

**Міністерство освіти і науки України
Національний лісотехнічний університет України**

**Микола Сало,
Софія Поберейко,
Андрій Нечепуренко,
Михайло Дендюк**

Методи та системи штучного інтелекту

Лабораторний практикум

Львів – 2017

Сало М.Ф., Поберейко С.Б., Нечепуренко А.В., Дендюк М.В.

Методи і системи штучного інтелекту: Лаб. практи. – Львів: Вид-во НЛТУ України, 2017. – 86 с.

Мета дисципліни “Методи та системи штучного інтелекту” – це формування знань, вмінь та навичок, необхідних для розроблення і застосування моделей відображення знань, стратегій логічного виведення, технологій інженерії знань, технологій і інструментальних засобів інтелектуальних систем.

Завдання курсу – вивчення сучасних методів і технологій штучного інтелекту, розроблення та застосування моделей представлення знань для побудови інтелектуальних систем.

Практикум призначено для студентів ВУЗів, що вивчають комп'ютерні науки та інформаційні системи і технології.

Навчальне видання

Микола Федорович САЛО – ст.викладач каф. інформаційних технологій НЛТУ України

Софія Богданівна Поберейко – асистент каф. інформаційних технологій НЛТУ України

Нечепуренко Андрій Валерійович – асистент каф. інформаційних технологій НЛТУ України

Михайло Володимирович ДЕНДЮК – кандидат техн. наук, доцент каф. інформаційних технологій НЛТУ України

Методи та системи штучного інтелекту

ЗМІСТ

ВСТУП	5
ЛАБОРАТОРНА РОБОТА №1. РОЗПІЗНАВАННЯ СИМВОЛІВ НА ДЕВЯТИСЕГМЕНТНОМУ ТАБЛО	6
1.1. Програма роботи	6
1.2. Вказівки до виконання роботи	6
1.3. Теоретичні відомості	7
1.4. Хід роботи	9
1.5. Індивідуальні завдання	11
ЛАБОРАТОРНА РОБОТА №2. РОЗПІЗНАВАННЯ СИМВОЛІВ НА ДЕВЯТИСЕГМЕНТНОМУ ТАБЛО В УМОВАХ НЕПОВНОТИ ІНФОРМАЦІЇ	13
2.1. Програма роботи	13
2.2. Вказівки до виконання роботи	13
2.3. Теоретичні відомості	13
2.4. Індивідуальні завдання	14
Лабораторна робота №3 ВИЯВЛЕННЯ ЛОГІЧНИХ ЗАКОНОМІРНОСТЕЙ В ДАНИХ	15
3.1. Програма роботи	15
3.2. Вказівки до виконання роботи	15
3.3. Теоретичні відомості	15
3.4. Індивідуальні завдання	21
ЛАБОРАТОРНА РОБОТА №4. ОБХІД ГРАФІВ ВШИР ТА ВГЛИБ	26
4.1. Програма роботи	26
4.2. Вказівки до виконання роботи	26
4.3. Теоретичні відомості	26
4.4. Індивідуальні завдання	30
ЛАБОРАТОРНА РОБОТА №5. ЖАДІБНИЙ АЛГОРИТМ	35
5.1. Програма роботи	35
5.2. Вказівки до виконання роботи	35
5.3. Теоретичні відомості	35
5.4. Індивідуальні завдання	38

ЛАБОРАТОРНА РОБОТА №6. АЛГОРИТМ А*	41
6.1. Програма роботи	41
6.2. Вказівки до виконання роботи	41
6.3. Теоретичні відомості	41
6.4. Індивідуальні завдання	49
Лабораторна робота №7. ГЕНЕТИЧНІ АЛГОРИТМИ	53
7.1. Програма роботи	53
7.2. Вказівки до виконання роботи	53
7.3. Теоретичні відомості	53
7.4. Індивідуальні завдання	56
ЛАБОРАТОРНА РОБОТА №8. ПОБУДОВА ФУНКЦІЙ НАЛЕЖНОСТІ ЛІНГВІСТИЧНОЇ ЗМІННОЇ	58
8.1. Програма роботи	58
8.2. Вказівки до виконання роботи	58
8.3. Теоретичні відомості	58
8.4. Індивідуальні завдання	65
ЛАБОРАТОРНА РОБОТА №9. ПОБУДОВА БАЗИ ЗНАНЬ НЕЧІТКОЇ ЕКСПЕРТНОЇ СИСТЕМИ	66
9.1. Програма роботи	66
9.2. Вказівки до виконання роботи	66
ЛАБОРАТОРНА РОБОТА №10. МОДЕЛІ ШТУЧНОГО НЕЙРОНА	67
9.1. Програма роботи	67
9.2. Вказівки до виконання роботи	67
9.3. Теоретичні відомості	68
9.4. Індивідуальні завдання	72
ЛАБОРАТОРНА РОБОТА №11. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ	78
10.1. Програма роботи	78
10.2. Вказівки до виконання роботи	78
10.3. Теоретичні відомості	79
10.4. Індивідуальні завдання	81



ВСТУП

Система штучного інтелекту (СШІ) – галузь науки, яка займається теоретичними дослідженнями, розробленням і застосуванням алгоритмічних та програмно-апаратних систем і комплексів з елементами штучного інтелекту та моделюванням інтелектуальної діяльності людини.

Можна виділити два напрямки розвитку СШІ :

- рішення проблем, пов'язаних з наближенням спеціалізованих систем ШІ до можливостей людини, та їх інтеграції, яка реалізована природою людини (посилення інтелекту);
- створення штучного розуму, що представляє інтеграцію вже створених систем ШІ в єдину систему, здатну вирішувати проблеми людства (слабкий і сильний штучний інтелект).

Для першого напрямку характерні та експертні системи.

У другому напрямі використовується нечітка логіка та штучні нейронні мережі.



ЛАБОРАТОРНА РОБОТА №1. РОЗПІЗНАВАННЯ СИМВОЛІВ НА ДЕВЯТИСЕГМЕНТНОМУ ТАБЛО

МЕТА РОБОТИ: Ознайомитись із основними принципами розпізнавання певної абетки символів, що задається на девятисегментному табло.

1.1. Програма роботи

1.1.1. Отримати завдання.

1.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

1.1.3. Підготувати власні коректні входні дані (вказати їх формат і значення) і проаналізувати їх.

1.1.4. Оформити електронний звіт про роботу та захистити її.

1.2. Вказівки до виконання роботи

1.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 1.5 і записує його до звіту.

1.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 1.4.

1.2.3. Власних входних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

1.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних .h і .cpp файлів, а також пояснення до них;
- результати реалізації програми, які виведені в вікно форми;
- діаграму класів з поясненням;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

1.3. Теоретичні відомості

Сегментний індикатор – це індикатор, елементи відображення якого є сегментами, згрупованими в одне або кілька знакісць. Сегментом називається елемент відображення інформації індикатора, контур якого представляє собою прямі і (або) криві лінії. На відміну від матричного індикатора, в якому всі елементи зображення однакові за формою, в сегментному індикаторі кожен сегмент унікальний. Форма і положення сегментів на індикаторі розробляється спеціально для передачі певного набору символів або знаків. Символи на таких індикаторах формуються сукупністю кількох сегментів. Основна відмінність сегментного індикатора від матричного – це порівняно невелика кількість елементів індикації і відповідно спрощена схема управління.

Найбільш часто використовуються два типи сегментних індикаторів:

Цифрові семисегментні індикатори, що мають вісім (це не помилка) елементів – сім сегментів для індикації цифри і один – для точки.

Буквено-цифрові індикатори, мають дев'ять, чотирнадцять або шістнадцять сегментів. Такі індикатори мають можливість показати більшість символів латинського алфавіту.

Для проведення розпізнавання виведених на 9-ти сегментне табло символів, як і при розробці будь-якої системи розпізнавання, найперше потрібно сформувати базу знань. Полями бази знань потрібно вибирати характеристики, які є водночас однакові за наявністю у різних символах, проте різні за значеннями. В умовах девятисегментного табло варто застосувати наступні поля:

- x_1 – кількість вертикальних ліній мінімального розміру;
- x_2 – кількість горизонтальних ліній;
- x_3 – кількість похилих ліній;

Якщо після побудови у базі знань серез записів будуть однакові, то виникає необхідність введення ще одного поля, ним може бути ознака горіння певного сегмента, яким різняться символи з однаковими записами у БЗ.

До прикладу, нехай задано написання цифр у діапазоні 0-9 для девятисегментного табло (рис.1.1), проведемо побудову бази знань.

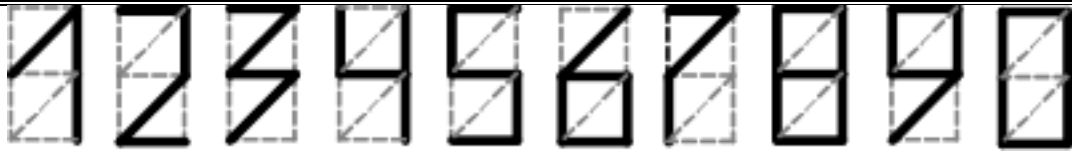


Рис. 1.1. Приклад запису символів на 9-сегментному табло

За вказаними вище полями отримаємо табл. 1.

Табл. 1. Спрощена БЗ опису символів на 9-сегментному табло

СИМВОЛ	x_1	x_2	x_3
1	2	0	1
2	1	2	1
3	0	2	2
4	3	1	0
5	2	3	0
6	2	2	1
7	1	1	1
8	4	3	0
9	2	2	1
0	4	2	0

Як бачимо записи для цифр «6» та «9» є однаковими, отже введемо ще одну характеристику x_4 – кількість горизонтальних ліній знизу об'єкту, і побудуємо знову базу знань (табл. 2).

Табл. 2. Розширена БЗ опису символів на 9-сегментному табло

СИМВОЛ	x_1	x_2	x_3	x_4
1	2	0	1	0
2	1	2	1	1
3	0	2	2	0
4	3	1	0	0
5	2	3	0	1
6	2	2	1	1
7	1	1	1	0
8	4	3	0	1
9	2	2	1	0
0	4	2	0	1

Таким чином ми отримали БЗ, яка однозначно описує кожен із перелічених символів, що дає змогу виконати зворотну задачу, а саме ідентифікації цифри за кількісними характеристиками величин x_1 , x_2 , x_3 та x_4 .

1.4. Хід роботи

1. Згідно індивідуального завдання розробити базу знань (табл. 2) для об'єктів, які необхідно розпізнавати.

2. У блокноті створити файл <ім'я>.txt із зображенням символу, який будемо розпізнавати та зберегти його. Розмір матриці, у яку заноситься зображення цього символу, 9 рядків та 5 стовпців (рис. 1.2). Горизонтальні сегменти позначені голубим кольором, вертикальні – жовтим, а діагональні – зеленим. Кутові елементи (клітинки) сегментів можуть належати до як горизонтальних або до вертикальних сегментів, так і бути порожніми.

ж	ж	ж	ж	ж
ж			ж	ж
ж		ж		ж
ж	ж			ж
ж	ж	ж	ж	ж
ж			ж	ж
ж		ж		ж
ж	ж			ж
ж	ж	ж	ж	ж

Рис. 1.2. Приклад розмітки сегментів у 9-сегментному табло

Для прикладу, файл із зображенням символу «А», наведено на рис. 1.3.

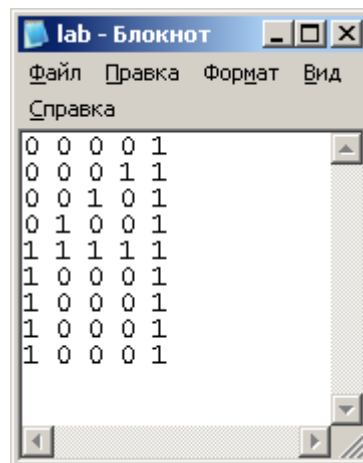


Рис. 1.3. Приклад запису символу «А» у файлі для подальшого розпізнавання

1. Під час написання коду програми необхідно:

- ініціалізувати розроблену БЗ;
- прочитати із файлу зображення символу, який треба розпізнати та записати його в матрицю символу;
- визначити кількість x_1 , x_2 , x_3 та x_4 сегментів у матриці символу. При цьому необхідно враховувати, що сегмент не засвічується, якщо кількість символів у сегменті (одиниць) менша за 3;
- обчислити ключ для розпізнавання символу :
$$\text{key} = 1000 * x_1 + 100 * x_2 + 10 * x_3 + x_4;$$
- у циклі знайти індекс співпадання ключа заданого символу та ключів БД;
- вивести результати.

Фрагмент коду для реалізації у VS C++:

```

#pragma hdrstop
#include <iostream>
#include <fstream>
using namespace std;
//-----
#pragma argsused
void main(int argc, char* argv[])
{

int mas[11][6];
int base[9]={310,210,420,320,110,430,410,301,201};
char base_chars [9]= {'J','L','O','P','r','S','U','V','Y'};
int ver = 0, hor = 0, diag = 0;
int Smm[10];
int bv, bh, bd;

ifstream potik;
potik.open("l1.txt");
for (int i=0;i<9;i++)
    for (int j=0;j<5;j++)
        potik>>mas[i][j];

//горизонталь
for (int i=0;i<9;i+=4)
    for(int j=1;j<4;j++){
        if (mas[i][j]==1)
            hor++;}

hor/=3;
//вертикаль
for (int i=1;i<4;i++)
    for(int j=0;j<4;j+=4)
        if (mas[i][j]==1)
            ver++;

for (int i=5;i<8;i++)
    for(int j=0;j<4;j+=4)
        if (mas[i][j]==1)
            ver++;

ver/=3;

//похили
for(int i=1;i<4;i++)
    if (mas[i][4-i]==1)
        diag++;
    for(int i=5;i<8;i++)
        if (mas[i][8-i]==1)
            diag++;
diag/=3;

int key;
key=100*ver+10*hor+diag;
for (int i=0;i<8;i++)
{
    if(key==base[i])
    {
        cout<<base_chars[i];
    }
}
}

```

1.5. Індивідуальні завдання

Згідно номера варіанту представити задані символи у рамках 9-ти сегментного індикатора. Сформувати базу знань для системи розпізнання. Розро-

бити програмний код, що дозволяв би визначити представлений на табло символ.

- 1) A, Б, В, Г, Е, З, І, К, Л, Н;
- 2) О, 4, П, Р, С, 7, Ч, 2, Ї, 9;
- 3) Б, Г, Л, І, Р, С, К, П, Е, Ї;
- 4) A, B, C, d, G, E, F, J, I, H;
- 5) L, 3, O, P, r, S, 8, U, Y, 4;
- 6) 1, O, 2, D, 3, F, 4, Ч, 5, П;
- 7) A, B, E, I, Л, O, П, C, Ч, Ї;
- 8) Б, Г, Е, І, Н, П, Р, С, 2, 5;
- 9) A, C, G, F, J, H, L, S, U, Y;
- 10) B, C, d, G, E, I, 3, O, P, S;
- 11) 2, 3, C, d, G, E, F, J, I, n;
- 12) C, d, L, O, P, r, S, A, B, G;
- 13) A, d, G, F, H, L, O, P, r, U;
- 14) B, O, P, r, S, C, d, G, E, F;
- 15) L, U, C, d, G, S, 8, A, I, H.

ЛАБОРАТОРНА РОБОТА №2. РОЗПІЗНАВАННЯ СИМВОЛІВ НА ДЕВЯТИСЕГМЕНТНОМУ ТАБЛО В УМОВАХ НЕПОВНОТИ ІНФОРМАЦІЇ

МЕТА РОБОТИ: освоїти технологію створення програм в консолі з використанням класів, розв'язувати поставлені задачі з масивами об'єктів.

2.1. Програма роботи

2.1.1 Отримати завдання.

2.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

2.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

2.1.4. Оформити електронний звіт про роботу та захистити її.

2.2. Вказівки до виконання роботи

2.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 2.5 і записує його до звіту.

2.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 2.4.

2.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

2.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних .h і .cpp файлів, а також пояснення до них;
- результати реалізації програми;
- діаграму класів та діаграму варіантів використання з поясненням;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

2.3. Теоретичні відомості

Дуже часто під час читання інформації відбувається її спотворення. Для розпізнавання спотвореної інформації використовуються різноманітні методи, один з яких – мінімальна оцінка близькості.

Щоб визначити символ для 9-ти сегментного індикатора в умовах неповноти інформації або її невизначеності, необхідно спершу побудувати таблицю навчання розпізнаванню образів (базу знань) для можливих варіантів.

Ознаки таблиці навчання: x_1 – кількість вертикальних ліній мінімального розміру; x_2 – кількість горизонтальних ліній; x_3 – кількість похилих ліній; x_4 – кількість горизонтальних ліній знизу об'єкту.

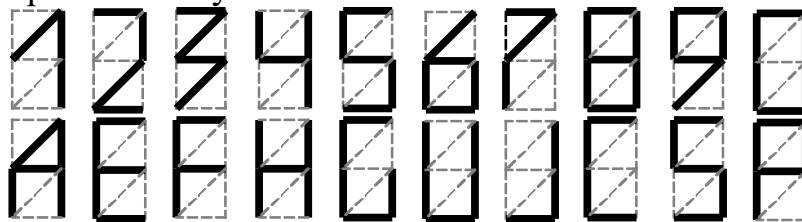
За отриманою таблицею необхідно обчислити суму різниць модулів кожної ознаки кожного з можливих варіантів та символу, який розпізнається.

Найменша сума вкаже на можливий варіант відповіді.

Приклад. Побудувати таблицю навчання розпізнаванню образів за 9-ти сегментним таблом для символів: 1, 2, 4, 5 та за мінімальною оцінкою близькості визначити розпізнаний символ:



Зразок запису символів:



Таблиця навчання:

	x_1	x_2	x_3	x_4	$\min \Sigma$
1	2	0	1	0	1
2	2	1	1	1	3
4	3	1	0	0	2
5	2	3	0	1	5
	3	0	1	0	

2.4. Індивідуальні завдання

1. Згідно номера варіанту (лабораторна робота №1) представити задані символи у рамках 9-ти сегментного індикатора.

2. Сформувати базу знань для системи розпізнання.

3. У якості символу, який необхідно розпізнати, задати один із символів БЗ спотвореним.

4. Розробити програмний код, що дозволяв би визначити спотворений символ за мінімальною оцінкою близькості з таблиці навчання розпізнаванню образів.

Лабораторна робота №3 ВИЯВЛЕННЯ ЛОГІЧНИХ ЗАКОНОМІРНОСТЕЙ В ДАНИХ

МЕТА РОБОТИ: засвоїти основні принципи виявлення логічних закономірностей у даних.

3.1. Програма роботи

3.1.1 Отримати завдання.

3.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

3.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

3.1.4. Оформити електронний звіт про роботу та захистити її.

3.2. Вказівки до виконання роботи

3.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 3.5 і записує його до звіту.

3.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 3.4.

3.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

3.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних .h і .cpp файлів, а також пояснення до них;
- результати реалізації програми;
- діаграму класів та діаграму варіантів використання з поясненням;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

3.3. Теоретичні відомості

Розглянемо рис. 3.1. На ньому схематично зображено лице людей. Ці лиця із якихось причин, можливо, важливим, розділені на два класи. Ставиться задача знайти закономірності проведеного розділення.

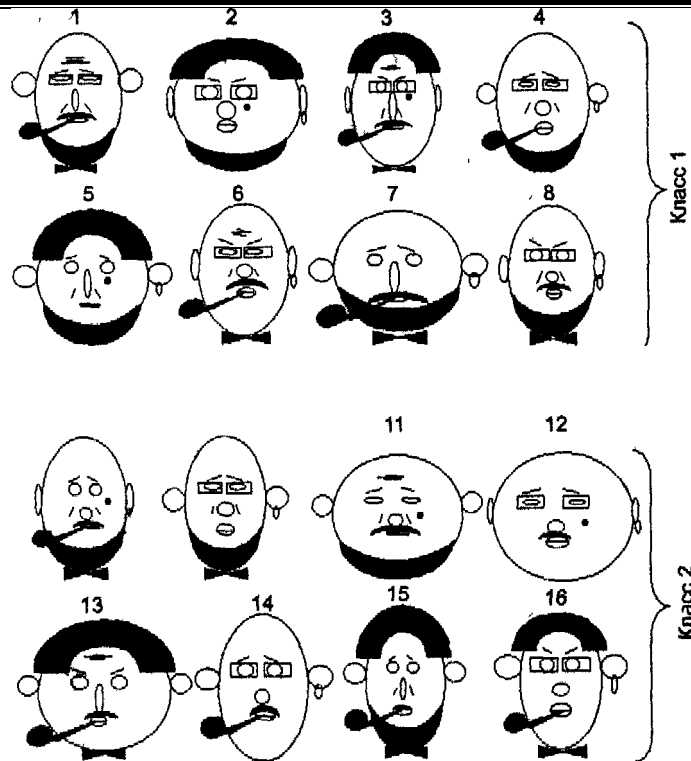


Рис. 3.1. Зображення облич людей

Спробуйте візуально визначити, чим лиця людей різних класів відрізняються один від одного і що об'єднує лиця одного класу. Відразу помітимо — рішення існує. Але ваш візуальний аналіз, швидше за все, не дасть відповіді на поставлене питання. Звичайний людський розум не в змозі вирішити навіть таку, на перший погляд просту, задачу виявлення прихованих закономірностей. Тут необхідне застосування комп'ютерних методів аналізу даних.

Перш за все виділимо ознаки, що характеризують зображені лиця. Це такі характеристики:

- x_1 (голова) — кругла — 1, овальна — 0;
- x_2 (вуха) — відтопирені — 1, притиснуті — 0;
- x_3 (ніс) — круглий — 1, довгий — 0;
- x_4 (очі) — круглі — 1, вузькі — 0;
- x_5 (лоб) — із зморшками — 1, без зморшок — 0;
- x_6 (складка) — носогубна складка є — 1, носогубної складки немає — 0;
- x_7 (губи) — товсті — 1, тонкі — 0;
- x_8 (волосся) — є — 1, немає — 0;
- x_9 (вуса) — є — 1, немає — 0;
- x_{10} (борода) — є — 1, немає — 0;
- x_{11} (окуляри) — є — 1, немає — 0;
- x_{12} (родимка) — родимка на щоці є — 1, родимки на щоці немає — 0;
- x_{13} (метелик) — є — 1, немає — 0;
- x_{14} (брови) — підняті догори — 1, опущені донизу — 0;
- x_{15} (сережка) — є — 1, немає — 0;
- x_{16} (трубка) — курильна трубка є — 1, немає — 0.

