \cdot \bar{x}_2 \sim (x_3 \oplus \bar{x}_2)$.
9.	$f = x_2 \cdot \bar{x}_1 \oplus (x_3 \to \bar{x}_2)$.	24.	$f = x_2 \cdot \bar{x}_1 \to (x_3 \downarrow x_2)$.
10.	$f = (x_1 \to x_2) \oplus x_3$.	25.	$f = (x_1 \sim x_2) + x_3 \downarrow \bar{x}_2$.
11.	$f = (x_1 \downarrow \bar{x}_2) + (x_3 \downarrow x_2)$.	26.	$f = (x_1 \Delta \bar{x}_2) \sim (x_3 \to x_2)$.
12.	$f = (x_1 \cdot \bar{x}_3 \to x_2) \oplus x_3$.	27.	$f = (x_1 \cdot \bar{x}_3 \Delta x_2) \sim x_3$.
13.	$f = (x_1 \downarrow \bar{x}_2 \downarrow x_3) \sim \bar{x}_2$.	28.	$f = (x_1 \Delta \bar{x}_2 \downarrow x_3) \oplus \bar{x}_2$.
14.	$f = (x_1 \Delta \bar{x}_2) \cdot (x_3 \sim x_2)$.	29.	$f = (x_1 \oplus \bar{x}_2) \cdot (\bar{x}_3 \downarrow x_2)$.
15.	$f = (x_1 \to \bar{x}_2) \downarrow (x_3 \oplus x_2)$.	30.	$f = (x_1 \oplus \bar{x}_2) \downarrow (x_3 \Delta \bar{x}_2)$.

Таблиця 5.1 – Варіанти завдань

Номер варіанта	Функція	Номер варіанта	Функція
1.	$f = (A \to B) \sim (C \mid B) \oplus D;$	16.	$f = A \sim C \mid B \to A \oplus D;$
2.	$f = A \sim B \wedge C \mid A \to D;$	17.	$f = A \sim (C \mid B \vee \overline{C}) \to D;$
3.	$f = A \to B \sim C \vee \overline{A} \oplus D;$	18.	$f = (A \mid B) \oplus (A \mid C) \sim D;$
4.	$f = (\overline{A} \sim B) \wedge (A \sim \overline{C}) \oplus D;$	19.	$f = (A \sim B) \vee C \to (A \wedge D);$
5.	$f = (C \vee \overline{A}) \to (A \sim B) \oplus D;$	20.	$f = A \sim B \vee C \to B \wedge D;$
6.	$f = A \sim B \to C \vee \overline{A} \oplus D;$	21.	$f = A \sim B \to C \vee A \oplus D;$
7.	$f = A \sim C \vee (B \mid A) \oplus D;$	22.	$f = (A \sim B) \to C \vee B \downarrow D;$
8.	$f = (A \sim B) \vee \overline{A} \to (C \oplus \overline{D});$	23.	$f = A \sim (B \mid C) \to B \oplus D;$
9.	$f = A \to B \wedge C \sim (A \oplus \overline{D});$	24.	$f = (A \to B) \wedge C \sim A \mid D;$
10.	$f = A \to B \mid (C \oplus A) \sim D;$	25.	$f = (A \sim B) \mid C \vee A \oplus D;$
11.	$f = (A \sim B) \wedge (\overline{A} \to C) \oplus D;$	26.	$f = (A \to B) \to C \vee A \sim D;$
12.	$f = (A \sim B) \oplus C \vee (\overline{A} \to D);$	27.	$f = (A \to B) \sim C \wedge (\overline{B} \oplus D);$
13.	$f = C \vee B \sim A \vee (\overline{B} \to D);$	28.	$f = C \vee \overline{A} \to B \sim (C \oplus D);$
14.	$f = A \to B \sim C \wedge (\overline{B} \oplus D);$	29.	$f = A \to B \sim (C \mid A) \oplus \overline{D};$
15.	$f = B \to C \sim A \vee (\overline{B} \oplus D);$	30.	$f = (A \sim B) \wedge C \to \overline{B} \oplus D.$

Порядок виконання лабораторної роботи

Завдання 1. Для прикладу розглянемо функцію $F = (x_1 \oplus x_2)(x_1 + \bar{x}_3)$.

