МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА"

ОСНОВИ ПОБУДОВИ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт та самостійної роботи
для здобувачів вищої освіти
за освітньою програмою "Комп'ютерна інженерія"
(освітній ступінь бакалавр)

Обговорено і рекомендовано
на засіданні кафедри
інформаційних та комп'ютерних
систем
Протокол № 5
від 2 травня 2024 р.

Чернігів 2024

Основи побудови систем штучного інтелекту. Методичні вказівки до лабораторних робіт та самостійної роботи для здобувачів вищої освіти за освітньою програмою “Комп’ютерна інженерія” (освітній ступінь бакалавр) / Укл. Бичко В.А. Бівойно П.Г – Чернігів: Чернігів: НУ «Чернігівська політехніка», 2024. – 43 с., укр. мовою.

Укладачі: БИЧКО ВОЛОДИМИР АНАТОЛІЙОВИЧ, кандидат фізико-математичних наук, доцент кафедри інформаційних та комп’ютерних систем
БІВОЙНО ПАВЛО ГЕОРГІЄВИЧ, кандидат технічних наук, доцент кафедри інформаційних та комп’ютерних систем

Відповідальний за випуск: Базилевич Володимир Маркович, завідувач кафедри інформаційних та комп’ютерних систем Національний університет “Чернігівська політехніка”, кандидат економічних наук, доцент

Рецензент: Пріла Ольга Анатоліївна, кандидат технічних наук, доцент кафедри інформаційних та комп’ютерних систем

ЗМІСТ

Вступ

Дисципліна "Основи побудови систем штучного інтелекту" відноситься до розряду дисциплін по вибору здобувачів вищої освіти за освітньою програмою “Комп’ютерна інженерія” (освітній ступінь бакалавр).

Необхідною передумовою для освоєння даної дисципліни є знання студентами таких навчальних курсів, як "Програмування".

Вивчення дисципліни сприяє більш глибокому розумінню інженерних задач у сфері Основи побудови систем штучного інтелекту.і освоєнню сучасних методів їх застосування.

1 Лабораторна робота №1

МЕТОДИ ПОШУКУ У ПРОСТОРІ СТАНІВ

Мета - отримати теоретичні та практичні навички по роботі з методами пошкуку розв‘язків задач.

1.1 Теоретичні відомості

Розрізняють локальний та системний підходи до подання інтелектуальних задач.

Локальний або «задачний» підхід заснований на точці зору, що для кожної задачі, властивої творчій діяльності людини, можна знайти спосіб її вирішення на електронній обчислювальній машині (ЕОМ), який, будучи реалізований у вигляді програми, дає результат, або подібний результату, отриманому людиною, або навіть кращий.

Системний або заснований на знаннях підхід пов’язаний із уявленням про те, що розв’язання окремих творчих задач не вичерпує всієї проблематики ШІ. Природний інтелект людини здатний не лише розв’язувати творчі завдання, а за потреби навчається того чи іншого виду творчої діяльності. Тому і програми ШІ повинні бути орієнтовані не лише або не стільки на вирішення конкретних інтелектуальних задач, скільки на створення засобів, що дозволяють автоматично будувати програми вирішення інтелектуальних задач, коли в таких програмах виникне потреба.

У такому підході проблема створення інтелектуальних систем розглядається як частина загальної теорії програмування. При цьому підході для складання інтелектуальних програм використовуються звичайні програмні засоби, що дозволяють писати потрібні програми за описами задач професійною природною мовою. Усі метазасоби, що виникають при цьому на базі часткового аналізу природного інтелекту, розглядаються тут лише з точки зору створення інтелектуального програмного забезпечення, тобто комплексу засобів, що автоматизують діяльність самого програміста.

Оскільки поняттям знання про задачи можуть відповідати стани задачи, а правилам виведення - оператори переходу з одного стану в інший від задачи до підзадачі, то процедура пошуку рішення називається *пошуком у просторі станів.*

Пошук рішень у просторі станів зводиться до визначення послідовності операторів, які відображають початкові стани в цільові. Причому якщо така послідовність не одна й задано критерій опти-

мальності, то пошук зводиться до визначення оптимальної послідовності операторів, які забезпечують оптимум заданого критерію оптимальності.

Методи пошуку рішень у просторі станів зручно розглянути, використовуючи дерево (граф) станів. На дереві станів (рис. 1.1) пошук рішення зводиться до визначення шляху (оптимального, якщо задано критерій опти¬мальності) від кореня дерева до цільової вершини A, тобто до вершини, яка відповідає цільовому стану. Вершини B і C є вершинами, що розкриваються (обчислюваними, проміжними). Вершина D термінальна, тобто завершальна. Ребра, що з'єднують вершини, означають приєднані процедури, які необхідно виконати, щоб перейти до наступного стану.