Початкова матриця даних, що відповідає зображенням лицям, представлена в табл. Ошибка! Текст указанного стиля в документе отсутствует..1. Рядки відповідають об'єктам ($N=16$), стовпці — виділеним бінарним ознакам ($p=16$). Об'єкти з номерами 1-8 відносяться до класу 1, а з номерами 9-16 — до класу 2.

Табл. Ошибка! Текст указанного стиля в документе отсутствует..1. Матриця початкових даних

№ п/п	Голова	Вуха	Ніс	Очі	Лоб	Складка	Губи	Волося	Вуса	Борода	Окуляри	Родимка	Метелик	Брови	Сережка	Трубка	Class
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	
	0	1	0	0	1	1	0	0	1	1	1	0	1	1	0	1	1
	1	0	1	1	0	0	1	1	0	1	1	1	0	0	1	0	1
	0	0	0	1	1	1	0	1	1	0	1	1	1	0	0	1	1
	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1	1
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	1	0	1
	0	0	1	0	1	1	1	0	1	0	1	0	1	0	1	1	1
	1	1	0	1	0	0	0	0	1	1	0	0	1	1	1	1	1
	0	0	1	1	0	1	1	0	1	1	1	0	1	0	1	0	1
	0	0	1	1	0	1	0	0	1	1	0	1	1	1	0	1	2
	0	1	1	0	0	1	1	0	0	1	1	0	1	1	1	0	2
	1	1	1	0	1	1	0	0	1	1	0	1	0	1	0	0	2
	1	0	1	0	1	0	1	0	1	0	1	1	0	1	1	0	2
	1	1	0	1	1	0	1	1	1	0	0	0	1	0	0	1	2
	0	1	1	1	0	0	1	0	1	0	1	0	0	1	1	1	2
	0	1	0	1	0	1	1	1	0	1	0	0	1	1	0	1	2
	0	1	1	1	0	0	1	1	0	0	1	0	1	0	1	1	2

Основна вимога до математичного апарату виявлення закономірностей в даних (окрім, звичайно, вимоги ефективності) полягає в інтерпретації результатів. Правила, що відображають знайдені закономірності, повинні формулюватися на простій і зрозумілій людині мові логічних висловлювань. Наприклад, **ЯКЩО** {(подія 1) і (подія 2) і... і (подія N)} **ТО**... Іншими словами, це повинні бути логічні правила.

Так, класифікація лиця в розглянутому прикладі може бути проведений за допомогою чотирьох логічних правил:

1. **ЯКЩО** {(голова овальна) і (є носогубна складка) і (є окуляри) і (є трубка)} **ТО** (Клас 1).
2. **ЯКЩО** {(очі круглі) і (лоб без зморшок) і (є борода) і (є сережка)} **ТО** (Клас 1).
3. **ЯКЩО** {(ніс круглий) і (лисий) і (є вуса) і (брови підняті догори)} **ТО** (Клас 2).

4. ЯКЩО {(відтопирені вуха) і (товсті губи) і (немає родимки на щоці) і (є метелик)} **ТО** (Клас 2).

Математичний запис цих правил виглядає таким чином:

$$[(x_1=0) \wedge (x_6=1) \wedge (x_{11}=1) \wedge (x_{16}=1)] \vee [(x_4=1) \wedge (x_5=0) \wedge (x_{10}=1) \wedge (x_{15}=1)] \Rightarrow \omega_1,$$

$$[(x_3=1) \wedge (x_8=1) \wedge (x_9=1) \wedge (x_{14}=1)] \vee [(x_2=1) \wedge (x_7=1) \wedge (x_{12}=0) \wedge (x_{13}=1)] \Rightarrow \omega_2.$$

Тут значки \wedge — кон'юнкція (і), \vee — диз'юнкція (або), \Rightarrow — імплікація (якщо, то).

Під правило 1 підпадають перша, третя, четверта і шоста особи з першого класу; під правило 2 — друга, п'ята, сьома і восьма особа з першого класу; під правило 3 — дев'ята, одинадцята, дванадцята і чотирнадцята особа з другого класу; під правило 4 — десята, тринадцята, п'ятнадцята і шістнадцята особа з другого класу.

Дерева рішень (decision trees) є найпоширенішим в даний час підходом до виявлення і зображення логічних закономірностей в даних. Знані представники цього підходу — системи CHAID (chi square automatic interaction detection), CART (classification and regression trees) і ID3 (Interactive Dichotomizer — інтерактивний дихотомайзер). Розглянемо більш детально процедуру побудови дерев рішень на прикладі системи ID3.

В основі системи ID3 лежить алгоритм CLS. Цей алгоритм циклічно розбиває навчальні приклади на класи відповідно до змінної, що має найбільшу класифікуючу силу. Кожна підмножина прикладів (об'єктів), що виділяється такою змінною, знову розбивається на класи з використанням наступної змінної з найбільшою класифікуючою здатністю і т.д. Розбиття закінчується, коли в підмножині опиняються об'єкти лише одного класу. В ході процесу утворюється дерево рішень. Шляхи руху по цьому дереву з верхнього рівня на самі нижні визначають логічні правила у вигляді послідовностей кон'юнкцій.

Для ілюстрації роботи алгоритму CLS звернемося наприклад по класифікації лиць людей (див. рис. 3.1 і табл. **Ошибка! Текст указанного стиля в документе отсутствует..1**).

На першому кроці алгоритму визначається ознака з найбільшою дискримінуючою силою. В нашому випадку однаковою і максимальною силою володіють відразу 7 ознак — x_2 , x_3 , x_6 , x_{10} , x_{11} , x_{14} і x_{15} (табл. **Ошибка! Текст указанного стиля в документе отсутствует..2**). Тому тут ми ухвалюємо волевирішення і призначаємо першою ознакою, наприклад, x_6 .

*Табл. **Ошибка! Текст указанного стиля в документе отсутствует..2**. Відношення одиниць (1) в різних класах об'єктів для різних ознак*

Ознаки	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
Клас1/Клас2	3/3	4/6	4/6	5/5	3/3	6/4	4/6	3/3	5/5	6/4	6/4	3/3	5/5	4/6	4/6	5/5

Від першої ознаки відходять дві гілки. Перша для значення $x_6=0$, а друга $x_6=1$. В табл. 5.3 і 5.4 містяться дані, відповідні цим гілкам.

Табл. **Ошибка! Текст указанного стиля в документе отсутствует..3.** Таблиця даних, що відповідає гілці $x_6=0$

Ознаки																
Об'єкт	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
2	1	0	1	1	0	0	1	1	0	1	1	1	0	0	1	0
7	1	1	0	1	0	0	0	0	1	1	0	0	1	1	1	1
12	1	0	1	0	1	0	1	0	1	0	1	1	0	1	1	0
13	1	1	0	1	1	0	1	1	1	0	0	0	1	0	0	1
14	0	1	1	1	0	0	1	0	1	0	1	0	0	1	1	1
16	0	1	1	1	0	0	1	1	0	0	1	0	1	0	1	1
Клас1/Клас2	2/2	1/3	1/3	2/3	2/2	2/4	1/4	1/2	1/3	2/0	1/3	1/1	1/2	1/2	2/3	1/3

Для гілки $x_6=0$ остаточне рішення дає ознаку x_{10} . Вона приймає значення 1 на об'єктах 2 і 7 з першого класу і значення 0 на об'єктах 12,13,14 і 16 з другого класу.

Табл. **Ошибка! Текст указанного стиля в документе отсутствует..4.** Таблиця даних, відповідна гілці $x_6=1$

Ознаки																
Об'єкт	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
1	0	1	0	0	1	1	0	0	1	1	1	0	1	1	0	1
3	0	0	0	1	1	1	0	1	1	0	1	1	1	0	0	1
4	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1
5	1	1	0	1	0	1	0	1	0	1	0	1	0	1	1	0
6	0	0	1	0	1	1	1	0	1	0	1	0	1	0	1	1
8	0	0	1	1	0	1	1	0	1	1	1	0	1	0	1	0
9	0	0	1	1	0	1	0	0	1	1	0	1	1	1	0	1
10	0	1	1	0	0	1	1	0	0	1	1	0	1	1	1	0
11	1	1	1	0	1	1	0	0	1	1	0	1	0	1	0	0
15	0	1	0	1	0	1	1	1	0	1	0	0	1	1	0	1
Клас1/Клас2	2/2	1/3	1/3	2/3	2/2	2/4	1/4	1/2	1/3	2/0	1/3	1/1	1/2	1/2	2/3	1/3

Гілка $x_6=1$ влаштована складніше. На цій гілці найбільшою дискримінуючою силою володіє ознака x_{11} (табл. **Ошибка! Текст указанного стиля в документе отсутствует..4.**). Вона має значення 0 у об'єкту 5 з першого класу і об'єктів 9, 11, 15 з другого класу; і значення 1 у об'єктів 1, 3, 4, 6 з першого класу і об'єкту 10 з другого класу. Таким чином, потрібне додаткове розгалуження, яке здійснюється за допомогою ознак x_{15} , x_{16} і x_2 .

Дерево логічного висновку, що вирросло з ознаки x_6 (носогубна складка), має 6 результатів. Тільки два з цих результатів включають по чотири об'єкти (повнота 4/8). Один результат групує три об'єкти свого класу (повнота 3/8), один результат — два об'єкти (повнота 2/8) і три результати включають по одному об'єкту (повнота 1/8). Природно, не можна вважати логічною закономірністю шлях по дереву, для якого результат охоплює таке мале відносне число об'єктів одного класу (володіє малою повнотою). Як бачимо, рішення, що дається алгоритмом CLS, далеко від того, що вимагається.

Якщо спробувати побудувати дерево рішень з будь-якої іншої ознаки, наприклад, $x_2, x_3, x_{10}, x_{11}, x_{14}$ або x_{15} , то результат також буде далекий від оптимального.

Алгоритм CLS здатний приводити до якісних рішень задачі пошуку логічних закономірностей тільки у разі незалежних ознак. В протилежному випадку він нерідко направляє хід логічного виведення по помилковому шляху і створює лише ілюзію правильного міркування.

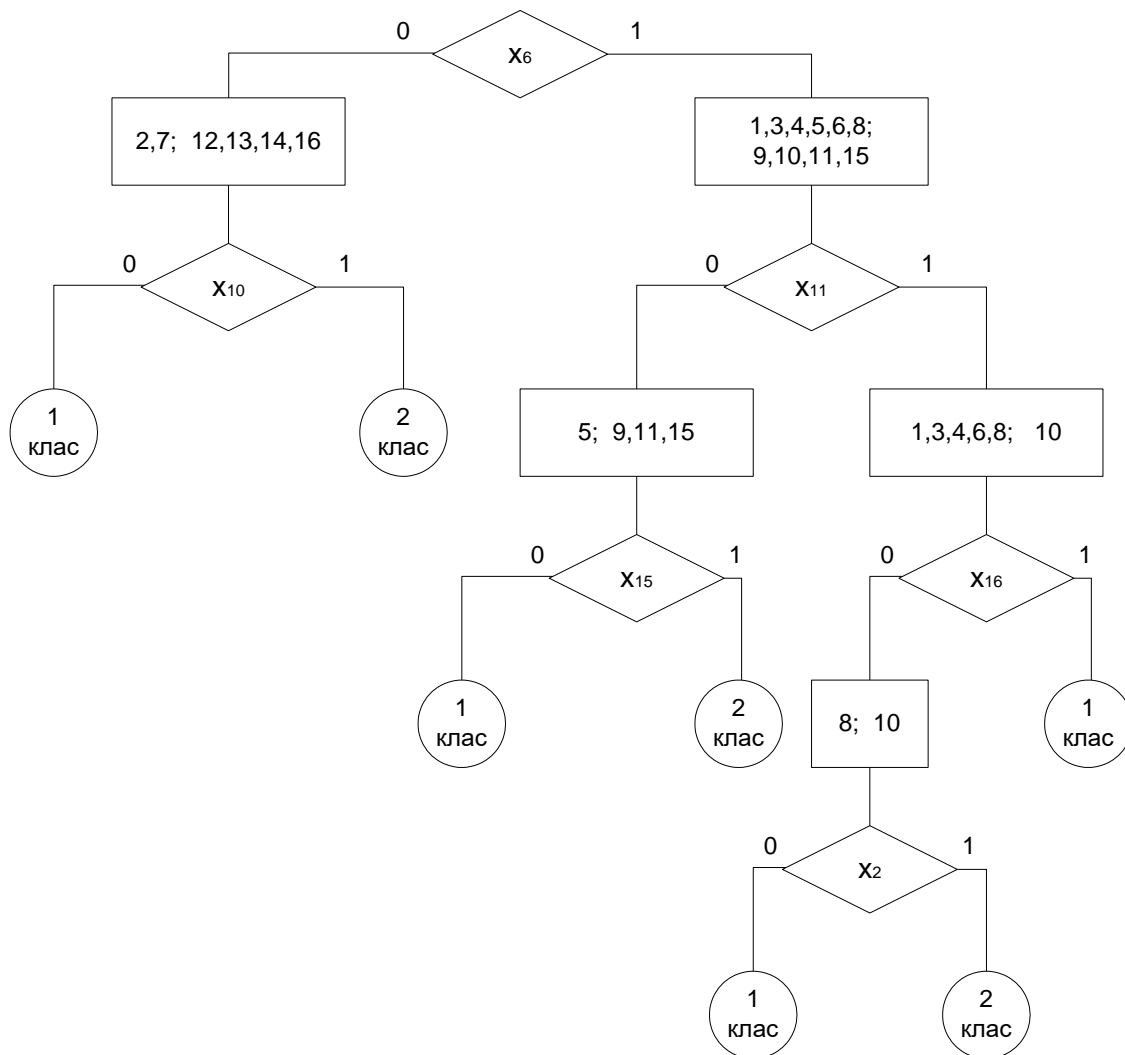


Рис. Ошибка! Текст указанного стиля в документе отсутствует..2.

Отримані правила:

Правило1. ЯКЩО ($x_6=0$) І ($x_{10}=0$) ТО клас 1

Правило2. ЯКЩО $(x_6=0)$ І $(x_{10}=1)$ ТО клас 2

Правило3. ЯКЩО $(x_6=1)$ І $(x_{11}=0)$ І $(x_{15}=0)$ ТО клас 1

Правило4. ЯКЩО $(x_6=1)$ І $(x_{11}=0)$ І $(x_{15}=1)$ ТО клас 2







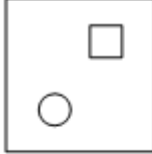


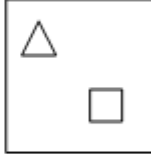


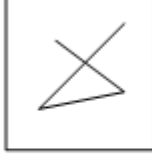





Правило5. ЯКЩО $(x_6=1)$ І $(x_{11}=1)$ І $(x_{16}=1)$ ТО клас 1

Правило6. ЯКЩО $(x_6=1)$ І $(x_{11}=1)$ І $(x_{16}=0)$ І $(x_2=0)$ ТО клас 1

Правило7. ЯКЩО $(x_6=1)$ І $(x_{11}=1)$ І $(x_{16}=0)$ І $(x_2=1)$ ТО клас 2

3.4. Індивідуальні завдання

Згідно номеру варіанту провести формування правил розподілу графічних об'єктів по групах

		
		
		
1.		
		
		
2.		
		
3.		

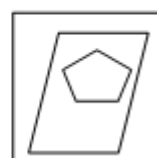
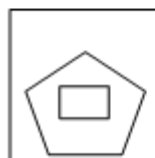
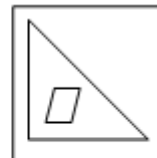
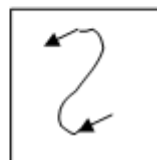
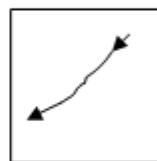
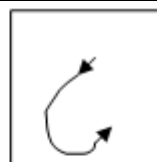
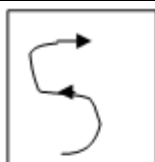
4.

5.

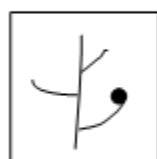
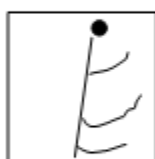
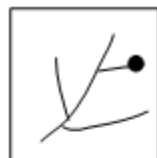
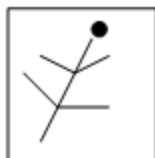
6.

--	--

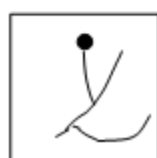
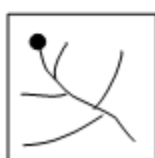
7.

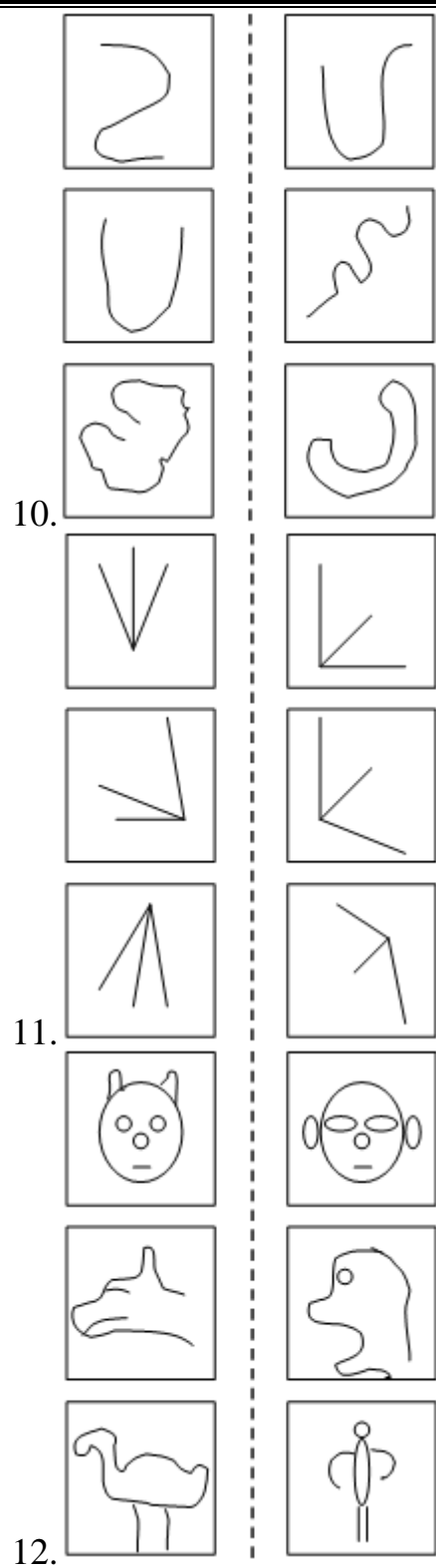


8.

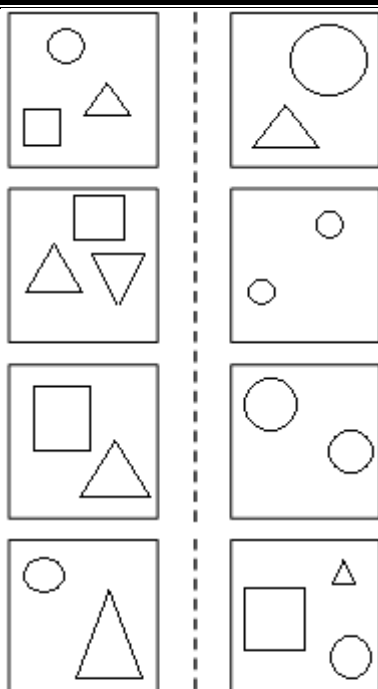


9.

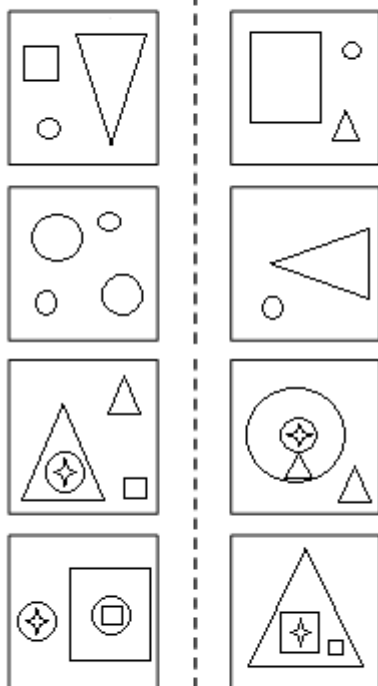




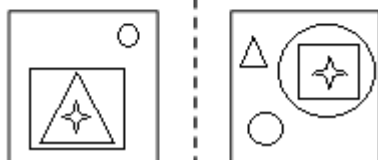
13.



14.



15.



ЛАБОРАТОРНА РОБОТА №4. ОБХІД ГРАФІВ ВШИР ТА ВГЛИБ

МЕТА РОБОТИ: засвоїти механізм перевантаження операторів та здобути практичні навички при роботі з ними.

4.1. Програма роботи

4.1.1 Отримати завдання.

4.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

4.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

4.1.4. Оформити електронний звіт про роботу та захистити її.

4.2. Вказівки до виконання роботи

4.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 4.5 і записує його до індивідуального бланку.

4.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 4.4.

4.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання. Після проведення розрахунків отримані результати необхідно проаналізувати і занести до звіту.

4.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних .h і .cpp файлів, а також пояснення до них;
- результати реалізації програми;
- діаграму класів та діаграму варіантів використання з поясненням;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

4.3. Теоретичні відомості

Роком виникнення теорії графів одностайно вважається рік 1736, коли Леонард Ейлер опублікував розв'язок так званої задачі про кенігсберзькі мости, а також знайшов загальний критерій існування ейлерового циклу в графі (див. розділ 12).

Отримання дальших суттєвих результатів у цій галузі датують серединою XIX століття. Однак початок проведення активних систематичних дослід-

джень та становлення теорії графів як окремішнього авторитетного розділу сучасної математики відбулося ще майже 100 років по тому, тобто в середині XX століття. Саме з цього часу граф стає однією з найпоширеніших і найпопулярніших математичних моделей у багатьох сферах науки і техніки. Картинка у вигляді набору точок на площині та ліній, проведених між деякими з них, стала зручною і наочною формою зображення найрізноманітніших об'єктів, процесів та явищ.

Графом (неорієнтованим графом) G називається пара множин (V, E) , де E – довільна підмножина множини $V(2)$ ($E \subseteq V(2)$); позначається $G = (V, E)$.

Елементи множини V називаються вершинами графа G , а елементи множини E – ребрами графа G . Відповідно V називається множиною вершин і E – множиною ребер графа G .