1. У робочому вікні Jupyter.Notebook імпортуємо бібліотеки Numpy та Matplotlib:

```
import numpy as np
import matplotlib.pyplot as plt
```

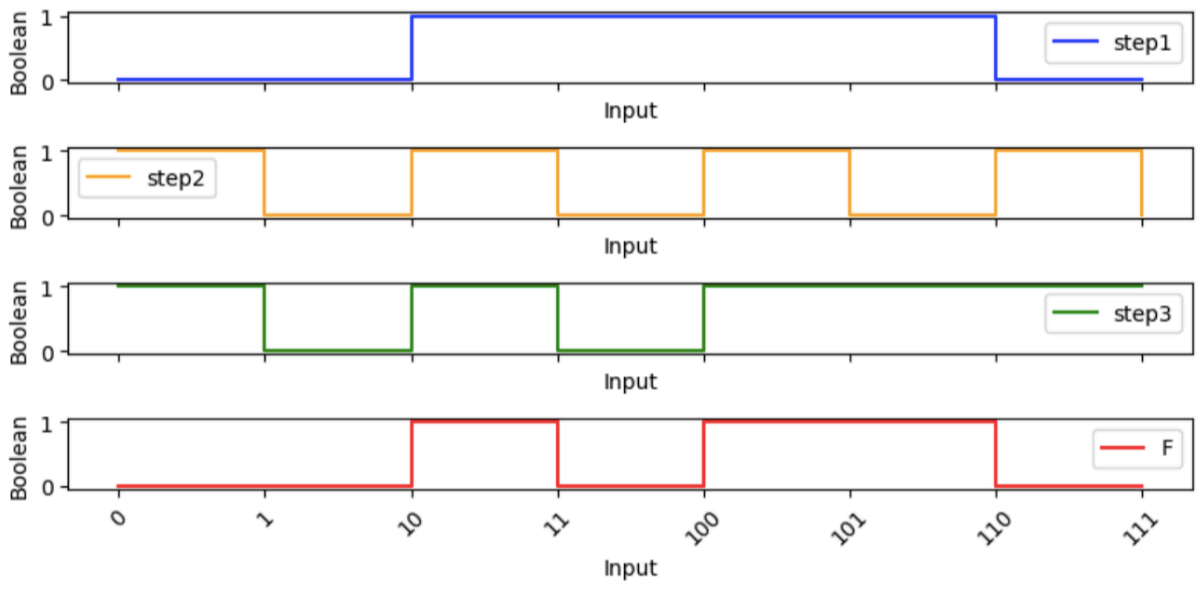
2. Задаємо загальну функцію для побудови графіків часових діаграм:

```
def plot_boolean_function(ax, F, label=None, color='green'):
    x = np.arange(len(F))
    ax.step(x, F, where='post', label=label, color=color)
    ax.set_xticks(x)
    ax.set_xticklabels([f'{i:b}' for i in range(len(F))], rotation=45)
    ax.set_xlabel('Input')
    ax.set_ylabel('Boolean')
    ax.legend()
```

3. Задаємо аргументи булевої функції та послідовно виконуємо необхідні логічні операції:

```
# Задаємо набір значень аргументів функції F
x3 = np.array([0, 1, 0, 1, 0, 1, 0, 1])
x2 = np.array([0, 0, 1, 1, 0, 0, 1, 1])
x1 = np.array([0, 0, 0, 0, 1, 1, 1, 1])

# Задаємо розміри загального графіка та визначаємо кількість підграфіків
fig, axs = plt.subplots(4, 1, figsize=(8, 4), sharex=True)

# Покроково виконуємо операції для знаходження F
step1 = np.logical_xor(x1, x2)
step2 = np.logical_not(x3)
step3 = np.logical_or(x1, step2)
F = np.logical_and(step1, step3).astype(int)

# Визначаємо підграфіки
plot_boolean_function(axs[0], step1, label='step1', color='blue')
plot_boolean_function(axs[1], step2, label='step2', color='orange')
plot_boolean_function(axs[2], step3, label='step3', color='green')
plot_boolean_function(axs[3], F, label='F', color='red')

plt.tight_layout()
plt.show()

print("F = (x1 ⊕ x2) ∧ (x1 ∨ ¬x3) =", F)
```