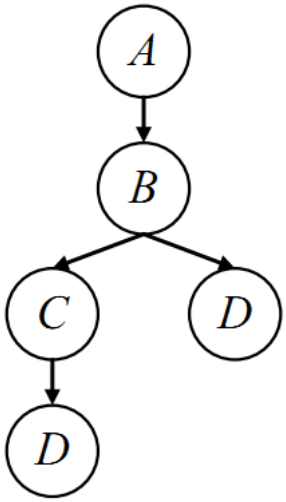

Рис. 1.1. Приклад дерева станів

Процес застосування приєднаних процедур називають породженням вершин або *перебором варіантів.* При породженні нової вершини обов'язково запам'ятовується покажчик на стару. У кінці перебору сукупність цих покажчиків утворює шлях вирішення задачі, який записується разом із іменами виконаних приєднаних процедур.

Для знаходження рішення необхідно знову і знову вибирати, перевіряти й розкривати вузли доти, поки не буде знайдено розв'язок або не залишиться більше станів, які можна було б розгорнути. Порядок, у якому відбувається розгортання станів, визначається *стратегією пошуку.*

Результатом застосування будь-якого алгоритму вирішення задачі є або невдале завершення, або отримання рішення. Деякі алгоритми можуть входити до нескінченого циклу й не повертати ніякого результату. Продуктивність алгоритму оцінюється за допомогою чотирьох показників:

- *повнота* дає відповідь на запитання, чи гарантує алгоритм виявлення рішення, якщо воно є;
- *оптимальнсть* відповідає за те, чи забезпечує дана стратегія знаходження оптимального рішення (тобто такого, яке має найменшу вартість шляху серед всіх інших рішень);

- *затрачений час,* за який алгоритм знаходить рішення;
- *необхідні ресурси,* тобто який обсяг пам’яті необхідний для здійснення пошуку.

Методи пошуку в одному просторі призначені для використання за таких умов:

- області невеликої розмірності,
- повнота моделі,
- точні та повні дані.

Стратегії пошуку в одному просторі можна класифікувати так:

1)Неінформований пошук («сліпі» методи):

- в ширину;
- в глибину (з обмеженням глибини, з ітеративним поглибленням).

2)Інформований пошук (евристичний пошук).

Пошук може здійснюватись у різних напрямках.

Прямий пошук йде від вихідного стану і, як правило, використовується тоді, коли цільовий стан задано неявно.

Зворотний пошук йде від цільового стану і використовується тоді, коли початковий стан задано неявно, а цільовий - явно.

Двонаправлений метод пошуку дозволяє одночасно проводити два пошуки в прямому і у зворотному напрямку, зупиняючись після того, як два процеси пошуку зустрінуться на середині. У такому разі передбачається перевірка в одному або в обох процесах пошуку кожного вузла перед його розгортанням для визначення того, чи не знаходиться він на периферії іншого дерева пошуку. У разі позитивного результату перевірки рішення знайдено і пошук припиняється.

Переваги цього методу:

- незначний час пошуку, оскільки перевірка приналежності вузла до іншого дерева пошуку може бути виконана за сталий час за допомогою хеш-таблиці;
- повнота методу;
- оптимальність.

Серед недоліків слід відзначити значну витрату пам’яті, оскільки необхідно зберігати принаймні одне з дерев пошуку, для того, щоб можна було виконати перевірку приналежності до іншого дерева.

На рис. 1.2 показано схематичне зображення двонаправленого пошуку в тому стані, коли він має успішно закінчитися після того, як одна з гілок, яка виходить із початкового вузла, зустрінеться з гілкою, що виходить із цільового вузла.

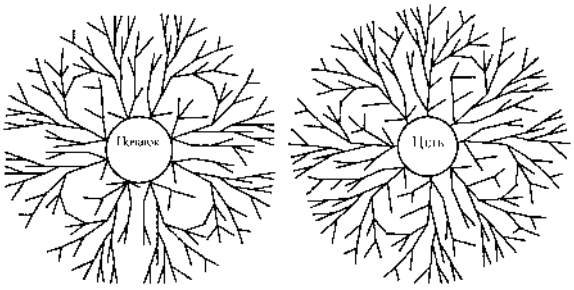

Рис. 1.2. Приклад двонаправленого пошуку

1.1.1 Методи «сліпого» пошуку

Методи *неінформованого* або «сліпого» пошуку (повного перебору) означають, що в даних стратегіях не використовується додаткова інформація про стани, крім тієї, яка подана у задачі. Такі

стратегії можуть лише виробляти наступників і відрізняти цільовий стан від нецільового. Потребують великої витрати часу.