Традиційно ребра $\{v, w\}$ записуються за допомогою круглих дужок (v, w) (іноді просто vw).

Граф, який складається з однієї вершини, називається тривіальним.

Оскільки для тривіального графа або так званих порожніх графів $G = (V, \emptyset)$ переважну більшість властивостей та тверджень перевірити неважко, то надалі не будемо кожен раз при формулюванні та доведенні тих чи інших загальних тверджень теорії графів спеціально обумовлювати, що йдеться про нетривіальні графи (при цьому для тривіального або порожнього графів результат може бути дещо іншим).

Нехай задано граф $G = (V, E)$. Якщо $(v, w) \in E$, то кажуть, що вершини v і w є суміжними, у противному разі вершини v і w є несуміжними. Якщо $e = (v, w)$ – ребро графа, то вершини v і w називаються кінцями ребра e . У цьому випадку кажуть також, що ребро e з'єднує вершини v і w . Вершина v і ребро e називаються інцидентними, якщо v є кінцем e .

Два ребра називаються суміжними, якщо вони мають спільну вершину.

Існує декілька способів завдання графів.

Одним зі способів завдання графа $G = (V, E)$ є завдання кожної з множин V і E за допомогою переліку їх елементів.

Приклад – граф $G_1 = (V_1, E_1)$, $V_1 = \{v_1, v_2, v_3, v_4\}$ і $E_1 = \{(v_1, v_3), (v_1, v_4), (v_2, v_3), (v_2, v_4), (v_3, v_4)\}$ – це граф із чотирма вершинами і п'ятьма ребрами, граф $G_2 = (V_2, E_2)$, $V_2 = \{v_1, v_2, v_3, v_4, v_5\}$ і $E_2 = \{(v_1, v_2), (v_2, v_4), (v_1, v_5), (v_3, v_2), (v_3, v_5), (v_4, v_1), (v_5, v_4)\}$ – граф із п'ятьма вершинами і сімома ребрами.

Граф $G = (V, E)$ зручно зображати за допомогою рисунка на площині, який називають діаграмою графа G . Вершинам графа G ставляться у бієктивну відповідність точки площини; точки, що відповідають вершинам v і w , з'єднуються лінією (відрізком або кривою) тоді і тільки тоді, коли v і w суміжні вершини. Зрозуміло, що діаграма графа змінюватиме свій вигляд у залежності від вибору відповідних точок на площині.

На рисунку 4.1 зображені діаграми графів G_1 і G_2 з попереднього прикладу.



Рис 4.1

Графи можна задавати також за допомогою матриць.

Занумеруємо всі вершини графа G натуральними числами від 1 до n . Матрицею суміжності A графа G називається квадратна $n \times n$ -матриця, в якій елемент a_{ij} i -го рядка і j -го стовпчика дорівнює 1, якщо вершини v_i та v_j з номерами i та j суміжні, і дорівнює 0 у протилежному разі.

Для графів $G1$ і $G2$ маємо відповідно

$$A1 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad \text{і} \quad A2 = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Очевидно, що матриці суміжності графів – симетричні.

Занумеруємо всі вершини графа G числами від 1 до n і всі його ребра – числами від 1 до m . Матрицею інцидентності B графа G називається $n \times m$ -матриця, в якій елемент b_{ij} i -го рядка і j -го стовпчика дорівнює 1, якщо вершина v_i з номером i інцидентна ребру e_j з номером j , і дорівнює 0 у протилежному разі.

Для графів $G1$ і $G2$ маємо

$$B1 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \quad \text{і} \quad B2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Нарешті, ще одним способом завдання графів є списки суміжності. Кожній вершині графа відповідає свій список. У список, що відповідає вершині v , послідовно записуються всі суміжні їй вершини.

Для графів $G1$ і $G2$ маємо списки

G1:	G2:
v1: v3, v4	v1: v2, v4, v5
v2: v3, v4	v2: v1, v3, v4
v3: v1, v2, v4	v3: v2, v5
v4: v1, v2, v3	v4: v1, v2, v5
	v5: v1, v3, v4

Вибір та зручність того чи іншого зі способів завдання графів залежать від особливостей задачі, яка розв'язується.

Для різних практичних застосувань теорії графів важливою є проблема систематичного обходу (перебору) всіх вершин і/або ребер графа.

Двома класичними методами розв'язання цих проблем є: *метод пошуку вишир* та *метод пошуку вглиб*.

Для опису алгоритмів нам знадобляться три списки ребер ВІДКР, ЗАКР і РОЗВ. Крім того, для вершини $v \in V$ через $S(v)$ будемо позначати множину всіх вершин $w \in V$ таких, що в графі G існує ребро $(v,w) \in E$. Такі вершини w часто називають *синами* вершини v , а множину $S(v)$ – множиною синів вершини v .

Алгоритм пошуку вишир / вглиб.

1. Всі "порожні" ребра розмістити у списку ВІДКР (у довільному порядку).

2. Якщо ВІДКР = \emptyset , то РОЗВ = \emptyset (тобто сформульована задача не має розв'язку) і алгоритм завершує свою роботу.

3. Закрити перше ребро (v,w) з ВІДКР, тобто перенести ребро (v,w) зі списку ВІДКР у список ЗАКР.

4. Якщо вершина w закритого ребра є кінцевою вершиною ($w \in V_K$), то шуканий список РОЗВ (тобто шуканий шлях з V_O у V_K) міститься серед ребер списку ЗАКР і може бути виділений з нього послідовно, починаючи з останнього закритого ребра шляху. Зауважимо, що при побудові результуючого шляху для кожного з ребер (v,w) списку ЗАКР необхідно вибирати ребро-попередника (u,v) так, що воно є першим ребром з кінцем v у списку ЗАКР. Алгоритм завершує свою роботу.

У протилежному разі ($w \notin V_K$) перейти до пункту 5.

5. Визначити $S(w)$ – множину синів вершини w останнього закритого ребра, а також множину ребер $R(w) = \{(w,z) \mid z \in S(w)\}$.

Розмістити у списку ВІДКР усі ребра з множини $R(w) \setminus (ВІДКР \cup ЗАКР)$ після / перед усіх ребер, що вже містяться в цьому списку.

6. Перейти до пункту 2.

Відмінність між обома алгоритмами пошуку знаходиться в позиції 5 і полягає в тому, що в алгоритмі пошуку вишир необхідно розміщувати відповідні ребра **після**, а в алгоритмі пошуку вглиб – **перед** усіма ребрами, що знаходяться в списку ВІДКР.

Для наведених алгоритмів вживають скорочені назви АПШ і АПГ (відповідні англійські назви – BFS (breadth first search) і DFS (depth first search)).

Таким чином, для АПШ список ВІДКР є *чергою*, тобто такою сукупністю елементів, в якій нові елементи розміщуються в кінці сукупності, а елемент, що "обслуговується" (закривається), вибирається з голови (початку) цієї сукупності. У той час для АПГ список ВІДКР є так званим *стеком*, тобто сукупністю, в якій елементи, що додаються до сукупності, і елементи, що відбираються для "обслуговування", розміщуються тільки на початку сукупності - у верхівці стеку (за принципом: "останній прийшов - перший обслуговується").

Приклад 3.8. Розглянемо дію алгоритму пошуку вишир для графа, зображеного на рис.4.2 (див. таблицю 4.1). Вважаємо, що $V_O = \{3\}$ і $V_K = \{11\}$.

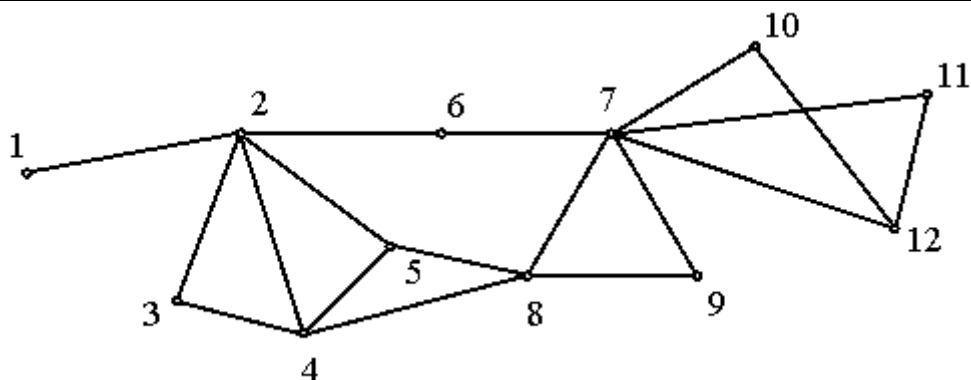


Рис.4.2

Кожний рядок таблиці описує результат виконання одного циклічного кроку (позиції 2–6) алгоритму. Оскільки список ЗАКР тільки поповнюється і на кожному кроці поповнюється тільки одним ребром, то в таблиці записуємо лише це ребро (не повторюючи всі елементи, які ввійшли до складу ЗАКР на попередніх кроках). Нагадаємо також, що ребра (v,w) і (w,v) збігаються.

Таблиця 3.1.

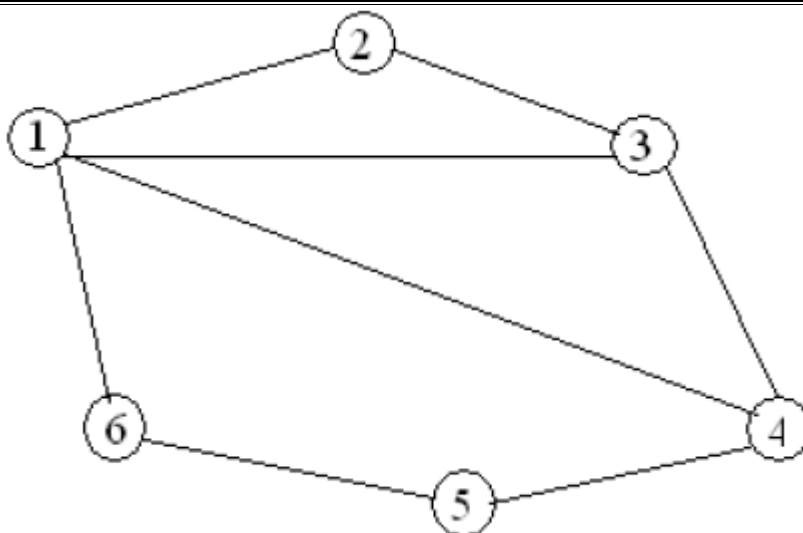
Алгоритм пошуку вшир (АПШ)

Крок	ВІДКР	ЗАКР	w	$R(w)$
0	$(p,3)$			
1	$(3,2),(3,4)$	$(p,3)$	3	$(3,2),(3,4)$
2	$(3,4),(2,1),(2,4),(2,5),(2,6)$	$(3,2)$	2	$(2,1),(2,4),(2,5),(2,6)$
3	$(2,1),(2,4),(2,5),(2,6),(4,5),(4,8)$	$(3,4)$	4	$(4,5),(4,8)$
4	$(2,4),(2,5),(2,6),(4,5),(4,8)$	$(2,1)$	1	
5	$(2,5),(2,6),(4,5),(4,8)$	$(2,4)$	4	
6	$(2,6),(4,5),(4,8),(5,8)$	$(2,5)$	5	$(5,8)$
7	$(4,5),(4,8),(5,8),(6,7)$	$(2,6)$	6	$(6,7)$
8	$(4,8),(5,8),(6,7)$	$(4,5)$	5	
9	$(5,8),(6,7),(8,7),(8,9)$	$(4,8)$	8	$(8,7),(8,9)$
10	$(6,7),(8,7),(8,9)$	$(5,8)$	8	
11	$(8,7),(8,9)$	$(6,7)$	7	$(7,12),(7,11),(7,9),(7,10)$
12	$(8,9),(7,12),(7,11),(7,9),(7,10)$	$(8,7)$	7	
13	$(7,12),(7,11),(7,9),(7,10)$	$(8,9)$	9	
14	$(7,11),(7,9),(7,10),(12,10),(12,11)$	$(7,12)$	12	$(12,10),(12,11)$
15	$(7,9),(7,10),(12,10),(12,11)$	$(7,11)$	11	

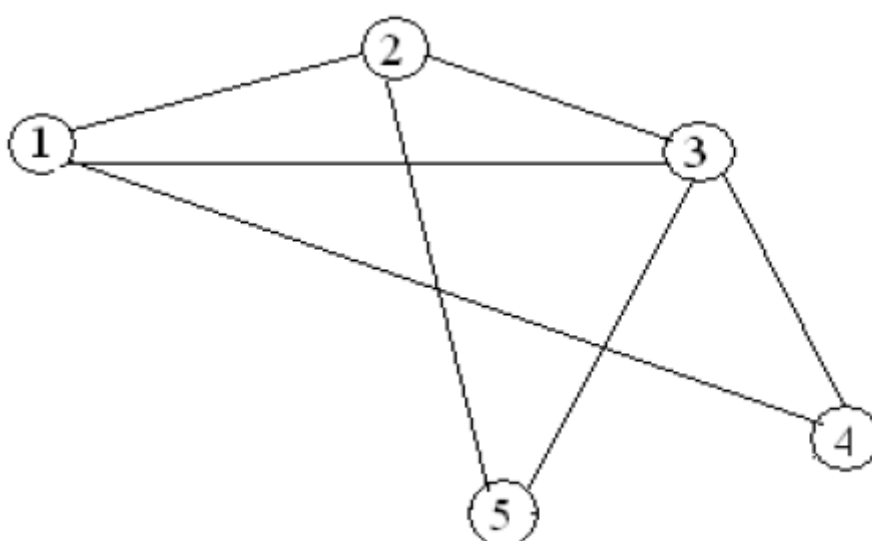
Алгоритм завершує свою роботу на п'ятнадцятому кроці. Аналізуючи список ребер ЗАКР від його кінця до початку, будуємо список РОЗВ, тобто знаходимо шуканий шлях від вершини 3 у вершину 11: $3,(3,2),2,(2,6),6,(6,7),7,(7,11),11$. Довжина цього шляху – 4.

4.4. Індивідуальні завдання

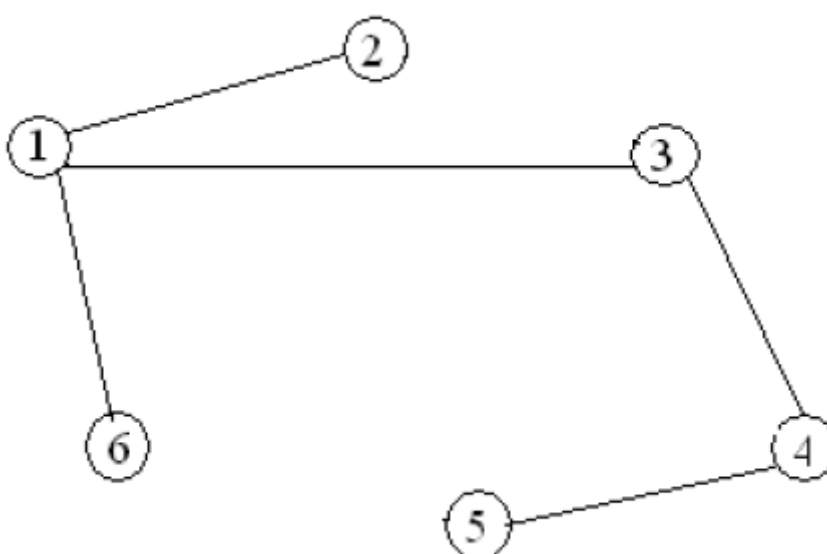
Розробити програму для обходу графу згідно варіанту вшир та вглиб. Метод задання графа обрати самостійно та обґрунтувати його.



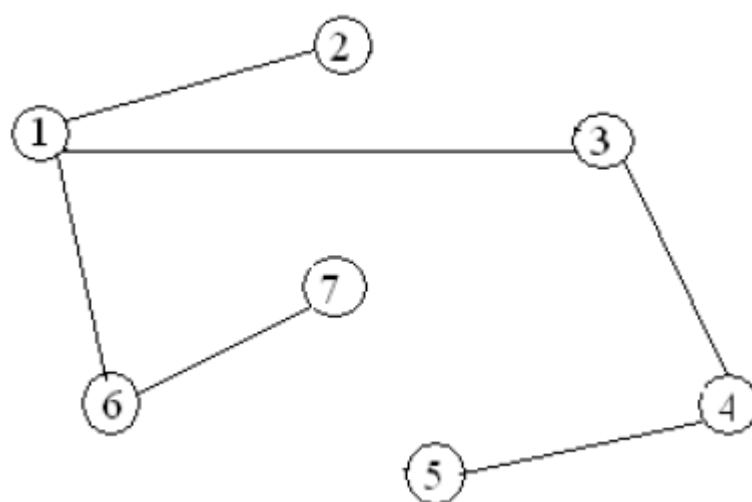
2.



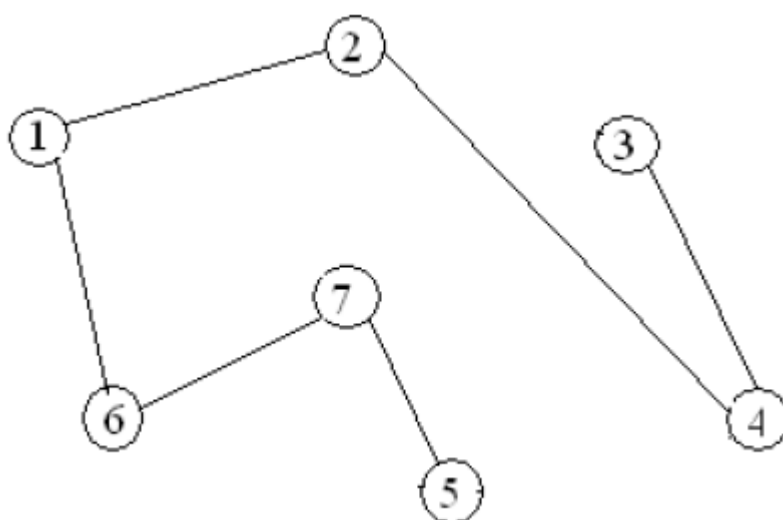
3.



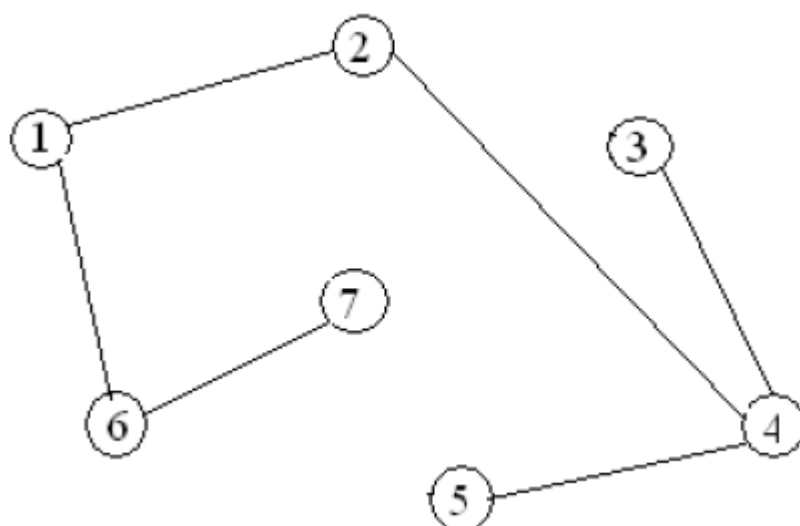
4.



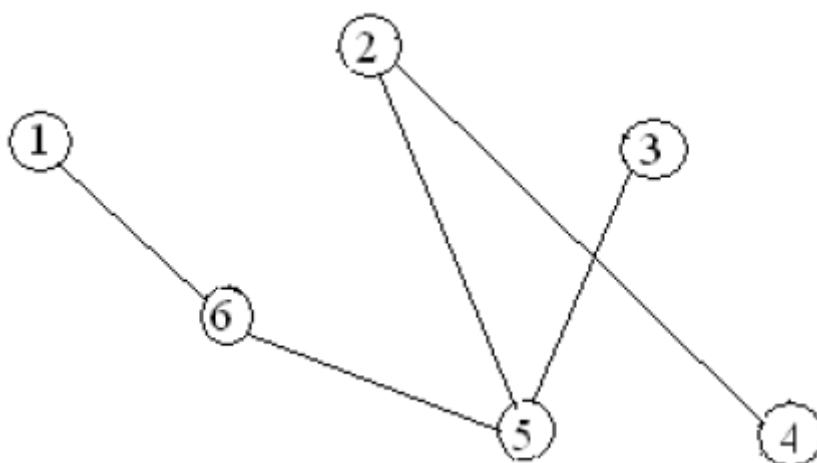
5.



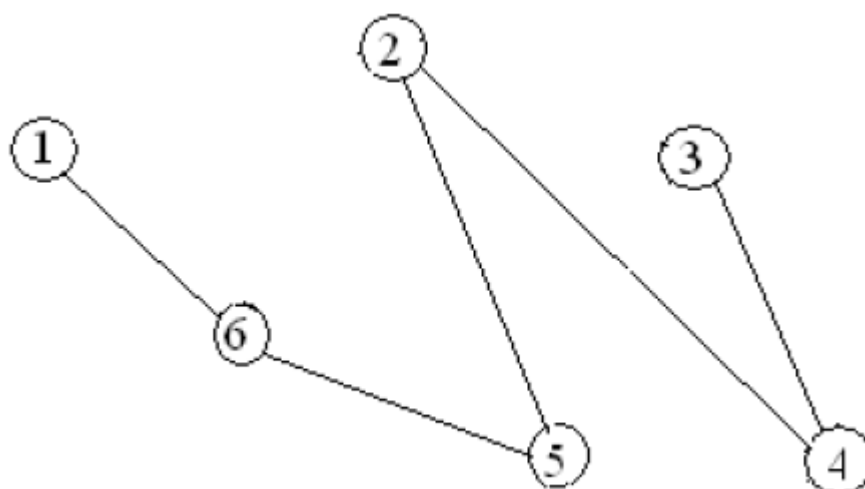
6.



