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 №2

З дисципліни «Теорія прийняття рішень»

 на тему «Оптимізація за бінарним відношення»

1 варіант

|  |  |
| --- | --- |
| Перевірила: | Виконав: |
| Жураковська О.С. | студент гр. ІС-01  Адамов Д.І. |

Київ-2023

**Завдання 1**

Для кожного з бінарних відношень R1-R8 (із завдання 1

практикуму 1) визначити множину найкращих альтернатив за

принципами домінування та блокування.

|  |  |  |  |
| --- | --- | --- | --- |
| Відношення | Клас, до якого належить БВ | Опт. альтернативи за принципом домінування | Опт. Альтернативи за принципом блокування |
| R1 | Квазіпорядок | X\*R = {2, 3, 8}  X\*\*R = ∅ | X0R = {2, 3, 8}  X00R = ∅ |
| R2 | - | X\*R = ∅  X\*\*R = ∅ | X0R = {6, 8}  X00R = ∅ |
| R3 | - | X\*R = ∅  X\*\*R = ∅ | X0R = {1, 6}  X00R = ∅ |
| R4 | Нестрогий порядок | X\*R = {3}  X\*\*R = {3} | X0R = {3}  X00R = {3} |
| R5 | Строгий порядок | X\*P = {6} | X0P = {6} |
| R6 | Нестрогий порядок | X\*R = {8}  X\*\*R = {8} | X0R = {8}  X00R = {8} |
| R7 | Еквівалентність | X\*R = ∅  X\*\*R = ∅ | X0R = {1, 2, 3, 4, 5, 6, 7, 8}  X00R = ∅ |
| R8 | Строгий порядок | X\*P = ∅ | X0P = {1, 5, 6} |

**Завдання 2** (варіанти завдань в файлі Практикум 2. Варіанти для завдання 2 Файл)

На множині із 15 альтернатив задано 10 бінарних відношень R1-R10

матрицями відношень. Для кожного із БВ R*i* необхідно:

1. перевірити наявність властивості ациклічності;
   1. якщо відношення R*i* є ациклічним - знайти множину Неймана-Моргенштерна (отриманий результат обгрунтувати – показати, що отримана множина відповідає означенню розв’язка Неймана-Моргенштерна);
   2. якщо відношення R*i* не ациклічне - знайти множини оптимальних альтернатив за принципом К-оптимізації (k=1, k=2, k=3, k=4)

|  |  |  |  |
| --- | --- | --- | --- |
| Відношення | Ациклічне/  неациклічне | Розв’язок Неймана-Моргенштерна | Опт. Альтернативи за принципом К-оптимізації |
| R1 | + | XНМ = {0, 8, 3, 11} |  |
| R2 | + | XНМ = {0, 3} |  |
| R3 | + | XНМ = {0, 3, 13} |  |
| R4 | + | XНМ = {2, 4, 6, 8} |  |
| R5 | + | XНМ = {4, 6, 12, 13} |  |
| R6 | - |  | 1-max: {0, 2, 4, 11, 12, 13}  1-opt: {0, 2, 4, 11, 12, 13}  2-max: {0, 2, 4, 11, 12, 13}  2-opt: {0, 2, 4, 11, 12, 13}  3-max: {0, 6, 7, 11, 12}  3-opt: {}  4-max: {0, 11, 12}  4-opt: {} |
| R7 | + | XНМ = {0, 4, 8, 12} |  |
| R8 | - |  | 1-max: {0, 1, 3, 6, 9, 13}  1-opt: {0, 1, 3, 6, 9, 13}  2-max: {0, 1, 3, 6, 9, 13}  2-opt: {0, 1, 3, 6, 9, 13}  3-max: {0, 1, 2, 11, 13}  3-opt: {}  4-max: {0, 1, 13}  4-opt: {} |
| R9 | - |  | 1-max: {2, 3, 6, 10, 11, 13}  1-opt: {2, 3, 6, 10, 11, 13}  2-max: {2, 6, 7, 13, 14}  2-opt: {}  3-max: {0, 1, 2, 11, 13}  3-opt: {}  4-max: {2, 13, 6}  4-opt: {} |
| R10 | + | XНМ = {1, 2, 6} |  |

**Результати виконання завдання 1**

1 Квазіпорядок

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| I |  |  | I | I | I |  |  |
| P | I | I | P | P | P | P | I |
| P | I | I | P | P | P | P | I |
| I |  |  | I | I | I |  |  |
| I |  |  | I | I | I |  |  |
| I |  |  | I | I | I |  |  |
| P |  |  | P | P | P | I |  |
| P | I | I | P | P | P | P | I |

Відношення не є асиметричним

Оптимізація за домінуванням:

X\*R = {2, 3, 8} – у рядках всі елементи є одиницями

X\*\*R = ∅, оскільки немає відповідних до X\*R стовпчиків, де одиниця лише на головній діагоналі

Оптимізація за блокуванням:

X0R = {2, 3, 8} – стовпці містять лише I або 0

X00R = ∅ – немає стовпців, які містять усі 0, без врахування головної діагоналі

2 Не належить до класу

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | | 0 | | 1 | | 0 | | 0 | | 1 | | 0 | |
| 0 | 0 | | 1 | | 0 | | 0 | | 1 | | 0 | | 0 | |
| 1 | 0 | | 0 | | 1 | | 1 | | 0 | | 0 | | 0 | |
| 1 | 0 | | 1 | | 0 | | 1 | | 1 | | 0 | | 1 | |
| 1 | 0 | | 0 | | 0 | | 1 | | 0 | | 1 | | 1 | |
| 0 | 1 | | 0 | | 1 | | 0 | | 1 | | 1 | | 0 | |
| 0 | 1 | | 0 | | 1 | | 0 | | 0 | | 1 | | 0 | |
| 0 | 0 | | 0 | | 1 | | 1 | | 0 | | 0 | | 1 | |
| I | | N | |  | | I | |  | | N | | P | | N | |
| N | | N | | P | | N | | N | | I | |  | | N | |
| P | |  | | N | | I | | P | | N | | N | | N | |
| I | | N | | I | | N | | P | | I | |  | | I | |
| P | | N | |  | |  | | I | | N | | P | | I | |
| N | | I | | N | | I | | N | | I | | P | | N | |
|  | | P | | N | | P | |  | |  | | I | | N | |
| N | | N | | N | | I | | I | | N | | N | | I | |

Відношення не є асиметричним

Оптимізація за домінуванням:

X\*R = ∅ – немає рядка, що містить усі одиниці

X\*\*R = ∅, оскільки X\*R = ∅

Оптимізація за блокуванням:

X0R = {6, 8} – стовпці не містять P

X00R = ∅ – немає стовпців, які містять усі 0, без врахування діагоналі

3 Не належить до класу

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | | 1 | | 0 | | 0 | | 1 | | 1 | | 1 | |
| 1 | 0 | | 0 | | 0 | | 0 | | 1 | | 1 | | 1 | |
| 0 | 0 | | 0 | | 0 | | 0 | | 0 | | 1 | | 1 | |
| 0 | 1 | | 0 | | 0 | | 0 | | 0 | | 1 | | 0 | |
| 0 | 1 | | 0 | | 1 | | 1 | | 0 | | 0 | | 0 | |
| 1 | 1 | | 0 | | 0 | | 0 | | 1 | | 0 | | 1 | |
| 1 | 1 | | 1 | | 0 | | 1 | | 0 | | 1 | | 0 | |
| 0 | 0 | | 0 | | 0 | | 0 | | 1 | | 1 | | 1 | |
| I | | I | | P | | N | | N | | I | | I | | P | |
| I | | N | | N | | P | | P | | I | | I | | P | |
|  | | N | | N | | N | | N | | N | | I | | P | |
| N | | P | | N | | N | |  | | N | | P | | N | |
| N | | P | | N | | P | | I | | N | |  | | N | |
| I | | I | | N | | N | | N | | I | | N | | I | |
| I | | I | | I | |  | | P | | N | | I | |  | |
|  | |  | |  | | N | | N | | I | | P | | I | |

Відношення не є асиметричним

Оптимізація за домінуванням:

X\*R = ∅ – немає рядка, що містить усі одиниці

X\*\*R = ∅, оскільки X\*R = ∅

Оптимізація за блокуванням:

X0R = {1, 6} – стовпці не містять P

X00R = ∅ – немає стовпців, які містять усі 0, без врахування головної діагоналі

4 Нестрогий порядок

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | | 0 | | 1 | | 1 | | 1 | | 1 | | 1 | |
| 0 | 1 | | 0 | | 1 | | 1 | | 1 | | 0 | | 0 | |
| 1 | 1 | | 1 | | 1 | | 1 | | 1 | | 1 | | 1 | |
| 0 | 0 | | 0 | | 1 | | 0 | | 0 | | 0 | | 0 | |
| 0 | 0 | | 0 | | 1 | | 1 | | 0 | | 0 | | 0 | |
| 0 | 0 | | 0 | | 1 | | 1 | | 1 | | 0 | | 0 | |
| 0 | 1 | | 0 | | 1 | | 1 | | 1 | | 1 | | 0 | |
| 0 | 1 | | 0 | | 1 | | 1 | | 1 | | 1 | | 1 | |
| I | | P | |  | | P | | P | | P | | P | | P | |
|  | | I | |  | | P | | P | | P | |  | |  | |
| P | | P | | I | | P | | P | | P | | P | | P | |
|  | |  | |  | | I | |  | |  | |  | |  | |
|  | |  | |  | | P | | I | |  | |  | |  | |
|  | |  | |  | | P | | P | | I | |  | |  | |
|  | | P | |  | | P | | P | | P | | I | |  | |
|  | | P | |  | | P | | P | | P | | P | | I | |

Відношення не є асиметричним

Оптимізація за домінуванням:

X\*R = {3} – у рядку всі одиниці

X\*\*R = {3} – у стовпці всі нулі, без врахування головної діагоналі

Оптимізація за блокуванням:

X0R = {3} – у стовпці немає P

X00R = {3} – у стовпці всі нулі, без врахування головної діагоналі

5 Строгий порядок

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | | 1 | | 1 | | 0 | | 0 | | 0 | | 1 | |
| 0 | 0 | | 0 | | 0 | | 0 | | 0 | | 0 | | 0 | |
| 0 | 1 | | 0 | | 1 | | 0 | | 0 | | 0 | | 1 | |
| 0 | 1 | | 0 | | 0 | | 0 | | 0 | | 0 | | 0 | |
| 1 | 1 | | 1 | | 1 | | 0 | | 0 | | 0 | | 1 | |
| 1 | 1 | | 1 | | 1 | | 1 | | 0 | | 1 | | 1 | |
| 1 | 1 | | 1 | | 1 | | 1 | | 0 | | 0 | | 1 | |
| 0 | 1 | | 0 | | 1 | | 0 | | 0 | | 0 | | 0 | |
| N | | P | | P | | P | |  | |  | |  | | P | |
|  | | N | |  | |  | |  | |  | |  | |  | |
|  | | P | | N | | P | |  | |  | |  | | P | |
|  | | P | |  | | N | |  | |  | |  | |  | |
| P | | P | | P | | P | | N | |  | |  | | P | |
| P | | P | | P | | P | | P | | N | | P | | P | |
| P | | P | | P | | P | | P | |  | | N | | P | |
|  | | P | |  | | P | |  | |  | |  | | N | |

Відношення асиметричне

Оптимізація за домінуванням:

X\*P = {6} – у відповідному рядку всі одиниці, крім головної діагоналі

Оптимізація за блокуванням:

X0P = {6} – у відповідному стовпці всі нулі

6 Нестрогий порядок

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | | 1 | | 1 | | 1 | | 1 | | 1 | | 0 | |
| 0 | 1 | | 1 | | 1 | | 0 | | 1 | | 1 | | 0 | |
| 0 | 0 | | 1 | | 0 | | 0 | | 1 | | 0 | | 0 | |
| 0 | 0 | | 1 | | 1 | | 0 | | 1 | | 1 | | 0 | |
| 0 | 1 | | 1 | | 1 | | 1 | | 1 | | 1 | | 0 | |
| 0 | 0 | | 0 | | 0 | | 0 | | 1 | | 0 | | 0 | |
| 0 | 0 | | 1 | | 0 | | 0 | | 1 | | 1 | | 0 | |
| 1 | 1 | | 1 | | 1 | | 1 | | 1 | | 1 | | 1 | |
| I | | P | | P | | P | | P | | P | | P | | 0 | |
| 0 | | I | | P | | P | | 0 | | P | | P | | 0 | |
| 0 | | 0 | | I | | 0 | | 0 | | P | | 0 | | 0 | |
| 0 | | 0 | | P | | I | | 0 | | P | | P | | 0 | |
| 0 | | P | | P | | P | | I | | P | | P | | 0 | |
| 0 | | 0 | | 0 | | 0 | | 0 | | I | | 0 | | 0 | |
| 0 | | 0 | | P | | 0 | | 0 | | P | | I | | 0 | |
| P | | P | | P | | P | | P | | P | | P | | I | |

Відношення не є асиметричним

Оптимізація за домінуванням:

X\*R = {8} – рядок містить усі одиниці

X\*\*R = {8} – стовпець містить усі нулі, без врахування головної діагоналі

Оптимізація за блокуванням:

X0R = {8} – стовпець не містить P

X00R = {8} – стовпець містить усі нулі, без врахування головної діагоналі

7 Еквівалентність

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| I | 0 | 0 | 0 | 0 | 0 | I | 0 |
| 0 | I | I | I | 0 | 0 | 0 | I |
| 0 | I | I | I | 0 | 0 | 0 | I |
| 0 | I | I | I | 0 | 0 | 0 | I |
| 0 | 0 | 0 | 0 | I | I | 0 | 0 |
| 0 | 0 | 0 | 0 | I | I | 0 | 0 |
| I | 0 | 0 | 0 | 0 | 0 | I | 0 |
| 0 | I | I | I | 0 | 0 | 0 | I |

Відношення не є асиметричним

Оптимізація за домінуванням:

X\*R = ∅ – немає рядка, що містить усі одиниці

X\*\*R = ∅, оскільки X\*R = ∅

Оптимізація за блокуванням:

X0R = {1, 2, 3, 4, 5, 6, 7, 8} – стовпці не мають P

X00R = ∅ – немає стовпців, у яких всі нулі, без врахування головної діагоналі

8 Строгий порядок

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | P | P | P | 0 | 0 | P | P |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | P | P | P | 0 | 0 | P | P |
| 0 | P | P | P | 0 | 0 | P | P |
| 0 | P | P | P | 0 | 0 | 0 | 0 |
| 0 | P | P | P | 0 | 0 | 0 | 0 |

Відношення є асиметричним

Оптимізація за домінуванням:

X\*P = ∅ – немає рядка, що містить усі одиниці, крім головної діагоналі

Оптимізація за блокуванням:

X0P = {1, 5, 6} – стовпці містять всі нулі

**Результати виконання завдання 2**

**Relation #1**

0 1 1 0 0 1 1 0 0 1 1 0 1 1 0

0 0 0 1 0 1 0 0 1 1 1 1 0 0 0

0 0 0 0 0 0 0 1 0 1 0 0 0 0 1

0 0 1 0 1 1 1 1 0 1 1 0 0 1 1

0 0 0 0 0 1 0 1 1 1 0 0 0 1 0

0 0 1 0 0 0 1 1 0 1 1 1 0 1 0

0 0 1 0 0 0 0 1 1 1 1 1 1 1 0

0 0 0 0 0 0 0 0 0 1 1 0 0 1 0

0 0 1 0 0 0 0 0 0 1 1 0 0 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 0 0 0 0 0 0 1 1 0 0

0 0 0 0 0 0 0 0 0 1 0 0 1 1 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Relation #1 is acyclic. Neumann-Morgenstern optimization will be used.