4. У результаті виконання коду отримуємо набір часових графіків для кожного кроку та заданої функції *F* (рис. Л3.3).




```
F = (x1 ⊕ x2) ∧ (x1 ∨ ¬x3) = [0 0 1 0 1 1 0 0]
```

Рисунок Л3.3

Завдання 2. Для виконання завдання 2 необхідно модифікувати попередній код таким чином, щоб результуюча функція f відповідала булевій функції чотирьох змінних A, B, C, D.

Контрольні питання

1) Що таке перемикаюча функція?
2) На скількох наборах визначена перемикаюча функція ?
3) Які способи задання перемикаючої функції існують?
4) Яке призначення функції `plot_boolean_function` у коді? Як її використовувати для побудови графіків?
5) Що означає логічна операція XOR (виключне АБО), і як вона реалізується у Python?
6) Як змінюється результат булевої функції при застосуванні операції NOT до одного з її аргументів?
7) Які особливості потрібно враховувати при розширенні булевої функції до чотирьох аргументів?
8) Який результат роботи функції `np.logical_and` і як вона використовується для булевих функцій?
9) Чим відрізняються логічні операції OR і XOR? Наведіть приклади.

Лабораторна робота №4

Тема: Мінімізація булевих функцій алгебри логіки за допомогою методів спрощення логічних виразів.

Мета роботи: Набуття практичних навичок мінімізації булевих функцій алгебри логіки з використанням мови програмування Python.

Методичні рекомендації

Теоретичні відомості та основні прийоми, що використовуються для мінімізації логічних функцій, були детально розглянуті в шостій темі даного посібника. У цій лабораторній роботі основну увагу приділено практичній реалізації даних методів за допомогою мови програмування Python із використанням функцій `simplify_logic`, `SOPform` і `POSform` бібліотеки SymPy (табл. Л4.1).

Таблиця Л4.1. Функції спрощення булевих виразів

Функція	Дія	Умова використання
`simplify_logic`	Мінімізує булевий вираз у найкоротшій формі	Для автоматичного спрощення булевої функції без жорстких вимог до форми (ДНФ чи КНФ).
`SOPform`	Побудова мінДНФ	Коли необхідно отримати функцію у вигляді суми добутків, використовуючи мінтерми.
`POSform`	Побудова мінКНФ	Коли необхідно отримати функцію у вигляді добутку сум, використовуючи макстерми.

Функція `simplify_logic` із бібліотеки SymPy використовується для мінімізації булевих функцій. Вона приймає як вхідний параметр булевий вираз у символьному вигляді та повертає його спрощену форму, зводячи кількість логічних операцій до мінімуму.

Ця функція підтримує як стандартну форму представлення логічних виразів, так і роботу з істинними таблицями. Її можна застосовувати для отримання мінімізованих форм у вигляді диз'юнктивної нормальної форми (ДНФ) або кон'юнктивної нормальної форми (КНФ) залежно від завдання. Для цього використовується параметр `form`, який дозволяє обирати між `'dnf'` (диз'юнктивна форма) і `'cnf'` (кон'юнктивна форма). На рис. Л4.1 наведено приклад використання цієї функції.

`Simplify_logic` найкраще використовувати для задач, у яких необхідно отримати компактне подання булевих функцій, таких як

оптимізація електронних схем чи зменшення складності логічних виразів у програмному коді.

```python
from sympy import symbols, simplify_logic

# Оголошення змінних
A, B, C = symbols('A B C')

# Початковий булевий вираз
F = (A & ~B & ~C) | (A & B & ~C) | (~A & B & C)

# Отримання мінімальної ДНФ
F_dnf_min = simplify_logic(F, form='dnf')

# Отримання мінімальної КНФ (кон'юнктивної нормальної форми)
F_cnf_min = simplify_logic(F, form='cnf')

print("Початкова функція: F =", F)
print("Мінімальна диз'юнктивна нормальна форма (мінДНФ):", F_dnf_min)
print("Мінімальна кон'юнктивна нормальна форма (мінКНФ):", F_cnf_min)
```

```
Початкова функція: F = (A & B & ~C) | (B & C & ~A) | (A & ~B & ~C)
Мінімальна диз'юнктивна нормальна форма (мінДНФ): (A & ~C) | (B & C & ~A)
Мінімальна кон'юнктивна нормальна форма (мінКНФ): (A | B) & (A | C) & (~A | ~C)
```