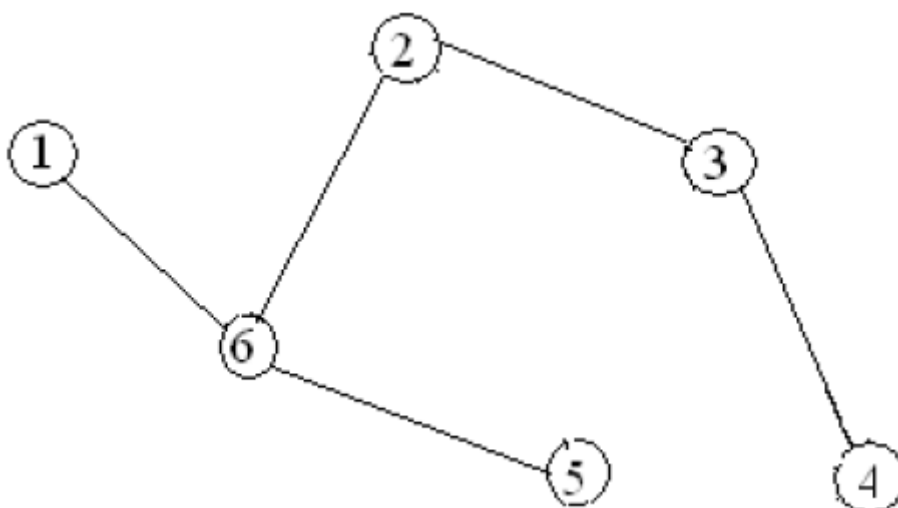
7.



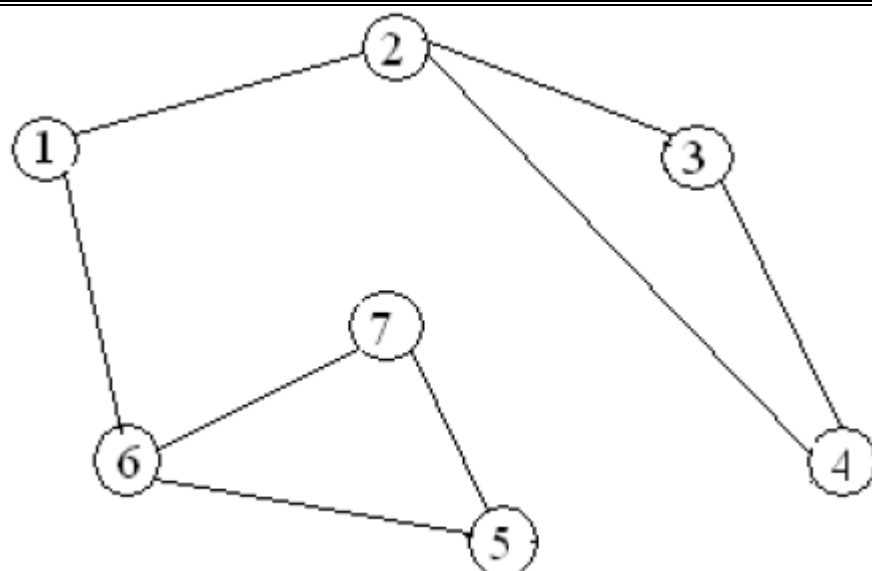
8.



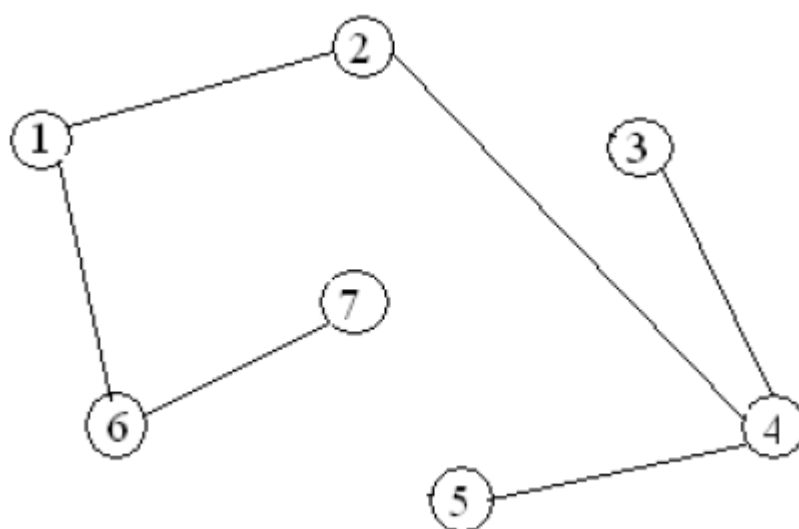
9.



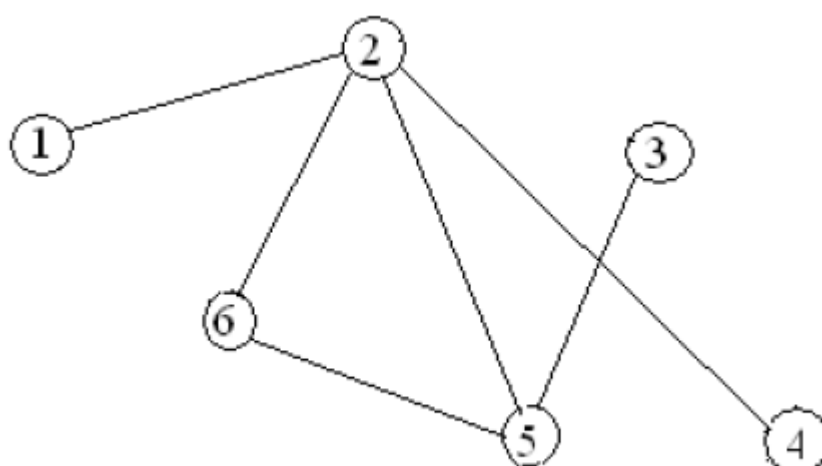
10.



11.



12.



ЛАБОРАТОРНА РОБОТА №5. ЖАДІБНИЙ АЛГОРИТМ

МЕТА РОБОТИ: засвоїти принцип роботи жадібного алгоритму та з його допомогою розв'язати задачу комівояжера.

5.1. Програма роботи

5.1.1 Отримати завдання.

5.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

5.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

5.1.4. Оформити електронний звіт про роботу та захистити її.

5.2. Вказівки до виконання роботи

5.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання.

5.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано

5.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання. Після проведення розрахунків отримані результати необхідно проаналізувати і занести до звіту.

5.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних .h і .cpp файлів, а також пояснення до них;
- результати реалізації програми;
- діаграму класів та діаграму варіантів використання з поясненням;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

5.3. Теоретичні відомості

Для багатьох оптимізаційних задач є простіші і швидкі алгоритми, ніж динамічне програмування. До таких алгоритмів відносяться жадібні алгоритми. Жадібний алгоритм на кожному кроці робить локально оптимальний вибір в розрахунку на те, що підсумкове рішення також буде оптимальним. Це не завжди виправдано, але для багатьох завдань жадібні алгоритми дійсно дають оптимум.

Принцип жодного вибору

Кажуть, що до оптимізаційної задачі застосуємо принцип жодного вибору (greedy choice property), якщо послідовність локально оптимальних (жодних) виборів дає глобально оптимальне рішення. Різниця між жадібними алгоритмами і динамічним програмуванням можна пояснити так: на кожному кроці жодній алгоритм бере " самий жирний шматок ", а потім вже намагається зробити найкращий вибір серед варіантів, що залишилися, які б вони не були. Алгоритм динамічного програмування приймає рішення, прорахувавши наперед наслідок всіх варіантів.

Оптимальність для підзадач

Вирішувані за допомогою жадібних алгоритмів завдання мають властивість оптимальності для підзадач : оптимальне рішення всієї задачі містить оптимальне рішення підзадач.

І жадібні алгоритми, і динамічне програмування ґрунтуються на властивості оптимальності для підзадач, тому може виникнути бажання застосувати жодній алгоритм замість динамічного, і навпаки. В одному випадку це може не дати оптимального рішення, в другому може привести до менш ефективного вирішення.

Прикладом може служити завдання про рюкзак 3 : вона полягає в тому, щоб укласти в рюкзак речі таким чином, щоб їх сумарна вага не перевищувала граничного значення W і сумарна вартість речей була максимальна. Завдання має два різновиди - безперервна завдання і дискретна. Наприклад, речі неподільні (золотий злиток) і подільні (золотий пісок).

Нехай є n речей, кожна з яких має вартість v_i і вага w_i .

Стратегія жодного алгоритму полягає в наступному: розраховується питома ціна речі $c_i = v_i / w_i$. Речі сортуються в порядку зменшення питомої ціни. Першою вибирається річ з максимальною питомою ціною, потім з безлічі елементів, що залишились, знову вибирається річ з максимальною ціною і т.д. При цьому умовою вибору є те, що кожна нова річ, що додається, річ не приводить до перевищення граничної ваги W .

У разі безперервної задачі про рюкзак виходить загальне оптимальне рішення, у разі дискретної - ні.

Існують завдання, для яких жоден з відомих жодних алгоритмів не дозволяє отримати оптимального рішення. Проте є жадібні алгоритми, які з великою ймовірністю дозволяють отримати "гарне" рішення, тобто майже оптимальне рішення, що характеризується вартістю, яке лише не кілька відсотків перевищує оптимальну. У таких випадках " жадібний " алгоритм виявляється найшвидшим способом отримати "гарне" рішення.

Розглянемо задачу " комівояжера ".

Для отримання оптимального рішення підходить лише алгоритм вичерпного пошуку, час виконання якого експоненціально.

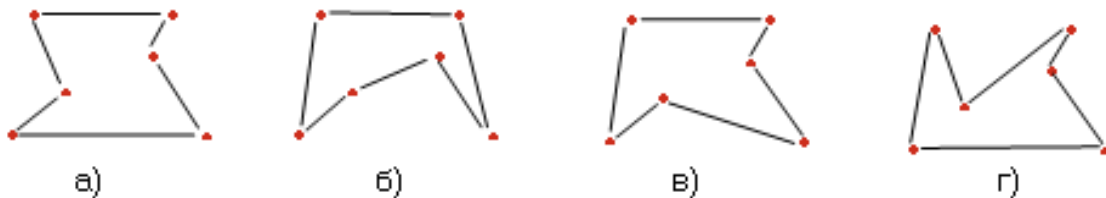
Задачу комівояжера можна вирішити, використовуючи евристичні рішення. Евристика - це засновані на здоровому глузді правила, які з досвіду, а не з чітко доведених математичних положень

Завдання комівояжера зводиться до пошуку в неорієнтованому графі з ваговими значеннями ребер маршруту, у якого сума ваг складових ребер буде мінімальною. Такий маршрут називається гамільтоновим циклом.

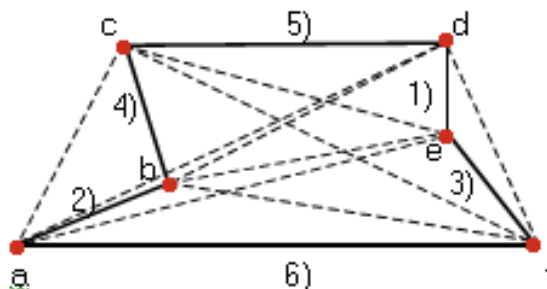
Нехай дано граф з шістьма вершинами (" містами "). Для кожної вершини задані координати, вагою ребра вважається його довжина. Припускаємо, що існують всі ребра, тобто граф є повним. У більш загальному випадку, коли вага ребер щире за евклідовим відстанню, вага ребра може дорівнювати нескінченності.

$$\begin{array}{ll} c \cdot (1, 7) & d \cdot (15, 7) \\ e \cdot (15, 4) & \\ b \cdot (4, 3) & \\ a \cdot (0, 0) & f \cdot (18, 0) \end{array}$$

Можна прокласти різні маршрути:



Жадібний алгоритм задачі комівояжера вибирає на кожному кроці ребро з мінімальною вагою, таке щоб воно не утворило циклу з раніше вибраними ребрами, і не породжувало вершини зі ступенем 3 і більше, що неприйнятно для маршруту. Цикл допустимо, тільки якщо кількість ребер відповідає кількості вершин, тобто ми приходимо до вихідної вершини маршруту.



- 1) Додається ребро (d, e) (довжина = 3).
- 2) Додаються ребра (a, b), (b, c), (e, f) (довжина = 5).
- 3) Відхиляється ребро (a, c), оскільки утворює цикл.
- 4) Відхиляється ребро (d, f), оскільки утворює цикл.
- 5) Відхиляється ребро (b, e), оскільки утворює ступінь 3 у вершин b і e.
- 6) Відхиляється ребро (b, d), оскільки утворює ступінь 3 у вершини b.
- 7) Додається ребро (c, d).
- 8) Додається ребро (a, f).

На малюнку спочатку вибирається ребро (d, e) з мінімальною вагою 3. Потім розглядаються ребра (b, c), (a, b) і (e, f) з вагою 5. Всі ці ребра задовольняють умовам вибору і можна вибрати будь-яке з них в будь-якому порядку (

наприме, (a, b) , (e, f) , (b, c)). Всі ці ребра включаються до рішення. Наступним найкоротшим шляхом є (a, c) з довжиною $(7, 08)$, але це ребро утворює цикл з ребрами (a, b) і (a, c) , і тому відкидається. Потім з тієї ж причини відкидається ребро (d, f) . Далі ребро (b, e) відкидається, бо інакше вершини b і e будуть мати ступінь 3. З цієї ж причини буде відкинута наступне ребро (b, d) . Потім розглянемо ребро (c, d) і додамо його до Гамільтона циклу. Кількість ребер відповідає кількості вершин - 1 і побудований маршрут: $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow f$. Для завершення додаємо ребро (a, f) для замикання циклу.

За оптимальності маршрут на четвертому місці серед можливих маршрутів. Його вартість на 4 % більше вартості оптимального маршруту.

5.4. Індивідуальні завдання

Згідно жадібного алгоритму провести розв'язок задачі комівояжера. Населені пункти, що необхідно відвідати комівояжеру обрати згідно номера варіанту, а відстані між ними – із таблиці.

1. Вінниця, Дніпропетровськ, Житомир, Запоріжжя, Кіровоград, Луцьк, Львів, Миколаїв, Одеса, Полтава, Сімферополь, Суми, Тернопіль, Ужгород, Хмельницький, Черкаси, Чернівці;

2. Вінниця, Дніпропетровськ, Донецьк, Житомир, Запоріжжя, Івано-Франківськ, Київ, Кіровоград, Луганськ, Львів, Миколаїв, Одеса, Полтава, Рівне, Сімферополь, Суми, Черкаси, Чернівці, Чернігів.

3. Запоріжжя, Івано-Франківськ, Луганськ, Миколаїв, Рівне, Сімферополь, Суми, Тернопіль, Ужгород, Харків, Херсон, Хмельницький, Черкаси, Чернівці, Чернігів.

4. Вінниця, Житомир, Запоріжжя, Івано-Франківськ, Київ, Кіровоград, Луганськ, Львів, Миколаїв, Одеса, Полтава, Рівне, Сімферополь, Суми, Тернопіль, Ужгород, Харків, Черкаси, Чернівці, Чернігів.

5. Вінниця, Дніпропетровськ, Донецьк, Житомир, Івано-Франківськ, Кіровоград, Луганськ, Луцьк, Львів, Миколаїв, Полтава, Рівне, Сімферополь, Тернопіль, Ужгород, Харків, Херсон, Хмельницький, Черкаси, Чернівці.

6. Дніпропетровськ, Житомир, Київ, Кіровоград, Миколаїв, Одеса, Полтава, Рівне, Сімферополь, Тернопіль, Харків, Херсон, Черкаси, Чернівці.

7. Вінниця, Дніпропетровськ, Донецьк, Житомир, Запоріжжя, Івано-Франківськ, Кіровоград, Луганськ, Луцьк, Миколаїв, Полтава, Сімферополь, Тернопіль, Ужгород, Харків, Херсон, Хмельницький, Черкаси, Чернівці, Чернігів.

8. Вінниця, Донецьк, Житомир, Запоріжжя, Івано-Франківськ, Київ, Кіровоград, Луганськ, Львів, Миколаїв, Одеса, Полтава, Рівне, Сімферополь, Тернопіль, Харків, Херсон, Хмельницький, Черкаси, Чернівці;

9. Вінниця, Дніпропетровськ, Донецьк, Житомир, Запоріжжя, Івано-Франківськ, Київ, Луганськ, Луцьк, Львів, Миколаїв, Одеса, Полтава, Рівне, Сімферополь, Тернопіль, Ужгород, Херсон, Черкаси, Чернівці.

10. Дніпропетровськ, Донецьк, Житомир, Запоріжжя, Ів-Франківськ, Київ, Кіровоград, Луганськ, Луцьк, Миколаїв, Полтава, Рівне, Тернопіль, Ужгород, Харків, Херсон, Хмельницький, Чернівці, Чернігів.

11. Вінниця, Дніпропетровськ, Донецьк, Запоріжжя, Ів-Франківськ, Київ, Луцьк, Львів, Миколаїв, Полтава, Рівне, Суми, Тернопіль, Ужгород, Харків, Херсон, Хмельницький, Чернівці, Чернігів.

12. Вінниця, Донецьк, Житомир, Запоріжжя, Київ, Кіровоград, Луганськ, Луцьк, Львів, Одеса, Полтава, Рівне, Сімферополь, Суми, Тернопіль, Ужгород, Хмельницький, Черкаси, Чернівці, Чернігів.

Відстані між обласними центрами України																										
	Місто	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	Вінниця	-	645	868	125	748	366	256	316	1057	382	360	471	428	593	311	844	602	232	575	734	521	120	343	312	396
2	Дніпропетровськ	645	-	252	664	81	901	533	294	394	805	975	343	468	196	957	446	430	877	1130	213	376	765	324	891	672
3	Донецьк	868	252	-	858	217	1171	727	520	148	1111	1221	611	731	390	1045	591	706	1100	1391	335	560	988	547	1141	867
4	Житомир	125	664	858	-	738	431	131	407	1182	257	423	677	557	468	187	803	477	298	671	690	624	185	321	389	271
5	Запоріжжя	748	81	217	738	-	1119	607	303	365	681	833	377	497	270	925	365	477	977	1488	287	297	875	405	957	747
6	Ів-Франківськ	366	901	1171	431	1119	-	561	618	1402	328	135	747	627	898	296	1070	908	134	280	1040	798	246	709	143	701
7	Київ	256	533	727	131	607	561	-	298	811	388	550	490	489	337	318	972	346	427	806	478	551	315	190	538	149
8	Кіровоград	316	294	520	407	303	618	298	-	668	664	710	174	294	246	627	570	506	547	883	387	225	435	126	637	363
9	Луганськ	1057	394	148	1182	365	1402	811	668	-	1199	1379	857	977	474	1129	739	253	1289	1539	333	806	1177	706	1292	951
10	Луцьк	382	805	1111	257	681	328	388	664	1199	-	152	780	856	725	70	1052	734	159	413	866	869	263	578	336	949
11	Львів	360	975	1221	423	833	135	550	710	1379	152	-	850	970	891	232	1173	896	128	261	1028	1141	240	740	278	690
12	Миколаїв	471	343	611	677	377	747	490	174	857	780	850	-	120	420	864	282	681	754	999	556	51	590	300	642	640
13	Одеса	428	468	731	557	497	627	489	294	977	856	970	120	-	540	741	392	800	660	1009	831	171	548	420	515	529
14	Полтава	593	196	390	468	270	898	337	246	474	725	891	420	540	-	665	635	261	825	1149	141	471	653	279	892	477
15	Рівне	311	957	1045	187	925	296	318	627	1129	70	232	864	741	665	-	1157	664	162	484	805	834	193	508	331	458
16	Сімферополь	844	446	591	803	365	1070	972	570	739	1052	1173	282	392	635	1157	-	896	1097	1363	652	221	964	696	981	1112
17	Суми	602	430	706	477	477	908	346	506	253	734	896	681	800	261	664	896	-	774	1138	190	732	662	540	883	350
18	Тернопіль	232	877	1100	298	977	134	427	547	1289	159	128	754	660	825	162	1097	774	-	338	987	831	112	575	176	568
19	Ужгород	575	1130	1391	671	1488	280	806	883	1539	413	261	999	1009	1149	484	1363	1138	338	-	1299	1065	455	984	444	951
20	Харків	734	213	335	690	287	1040	478	387	333	866	1028	556	831	141	805	652	190	987	1299	-	576	854	420	1036	608
21	Херсон	521	376	560	624	297	798	551	225	806	869	1141	51	171	471	834	221	732	831	1065	576	-	641	351	713	691
22	Хмельницький	120	765	988	185	875	246	315	435	1177	263	240	590	548	653	193	964	662	112	455	854	641	-	463	190	455
23	Черкаси	343	324	547	321	405	709	190	126	706	578	740	300	420	279	508	696	540	575	984	420	351	463	-	660	330
24	Чернівці	312	891	1141	389	957	143	538	637	1292	336	278	642	515	892	331	981	883	176	444	1036	713	190	660	-	695
25	Чернігів	396	672	867	271	747	701	149	363	951	949	690	640	529	477	458	1112	350	568	951	608	691	455	330	695	-

ЛАБОРАТОРНА РОБОТА №6. АЛГОРИТМ А*

МЕТА РОБОТИ: ознайомитись із алгоритмом А* та використати його для пошуку оптимального шляху.

6.1. Програма роботи

6.1.1. Отримати завдання. Ознайомитись з правилами оголошення та використання віртуальних функцій.

6.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

6.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

6.1.4. Оформити електронний звіт про роботу та захистити її.

6.2. Вказівки до виконання роботи

6.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 6.5 і записує його до звіту.

6.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 6.4.

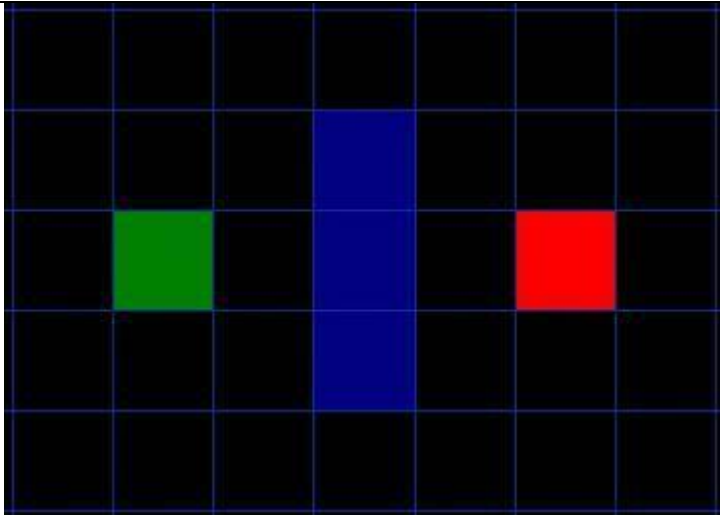
6.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

6.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних .h і .cpp файлів, а також пояснення до них;
- результати реалізації програми, які виведені в вікно форми;
- діаграму класів та варіантів використання з поясненням;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

6.3. Теоретичні відомості

Давайте уявимо собі, що у нас є хтось, хто хоче потрапити з точки А в точку В. Ці дві точки розділені стіною. На ілюстрації нижче зелений квадрат це стартова точка А, червоний квадрат - цільова точка В, а кілька синіх квадратів - стіна між ними.