Пошук *у ширину* є досить простою стратегією, в якій спочатку розгортається кореневий вузол, потім - усі його наступники (рис. 2.3). Після цього розгортаються наступники цих наступників і т. д. Тобто перш ніж відбувається розгортання будь- яких вузлів на наступному рівні, розгортаються всі вузли на даній конкретній глибині в дереві пошуку. Перевага - повнота. Недоліки: неоптимальний, значні витрати часу та пам’яті, оскільки необхідно зберігати всі проміжні значення.

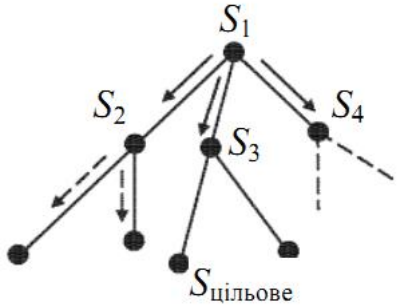

Рис. 1.3. Дерево пошуку в ширину

Пошук *у глибину* завжди розгортає найглибший вузол у дереві пошуку (рис. 1.4). Пошук безпосередньо переходить на найглибший рівень дерева пошуку, на якому вузли не мають наступників.

У міру того, як ці вузли розгортаються, вони видаляються з периферії, тому надалі пошук «відновлюється» з наступного поверхневого вузла, який все ще має недосліджених наступників.

Перевага - незначні потреби в пам’яті (зберігання лише єдиного шляху від кореня до листового вузла/

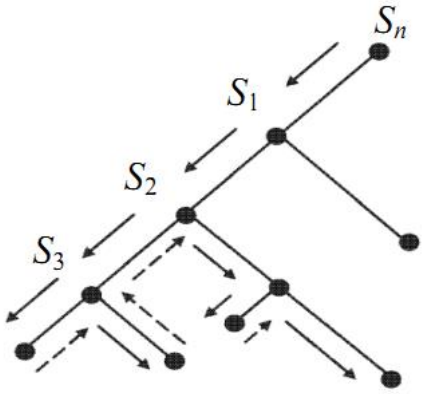

Рис. 1.4. Дерево пошуку

Недолік: може бути зроблений неправильний вибір і перехід у тупикову ситуацію, пов'язану з проходженням вниз по дуже довгому (чи навіть нескінченному) шляху, при тому, що інший варіант міг би привести до рішення, яке знаходиться недалеко від кореня дерева пошуку.

Пошук *із обмеженням глибини* передбачає застосування під час пошуку заздалегідь певної межі глибини I. Це дозволяє вирішити проблему необмежених дерев (тобто нескінченного шляху). Вузли на глибині *I* розглядаються як такі, що не мають наступників. Однак при неправильному виборі $I < d$, коли поверхнева ціль виходить за межі глибини, рішення не буде знайдено. Така ситуація цілком ймовірна, якщо значення *d* невідомо. Крім того, пошук із обмеженням глибини буде неоптимальним при виборі значення $I > d$.

Пошук у глибину *з ітеративним поглибленням* передбачає поступове збільшення глибини пошуку (яка спочатку дорівнює 1, потім 2, 3 і т. д.) доти, поки не буде знайдено ціль. Така подія відбувається після того, як межа глибини досягає значення d, глибини поверхневого цільового вузла.

У пошуку з ітеративним поглибленням поєднуються переваги пошуку в глибину та пошуку в ширину. Основні переваги методу:

незначні вимоги до пам'яті, як у пошуку в глибину;

повнота, як і в пошуку в ширину, якщо коефіцієнт розгалуження кінцевий;

оптимальність, якщо вартість шляху становить неспадну функцію глибини вузла.

Дана стратегія може здатися занадто витратною, оскільки одні й ті ж стани формуються декілька разів. Але, як виявилося, такі повторні операції не є надто дорогими. Причина цього полягає в тому, що в дереві пошуку з одним і тим же (або майже одним і тим же) коефіцієнтом розгалуження на кожному рівні більшість вузлів перебуває на нижньому рівні, тому не має великого значення те, що вузли на верхніх рівнях формуються багаторазово. У пошуку з ітеративним поглибленням вузли на нижньому рівні (з глибиною *d)* формуються один раз. Вузли, які перебувають на

рівні, що передує нижньому, формуються двічі і т. д., до дочірніх вузлів кореневого вузла, які формуються *d* разів.