Рисунок Л4.1 – Приклад використання функції `Simplify_logic`

Функції `SOPform` (Sum of Products) і `POSform` (Product of Sums) із бібліотеки SymPy використовуються для отримання мінімізованих форм булевих функцій у вигляді диз'юнктивної нормальної форми (мінДНФ) та кон'юнктивної нормальної форми (мінКНФ).

Обидві функції приймають три основні параметри: список змінних, набір мінтермів або макстермів (для яких функція відповідно приймає істинність або хибність), а також, за потреби, набір неважливих значень. Результатом є логічний вираз у відповідній мінімальній формі – компактний і зручний для аналізу або реалізації.

Мінтерми (minterms) – це окремі логічні вирази, які відповідають наборам значень змінних, за яких булева функція набуває значення 1 (істина).

У канонічній диз’юнктивній нормальній формі (ДНФ) булева функція записується як сума (логічне "або") добутків (логічне "і") змінних, де кожен добуток представляє один мінтерм.

Макстерми (maxterms) – це логічні вирази, які відповідають тим комбінаціям змінних, при яких булева функція набуває хибного значення (0). Макстерми є основою для формування кон'юнктивної нормальної форми (КНФ), оскільки вони представляють набір умов, які повинні бути виконані одночасно, щоб функція була хибною.

Неважливі терми або терми "байдужості" (dontcares) – це значення булевої функції, для яких її результат не впливає на кінцеву поведінку системи чи результат роботи. Тобто, для цих термів можна вибрати

значення функції як 0 або 1, залежно від того, що забезпечує найпростішу форму функції

`SOPform` використовується для отримання булевої функції у формі мінДНФ, що складається з диз'юнкції кількох кон'юнкцій. Це корисно для задач, де необхідно описати функцію через істинні значення, наприклад, під час побудови комбінаційних схем. Програмний код з прикладом використання цієї функції наведено на рис. Л4.2.

```
from sympy import symbols
from sympy.logic import SOPform, POSform

x1, x2, x3, x4 = symbols('x1 x2 x3 x4') # Оголошення змінних

minterms = [1, 3, 7, 11, 15]  # Список мінтермів
dontcares = [0, 2, 5]         # Список неважливих (невизначених) термів

# Отримання мінімальної ДНФ без урахування неважливих термів
minDNF_1 = SOPform([x1, x2, x3, x4], minterms)

# Отримання мінімальної ДНФ з урахуванням неважливих термів
minDNF_2 = SOPform([x1, x2, x3, x4], minterms, dontcares)

print("мінДНФ без урахування неважливих термів F1 =", minDNF_1)
print("мінДНФ з урахуванням неважливих термів F2 =", minDNF_2)
```

```
мінДНФ без урахування неважливих термів F1 = (x3 & x4) | (x4 & ~x1 & ~x2)
мінДНФ з урахуванням неважливих термів F2 = (x3 & x4) | (~x1 & ~x2)
```

Рисунок Л4.2 – Приклад використання функції `SOPform`

`POSform` генерує булеву функцію у формі КНФ, представлену кон'юнкцією кількох диз'юнкцій. Такий підхід ідеально підходить для випадків, коли функцію потрібно описати через хибні значення, наприклад, при побудові схем на основі логічних елементів "OR-AND". Програмний код з прикладом використання `POSform` наведено на рис. Л4.3.

```
maxterms = [1, 3, 7, 11, 15]  # Список макстермів
dontcares = [0, 2, 5]         # Список неважливих (невизначених) термів

# Отримання мінімальної КНФ без урахування неважливих термів
minKNF_1 = POSform([x1, x2, x3, x4], minterms)

# Отримання мінімальної КНФ з урахуванням неважливих термів
minKNF_2 = POSform([x1, x2, x3, x4], minterms, dontcares)

print("мінКНФ без урахування неважливих термів F1 =", minKNF_1)
print("мінКНФ з урахуванням неважливих термів F2 =", minKNF_2)
```

```
мінКНФ без урахування неважливих термів F1 = x4 & (x3 | ~x1) & (x3 | ~x2)
мінКНФ з урахуванням неважливих термів F2 = x4 & (x3 | ~x1)
```