Перше, на що ви повинні звернути увагу, те, що ми розділили нашу область пошуку на квадратну сітку. Спрощення області пошуку – це перший крок у пошуку шляху. Цей метод спрощує нашу область пошуку до простого двовимірного масиву. Кожен елемент масиву представляє одну з клітин сітки, а його значенням буде прохідність цієї клітини (прохідна і непрохідна). Для знаходження шляху нам необхідно визначити які нам потрібні клітини для переміщення з точки А в точку В. Як тільки шлях буде знайдений, наш подорожній почне рухатися з центру однієї клітини на центр наступній доти, поки не досягне цільової клітини.

Початок Пошуку

Як тільки ми спростили нашу область пошуку до деякого числа вершин, нам потрібно почати пошук для знаходження найкоротшого шляху. Почнемо з точки А, перевіряючи сусідні клітини і рухаючись далі до тих пір, поки не знайдемо цільову точку.

Починаємо пошук шляху виконуючи наступне:

1. Починаємо зі стартовою точки А і додаємо її в " відкритий список " клітин, які потрібно обробити. Відкритий список це щось на зразок списку покупок. У даний момент є тільки один елемент у списку, але пізніше ми додамо ще. Список містить клітини, які може бути знаходяться вздовж шляху, який ви виберете, а може і ні. Простіше кажучи, це список клітин, які потрібно перевірити.

2. Шукаємо доступні або прохідні клітини, які межують із стартовою точкою, ігноруючи клітини зі стінами, водою або іншою непрохідною областю. І також додаємо їх у відкритий список. Для кожної з цих клітин зберігаємо точку А, як " батьківську клітину ". Ця батьківська клітина важлива, коли ми будемо простежувати наш шлях. Це буде описано набагато пізніше.

3. Видаляємо стартову точку А з вашого відкритого списку і додаємо її в "закритий список" клітин, які вам більше не потрібно перевіряти.

Тепер у вас має бути щось схоже на наступну ілюстрацію. На цій ілюстрації темно-зелений квадрат в центрі - ваша стартова точка. Вона виділена блакитним кольором для відображення того, що вона знаходиться в закритому списку. Всі сусідні клітини в даний момент знаходяться у відкритому списку. Вони виділені світло-зеленим. Кожна має сірий покажчик, спрямований на батьківську клітину, яка в нашому випадку є стартовою точкою.

Далі ми виберемо одну з сусідніх клітин у відкритому списку і практично повторимо вищеописаний процес. Але яку клітину ми виберемо? Ту, у якій менше вартість F .

оцінка шляху

Способом визначення того, яку ж клітку використовувати, є наступні вирази:

$$F = G + H$$

де

G = вартість пересування з стартовою точки A до даній клітині, слідуючи знайденому шляху до цієї клітки.

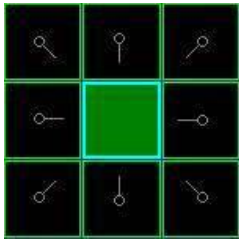
H = приблизна вартість пересування від даної клітини до цільової, тобто точки B . Вона зазвичай є евристичної функцією. Причина по якій вона так називається в тому, що це припущення. Ми дійсно не дізнаємося довжину шляху, поки не знайдемо сам шлях, тому що в процесі пошуку нам може зустрітися безліч речей (наприклад, стіни і вода). У цій статті вам запропонували один спосіб обчислити H , але існує безліч способів, які можна знайти в інших статтях.

Наш шлях генерується шляхом повторного проходу через відкритий список і вибору клітини з найменшою вартістю F . Цей процес буде описаний в статті більш детально, але трохи пізніше. Насамперед давайте уважно розглянемо обчислення вартості F .

Як описано вище, G є вартістю пересування зі стартовою клітини до поточної, використовуючи знайдений до неї шлях. У цьому прикладі ми признаємо вартість 10 до горизонтальних і вертикальних пересуванням, а до діагональних - 14. Ми використовуємо ці числа тому, що пройдене по діагоналі відстань приблизно в 1,414 раз (корінь з 2) більше вартості пересування по горизонталі або вертикалі. Для простоти ми використовуємо 10 і 14. Співвідношення дотримується і ми уникаємо обчислення квадратних коренів і десяткового дробу. Це не просто тому, що ми дурні і не любимо математику. Використання цілих чисел зразок цих, набагато швидше для комп'ютера. Як ви скоро дізнаєтеся, пошук шляху може бути дуже повільним якщо ви не використовуєте спрощення зразок цих.

Так як ми обчислюємо вартість G уздовж шляху до поточної точці, спо-

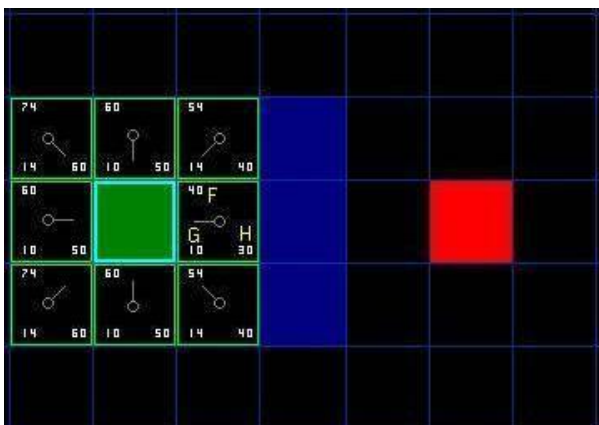
сіб її встановити полягає в тому, щоб взяти G батьківської клітини і додати 10 або 14, залежно від діагонального або ортогонального (Не діагонального) розташування поточної клітини щодо батьківської клітини. Необхідність використання цього методу стане очевидною трохи пізніше, коли ми віддалимося від стартової точки більш ніж на одну клітку.



Вартість H може бути обчислена безліччю способів. Метод, який ми використовуємо, називається методом Манхеттена (Manhattan method), де ви вважаєте загальна кількість клітин, необхідних для досягнення цільової точки від поточної, по горизонталі і вертикалі, ігноруючи діагональні переміщення і будь-які перешкоди, які можуть опинитися на шляху. Потім ми множимо загальну кількість отриманих клітин на 10.

Читаючи цей опис ви, повинно бути, вирішили, що евристика - просто приблизне визначення відстані, що залишилася між поточною кліткою і метою по прямій. Але це не так. Ми намагаємося встановити відстань, що залишилася вздовж шляху (який зазвичай йде не по прямій), але алгоритм вимагає від нас не переоцінити це відстань, інакше він може знайти не вірний шлях. Використаний тут метод гарантує надати нам правильний шлях. Хочете дізнатися про евристику більше? Ви знайдете вираження і додаткові відомості тут.

Вартість F обчислюється шляхом додавання вартостей G і H. Результати першого кроку нашого пошуку шляху проілюстровані нижче. Значення F, G і H записані в кожній клітині. Як бачимо по клітці справа від стартової точки, F виводиться у верхньому лівому кутку, G виводиться в нижньому лівому кутку, а H виводиться в нижньому правому куті.



Давайте подивимося на деякі з цих клітин. У клітці з буквами $G = 10$. Це тому, що вона знаходиться на відстані в одну клітку від стартової точки, при тому по горизонталі. Також $G = 10$ у клітин прямо зверху, знизу і зліва від стартової точки. У діагональних клітин $G = 14$.

Вартість H порахована за допомогою обчислення Манхеттенського відстані до червоної цільової клітці, рухаючись тільки по горизонталі і вертикалі, ігноруючи всі стіни на шляху. Біля клітки, прямо праворуч від стартової, відстань до цілі 3 клітини. Використовуючи цей метод бачимо, що $H = 30$. У клітині прямо над нею, відстань вже 4 клітини (пам'ятаєте, що треба рухатися тільки по горизонталі і вертикалі). І її значення вартості H дорівнюватиме 40. Ви, ймовірно, можете дога, як обчислюються вартості H для інших клітин.

Вартість F для кожної клітини обчислюється простим підсумовуванням G і H .

Продовжуємо пошук

Для продовження пошуку ми просто вибираємо клітку з найменшою вартістю F з усіх клітин, що знаходяться у відкритому списку. Потім з обраної кліткою ми виробляємо такі дії :

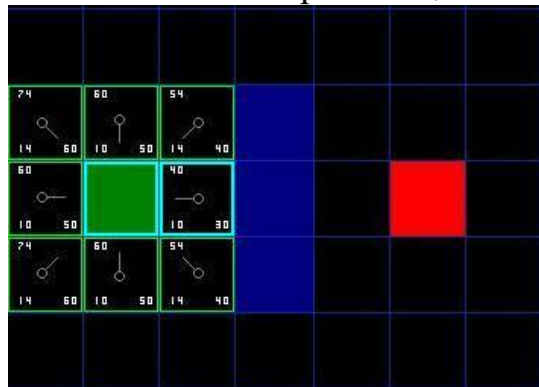
4) Видаляємо її з відкритого списку і додаємо в закритий список.

5) Перевіряємо всі сусідні клітини. Ігноруємо ті, які знаходяться в закритому списку або непрохідні (поверхня зі стінами, водою), решта додаємо у відкритий список, якщо вони там ще не знаходяться. Робимо обрану клітку "батьківською" для всіх цих клітин.

6) Якщо сусідня клітка вже знаходиться у відкритому списку, перевіряємо, а не коротше чи шлях по цій клітці ? Іншими словами, порівнюємо значення вартості G цих двох клітин. Якщо при використанні цієї клітини вартість G вище, ніж при використанні поточної клітини, то не робимо нічого.

Якщо ж вона нижча, то меням "батька" цієї клітини на обрану клітку. Потім обчислюємо вартості F і G цієї клітини. Якщо це виглядає для вас трохи заплутаним, далі ви можете побачити це на ілюстрації.

Добре, давайте подивимося, як це працює. З наших початкових 9 клітин, залишилося 8 у відкритому списку, а стартова клітка була внесена в закритий список. Клітка з найменшою вартістю F знаходиться прямо праворуч від стартової клітини, і її вартість $F = 40$. Тому ми вибираємо цю клітку як нашу наступну клітину. Вона виділена блакитним кольором на цій ілюстрації.



Спочатку ми видаляємо її з відкритого списку і додаємо в закритий список (ось чому вона виділена блакитним кольором). Потім ми перевіряємо сусідні клітини. Клітини, відразу праворуч від цієї клітини - стіни, тому ми їх ігноруємо. Клітка, прямо ліворуч - стартова клітина. Вона знаходиться в закритому

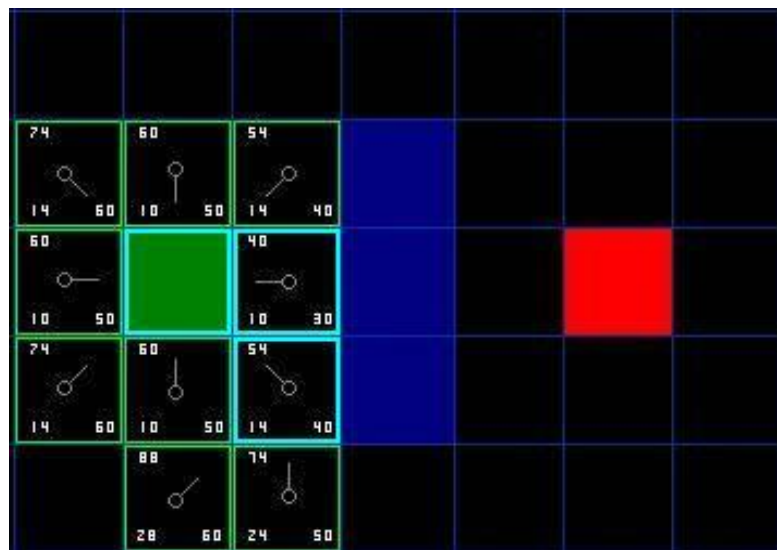
списку, тому ми її теж ігноруємо.

Решта 4 клітини вже знаходяться у відкритому списку, тому ми повинні перевірити, чи не коротше чи шляху по цим клітинам, використовуючи поточну клітку. Порівнювати будемо по соімость G . Давайте подивимося на клітку, прямо під нашою обраною кліткою. Її вартість G дорівнює 14. Якщо ми будемо рухатися по цій клітці, вартість G дорівнюватиме 20 (10, соімость G щоб дістатися до поточної клітці плюс 10 для руху вертикально вгору, до сусідній клітці). Вартість $G = 20$ більше, ніж $G = 14$, тому це буде не найкращий шлях. Це стане зрозумілим, якщо поглянути на діаграма. Більш доцільним буде рух по діагоналі на одну клітину, ніж рух на одну клітку по горизонталі, а потім одну по вертикалі.

Коли ми повторимо цей процес для всіх 4 -х сусідніх клітин, які знаходяться у відкритому списку, то дізнаємося, що жоден з шляхів не покращиться при русі по цим клітинам через обрану, тому нічого не змінюємо. Тепер, коли ми оглянули всі сусідні клітини, то закінчили з поточною кліткою і готові рухатися до наступної.

Тепер ми проходимо весь відкритий список, який зменшився до 7 -ми клітин, і вибираємо клітку з найменшою вартістю F . Цікаво, що в цьому випадку існує 2 клітини з вартістю 54. То яку ми виберемо ? Це не має ніякого значення. З метою збільшення швидкості пошуку можна вибрати останню клітку, яку ми додали у відкритий список. Це попередить пошук у виборі клітин, до яких можна буде звернутися пізніше, коли ми підберемося ближче до мети. Але насправді це не так уже й важливо. (Ось чому дві версії А * можуть знайти різні шляхи з однаковою довжиною.)

Так що давайте виберемо клітку, прямо вниз, СПАР від стартової, як показано на малюнку.

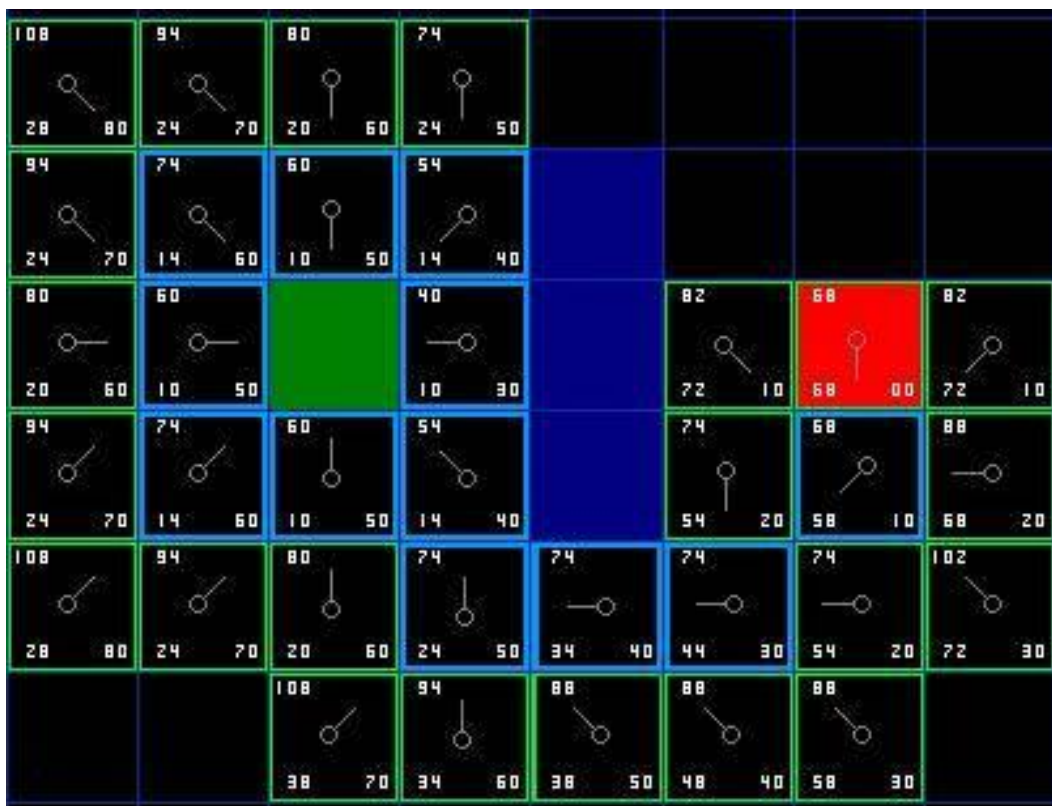


Цього разу, коли ми перевіряємо сусідні клітини, бачимо, що клітина, прямо праворуч - стіна і ми її пропускаємо. Так само чинимо і з кліткою, яка знаходиться прямо над нею. Так само ми ігноруємо клітку, яка знаходиться прямо під нею. Чому ? Тому, що ви не можете дістатися до тієї клітини без зрі-

зу кута найближчої стіни. Спочатку ви повинні спуститися вниз, а тільки потім рухатися на цю клітку. (Зауваження: Це правило зрізу кутів необов'язково. Його використання залежить від розташування ваших вершин.)

Залишається ще 5 клітин. 2 клітини, які знаходяться під поточною, ще не у відкритому списку, тому ми їх додаємо у відкритий список і призначаємо поточну клітку їх "батьком". З 3 -х інших клітин 2 вже знаходяться в закритому списку (стартова клітина і клітина, прямо над нею, на діаграмі обидві підсвічені блакитним кольором) і ми їх ігноруємо. Остання клітина, яка знаходиться прямо зліва від поточної, перевіряється на довжину шляху по поточній клітині через цю клітку за вартістю G. Ні, шлях буде не коротше. Так що ми тут закінчили і готові перевірити наступну клітину у відкритому списку.

Повторюємо цей процес до тих пір, поки не додамо цільову клітку у відкритий список. До цього часу у вас вийде щось схоже на ілюстрацію нижче.

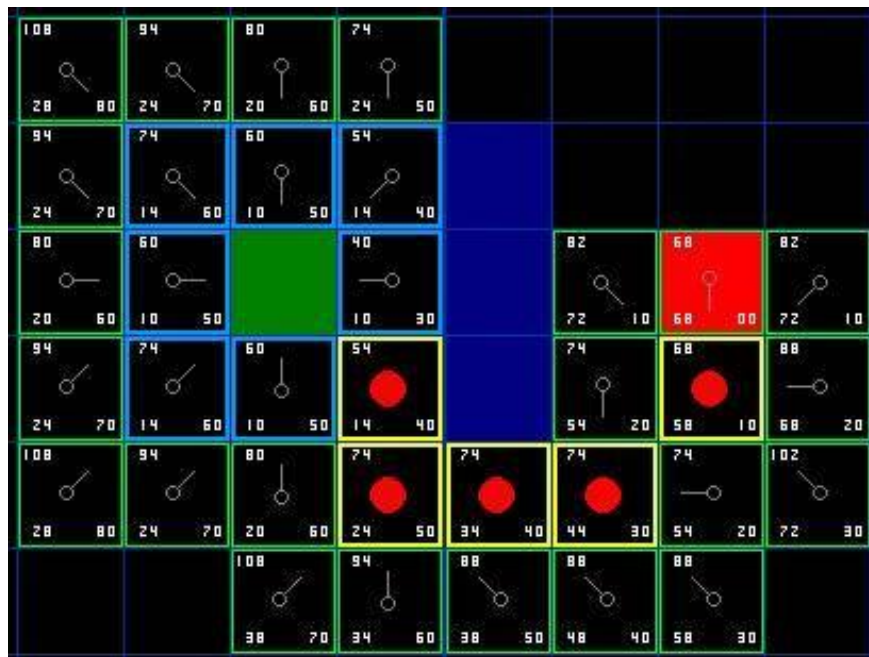


Зауважте, що батьківська клітина для клітини, що знаходиться в 2 -х клітинах під стартовою змінилася в порівнянні з передидущей ілюстрацією. Прерод цим у неї вартість G дорівнювала 28 і показчик був направлений вгору і вліво. Тепер вартість G дорівнює 20, а показчик спрямований прямо вгору. Це сталося десь в процесі нашого пошуку, коли була перевірена вартість G і виявилось, що шлях через цю клітку буде коротшим. Тому змінилася її батьківська клітина і були перераховані вартості G і F. І хоча в цьому прикладі це не здається дуже важливим, існує множество ситуацій, коли така перевірка

буде сильно впливати на вибір більш короткого шляху до мети.

Так як же ми визначимо сам шлях ? Дуже просто. Почнемо з червоною цільовою клітини і будемо рухатися назад з клітки на її батька, слідуючи вказів-

никами. Це доставить вас до стартової клітці і це і буде ваш шлях. Вийде як показано на ілюстрації нижче. Рух від стартової точки А до цільової точки В буде просто пересуванням від центру кожної клітини (вершини) до центру наступної клітини до тих пір, поки ви не досягнете мети.



Підсумки методу A *

Коротко словесний алгоритм A* виглядає наступним чином :

- 1) Додаємо стартову клітку у відкритий список.
- 2) Повторюємо наступне:
 - a. Шукаємо у відкритому списку клітку з найменшою вартістю F. Робимо її поточною кліткою.
 - b. Розміщуємо її в закритий список. (І видаляємо з відкритого)
 - c. Для кожної з сусідніх 8 -ми клітин...
 - ✖ Якщо клітина непрохідна або вона знаходиться в закритому списку, ігноруємо її. В іншому випадку робимо наступне.
 - ✖ Якщо клітина ще не в відкритому списку, то додаємо її туди. Робимо поточну клітку батьківською для цієї клітини. Розраховуємо вартості F, G і H клітини.
 - ✖ Якщо клітина вже у відкритому списку, то перевіряємо, чи не дешевше буде шлях через цю клітку. Для порівняння використовуємо вартість G. Більш низька вартість G вказує на те, що шлях буде дешевше. Якщо це так, то міняємо батька клітини на поточну клітку і перераховуємо для неї вартості G і F. Якщо ви сортуєте відкритий список за вартістю F, то вам треба відсортувати свесь список відповідно до змін.
 - d. Зупиняємося якщо :

- ✠ Додали цільову клітку у відкритий список, в цьому випадку шлях знайдений.
- ✠ Або відкритий список порожній і ми не дійшли до цільової клітини. У цьому випадку шлях відсутня.

3) Зберігаємо шлях. Рухаючись тому від цільової точки, проходячи від кожної точки до її батькові до тих пір, поки не дійдемо до стартової точки. Це і буде наш шлях.