1) Формуємо множину S0, в яку будуть входити елементи з порожнім верхнім перерізом (у матриці – у таких стовпчиках всі нулі).

2) Множина S1 будується на основі S0 – включаємо елементи, для яких верхній переріз включається в множину S0 (у матриці у стовпчиках одиниці стоять лише в тих рядках, що входять до множини S0).

3) Операція повторюється для Sn+1/Sn, допоки Sn не буде дорівнювати всій множині альтернатив Ω.

S0: {0}

S1: {0, 1}

S2: {0, 1, 3}

S3: {0, 1, 3, 4}

S4: {0, 1, 3, 4, 5}

S5: {0, 1, 3, 4, 5, 6}

S6: {0, 1, 3, 4, 5, 6, 8}

S7: {0, 1, 2, 3, 4, 5, 6, 8}

S8: {0, 1, 2, 3, 4, 5, 6, 7, 8}

S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 10}

S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11}

S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

S12: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}

S13: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Шукаємо множини Q0..Qn.

1. Q0 = S0.
2. Для знаходження Q1 ми перевіряємо кандидатів до цієї множини з S1/S0. Умова полягає в тому, що верхній переріз кожного з цих кандидатів має бути порожнім у рядках Q0 (S0). Якщо ця умова для кандидата виконується, то ми додаємо його в множину Q1.

Якщо множина Q1 виявиться порожньою, то Q1 = Q0.

1. Потрібно повторити ці дії для всіх множин S (кількість ітерацій співпадає), і множина Qn буде розв’язком Неймана-Моргенштерна.

Q0: {0}

Q1: {0}

Q2: {0, 3}

Q3: {0, 3}

Q4: {0, 3}

Q5: {0, 3}

Q6: {0, 8, 3}

Q7: {0, 8, 3}

Q8: {0, 8, 3}

Q9: {0, 8, 3}

Q10: {0, 8, 3, 11}

Q11: {0, 8, 3, 11}

Q12: {0, 8, 3, 11}

Q13: {0, 8, 3, 11}

Neumann-Morgenstern solution: {0, 8, 3, 11}

Internal stability: True

External stability: True

**Relation #2**

0 1 1 0 1 1 1 0 0 0 1 1 1 0 1

0 0 1 0 1 0 0 0 1 1 1 0 0 0 0

0 0 0 1 0 1 1 0 1 0 0 1 1 0 1

0 0 0 0 0 1 1 1 1 1 0 1 1 1 0

0 0 1 0 0 1 1 0 0 0 1 1 0 1 1

0 0 0 0 0 0 0 0 1 1 1 0 0 0 1

0 0 0 0 0 0 0 1 0 1 0 1 1 1 0

0 0 0 0 0 1 0 0 1 1 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 1 0 1 1

0 0 0 0 0 0 0 0 1 0 1 1 1 1 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 0 0 0 1 0 1 0 1

0 0 0 0 0 0 0 0 0 0 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 1 1 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Relation #2 is acyclic. Neumann-Morgenstern optimization will be used.

S0: {0}

S1: {0, 1}

S2: {0, 1, 4}

S3: {0, 1, 2, 4}

S4: {0, 1, 2, 3, 4}

S5: {0, 1, 2, 3, 4, 6}

S6: {0, 1, 2, 3, 4, 6, 7}

S7: {0, 1, 2, 3, 4, 5, 6, 7}

S8: {0, 1, 2, 3, 4, 5, 6, 7, 9}

S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 13}

S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13}

S12: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13}

S13: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}

S14: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {0}

Q1: {0}

Q2: {0}

Q3: {0}

Q4: {0, 3}

Q5: {0, 3}

Q6: {0, 3}

Q7: {0, 3}

Q8: {0, 3}

Q9: {0, 3}

Q10: {0, 3}

Q11: {0, 3}

Q12: {0, 3}

Q13: {0, 3}

Q14: {0, 3}

Neumann-Morgenstern solution: {0, 3}

Internal stability: True

External stability: True

**Relation #3**

0 1 1 0 0 1 1 0 1 1 0 1 1 0 1

0 0 1 0 0 1 0 0 1 0 1 0 1 0 0

0 0 0 1 0 1 1 0 1 1 0 1 1 0 1

0 0 0 0 1 0 1 1 0 0 1 1 1 0 0

0 0 0 0 0 0 1 1 1 1 0 1 1 1 0

0 0 0 1 1 0 1 1 0 1 0 0 1 1 0

0 0 0 0 0 0 0 1 0 0 0 1 1 0 1

0 0 0 0 0 0 0 0 1 1 1 1 0 1 0

0 0 0 0 0 0 0 0 0 0 0 1 1 1 0

0 0 0 0 0 0 0 0 1 0 0 1 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1 0 1 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 0 0 0 0 0 1 0 0 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Relation #3 is acyclic. Neumann-Morgenstern optimization will be used.