Рисунок Л4.3 – Приклад використання функції `POSform`

Обидві функції ефективні для мінімізації булевих функцій і широко використовуються в цифровій логіці, оптимізації схем і програмному моделюванні. Вибір між ними залежить від форми представлення, яка найбільше підходить для конкретного завдання.

Завдання до лабораторної роботи

Завдання 1. Мінімізувати логічну функцію, задану у табл. 5.1. використовуючи `simplify_logic` бібліотеки NumPy. Отриманий результат порівняти з результатом ручного виконання індивідуального практичного заняття № 6, задача 1.

Завдання 2. Мінімізувати логічну функцію, задану у табл. 6.2. використовуючи `SOPform` і `POSform` бібліотеки NumPy. Отриманий результат порівняти з результатом ручного виконання індивідуального практичного заняття №6, задача 2.

Порядок виконання лабораторної роботи

Завдання 1.

1. Для виконання завдання 1 необхідно обрати логічну функцію, яка відповідає варіанту виконання індивідуального практичного завдання №6, задача 1 (табл. 5.1).

2. Використовуючи функцію `simplify_logic` та приклад наведений на рис. Л4.1, розробити програму для знаходження мінімальної диз'юнктивної нормальної форми (мінДНФ) та мінімальної кон’юнктивної нормальної форми (мінКНФ) для обраної логічної функції. Порівняти отриманий результат з ручним виконанням.

Завдання 2.

1. Для виконання завдання 2 необхідно обрати логічну функцію, яка відповідає варіанту виконання індивідуального практичного завдання №6, задача 2 (табл. 6.2).

2. Використовуючи функції `SOPform` і `POSform` та приклади наведені на рис. Л4.2 та рис. Л4.3, розробити програму для знаходження мінімальної диз'юнктивної нормальної форми (мінДНФ) та мінімальної кон’юнктивної нормальної форми (мінКНФ) для обраної логічної функції. Порівняти отриманий результат з ручним виконанням.

3. Довільно обрати декілька неважливих термів та повторно розрахувати мінДНФ та мінКНФ для заданої логічної функції. Порівняти, як додавання неважливих термів вплинуло на отримані вирази.