6.4. Індивідуальні завдання

Розробити програму, яка б із зчитаного вхідного файла інформації прокладала шлях від точки А до точки В згідно алгоритму А*.

1.

A					
		*	*	*	*
		*			
				*	
				*	
				*	B

2.

A					
*	*	*	*		
		*	*	*	*
					B

3.

A					
		*	*	*	*

		*			
		*		*	
		*		*	
				*	B

4.

		*			
		*		*	
		*		*	
A		*		*	B

5.

			*	*	
	*				
	*		*	*	
	*			*	
	*			*	
A	*			*	B

6.

					В
*	*	*	*		
				*	
	*			*	
	*		*		
А	*				

7.

			*		В
			*		
			*	*	
				*	
*	*			*	
А					

8.

		*			В
				*	
	*	*	*	*	*
*	*	*	*	*	
А					

9.

					B
			*	*	*
			*		
	*	*	*	*	
A					

10.

				*	B
*		*		*	
		*		*	
	*	*			
		*	*	*	*
A					

Лабораторна робота №7. ГЕНЕТИЧНІ АЛГОРИТМИ

МЕТА РОБОТИ: ознайомитись з принципом роботи генетичного алгоритму та використати його для оптимізації заданої функції.

7.1. Програма роботи

7.1.1. Отримати завдання. Ознайомитись з правилами оголошення та використання віртуальних функцій.

7.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

7.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

6.1.4. Оформити електронний звіт про роботу та захистити її.

7.2. Вказівки до виконання роботи

7.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 7.5 і записує його до звіту.

7.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 7.4.

7.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

7.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних .h і .cpp файлів, а також пояснення до них;
- результати реалізації програми, які виведені в вікно форми;
- діаграму класів та варіантів використання з поясненням;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

7.3. Теоретичні відомості

Уявимо собі штучний світ, населений множиною істот (особин), причому кожна особина - це деяке рішення нашої задачі. Будемо вважати особину тим більше пристосованою, чим краще відповідне рішення (чим більше значення цільової функції воно дає). Тоді задача максимізації цільової функції зводиться до пошуку найбільш пристосованої істоти. Звичайно, ми не можемо поселити в наш віртуальний світ всі істоти відразу, тому що їх дуже багато. Замість цього ми будемо розглядати багато поколінь, що змінюють один одно-

го. Тепер, якщо ми зуміємо ввести в дію природний відбір і генетичне спадкування, тоді отримане середовище буде підкорятися законам еволюції. Помітимо, що, відповідно до нашого визначення пристосованості, метою цієї штучної еволюції буде саме створення найкращих рішень. Очевидно, еволюція - нескінченний процес, у ході якого пристосованість особин поступово підвищується. Примусово зупинивши цей процес через досить довгий час після його початку і вибравши найбільш пристосовану особину у поточному поколінні, ми одержимо не абсолютно точну, але близьку до оптимальної відповідь. Така, коротенько, ідея генетичного алгоритму. Перейдемо тепер до точних визначень і опишемо роботу генетичного алгоритму більш детально.

Для того щоб говорити про генетичне спадкування, потрібно наділити наші особини хромосомами. У генетичному алгоритмі хромосома - це деякий числовий вектор, що відповідає параметру, який підбирається, а набір хромосом даної особини визначає рішення задачі. Які саме вектори варто розглядати в конкретній задачі, вирішує сам користувач. Кожна з позицій вектора хромосоми називається ген.

Простий генетичний алгоритм випадковим образом генерує початкову популяцію структур. Робота генетичного алгоритму уявляє собою ітераційний процес, що продовжується доти, поки не виконаються задане число поколінь або будь-який інший критерій зупинки. В кожному поколінні генетичного алгоритму реалізується відбір пропорційно пристосованості, одноточковий кросинговер і мутація. Спочатку, пропорційний відбір призначає кожній структурі імовірність $P_s(i)$ рівну відношенню її пристосованості до сумарної пристосованості популяції:

$$P_s(i) = \frac{f(i)}{\sum_{i=1}^n f(i)}$$

Потім відбувається відбір (із заміщенням) усіх n особин для подальшої генетичної обробки, відповідно до величини $P_s(i)$.

При такому відборі члени популяції з більш високою пристосованістю з більшою імовірністю будуть частіше вибиратися, ніж особини з низькою пристосованістю. Після відбору, n обраних особин випадковим чином розбиваються на $n/2$ пари. Для кожної пари з імовірністю P_c може застосовуватися кросинговер. Відповідно з імовірністю $1-P_c$ кросинговер не відбувається і незмінні особини переходять на стадію мутації. Якщо кросинговер відбувається, отримані нащадки заміняють собою батьків і переходять до мутації.

Визначимо тепер поняття, що відповідають мутації і кросинговеру в генетичному алгоритмі.

Мутація - це перетворення хромосоми, що випадково змінює одну чи декілька її позицій (генів). Найбільш розповсюджений вид мутацій - випадкова зміна тільки одного з генів хромосоми.

Кросинговер (у літературі по генетичних алгоритмах також вживається назва кросовер або схрещування) - це операція, при якій із двох хромосом породжується одна чи декілька нових хромосом. Одноточковий кросинговер працює в такий спосіб. Спочатку, випадковим образом вибирається одна з $l-1$ то-

чок розриву. (Точка розриву - ділянка між сусідніми бітами в рядку.) Обидві батьківські структури розриваються на два сегменти по цій точці. Потім, відповідні сегменти різних батьків склеюються і виходять два генотипи нащадків.

Наприклад, припустимо, один батько складається з 10 нулів, а інший - з 10 одиниць. Нехай з 9 можливих точок розриву обрана точка 3. Батьки і їхні нащадки показані нижче.

Кросинговер

Батько 1 0000000000 000~0000000--> 111~0000000 1110000000 Нащадок

1

Батько 2 1111111111 111~1111111 --> 000~1111111 0001111111 Нащадок

2

Після того як закінчується стадія кросинговеру, виконуються оператори мутації. У кожному рядку, що піддається мутації, кожен біт з імовірністю P_m змінюється на протилежний.

Популяція, отримана після мутації записує поверх старої і цим цикл одного покоління завершується. Наступні покоління обробляються подібним чином: відбір, кросинговер і мутація.

В даний час дослідники ген пропонують багато інших операторів відбору, кросинговеру і мутації. От лише найбільш розповсюджені з них.

Елітні методи відбору гарантують, що при відборі обов'язково будуть виживати кращий чи кращі члени популяції сукупності. Найбільш поширена процедура обов'язкового збереження тільки одної кращої особини, якщо вона не пройшла як інші через процес відбору, кросинговеру і мутації. Елітизм може бути впроваджений практично в будь-який стандартний метод відбору.

Двоточковий кросинговер і рівномірний кросинговер - цілком гідні альтернативи одноточковому оператору. В двоточковому кросинговері вибираються дві точки розриву, і батьківські хромосоми обмінюються сегментом, що знаходиться між двома цими точками. У рівномірному кросинговері, кожен біт першого батька успадковується першим нащадком із заданою імовірністю; у протилежному випадку цей біт передається другому нащадку. І навпаки.

Блок-схема генетичного алгоритму зображена на рис. 7.1. Спочатку генерується початкова популяція особин (індивідуумів), тобто деякий набір рішень задачі. Як правило, це робиться випадковим образом. Потім ми повинні змоделювати розмноження всередині цієї популяції. Для цього випадково відбирається декілька пар індивідуумів, відбувається схрещування між хромосомами в кожній парі, а отримані нові хромосоми втілюються в популяцію нового покоління. У генетичному алгоритмі зберігається основний принцип природного відбору - чим пристосованіше індивідуум (чим більше відповідне йому значення цільової функції), тим з більшою імовірністю він буде брати участь у схрещуванні. Тепер моделюються мутації - у декількох випадково обраних особинах нового покоління змінюються деякі гени. Потім стара популяція частково або цілком знищується і ми переходимо до розгляду наступного покоління. Популяція наступного покоління в більшості реалізацій генетичних алгоритмів містить стільки ж особин, скільки початкова, але в силу відбору пристосованість у ній у середньому вище. Тепер описані процеси відбору, схрещу-

вання й мутації повторюються вже для цієї популяції і т.д.

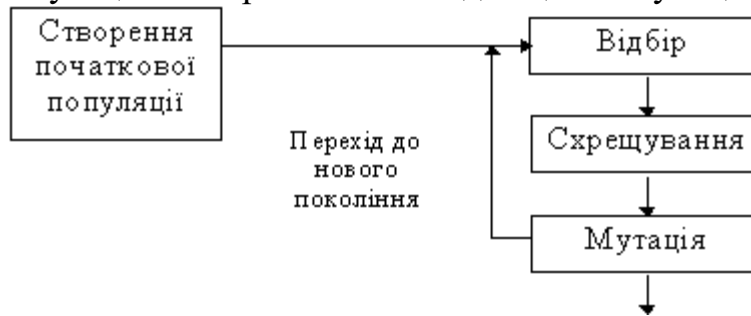


Рис. 7.1. Блок-схема генетичного алгоритму

В кожному наступному поколінні ми будемо спостерігати виникнення зовсім нових рішень нашої задачі. Серед них будуть як погані, так і хороші, але завдяки відбору число прийнятних рішень буде зростати. Помітимо, що в природі не буває абсолютних гарантій, і найпристосованіший тигр може загинути від рушничного пострілу, не залишивши нащадків. Імітуючи еволюцію на комп'ютері, ми можемо уникати подібних небажаних подій і завжди зберігати життя кращому з індивідуумів поточного покоління - така методика називається "стратегією елітизму".

7.4. Індивідуальні завдання

Задано $y=0.47 \cdot N_0$, $z=-1.32 \cdot N_0$ (де N_0 – номер варіанту). Відповідно до номера варіанту потрібно розробити програму для оптимізації функції $b[x,y,z]$ генетичним алгоритмом відносно змінної x (у діапазоні 0-1024) згідно з такими математичними виразами.

$$1) \ b[x,y,z] = x \left(\frac{y + \arctg|x^2 + z|^{0.1}}{3/x + \sin^2(y+z)^3} + ye^{-\frac{x+z}{y+z}} \right);$$

$$2) \ b[x,y,z] = \frac{2z + \cos|y - 3x|^{1/3}}{2.1 + \sin^2|z^3 - y|^{0.2}} + \ln^2 \left| \frac{z-x}{z+y} \right|;$$

$$3) \ b[x,y,z] = 1 + \frac{|y-x|^2}{|z-1|^{1.34}} + \frac{(z-x)^2}{\sin^2 z} + \frac{|y-z|^3}{\cos y^2};$$

$$4) \ b[x,y,z] = \left| \frac{x+y}{|z|^{0.6}} + \sin^2 \frac{x+z^2}{2x+y} \right|^{1/3} - ze^{\frac{x^2-y}{1+z}};$$

$$5) \ b[x,y,z] = \frac{x^2 + z^2 / \lg^2|x|^{0.3}}{3 + x + y^2 / 2! + z^3 / 3!} + \ln^{0.3} \left| \frac{y}{z} \right|^{1/3};$$

$$6) \ b[x,y,z] = \cos^2 \left(\arctg \frac{x^2 + y}{z+1} \right) + \frac{x}{z} e^{3x+y};$$

$$7) \quad b[x, y, z] = 1 - \frac{x + y}{|z|^{0.34}} + \frac{y^2}{3!} + \frac{z^3}{5!} + \frac{e^{x-y}}{z + y};$$

$$8) \quad b[x, y, z] = \frac{|y + x|^{0.2}}{|z|^{1.34}} + \frac{(y - z)^2}{1 + \sin^2 z} + \frac{|z - y|^3}{z / \cos x^2};$$

$$9) \quad b[x, y, z] = y \left| \frac{|x|^{0.3}}{z + y} + \operatorname{tg}^2 \frac{x + z^2}{2x - 1.4} \right|^{\frac{1}{3}} - ze^{x^2 - y};$$

$$10) \quad b[x, y, z] = x \left(\frac{y + \operatorname{arctg} |x^2 + z|^{0.1}}{2z + x + \sin^2 y^3} + e^{-\frac{x+z}{z+1}} \right);$$

ЛАБОРАТОРНА РОБОТА №8. ПОБУДОВА ФУНКЦІЙ НАЛЕЖНОСТІ ЛІНГВІСТИЧНОЇ ЗМІННОЇ

МЕТА РОБОТИ: вивчення основних способів побудови функцій належності лінгвістичної змінної, а також їх структурного та графічного представлення за допомогою табличного процесора Excel.

8.1. Програма роботи

8.1.1. Отримати завдання.

8.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

8.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

8.1.4. Оформити електронний звіт про роботу та захистити її.

8.2. Вказівки до виконання роботи

8.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 8.5 і записує його до звіту.

8.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 8.4.

8.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

8.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних .h і .cpp файлів, а також пояснення до них;
- результати реалізації програми;
- діаграму класів та діаграму варіантів використання з поясненням;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

8.3. Теоретичні відомості

Практичне використання теорії нечітких множин припускає наявність функцій належності, яким описуються лінгвістичні терми "низький", "середній", "високий" і тому подібне. Завдання побудови функцій належності ставиться таким чином: дано дві множини: множина термів і універсальна $U = \{u_1, u_2, \dots, u_n\}$. Нечітка множина, якою описується лінгвістичний терм, на

$$\tilde{f}_j = \left(\frac{\mu_j(u_1)}{u_1}, \frac{\mu_j(u_2)}{u_2}, \dots, \frac{\mu_j(u_n)}{u_n} \right).$$

універсальній множини U представляється у вигляді:

Необхідно визначити ступені належності елементів множини до елементів з множини L , тобто знайти для всіх $i = \overline{1, n}$.

Розглянемо два методи побудови функцій належності. Перший метод заснований на статистичній обробці думок групи експертів. Другий метод базується на парних порівняннях, що виконуються одним експертом.

8.3.1. Метод статистичної обробки експертної інформації

Кожен експерт заповнює опитувальника, в якому указує своя думка про наявність у елементів $u_i (i = \overline{1, n})$ властивостей нечіткої множини $\tilde{f}_j (j = \overline{1, m})$. Опитувальника має наступний вигляд:

	u_1	u_2	.	u_n
\tilde{f}_1				
\tilde{f}_2				
.				
\tilde{f}_m				

Введемо наступні позначення: K - кількість експертів; b_{ji}^k - думка k -го експерта про наявність у елементу властивостей нечіткої множини $j = \overline{1, m}$. Вважатимемо, що експертні оцінки бінарні, т.е., де 1 (0) указує на наявність (відсутність) у елементу властивостей нечіткої множини \tilde{f}_j . За наслідками опиту експертів, ступені належності нечіткій множині $\tilde{f}_j (j = \overline{1, m})$ розраховуються таким чином:

$$\mu_j(u_i) = \frac{1}{K} \sum_{k=1}^K b_{ji}^k, \quad i = \overline{1, n}. \quad (1)$$

Приклад 8.1. Побудувати функції належності термів "низький", "середній", "високий", використовуваних для лінгвістичної оцінки змінної "зростання чоловіка". Результати опиту п'яти експертів приведені в табл. 8.1.

Таблиця 8.1. Результати опиту експертів

k	терми	(160, 165)	(165, 170)	(170, 175)	(175, 180)	(180, 185)	(185, 190)	(190, 195)	(195, 200)

Лабораторна робота №8. Побудова функцій належності лінгвістичної змінної

Експерт 1	низький	1	1	1	0	0	0	0	0
	середній	0	0	1	1	1	0	0	0
	високий	0	0	0	0	0	1	1	1
Експерт 2	низький	1	1	1	0	0	0	0	0
	середній	0	0	1	1	0	0	0	0
	високий	0	0	0	0	1	1	1	1
Експерт 3	низький	1	0	0	0	0	0	0	0
	середній	0	1	1	1	1	1	0	0
	високий	0	0	0	0	0	1	1	1
Експерт 4	низький	1	1	1	0	0	0	0	0
	середній	0	0	0	1	1	1	0	0
	високий	0	0	0	0	0	0	1	1
Експерт 5	низький	1	1	0	0	0	0	0	0
	середній	0	1	1	1	0	0	0	0
	високий	0	0	0	1	1	1	1	1

Результати обробки експертних думок представлені в таблиці 8.2.

Таблиця 8.2. Результати обробки думок експертів

терми	[160, 165)	[165, 170)	[170, 175)	[175, 180)	[180, 185)	[185, 190)	[190, 195)	[195, 200)
низький	5	4	3	0	0	0	0	0
	1	0.8	0.6	0	0	0	0	0
середній	0	2	4	5	3	2	0	0
	0	0.4	0.8	1	0.6	0.4	0	0
високий	0	0	0	1	2	4	5	5
	0	0	0	0.2	0.4	0.8	1	1

Числа над пунктирною лінією - це кількість голосів, відданих експертами за приналежність нечіткій множини відповідного елементу універсальної множини. Числа під пунктирною лінією - ступені належності, розраховане по формулі (8.1). Графіки функцій належності показані на рис. 8.1.

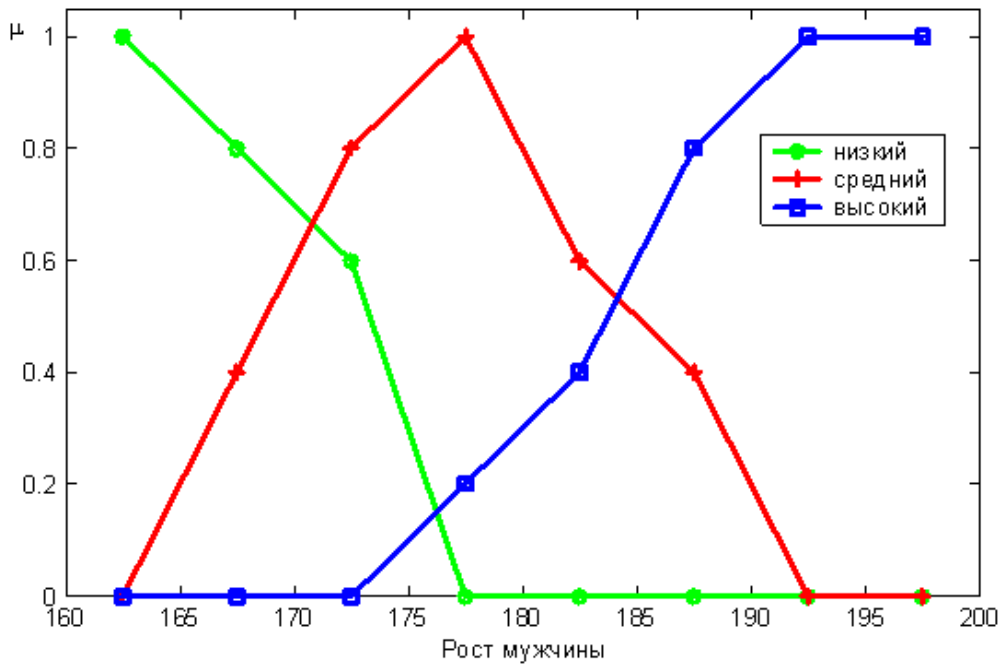


Рис. 8.1. Функції належності нечітких множин з прикладу 8.1

8.3.2. Побудова функцій належності на основі парних порівнянь

Початковою інформацією для побудови функцій належності є експертні парні порівняння. Для кожної пари елементів універсальної множини експерт оцінює перевагу одного елементу над іншим по відношенню до властивості нечіткої множини. Парні порівняння зручно представляти наступною матрицею:

$$A = \begin{matrix} & \begin{matrix} u_1 & u_2 & \dots & u_n \end{matrix} \\ \begin{matrix} u_1 \\ u_2 \\ \dots \\ u_n \end{matrix} & \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \end{matrix},$$

де a_{ij} – рівень переваги елементу над u_j , що визначається за дев'ятибальною шкалою Сааті:

- 1 - якщо відсутня перевага елементу над елементом u_j ;
- 3 - якщо є слабка перевага над u_j ;
- 5 - якщо є істотна перевага над u_j ;
- 7 - якщо є явна перевага над u_j ;
- 9 - якщо є абсолютна перевага над u_j ;
- 2,4,6,8 - проміжні порівняльні оцінки.

Матриця парних порівнянь є діагональною ($a_{ii} = 1, i = \overline{1, n}$) і обернено симетричною ($a_{ij} = \frac{1}{a_{ji}}, i, j = \overline{1, n}$).

Ступені належності приймаються рівними відповідним координатам власного вектора матриці парних порівнянь:

$$\mu(u_i) = w_i, \quad i = \overline{1, n}. \quad (8.2)$$

Власний вектор знаходиться з наступної системи рівнянь:

$$\begin{cases} A \cdot W = \lambda_{\max} \cdot W \\ w_1 + w_2 + \dots + w_n = 1 \end{cases}, \quad (8.3)$$

де λ_{\max} - максимальне власне значення матриці A.

Приклад 8.2. Побудувати функцію належності нечіткої множини "високий чоловік" на універсальній множині {170, 175, 180, 185, 190, 195}.

Парні порівняння задамо наступною матрицею:

$$A = \begin{matrix} & \begin{matrix} 170 & 175 & 180 & 185 & 190 & 195 \end{matrix} \\ \begin{matrix} 170 \\ 175 \\ 180 \\ 185 \\ 190 \\ 195 \end{matrix} & \begin{bmatrix} 1 & 1/2 & 1/4 & 1/6 & 1/8 & 1/9 \\ 2 & 1 & 1/3 & 1/5 & 1/7 & 1/8 \\ 4 & 3 & 1 & 1/4 & 1/4 & 1/5 \\ 6 & 5 & 4 & 1 & 1/3 & 1/3 \\ 8 & 7 & 4 & 3 & 1 & 1 \\ 9 & 8 & 5 & 3 & 1 & 1 \end{bmatrix} \end{matrix}.$$

Власні значення цієї матриці парних порівнянь рівні:

6.2494;

0.0318 + 1.2230i;

0.0318 - 1.2230i;

- 0.1567 + 0.2392i;

- 0.1567 - 0.2392i;

0.0004.

Отже $\lambda_{\max} = 6.2494$. Ступені належності, знайдене по формулах (8.3) і (8.2), приведені в табл. 8.3. Нечітка множина вийшла субнормальною. Для нормалізації розділимо всі ступені належності на максимальне значення, тобто на 0.3494. Графіки функцій належності субнормальної і нормальної нечіткої множини "високий чоловік" приведені на рис. 8.2.