S0: {0}

S1: {0, 1}

S2: {0, 1, 2}

S3: {0, 1, 2, 5}

S4: {0, 1, 2, 3, 5}

S5: {0, 1, 2, 3, 4, 5}

S6: {0, 1, 2, 3, 4, 5, 6}

S7: {0, 1, 2, 3, 4, 5, 6, 7}

S8: {0, 1, 2, 3, 4, 5, 6, 7, 9}

S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12}

S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12}

S12: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

S13: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}

S14: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {0}

Q1: {0}

Q2: {0}

Q3: {0}

Q4: {0, 3}

Q5: {0, 3}

Q6: {0, 3}

Q7: {0, 3}

Q8: {0, 3}

Q9: {0, 3}

Q10: {0, 3}

Q11: {0, 3}

Q12: {0, 3}

Q13: {0, 3, 13}

Q14: {0, 3, 13}

Neumann-Morgenstern solution: {0, 3, 13}

Internal stability: True

External stability: True

**Relation #4**

0 0 0 0 0 0 0 1 0 1 0 0 1 1 1

1 0 0 1 0 1 0 1 1 1 0 1 1 1 0

0 1 0 1 0 1 0 0 0 1 0 1 0 0 0

1 0 0 0 0 0 1 1 1 1 0 1 0 0 1

0 0 0 1 0 1 0 1 0 0 1 0 0 1 0

1 0 0 0 0 0 1 1 0 1 0 1 1 1 0

1 0 0 0 0 0 0 1 0 1 1 0 0 1 0

0 0 0 0 0 0 0 0 1 1 0 1 0 0 1

0 0 0 0 0 0 0 0 0 0 0 1 1 0 1

0 0 0 0 0 0 0 0 1 0 1 0 1 1 1

0 0 0 0 0 0 0 0 0 0 0 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 1 0

Relation #4 is acyclic. Neumann-Morgenstern optimization will be used.

S0: {2, 4}

S1: {1, 2, 4}

S2: {1, 2, 3, 4, 5}

S3: {1, 2, 3, 4, 5, 6}

S4: {0, 1, 2, 3, 4, 5, 6}

S5: {0, 1, 2, 3, 4, 5, 6, 7}

S6: {0, 1, 2, 3, 4, 5, 6, 7, 9}

S7: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

S8: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 14}

S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14}

S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {2, 4}

Q1: {2, 4}

Q2: {2, 4}

Q3: {2, 4, 6}

Q4: {2, 4, 6}

Q5: {2, 4, 6}

Q6: {2, 4, 6}

Q7: {8, 2, 4, 6}

Q8: {8, 2, 4, 6}

Q9: {8, 2, 4, 6}

Q10: {8, 2, 4, 6}

Q11: {8, 2, 4, 6}

Neumann-Morgenstern solution: {8, 2, 4, 6}

Internal stability: True

External stability: True

**Relation #5**

0 0 1 0 0 1 1 1 0 1 0 1 1 0 0

0 0 1 1 0 1 0 0 0 1 1 1 0 0 0

0 0 0 1 0 0 1 1 0 0 0 0 0 0 1

0 0 0 0 0 0 0 0 1 1 0 0 0 0 0

1 1 1 1 0 1 0 1 1 1 1 1 0 0 1

0 0 0 1 0 0 0 1 1 0 0 1 0 0 0

0 0 0 1 0 0 0 1 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 1 0 0 1 0 0

0 0 0 0 0 0 0 0 0 0 0 1 1 0 0

0 0 0 0 0 0 0 0 1 0 0 1 1 1 1

1 0 0 1 0 0 0 1 1 1 0 0 1 1 0

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 0 0 1 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

Relation #5 is acyclic. Neumann-Morgenstern optimization will be used.

S0: {4}

S1: {1, 4}

S2: {1, 4, 10}

S3: {0, 1, 4, 10}

S4: {0, 1, 2, 4, 5, 10}

S5: {0, 1, 2, 4, 5, 6, 10}

S6: {0, 1, 2, 3, 4, 5, 6, 7, 10}

S7: {0, 1, 2, 3, 4, 5, 6, 7, 9, 10}

S8: {0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 13}

S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14}

S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14}

S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {4}

Q1: {4}

Q2: {4}

Q3: {4}

Q4: {4}

Q5: {4, 6}

Q6: {4, 6}

Q7: {4, 6}

Q8: {4, 13, 6}

Q9: {4, 13, 6}

Q10: {4, 13, 6}

Q11: {12, 4, 13, 6}

Neumann-Morgenstern solution: {12, 4, 13, 6}

Internal stability: True

External stability: True

**Relation #6**

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Relation #6 has a cycle. k-optimization will be used.

PIN matrix:

N P N P N P P P P P P N N N P

N N N N N N N N N N

N N N N N N N N N N N N N N N

N N N N N N N N N N

N N N N N N N N N N N N N N N

N N N N N N N N N N

P N P N P I I P P P N P

P N P N P I I P P P N P

N N N N N N N N N N

N N N N N N N N N N

N N N N N N N N N N

N P N P N P P P P P P N N N P

N P N P N P P P P P P N N N P

N N N N N N N N N N N N N N N

N N N N N N N N N N

k1 (R (P, I, N)).

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

0 1 1 1 1 1 1 1 1 1 1 0 0 1 1

0 1 1 1 1 1 1 1 1 1 1 0 0 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

Матриця для k1 будується таким чином: 1, якщо елемент це P, I або N, інакше - 0.

Сім’я множин S1 R(x) представлена як рядки матриці.

Щоб обрати найкращу альтернативу, треба обрати множину, що є

максимальною за включенням, тобто такий рядок включає в себе всі інші рядки і при цьому жоден інший рядок не включає його як власну підмножину. Таких альтернатив може бути декілька.

1-оптимальні альтернативи – це такі альтернативи, S1 R(x) яких співпадає з усією множиною альтернатив Ω. Їх може не існувати.

k1 max elements: {0, 2, 4, 11, 12, 13}

k1 opt elements: {0, 2, 4, 11, 12, 13}

k2 (R (P, N)).