Таблиця 6.2 – Варіанти завдань

Варіант	Функція	Варіант	Функція
1.	F = (0, 1, 2, 3, 5, 7, 10, 11, 14)	31.	F = (0, 2, 4, 5, 6, 8, 11, 13)
2.	F = (1, 2, 3, 5, 10, 14, 15)	32.	F = (1, 2, 3, 4, 5, 10, 11, 12, 13)
3.	F = (0, 1, 7, 11, 12, 13, 15)	33.	F = (0, 1, 2, 7, 8, 9, 10, 11)
4.	F = (1, 3, 5, 6, 8, 9, 11, 15)	34.	F = (0, 1, 2, 4, 5, 7, 8, 15)
5.	F = (2, 4, 6, 8, 10, 12, 14, 15)	35.	F = (1, 3, 4, 7, 8, 10, 13, 15)
6.	F = (0, 1, 2, 4, 7, 10, 12, 13, 14)	36.	F = (1, 2, 4, 5, 10, 11, 14)
7.	F = (1, 2, 3, 4, 7, 10, 12, 15)	37.	F = (0, 1, 3, 6, 7, 9, 10, 13, 15)
8.	F = (3, 4, 5, 7, 10, 11, 14, 15)	38.	F = (1, 2, 5, 6, 9, 10, 11, 12)
9.	F = (5, 7, 8, 10, 11, 12, 14, 15)	39.	F = (0, 1, 4, 5, 7, 9, 11, 13)
10.	F = (2, 5, 6, 10, 12, 13, 14)	40.	F = (1, 5, 6, 10, 12, 13, 14, 15)
11.	F = (1, 2, 3, 4, 6, 13, 14, 15)	41.	F = (2, 5, 6, 10, 12, 14, 15)
12.	F = (1, 3, 7, 9, 10, 12, 14, 15)	42.	F = (0, 1, 3, 9, 10, 11, 12, 15)
13.	F = (1, 6, 7, 8, 9, 11, 13, 15)	43.	F = (1, 2, 3, 6, 7, 9, 11, 13, 15)
14.	F = (3, 4, 5, 6, 12, 13, 14, 15)	44.	F = (0, 1, 2, 3, 6, 7, 12, 14, 15)
15.	F = (3, 6, 9, 10, 11, 12, 13, 14)	45.	F = (3, 6, 8, 9, 11, 13, 14, 15)
16.	F = (2, 5, 6, 9, 11, 12, 14)	46.	F = (1, 2, 3, 4, 5, 10, 11, 12)
17.	F = (0, 3, 6, 7, 8, 9, 12, 13, 15)	47.	F = (2, 5, 7, 9, 11, 13, 14, 15)
18.	F = (2, 4, 5, 8, 9, 11, 12, 13)	48.	F = (2, 5, 6, 10, 12, 13, 14, 15)
19.	F = (0, 3, 4, 7, 8, 13, 14, 15)	49.	F = (1, 2, 4, 6, 8, 10, 13, 14)
20.	F = (2, 3, 4, 6, 8, 12, 13, 14, 15)	50.	F = (0, 1, 3, 9, 10, 11, 12, 14)
21.	F = (0, 1, 3, 4, 7, 8, 9, 12, 13)	51.	F = (3, 5, 6, 7, 8, 9, 11, 15)
22.	F = (0, 1, 3, 4, 5, 7, 8, 9, 13)	52.	F = (0, 5, 6, 8, 9, 10, 11, 13, 14)
23.	F = (0, 1, 2, 4, 5, 6, 7, 8, 14)	53.	F = (0, 1, 2, 6, 8, 9, 10, 13, 15)
24.	F = (0, 4, 5, 8, 9, 10, 12, 13, 14)	54.	F = (0, 1, 3, 4, 6, 8, 9, 13)
25.	F = (4, 5, 8, 9, 10, 11, 14, 15)	55.	F = (1, 3, 4, 7, 8, 9, 11, 13, 15)
26.	F = (0, 1, 2, 3, 4, 5, 7, 10, 12)	56.	F = (0, 5, 7, 8, 9, 10, 11, 12)
27.	F = (0, 6, 7, 8, 9, 11, 13, 14, 15)	57.	F = (2, 4, 5, 7, 9, 11, 13, 15)
28.	F = (0, 1, 3, 4, 6, 10, 11, 12, 14)	58.	F = (2, 5, 6, 10, 12, 14, 15)
29.	F = (0, 1, 3, 4, 7, 8, 9, 11)	59.	F = (2, 4, 5, 6, 7, 9, 11, 13, 15)
30.	F = (0, 1, 3, 5, 7, 9, 10, 11, 12)	60.	F = (1, 2, 3, 5, 7, 13, 14, 15)

Контрольні питання

1) Що таке булевий вираз? Наведіть приклади.

2) У чому полягає різниця між ДНФ (СДНФ) і КНФ (СКНФ)?

3) Що таке мінтерми та макстерми? Як вони використовуються при побудові ДНФ і КНФ?

4) Яка мета мінімізації булевих виразів? Наведіть приклади її практичного застосування.

5) Чим відрізняється функція `SOPform` від `simplify_logic`?

6) Як враховувати неважливі значення булевої функції при мінімізації булевих функцій?

7) Які функції SymPy можна використати для автоматичного спрощення складного булевого виразу? Чим вони відрізняються?

8) Які переваги має спрощення булевих виразів у ДНФ або КНФ перед їх використанням у цифровій логіці?

9) Як неважливі значення булевої функції впливають на результат мінімізації?

10) Порівняйте результати мінімізації булевого виразу за допомогою функцій `SOPform`, `POSform` і `simplify_logic`.