Таблиця 8.3. Приклад 8.2: функції належності нечіткої множини "високий чоловік"

u_i	170	175	180	185	190	195
$\mu_{\text{високий чоловік}}(u_i)$ (субнормальна нечітка множина)	0.0284	0.0399	0.0816	0.1754	0.3254	0.3494
$\mu_{\text{високий чоловік}}(u_i)$ (нормальна нечітка множина)	0.0813	0.1141	0.2335	0.5021	0.9314	1.0000

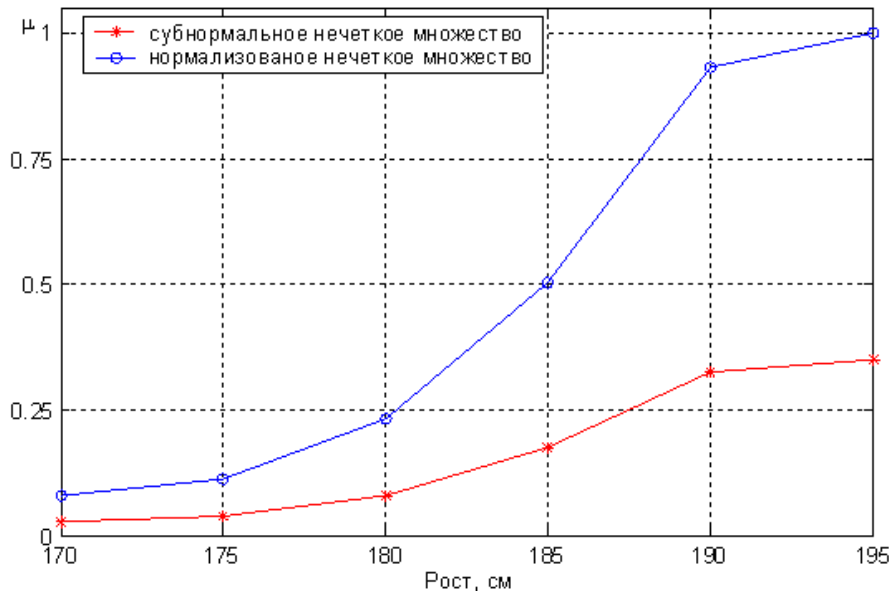


Рис. 8.2. Приклад 8.2: функції належності нечіткої множини "високий чоловік"

Відхилення від n може служити мірою неузгодженості парних порівнянь експерта. У прикладі 8.2, $n = 6$. Отже, міра неузгодженості рівна 0.2494. При узгоджених парних порівняннях процедура побудови функцій належності значно спрощується.

При узгоджених думках експерта матриця парних порівнянь володіє наступними властивостями:

- вона діагональна, тобто $a_{ii}=1$, $i=1..n$;
- вона назад симметрична, тобто елементи, симетричні щодо головної діагонали, зв'язані залежністю $a_{ij}=1/a_{ji}$, $i,j=1..n$;
- вона транзитивна, тобто $a_{ik}a_{kj}=a_{ij}$, $i,j,k=1..n$.

Наявність цих властивостей дозволяє визначити всі елементи матриці парних порівнянь, якщо відомі $(n-1)$ недіагональних елементів. Наприклад, якщо відомий k -тий рядок, тобто елементи a_{kj} , $i,j=1..n$, то довільний елемент a_{ij} визначається так:

$$a_{ij} = \frac{a_{kj}}{a_{ki}} \quad i,j,k = \overline{1,n}. \quad (8.4)$$

Після визначення всіх елементів матриці парних порівнянь ступеня приналежності, нечіткої множини обчислюються за формулою:

$$\mu(u_i) = \frac{1}{a_{1i} + a_{2i} + \dots + a_{ni}} \quad (8.5)$$

Формула (8.5), на відміну від формул (8.2) - (8.3) не вимагає виконання трудомістких обчислювальних процедур, пов'язаних із знаходженням власного вектора матриці A .

Приклад 8.3. Побудувати функцію належності нечіткої множини "високий чоловік" на універсальній множині $\{170, 175, 180, 185, 190, 195\}$, якщо відомі такі експертні парні порівняння:

- абсолютна перевага 195 над 170;
- явна перевага 195 над 175;
- істотна перевага 195 над 180;
- слабка перевага 195 над 185;
- відсутня перевага 195 над 190.

Приведеним експертним висловам відповідає така матриця парних порівнянь:

$$A = \begin{matrix} & \begin{matrix} 170 & 175 & 180 & 185 & 190 & 195 \end{matrix} \\ \begin{matrix} 170 \\ 175 \\ 180 \\ 185 \\ 190 \\ 195 \end{matrix} & \begin{bmatrix} 1 & 7/9 & 5/9 & 1/3 & 1/9 & 1/9 \\ 9/7 & 1 & 5/7 & 3/7 & 1/7 & 1/7 \\ 9/5 & 7/5 & 1 & 3/5 & 1/5 & 1/5 \\ 3 & 7/3 & 5/3 & 1 & 1/3 & 1/3 \\ 9 & 7 & 5 & 3 & 1 & 1 \\ \mathbf{9} & \mathbf{7} & \mathbf{5} & \mathbf{3} & \mathbf{1} & \mathbf{1} \end{bmatrix} \end{matrix}$$

Напівжирним шрифтом виділені елементи, відповідні парним порівнянням з умови прикладу. Решта елементів знайдена по формулі (8.4). Застосовуючи формули (8.5) знаходимо ступені належності (табл. 8.4). Для нормалізації нечіткої множини розділимо всі ступені належності на максимальне значення, тобто на 0.3588. Графіки функцій належності субнормальної і нормальної нечіткої множини "високий чоловік" приведені на мал. 8.3.

Таблиця 8.4. Функції належності нечіткої множини "високий чоловік"

u _i	170	175	180	185	190	195
м високий чоловік (u _i) (субнормальна нечітка множина)	0.0399	0.0513	0.0718	0.1196	0.3588	0.3588
м високий чоловік (u _i) (нормальна нечітка множина)	0.1111	0.1429	0.2000	0.3333	1.0000	1.0000

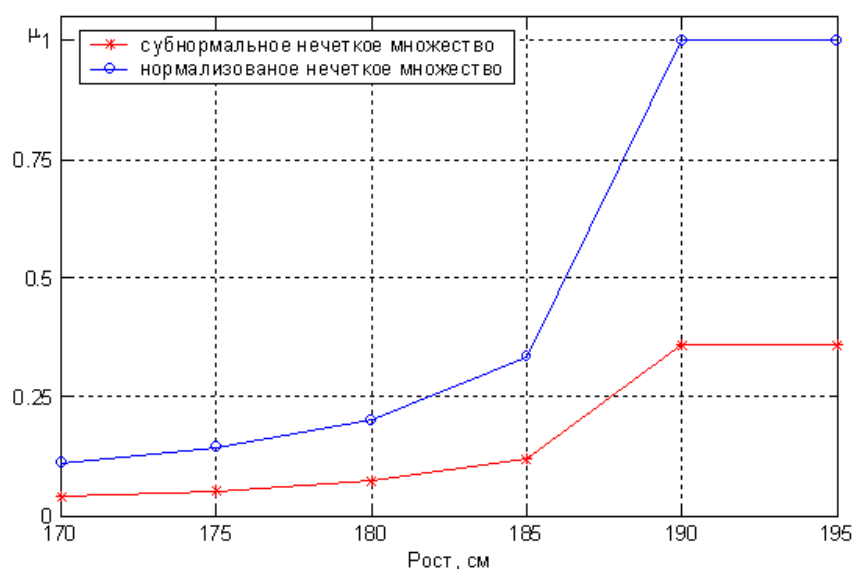


Рис. 8.3. Функції належності нечіткої множини "високий чоловік"

8.4. Індивідуальні завдання

Побудувати функції належності заданих лінгвістичних змінних згідно варіанту методом статичної обробки та парних порівнянь.

1. Досвід роботи в сфері комп'ютерних технологій,
2. Стан погодних умов
3. Температура води в душі
4. Температура повітря
5. Кут хитання контейнера на контейнерному крані
6. "Кредитоспроможність" клієнта $T4 = \{\text{"дуже низька"}, \text{"низька"}, \text{"середня"}, \text{"висока"}, \text{"дуже висока"}\}$.
7. Тиск у реакторі $[0,100]$. Початкова множина термів $\{\text{Високий}, \text{Середній}, \text{Низький}\}$.
8. Оцінка техніки гри: визначається в балах від 0 до 100, будується на основі нечітких (суб'єктивних) оцінок гравців. Множина визначення $[0,100]$. Базові терми - відмінна, дуже хороша, хороша, не дуже хороша, погана
9. Чинник поля - (розраховуватися як $HP/HG - GP/GG$, де HP - загальна кількість очок, набрана командою господарем поля в домашніх іграх поточного чемпіонату; HG - загальна кількість домашніх ігор, проведених командою господарем поля в поточному чемпіонаті; GP - загальна кількість очок, набрана гостьовою командою в поточного чемпіонату на виїзді; GG - загальна кількість виїзних ігор, проведених гостьовою командою в поточному чемпіонаті).
10. "Ступінь ризику" фінансування даного проекту $T1 = \{\text{«допустимий»}, \text{«малий»}, \text{«середній»}, \text{«високий»}\}$.
11. Ступінь забруднення важкими металами узбіччя автострад
12. Оцінка небезпеки від забруднення води у міському водогоні
13. Ризик підтоплення заплавлених територій річок при таненні снігу
14. Оцінка радіоактивного забруднення ґрунтів навколо промислових об'єктів
15. Ступінь забруднення повітря небезпечними сполуками навколо сміттєзвалища

ЛАБОРАТОРНА РОБОТА №9. ПОБУДОВА БАЗИ ЗНАНЬ НЕЧІТКОЇ ЕКСПЕРТНОЇ СИСТЕМИ

МЕТА РОБОТИ: вивчення основних способів представлень та інструментів створення баз знань за допомогою інструментального пакету Fuzzy системи MATLAB

9.1. Програма роботи

9.1.1. Отримати завдання.

9.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

9.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

9.1.4. Оформити електронний звіт про роботу та захистити її.

9.2. Вказівки до виконання роботи

9.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 9.5 і записує його до звіту.

9.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 9.4.

9.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

9.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних .h і .cpp файлів, а також пояснення до них;
- результати реалізації програми;
- діаграму класів та діаграму варіантів використання з поясненням;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

ЛАБОРАТОРНА РОБОТА №10. МОДЕЛІ ШТУЧНОГО НЕЙРОНА

МЕТА РОБОТИ: вивчення основних моделей штучного нейрона, їх математичного опису, а також функціонального і структурного графічних представлень, дослідження функцій активації і розглянутих моделей нейронів за допомогою інструментального пакету імітаційного моделювання Simulink системи MATLAB

10.1. Програма роботи

10.1.1. Отримати завдання.

10.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

10.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

10.1.4. Оформити електронний звіт про роботу та захистити її.

10.2. Вказівки до виконання роботи

10.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 10.5 і записує його до звіту.

10.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 10.4.

10.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

10.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних .h і .cpp файлів, а також пояснення до них;
- результати реалізації програми;
- діаграму класів та діаграму варіантів використання з поясненням;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

10.3. Теоретичні відомості

Простий нейрон

Найпростішим елементом будь-якої нейронної мережі є нейрон. Структура нейрона з єдиним скалярним входом показана на рис. 10.1,а.

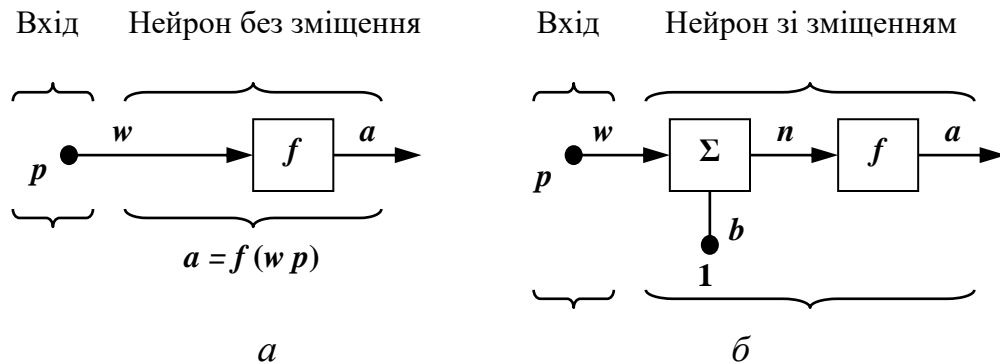


Рис. 10.1. Простий нейрон

Скалярний вхідний сигнал p перемножується на скалярний ваговий коефіцієнт w , і результуючий зважений вхід $w \cdot p$ є аргументом функції активації нейрона, яка породжує скалярний вихід a .

Нейрон, показаний на рис. 10.1,б, доповнений скалярним зсувом b . Зміщення підсумовується із зваженим входом $w \cdot p$ і приводить до зміщення аргументу функції на величину b . Дію зсуву можна звести до схеми зважування, якщо представити що нейрон має другий вхідний сигнал із значенням, рівним 1. Вхід n функції активації нейрона, як і раніше, залишається скалярним і рівним сумі зваженого входу і зсуву b . Ця сума є аргументом функції активації f ; виходом функції активації є сигнал a . Константи w і b є скалярними параметрами нейрона. Основний принцип роботи нейронної мережі полягає в налаштуванні параметрів нейрона для того, щоб функціонування мережі відповідало деякій бажаній поведінці. Регулюючи ваги або параметри зсуву, можна "навчити" мережу виконувати конкретну роботу; можливо також, що мережа сама коректуватиме свої параметри, щоб досягти необхідного результату.

Рівняння нейрона із зсувом має вигляд

$$a = f(w \cdot p + b \cdot 1).$$

Як вже наголошувалося, зсув b – скалярний параметр нейрона, який не є входом, що настраюється, а константа 1 , яка управляє зсувом, розглядається як вхід і може бути врахована у вигляді лінійної комбінації векторів входу

$$a = [w \ b] \cdot \begin{bmatrix} p \\ 1 \end{bmatrix}.$$

Функція активації

Функції активації (передавальні функції) нейрона можуть мати найрізноманітніший вигляд. Функція активації f , як правило, належить класу сигмоїдальних функцій з аргументом n і виходом a .

Нижче розглянуті три найбільш поширені функції активації.

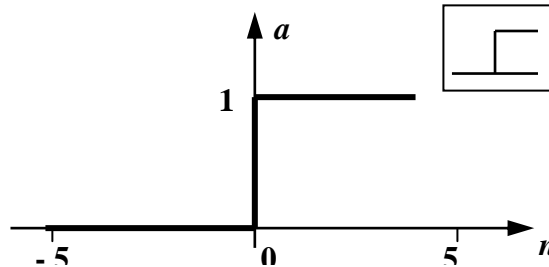


Рис. 10.2. Функція активації *hardlim*

Одинична функція активації з жорстким обмеженням *hardlim*. Ця функція описується співвідношенням $a = \text{hardlim}(n) = 1(n)$ і показана на рис. 10.2. Вона рівна 0 , якщо $n < 0$, і 1 , якщо $n \geq 0$.

Символ у квадраті в правому верхньому кутку графіка характеризує функцію активації. Це зображення використовується на структурних схемах нейронних мереж.

До складу пакету *Neural Network Toolbox* входить М-функція *hardlim*, що реалізовує функцію активації з жорсткими обмеженнями.

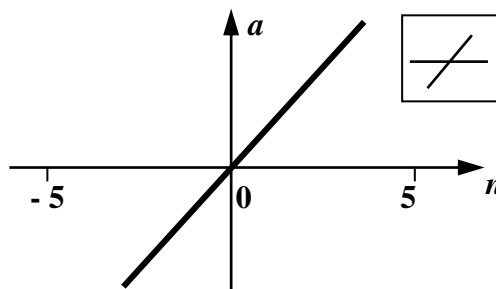


Рис. 10.3. Лінійна функція активації *purelin*

Лінійна функція активації **purelin**. Ця функція описується співвідношенням

$a = \text{purelin}(n) = n$ і показана на рис. 10.3.

Логістична функція активації **logsig**. Ця функція описується співвідношенням $a = \text{logsig}(n) = 1/(1 + \exp(-n))$ і показана на рис. 10.4.

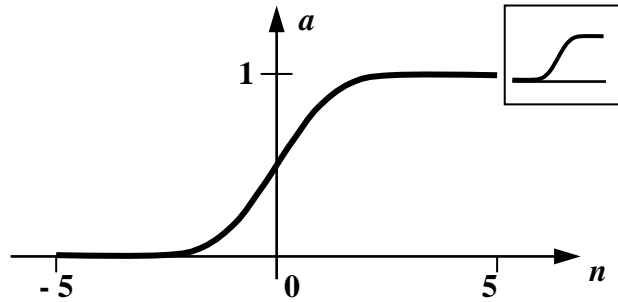


Рис. 10.4. Функція **logsig**

Вона належить до класу сигмоїдальних функцій, і її аргумент може приймати будь-яке значення в діапазоні від $-\infty$ до $+\infty$, а вихід змінюється в діапазоні від **0** до **1**. У пакеті **Neural Network Toolbox** вона представлена М-функцією **logsig**. Завдяки властивості диференціювання ця функція часто використовується в мережах з навчанням на основі методу зворотного розповсюдження помилки.

У пакет **Neural Network Toolbox** включені і інші функції активації. Використовуючи мову MATLAB, користувач може створювати і свої власні унікальні функції.

Нейрон з векторним входом

Нейрон з одним вектором входу p і з R елементами p_1, p_2, \dots, p_R , показаний на рис. 10.5. Тут кожен елемент входу перемножується на ваги $w_{11}, w_{12}, \dots, w_{1R}$ відповідно і зважені значення передаються на суматор. Їх сума рівна скалярному добутку вектора-рядка W на вектор входу p .

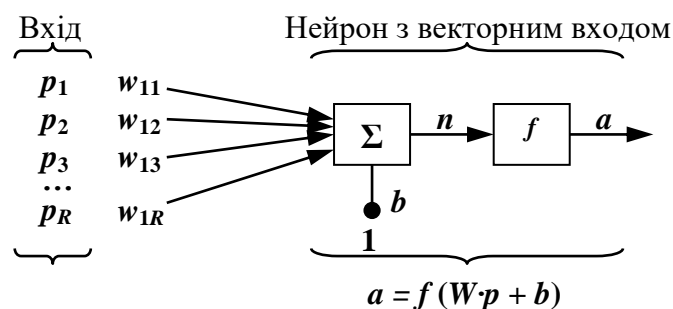


Рис. 10.5. Функціональна схема нейрона

Нейрон має зсув b , який підсумовується із зваженою сумою входів. Результуюча сума n визначається як

$$n = w_{11}p_1 + w_{12}p_2 + \dots + w_{1R}p_R + b$$

і служить аргументом функції активації f . У MATLAB цей вираз записується так:

$$n = W \cdot p + b.$$

Структура нейрона, показана на рис. 10.5, містить багато зайвих деталей. При розгляді мереж, що складаються з великого числа нейронів, використовуватиметься укрупнена структурна схема нейрона (рис. 10.6).

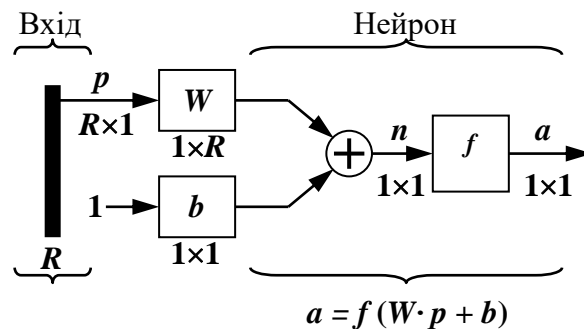


Рис. 10.6. Структурна схема нейрона

Вхід нейрона зображається у вигляді темної вертикальної риски, під якою вказується кількість елементів входу R . Розмір вектора входу p вказується нижче за символ p і рівний $R \times 1$. Вектор входу перемножується на вектор-рядок W довжини R . Як і раніше, константа 1 розглядається як вхід, який перемножується на скалярний зсув b . Входом n функції активації нейрона служить сума зсуву b і добутки $W \cdot p$. Ця сума перетворюється функцією активації f , на виході якої отримуємо вихідну величину нейрона a , яка в даному випадку є скалярною величиною. Структурна схема, приведена на рис. 10.6, називається **шаром мережі**. Шар характеризується матрицею ваг W , зсувом b , операціями множення $W \cdot p$, підсумовування і функцією активації f . Вектор входів p зазвичай не включається в характеристики шару.

Кожного разу, коли використовується скорочене позначення мережі, роз-

мірність матриць вказується під іменами векторно-матричних змінних. Ця система позначень пояснює будову мережі і пов'язану з нею матричну математику.

На укрупненій структурній схемі для позначення типу функції активації застосовуються спеціальні графічні символи, деякі з яких наведено на рис. 10.7.

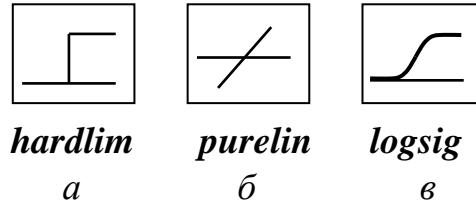


Рис. 10.7. Функції активації, де: *a* – ступінчаста; *б* – лінійна; *в* – логістична

10.4. Індивідуальні завдання

Завдання 1. Для функції активації з жорсткими обмеженнями ***hardlim*** і її похідної ***dhardlim***, які визначаються наступними співвідношеннями:

$$hardlim(n) = \begin{cases} 0, & n < 0; \\ 1, & n \geq 0; \end{cases}$$

$$dhardlim(n) = \begin{cases} 0, & n < 0; \\ 1, & n \geq 0; \end{cases}$$

виконати наступні дії:

1. Видати на екран інформацію про ці функції за допомогою наступних команд:

```
name=hardlim('name')           % - повна назва функції;
```

```
dname=hardlim('deriv')         % - назва похідної;
```

```
inrange=hardlim('active')      % - діапазон входу;
```

```
outrange=hardlim('output')     % - діапазон виходу;
```

2. Побудувати графіки функцій:

```
n=-5:0,1:5;
```

```
a=hardlim(n);
```

```
da=dhardlim(n);
```

```
plot(n,a,'r') %графік функції активації - черво-  
ний;
```

```
hard on
```

```
plot (n,da,'c') %графік похідної - блакитний;
```

3. Обчислити вектори виходу A і похідної dA_dN для шару з трьох нейронів з вектором входу N , що складається з трьох компонентів:

```
N=[-0,7; 0,1; 0,8];
```

```
A=hardlim(N) % - вектор виходу функції активу;
```

```
dA_dN= dhardlim(N,A) % - вектор виходу похідної.
```

4. Розглянуту послідовність команд оформити у вигляді скрипта і записати в М-файл з ім'ям *hardlimfle*.