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 0 0 1 1 1 0 0 1 1

k2 max elements: {0, 2, 4, 11, 12, 13}

k2 opt elements: {0, 2, 4, 11, 12, 13}

k3 (R (P, I))

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

k3 max elements: {0, 6, 7, 11, 12}

k3 opt elements: {}

k4 (R (P))

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 0 1 0 1 0 0 1 1 1 0 0 0 1

0 1 0 1 0 1 0 0 1 1 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 1 0 1 0 1 1 1 1 1 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

k4 max elements: {0, 11, 12}

k4 opt elements: {}

**Relation #7**

0 1 1 1 0 1 1 0 0 1 1 0 0 0 1

0 0 1 1 1 1 0 0 0 1 0 0 1 1 1

0 0 0 1 0 1 1 0 0 0 0 1 0 1 0

0 0 0 0 0 0 1 1 1 0 1 0 0 0 0

0 0 1 1 0 1 1 1 0 1 0 0 0 0 0

0 0 0 1 0 0 1 1 1 0 1 0 0 0 0

0 0 0 0 0 0 0 1 0 1 1 1 0 0 1

0 0 0 0 0 0 0 0 0 1 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 1 1 0 1 1

0 0 0 0 0 0 0 0 0 0 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 1 1 0 0 0 1 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Relation #7 is acyclic. Neumann-Morgenstern optimization will be used.

S0: {0}

S1: {0, 1}

S2: {0, 1, 4}

S3: {0, 1, 2, 4}

S4: {0, 1, 2, 4, 5}

S5: {0, 1, 2, 3, 4, 5}

S6: {0, 1, 2, 3, 4, 5, 8}

S7: {0, 1, 2, 3, 4, 5, 8, 13}

S8: {0, 1, 2, 3, 4, 5, 6, 8, 13}

S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 13}

S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 13}

S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13}

S12: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13}

S13: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {0}

Q1: {0}

Q2: {0, 4}

Q3: {0, 4}

Q4: {0, 4}

Q5: {0, 4}

Q6: {0, 8, 4}

Q7: {0, 8, 4}

Q8: {0, 8, 4}

Q9: {0, 8, 4}

Q10: {0, 8, 4}

Q11: {0, 8, 4}

Q12: {0, 8, 4}

Q13: {0, 8, 4, 12}

Neumann-Morgenstern solution: {0, 8, 4, 12}

Internal stability: True

External stability: True

**Relation #8**

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Relation #8 has a cycle. k-optimization will be used.

PIN matrix:

N N P N P P N P P N P P P N P

N N P N P P N P P N P P P N P

I N P P N P P N P I P P

N N N N N N N N N N N N N N N

N N N N N N N N N N

N N N N N N N N N N

N N N N N N N N N N N N N N N

N N N N N N N N N N

N N N N N N N N N N

N N N N N N N N N N N N N N N

N N N N N N N N N N

I N P P N P P N P I P P

N N N N N N N N N N

N N P N P P N P P N P P P N P

N N N N N N N N N N

k1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 1 1 1 1 1 1 1 1 1 1 1 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

0 0 1 1 1 1 1 1 1 1 1 1 1 0 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

k1 max elements: {0, 1, 3, 6, 9, 13}

k1 opt elements: {0, 1, 3, 6, 9, 13}

k2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 1 1 1 1 1 1 1 1 0 1 0 1

k2 max elements: {0, 1, 3, 6, 9, 13}

k2 opt elements: {0, 1, 3, 6, 9, 13}

k3

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

k3 max elements: {0, 1, 2, 11, 13}

k3 opt elements: {}

k4

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 0 0 1 1 0 1 1 0 1 0 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 0 1 1 0 1 0 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 1 0 1 1 0 1 1 0 1 1 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

k4 max elements: {0, 1, 13}

k4 opt elements: {}

**Relation #9**

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

Relation #9 has a cycle. k-optimization will be used.

PIN matrix:

N N N N N N N N N N

N N N N N N N N N N

P P N N P P N P P P N N P N P

N N N N N N N N N N N N N N N

N N N N N N N N N N

N N N N N N N N N N

P P N N P P N P P P N N P N P

P P N P P I P P N N P I

N N N N N N N N N N

N N N N N N N N N N

N N N N N N N N N N N N N N N

N N N N N N N N N N N N N N N

N N N N N N N N N N

P P N N P P N P P P N N P N P

P P N P P I P P N N P I

k1

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 0 1 1 1 0 1 1 1 1 1 1 0 1

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 0 1 1 1 0 1 1 1 1 1 1 0 1

k1 max elements: {2, 3, 6, 10, 11, 13}

k1 opt elements: {2, 3, 6, 10, 11, 13}

k2

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 0 1 1 1 0 0 1 1 1 1 1 0 0

k2 max elements: {2, 3, 6, 10, 11, 13}

k2 opt elements: {2, 3, 6, 10, 11, 13}

k3

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

k3 max elements: {2, 6, 7, 13, 14}

k3 opt elements: {}

k4

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

1 1 0 0 1 1 0 0 1 1 0 0 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 1 1 0 1 1 1 0 0 1 0 1