Завдання 2. Для симетричної функції активації з жорсткими обмеженнями *hardlims* і її похідної *dhardlims*, які визначаються співвідношеннями

$$\text{hardlims}(n) = \begin{cases} -1, & n < 0; \\ 1, & n \geq 0; \end{cases}$$

$$\text{dhardlims}(n) = \begin{cases} 0, & n < 0; \\ 0, & n \geq 0; \end{cases}$$

видати на екран інформацію про ці функції, побудувати їх графіки і розрахувати вектори виходу, скориставшись скриптом з М-файлу *hardlimfle*. Новий скрипт записати у файл під ім'ям *hardlimsfile*.

Завдання 3. Для лінійної функції активації *purelin* і її похідної *dpurelin*, які визначаються співвідношеннями

$$\begin{aligned} \text{purelin} &= n; \\ \text{dpurelin} &= 1, \end{aligned}$$

видати на екран інформацію про ці функції, побудувати їх графіки і розрахувати вектори виходу, скориставшись скриптом з М-файлу *hardlimfle*. Новий скрипт записати у файл під ім'ям *purelinfile*.

Завдання 4. Для позитивної лінійної функції активації *poslin* і її похідній *dposlin*, які визначаються співвідношеннями

$$\text{poslin} = \begin{cases} 0, & n < 0; \\ n, & n \geq 0; \end{cases}$$

$$dposlin = \begin{cases} 0, & n < 0; \\ 1, & n \geq 0; \end{cases}$$

видати на екран інформацію про ці функції, побудувати їх графіки і розрахувати вектори виходу, скориставшись скриптом з М-файлу *hardlimfle*. Новий скрипт записати у файл під ім'ям *poslinfile*.

Завдання 5. Для лінійної функції активації з обмеженнями *saflin* і її похідної *dsaflin*, які визначаються співвідношеннями

$$saflin(n) = \begin{cases} 0, & n < 0; \\ n, & 0 \leq n \leq 1; \\ n, & n > 1, \end{cases}$$
$$dsaflin(n) = \begin{cases} 0, & n < 0; \\ 1, & 0 \leq n \leq 1; \\ 1, & n > 1; \end{cases}$$

видати на екран інформацію про ці функції, побудувати їх графіки і розрахувати вектори виходу, скориставшись скриптом з М-файлу *hardlimfle*. Новий скрипт записати у файл під ім'ям *satlinfile*.

Завдання 6. Для симетричної лінійної функції активації *satlins* з обмеженнями і її похідної *dsatlins*, які визначаються співвідношеннями

$$satlins(n) = \begin{cases} -1, & n < -1; \\ n, & -1 \leq n \leq 1; \\ 1, & n > 1, \end{cases}$$
$$dsatlins(n) = \begin{cases} 0, & n < -1; \\ 1, & -1 \leq n \leq 1; \\ 0, & n > 1; \end{cases}$$

видати на екран інформацію про ці функції, побудувати їх графіки і розрахувати вектори виходу, скориставшись скриптом з М-файлу *hardlimfle*. Новий скрипт записати у файл під ім'ям *satlinsfle*.

Завдання 7. Для радіальної базисної функції активації *radbas* і її похідної *dradbas*, які визначаються співвідношеннями

$$radbas = e^{-2n};$$
$$dradbas = -2ne^{-2n},$$

видати на екран інформацію про ці функції, побудувати їх графіки і розрахувати вектори виходу, скориставшись скриптом з М-файлу *hardlimfle*. Новий скрипт записати у файл під ім'ям *radbasfile*.

Завдання 8. Для трикутної функції активації *tribas* і її похідної *dtribas*, які визначаються співвідношеннями

$$\begin{aligned} \text{tribas}(n) &= \begin{cases} 0, & n < -1; \\ 1 - \text{abs}(n), & -1 \leq n \leq 1; \\ 0, & n > 1, \end{cases} \\ \text{dtribas}(n) &= \begin{cases} 0, & n < -1; \\ 1, & -1 \leq n \leq 0; \\ -1, & 0 < n \leq 1; \\ 0, & n > 1; \end{cases} \end{aligned}$$

видати на екран інформацію про ці функції, побудувати їх графіки і розрахувати вектори виходу, скориставшись скриптом з М-файлу *hardlimfle*. Новий скрипт записати у файл під ім'ям *tribasfile*.

Завдання 9. Для логістичної функції активації *logsig* і її похідній *dlogsig*, які визначаються співвідношеннями

$$\begin{aligned} \text{logsig}(n) &= 1 / (1 + e^{-n}); \\ \text{dlogsig}(n) &= e^{-n} / (1 + e^{-n})^2, \end{aligned}$$

видати на екран інформацію про ці функції, побудувати їх графіки і розрахувати вектори виходу скориставшись скриптом з М-файлу *hardlimfle*. Новий скрипт записати у файл під ім'ям *logsigfile*.

Завдання 10. Для гіперболічної тангенціальної функції активації *tansig* і її похідної *dtansig*, які визначаються співвідношеннями

$$\begin{aligned} \text{tansig}(n) &= 2 / (1 + e^{-2n}) - 1; \\ \text{dtansig}(n) &= 1 - \text{tansig}^2(n), \end{aligned}$$

видати на екран інформацію про ці функції, побудувати їх графіки і розрахувати вектори виходу, скориставшись скриптом з М-файлу *hardlimfle*. Новий скрипт записати у файл під ім'ям *tansigfile*.

Завдання 11. Для конкуруючої функції активації *compnet*, яка використовується для формування ймовірнісних нейронних мереж, і таких, які самоорганізуються, виконати наступні дії:

1. Видати на екран інформацію про цю функцію за допомогою наступного скрипта:

```
Name = compet('name')      %-competitive;
Dname = compet('deriv')      % - ";
Inrange = compet('active')    % - -in : inf;
Outrange = compet('outrut') % - 0 1.
```

2. Побудувати стовпцеві діаграми для вектора входу і для вектора виходу, використовуючи шар з чотирьох нейронів:

```
N = [0; 1; -0.5; 0.5];
A = compet(n);
subplot(2,1,1), % - підвікна 2x1; вивід в 1-е;
bar(n), % - стовпцева діаграма;
ylabet('n')      %- мітка осі ординат;
subplot(2,1,2), bar (a), ylabet('a') %у 2-му під-
вікні.
```

3. Розглянуту послідовність команд оформити у вигляді скрипта в М-файл з ім'ям *competlile*.

Завдання 12. Виконати дії 11-го завдання для конкуруючої функції активації з м'яким максимумом *softmax*, записавши при цьому новий скрипт в М-файл з ім'ям *softmaxfile*.

Завдання 13. Для простого нейрона з одним подвійним входом P і функції активації *hardlim* підібрати ваговий коефіцієнт W і зсув b так, щоб забезпечити інвертування вхідного сигналу, тобто заміну нуля одиницею, а одиниці нулем.

Завдання 14. Для нейрона з двома подвійними входами p_1 і p_2 і функцією активації *hardlim* підібрати вагові коефіцієнти і зсув так, щоб нейрон виконував функції логічного додавання і логічного множення.

Завдання 15. Для нейрона з двома подвійними входами p_1 і p_2 і функцією активації *hardlim* підібрати вагові коефіцієнти W_{11} , W_{12} і зсув b так, щоб класифікувати вхідні подвійні набори на два класи – нульовий і перший:

а) {00, 01} – нульовий клас, {10, 11} – перший клас;

б) $\{11\}$ – нульовий клас, $\{00, 01, 10\}$ – перший клас;

в) $\{00, 11\}$ – нульовий клас, $\{01, 10\}$ – перший клас;

г) $\{00, 11\}$ – перший клас, $\{01, 10\}$ – нульовий клас.

Завдання 16. Для нейрона з двома безперервними входами p_1 і p_2 і функції активації *hardlim* побудувати графік розділяючої лінії, яка визначається рівнянням

$$W_{11}p_1 + W_{12}p_2 + b = 0,$$

вважаючи, що значення вагових коефіцієнтів W_{11} , W_{12} і зсуву b задані. Переко-
натися, що набори входів p_1 і p_2 по різну сторону від розділяючої лінії належать
різним класам і що не всяку множину наборів значень входів можна розділити
на два класи, використовуючи нейрон розглянутого типу.

ЛАБОРАТОРНА РОБОТА №11. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ

МЕТА РОБОТИ: вивчення архітектури ШНМ, способів їх графічного зображення у вигляді функціональних і структурних схем і програмного уявлення у вигляді об'єктів спеціального класу `network`, що включають масив структур з атрибутами мережі і набір необхідних методів для створення, ініціалізації, навчання, моделювання і візуалізації мережі, а також придбання навиків побудови мереж різної архітектури за допомогою інструментального програмного пакету `Neural Network Toolbox` системи `MATLAB`.

11.1. Програма роботи

11.1.1. Отримати завдання.

11.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

11.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

11.1.4. Оформити електронний звіт про роботу та захистити її.

11.2. Вказівки до виконання роботи

11.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 8.5 і записує його до звіту.

11.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 8.4.

11.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

11.2.4. Звіт має містити такі розділи:

- мету роботи та завдання з записаною умовою задачі;
- коди всіх використовуваних `.h` і `.cpp` файлів, а також пояснення до них;
- результати реалізації програми;
- діаграму класів та діаграму варіантів використання з поясненням;
- висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

11.3. Теоретичні відомості

Хоча окремі нейрони і здатні після деякої процедури навчання вирішувати ряд завдань штучного інтелекту, все ж таки для ефективного вирішення складних завдань по розпізнаванню образів, ідентифікації і класифікації об'єктів, розпізнаванню і синтезу мови, оптимальному управлінню застосовують достатньо великі групи нейронів, утворюючи з них штучні нейронні мережі у вигляді зв'язаних між собою шарів, що нагадують біологічні нейронні (нервові) мережі людини і тварин.

Існує безліч способів організації ШНМ, які можуть містити різне число шарів нейронів. Нейрони можуть бути зв'язані між собою як всередині окремих шарів, так і між шарами. Залежно від напрямку передачі сигналу ці зв'язки можуть бути прямими або зворотними. Шар нейронів, що безпосередньо приймає інформацію із зовнішнього середовища, називається вхідним шаром, а шар, що передає інформацію в зовнішнє середовище, – вихідним шаром. Решта шарів, якщо вони є в мережі, називається проміжними, або прихованими. У ряді випадків такого функціонального розподілу шарів мережі не проводиться, так що входи і виходи можуть приєднуватися до будь-яких шарів і мати довільне число компонент.

Структура, або архітектура ШНМ залежить від того конкретного завдання, яке вона повинна вирішувати і може бути одношаровою без зворотних зв'язків або із зворотними зв'язками, двошаровою з прямими зв'язками, тришаровою із зворотними зв'язками і так далі. Мережі із зворотними зв'язками називають рекурентними.

Опис архітектури ШНМ, крім вказування числа шарів і зв'язків між ними, повинен включати інформацію про кількість нейронів в кожному шарі, вигляд функції активації в кожному шарі, наявності зсувів для кожного шару, наявності компонент вхідних, вихідних і цільових векторів, а у ряді випадків і характеристики топології шарів. Наприклад, для апроксимації будь-якої функції з кінцевим числом точок розриву широко використовується мережа з прямою пере-

дачею сигналів. У цій мережі є декілька шарів з сигмоїдальними функціями активації. Вихідний шар містить нейрони з лінійними функціями активації. Дана мережа не має зворотних зв'язків, тому її називають мережею з прямою передачею сигналів (FF-net).

Графічно штучна нейронна мережа зображається у вигляді функціональної або структурної схеми. На функціональній схемі мережі за допомогою геометричних фігур зображаються її функціональні блоки, а стрілками показуються входи, виходи і зв'язки. У блоках і на стрілках вказуються необхідні позначення і параметри.

Структурна схема мережі зображається за допомогою типового набору блоків, сполучних елементів і позначень, прийнятих в інструментальному програмному пакеті Neural Network Toolbox системи MATLAB і пакеті імітаційного моделювання Simulink тієї ж системи. Системи позначень блоків, елементів і параметрів мережі є векторно-матричною, прийнятою в системі MATLAB. Якщо в позначенні використовується два індекси, то, як правило, перший індекс (індекс рядка) вказує адресата, або пункт призначення, а другий індекс (індекс стовпця) – джерело структурної схеми мережі. Структурні схеми створюються системою автоматично за допомогою команди `gensim`. Якщо елементом вектора або матриці на структурній схемі є складний об'єкт, то використовуються відповідно клітинка і масив клітин.

Програмним уявленням, або обчислювальною моделлю штучної нейронної мережі, є об'єкт спеціального класу `network`, який включає масив структур з атрибутами мережі і набір методів, необхідних для створення мережі, а також для її ініціалізації, навчання, моделювання і візуалізації. Клас `Network` має два загальні конструктори, один з яких не має параметрів і забезпечує створення масиву структур з нульовими значеннями полів, а другий – має мінімальний набір параметрів для створення моделі нейронної мережі, що добудовується потім до потрібної конфігурації за допомогою операторів привласнення. Для створення нейронних мереж певного вигляду використовуються спеціальні конструктори.

11.4. Індивідуальні завдання

Завдання 1. Створити обчислювальну модель нейронної мережі з двома виходами, трьома шарами і одним цільовим входом, використовуючи загальний конструктор мережі з параметрами

```
Net=network(numInputs, numLayers, biasConnect,
inputConnect, layerConnect, outputConnect, targetConnect).
```

Зв'язки між шарами повинні бути тільки прямими, входи необхідно з'єднати з першим шаром, а вихід – з останнім. Вихід повинен бути цільовим, перший шар повинен мати зсуви.

Сенс і значення параметрів конструктора для створення моделі мережі заданої архітектури такі:

```
numInputs=2 - кількість входів мережі;
numLayers=3 - кількість шарів в мережі;
biasConnect=[1; 0; 0] - матриця зв'язності для зсувів розміру numLayers * 1;
inputConnect=[1 1; 0 0; 0 0]- матриця зв'язності для входів розміру numLayers * numInputs;
layerConnect=[0 0 0; 1 0 0 0; 0 1 0] - матриця зв'язності для шарів розміру numLayers * numLayers;
outputConnect=[0 0 1]- матриця зв'язності для виходів розміру 1* numLayers;
targetConnect=[0 0 1] - матриця зв'язності для цілей розміру 1 * numLayers.
```

Порядок виконання завдань наступний:

1. Створити шаблон мережі, виконавши команду

```
net = network(2,3[1; 0; 0][11; 00;00],...,
[0 0 0; 1 0 0; 0 1 0],[0 0 1])
```

2. Перевірити значення полів обчислювальної моделі нейронної мережі net і їх відповідність заданим значенням в списку параметрів.

3. Перевірити значення обчислюваних полів моделі, які доповнюють опис архітектури мережі

```
numOutputs = 1 - кількість виходів мережі;
```

numTargets = 1 - кількість цілей мережі;

numInputDelays = 0 - максимальне значення затримки для входів мережі.

numLayersDelays = 0 - максимальне значення затримки для шарів мережі.

Відмітимо, що кожен вихід і кожна мета приєднуються до одного або декількох шарів, при цьому кількість компонент виходу або мети рівно кількості нейронів у відповідному шарі. Для збільшення можливостей моделі в мережу включають або на її входах, або між шарами лінії затримки. Кожна лінія затримує сигнал на один такт. Параметри *numInputDelays* і *NumLayerDelays* визначають максимальне число ліній для якого-небудь входу або шару відповідно.

4. Проаналізувати структурну схему побудованої мережі, виконавши команду *gensim(net)* і деталізуючи блоки за допомогою подвійного клацання лівої клавіші миші по даному блоку. На структурних схемах штучних нейронних мереж в пакеті NNT використовуються наступні позначення:

- а) *Neural Network* – штучна нейронна мережа з позначеннями входів $p\{1\}, p\{2\}, \dots$ і виходу $y\{1\}$;
- б) входи *Input1* або $p\{1\}$ і *Input2* або $p\{2\}$;
- в) дисплей $y\{1\}$;
- г) *Layer 1, Layer 2, Layer 3...* шари нейронів з позначеннями входів $p\{1\}, p\{2\}, a\{1\}, a\{2\}, \dots$ і виходів $a\{1\}, a\{2\}, a\{3\}, \dots, y\{1\}$;
- д) *TDL* – лінії затримки (*Time Delay*) з іменами *Delays1, Delays2...*, які забезпечують затримку вхідних сигналів або сигналів між шарами нейронів на **1, 2, 3...** такти;
- е) *Weights* – вагова матриця для вхідних сигналів або сигналів між шарами нейронів; розмір матриці ваг для кожного вектора входу $S \times R$, де S – число нейронів вхідного шару, а R – число компонент вектора входу, помножене на число затримок; розмір матриці для сигналів від шару j до шару i рівний $S \times R$, де S – число нейронів в шарі i , а R – число нейронів в шарі j , помножене на число затримок;

- ж) **dotprod** – блок зважування вхідних сигналів і сигналів між шарами, на виході якого виходить сума зважених, тобто помножених на відповідних ваги компонент сигналу;
- з) **mux** – концентратор вхідних сигналів і сигналів між шарами, перетворює набір скалярних сигналів у вектор, а набір векторів у один вектор сумарної довжини;
- и) **netsum** – блок підсумовування компонент для кожного нейрона шару: компонент від декількох векторів входу зі врахуванням затримок, зсуву і т. д.;
- к) **hardlim**, **purelin** і так далі – блоки функцій активації;
- л) **pd{l, 1}**, **pd{l, 2}**, **ad{2, 1}**... – сигнали після ліній затримки (**d** – **delay**);
- м) **iz{l, 1}**, **iz{l, 2}**, **lz{2, 1}**, **lz{3, 2}** – вектор-сигнали з виходу концентратора;
- н) **bias** – блок ваг зсувів для шару нейронів;
- о) **IW** – масив клітин з матрицями ваг входів: **IW{i, j}** – матриці для шару **i** від вхідного вектора **j**;
- п) **LW** – масив клітин з матрицями ваг для шарів: **LW{i, j}** – матриці для шару **i** від шару **j**.

5. Проаналізувати всі параметри кожного блоку структурної схеми даної нейронної мережі і у разі потреби звернутися до довідкової системи пакету NNT.

6. Задати нульові послідовності сигналів для входів $P = [0 \ 0; \ 0 \ 0]$ і провести моделювання мережі $A = \text{sim}(\text{net}, P)$.

7. Задати діапазони вхідних сигналів і вагові матриці за допомогою наступних привласнень:

```
net.inputs{1}.range = [0 1];
net.inputs{2}.range = [0 1];
net.b{1} = -1/4;
net.IW{1,1}=[0.5];
net.IW{1,2}=[0.5];
```

```
net.LW{2,1}=[0.5];
```

```
net.LW{3,2}=[0.5].
```

Виконати команду *gensim(net)* і перевірити параметри блоку.

8. Вивести на екран поля обчислювальної моделі і їх вміст, використовуючи функцію *celldisp*. Переконайтеся у правильності значень полів моделі.

9. Змодельовати створену статичну мережу, тобто мережу без ліній затримки, використовуючи групове і послідовне представлення вхідних сигналів

```
PG = [0.5 1 ; 1 0.5];
```

```
PS = {[0.5 1][1 0.5]};
```

```
AG = sim(net,PG);
```

```
AS = sim(net, PS).
```

Переконайтеся, що для статичної мережі групове і послідовне представлення вхідних сигналів дають один і той же результат.

10. Доповнити архітектуру створеної нейронної мережі лініями затримки для вхідних сигналів і для сигналів між 2-м і 3-м шарами, перетворивши таким чином статичну мережу на динамічну:

```
net.inputWeights{1, 1}.delays =[0 1];
```

```
net.inputWeights{1, 2}.delays =[0 1];
```

```
net.layerWeights{3, 2}.delays = [0 1 2].
```

Побудувати структурну схему динамічної мережі і з'ясувати сенс використовуваних операторів привласнення.

11. Скоректувати вагові матриці:

```
net.IW{1, 1} =[0.5 0.5];
```

```
net.IW{1,2}=[0.5 0.25];
```

```
net.LW{3,2}=[0.5 0.25 1].
```

12. Про моделювати динамічну мережу, використовуючи групове і послідовне представлення вхідних сигналів:

```
AG = sim(net, PG);
```

```
AS = sim(net,PS).
```

Переконайтеся, що групове представлення вхідних сигналів спотворює результат, оскільки в цьому випадку робота однієї мережі замінюється паралель-

ною роботою двох (по числу послідовностей) однакових мереж з нульовими початковими значеннями сигналів на виходах ліній затримки.

13. Вивести на друк поля обчислювальної моделі і їх вміст, використовуючи функцію *celldisp*.

14. Зберегти вміст командного вікна в М-файлі для подальшого використання.

Завдання 2. Створити таку саму динамічну мережу *asgnet*, використовуючи конструктор класу *network* без параметрів і задаючи значення відповідних полів обчислювальної моделі з допомогою операторів привласнення. Перекоонатися в ідентичності мереж *net* і *asgnet*. Порівняти результати роботи отриманих мереж.

Завдання 3. Використовуючи блоки імітаційного моделювання інструментального пакету Simulink системи MATLAB, побудувати модель динамічної мережі *asgnet*, провести дослідження моделі, перевірити адекватність її поведінки поведінці моделі *net* і оформити електронний звіт за допомогою генератора *Report Generator*.

Завдання 4. Використовуючи конструктор класу *network* з параметрами і оператори присвоєння для полів і клітин об'єктів цього класу, побудувати, вивести на екран і змодельовати штучні нейронні мережі наступної архітектури:

- а) одношарова мережа з трьома нейронами, трьома двокомпонентними входами і одним цільовим виходом;
- б) тришарова мережа з прямою передачею сигналів і з трьома нейронами в кожному шарі; кількість входів – три з двома, п'ятьма і трьома компонентами; для всіх шарів є зсув; вихід – один;
- в) тришарова мережа, в якій кожен шар сполучений зі всіма останніми; вхід – один і складається з двох компонентів; кількість нейронів в кожному шарі – три; шари мають зсуви;
- г) тришарова динамічна мережа з трьома нейронами в кожному шарі; число входів – три, кожен із них складається з трьох компонентів; є зсуви на всіх шарах; лінії затримки затримують сигнали на один і два такти і

включені між всіма шарами, а також на вході;
квадратна мережа з десятима шарами і десятима нейронами в кожному шарі;
десять векторів підключаються поодинці до кожного шару; є десять виходів від
всіх шарів мережі; зсуви підключені до кожного шару.