1 1 0 0 1 1 0 0 1 1 0 0 1 0 0

k4 max elements: {2, 13, 6}

k4 opt elements: {}

**Relation #10**

0 0 0 1 1 0 1 0 1 1 1 0 0 1 1

1 0 0 0 1 0 0 0 1 0 0 0 1 1 1

1 0 0 1 0 1 0 1 0 0 1 1 1 0 0

0 0 0 0 0 1 0 1 0 0 1 1 1 0 1

0 0 0 0 0 1 1 1 1 1 1 0 1 0 0

0 0 0 0 0 0 1 1 0 1 0 0 1 1 0

0 0 0 0 0 0 0 0 1 1 1 0 1 0 1

0 0 0 0 0 0 0 0 1 1 0 0 1 1 1

0 0 0 0 0 0 0 0 0 1 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 0 0 1 0

0 0 0 0 0 0 0 0 0 0 0 1 0 1 1

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

Relation #10 is acyclic. Neumann-Morgenstern optimization will be used.

S0: {1, 2}

S1: {0, 1, 2}

S2: {0, 1, 2, 3, 4}

S3: {0, 1, 2, 3, 4, 5}

S4: {0, 1, 2, 3, 4, 5, 6, 7}

S5: {0, 1, 2, 3, 4, 5, 6, 7, 8}

S6: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

S7: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

S8: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

S9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

S10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14}

S11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Q0: {1, 2}

Q1: {1, 2}

Q2: {1, 2}

Q3: {1, 2}

Q4: {1, 2, 6}

Q5: {1, 2, 6}

Q6: {1, 2, 6}

Q7: {1, 2, 6}

Q8: {1, 2, 6}

Q9: {1, 2, 6}

Q10: {1, 2, 6}

Q11: {1, 2, 6}

Neumann-Morgenstern solution: {1, 2, 6}

Internal stability: True

External stability: True

Програмний код  
  
import numpy as np

from typing import List, Set

def read\_relations(filename, size, count):

    relations = [np.zeros((size, size)) for \_ in range(count)]

    def read\_one\_relation(file):

        # skip first row of a relation because it only contains name and dashes (---)

        file.readline()

        for row in range(size):

            line = file.readline()

            column = 0

            for char in line:

                if char != ' ' and (char == '1' or char == '0'):

                    relations[num\_rel][row][column] = int(char)

                    column += 1

    with open(filename, "r") as file:

        for num\_rel in range(count):

            read\_one\_relation(file)

    return relations

def print\_relation(relation, num\_rel=0):

    if num\_rel != 0:

        print\_to\_file(f"Relation #{num\_rel}")

    for row in relation:

        for element in row:

            if isinstance(element, int) or isinstance(element, float):

                print\_to\_file("{:.0f}".format(element) + " ", end="")

            else:

                print\_to\_file(element + " ", end="")

        print\_to\_file()

    print\_to\_file()

def print\_to\_file(data="", end="\n"):

    with open("results.txt", "a") as file:

        file.write(str(data))

        file.write(end)

# ----------

def has\_cycle(relation):

    def does\_vertex\_has\_cycle(vertex\_index):

        reachable\_vertexes = set()

        visited\_vertexes = set()

        for (index, value) in enumerate(relation[vertex\_index]):

            if value == 1:

                reachable\_vertexes.add(index)

        visited\_vertexes.add(vertex\_index)

        if reachable\_vertexes.\_\_contains\_\_(vertex\_index):

            return True

        while len(visited\_vertexes) != len(relation) and len(reachable\_vertexes) > 0:

            unvisited\_reachable\_vertexes = reachable\_vertexes.difference(

                visited\_vertexes)

            if len(unvisited\_reachable\_vertexes) == 0:

                return False

            current\_vertex = unvisited\_reachable\_vertexes.pop()

            visited\_vertexes.add(current\_vertex)

            for (index, value) in enumerate(relation[current\_vertex]):

                if value == 1:

                    if index == vertex\_index:

                        return True

                    reachable\_vertexes.add(index)

            visited\_vertexes.add(current\_vertex)

        return False

    for vertex\_index in range(len(relation)):

        if does\_vertex\_has\_cycle(vertex\_index):

            return True

    return False

def solve\_NM(relation):

    def get\_upper\_cut(vertex):

        return set(index for (index, value) in enumerate(relation[:, vertex]) if value == 1)

    def get\_S\_sets():

        all\_S\_sets = [set()]

        # S[0]

        for vertex in range(len(relation)):

            if len(get\_upper\_cut(vertex)) == 0:

                all\_S\_sets[0].add(vertex)

        # S[k]

        while len(all\_S\_sets[-1]) != len(relation):

            vertexes\_dominated\_only\_by\_prev\_S = set(vertex for vertex in range(len(relation))

                                                    if get\_upper\_cut(vertex).issubset(all\_S\_sets[-1]))

            all\_S\_sets.append(

                all\_S\_sets[-1].union(vertexes\_dominated\_only\_by\_prev\_S))

        return all\_S\_sets

    def get\_Q\_sets(all\_S\_sets: List[Set]):

        all\_Q\_sets = [all\_S\_sets[0]]  # need to make a copy explicitly?

        for i in range(1, len(all\_S\_sets)):

            diff\_S\_curr\_and\_S\_prev = all\_S\_sets[i].difference(all\_S\_sets[i-1])

            just\_the\_required\_condition = set(vertex for vertex in diff\_S\_curr\_and\_S\_prev

                                              if len(get\_upper\_cut(vertex).intersection(all\_Q\_sets[-1])) == 0)

            all\_Q\_sets.append(

                all\_Q\_sets[-1].union(just\_the\_required\_condition))

        return all\_Q\_sets

    def check\_internal\_stability(vertexes):

        for row in vertexes:

            for column in vertexes:

                if relation[row][column] == 1:

                    return False

        return True

    def check\_external\_stability(vertexes):

        other\_vertexes = set(range(len(relation))).difference(vertexes)

        for column in other\_vertexes:

            found\_better\_in\_set = False

            for row in vertexes:

                if relation[row][column] == 1:

                    found\_better\_in\_set = True

                    break

            if found\_better\_in\_set == False:

                return False

        return True

    def print\_sets(sets, type):

        for (index, set) in enumerate(sets):

            print\_to\_file(f"{type}{index}: {set}")

        print\_to\_file()

    S\_sets = get\_S\_sets()

    print\_sets(S\_sets, "S")

    Q\_sets = get\_Q\_sets(S\_sets)

    solution = Q\_sets[-1]

    print\_sets(Q\_sets, "Q")

    int\_stab = check\_internal\_stability(solution)

    ext\_stab = check\_external\_stability(solution)

    print\_to\_file(f"Neumann-Morgenstern solution: {solution}")

    print\_to\_file(f"Internal stability: {int\_stab}")

    print\_to\_file(f"External stability: {ext\_stab}")

    print\_to\_file()

def solve\_k\_optimization(relation):

    def get\_PIN\_matrix():

        PIN\_matrix = [[] for \_ in range(len(relation))]

        for row in range(len(relation)):

            for column in range(len(relation)):

                if relation[row][column] == 1 and relation[column][row] == 0:

                    PIN\_matrix[row].append("P")

                elif relation[row][column] == 1 and relation[column][row] == 1:

                    PIN\_matrix[row].append("I")

                elif relation[row][column] == 0 and relation[column][row] == 0:

                    PIN\_matrix[row].append("N")

                else:

                    PIN\_matrix[row].append(" ")

        return PIN\_matrix

    def get\_H\_matrix(k):

        H\_matrix = []

        PIN\_matrix = get\_PIN\_matrix()

        for row in range(len(relation)):

            H\_matrix.append([0 for \_ in range(len(relation))])

            for column in range(len(relation)):

                if k == 1 and (PIN\_matrix[row][column] == "P" or PIN\_matrix[row][column] == "I" or PIN\_matrix[row][column] == "N"):

                    H\_matrix[row][column] = 1

                elif k == 2 and (PIN\_matrix[row][column] == "P" or PIN\_matrix[row][column] == "N"):

                    H\_matrix[row][column] = 1

                elif k == 3 and (PIN\_matrix[row][column] == "P" or PIN\_matrix[row][column] == "I"):

                    H\_matrix[row][column] = 1

                elif k == 4 and (PIN\_matrix[row][column] == "P"):

                    H\_matrix[row][column] = 1

                else:

                    H\_matrix[row][column] = 0

        return H\_matrix

    def is\_subset\_or\_equal(list1, list2):

        for i in range(len(list1)):

            if list1[i] == 1 and list2[i] == 0:

                return False

        return True

    def get\_max\_and\_opt(k):

        max\_alts = set()

        max\_alt\_size = 0

        opt\_alts = set()

        H\_matrix = get\_H\_matrix(k)

        for vertex in range(len(relation)):

            if H\_matrix[vertex].count(1) > 0:

                if len(max\_alts) == 0:

                    max\_alts.add(vertex)

                    max\_alt\_size = H\_matrix[vertex].count(1)

                elif H\_matrix[vertex].count(1) > max\_alt\_size:

                    max\_alts = set([vertex])

                    max\_alt\_size = H\_matrix[vertex].count(1)

                elif H\_matrix[vertex].count(1) == max\_alt\_size:

                    if is\_subset\_or\_equal(H\_matrix[vertex],  H\_matrix[list(max\_alts)[0]]):

                        max\_alts.add(vertex)

        if max\_alt\_size > 0:

            for H\_row in H\_matrix:

                if not is\_subset\_or\_equal(H\_row, H\_matrix[list(max\_alts)[0]]):

                    max\_alts = set()

                    max\_alt\_size = 0

                    break

        if max\_alt\_size == len(relation):

            opt\_alts = max\_alts

        return max\_alts, opt\_alts

    print\_to\_file("PIN matrix:")

    print\_relation(get\_PIN\_matrix())

    for i in range(1, 5):

        print\_to\_file(f"k{i}")

        print\_relation(get\_H\_matrix(i))

        max\_alts, opt\_alts = get\_max\_and\_opt(i)

        print\_to\_file(f"k{i} max elements: {max\_alts}")

        print\_to\_file(f"k{i} opt elements: {opt\_alts}")

        print\_to\_file()

def main():

    size = 15

    count = 10

    with open("results.txt", "w") as file:

        file.truncate()

    relations = read\_relations("Варіант №1.txt", size, count)

    for (index, relation) in enumerate(relations):

        print\_relation(relation, index+1)

        if has\_cycle(relation):

            print\_to\_file(

                f"Relation #{index+1} has a cycle. k-optimization will be used.")

            solve\_k\_optimization(relation)

        else:

            print\_to\_file(

                f"Relation #{index+1} is acyclic. Neumann-Morgenstern optimization will be used.")

            solve\_NM(relation)

# --------------------------------

main()