1.2 Завданння до виконання роботи

1. Ознайомитися з теоретичними відомостями, розглянути та засвоїти описані тут методи пошуку у просторі станів.
2. За допомогою будь-якого середовища розробника реалізувати програмний модуль (ПМ), що виконує пошук цільової вершини у просторі станів. При цьому вхідний граф довільної конфігурації повинен бути заданим у формі матриці суміжності.
3. Протестувати роботу створеної Вами програми на 3-х різних початкових графах Провести аналіз роботи.
4. Зафіксувати результати роботи ПМ, опис та код ПМ у звіті.

1.3 Варіанти завдань до виконання роботи

№ варианту	Завдання
1	Реалізувати програму, що здійснює сліпий зворотній пошук початкової вершини в глибину на орієнтованому графі, (що містить 20 вершин).Тестові графи створити самостійно.
2	Реалізувати програму, що здійснює сліпий прямий пошук цільової вершини в глибину на орієнтованому графі, (що містить 15 вершин).Тестові графи створити самостійно.
3	Реалізувати програму, що здійснює сліпий двонаправлений пошук оптимального шляху між початковою та кінцевою вершинами в глибину на неорієнтованому графі, (що містить 30 вершин) вглибину..Тестові графи створити самостійно.
4	Реалізувати програму, що здійснює сліпий зворотній пошук початкової вершини в ширину на орієнтованому графі, (що містить 20 вершин).Тестові графи створити самостійно.
5	Реалізувати програму, що здійснює сліпий прямий пошук цільової вершини в ширину на орієнтованому графі, (що містить 15 вершин).Тестові графи створити самостійно.
6	Реалізувати програму, що здійснює сліпий двонаправлений пошук оптимального шляху між початковою та кінцевою вершинами в ширину на неорієнтованому графі, (що містить 30 вершин).Тестові графи створити самостійно.
7	Реалізувати програму, що здійснює спрямований (евристичний) зворотній пошук початкової вершини в глибину на орієнтованому графі, (що містить 20 вершин).Тестові графи створити самостійно.

8	Реалізувати програму, що здійснює спрямований (евристичний) прямий пошук цільової вершини в глибину на орієнтованому графі, (що містить 15 вершин).Тестові графи створити самостійно.
9	Реалізувати програму, що здійснює спрямований (евристичний) двонаправлений пошук оптимального шляху між початковою та кінцевою вершинами в глибину на неорієнтованому графі, (що містить 30 вершин) .Тестові графи створити самостійно.
10	Реалізувати програму, що здійснює спрямований (евристичний) зворотній пошук початкової вершини в глибину на орієнтованому графі, (що містить 20 вершин) .Тестові графи створити самостійно.
11	Реалізувати програму, що здійснює спрямований (евристичний) прямий пошук цільової вершини в ширину на орієнтованому графі, (що містить 15 вершин).Тестові графи створити самостійно.
12	Реалізувати програму, що здійснює спрямований (евристичний) двонаправлений пошук оптимального шляху між початковою та кінцевою вершинами в ширину на неорієнтованому графі, (що містить 30 вершин).Тестові графи створити самостійно.
13	Реалізувати програму, що здійснює сліпий зворотній пошук початкової вершини в глибину на неорієнтованому графі, (що містить 20 вершин).Тестові графи створити самостійно.
14	Реалізувати програму, що здійснює сліпий прямий пошук цільової вершини в глибину на неорієнтованому графі, (що містить 15 вершин).Тестові графи створити самостійно.
15	Реалізувати програму, що здійснює сліпий зворотній пошук початкової вершини в ширину на неорієнтованому графі, (що містить 20 вершин).Тестові графи створити самостійно.
16	Реалізувати програму, що здійснює сліпий прямий пошук цільової вершини в ширину на неорієнтованому графі, (що містить 15 вершин).Тестові графи створити самостійно.

1.4 Завдання для самостійної роботи

1. Складіть дерево пошуку шляху від вершини А до вершини З (рис. 1.5) за стратегією в глибину.

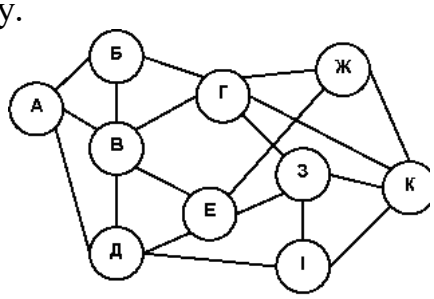

Рис. 1.9 Тестовий граф

Побудуйте дерево пошуку шляху від вершини А до вершини Е (див. рис. 1.5) за стратегією в ширину.

Побудуйте дерево пошуку шляху від вершини А до вершини К (див. рис. 1.5) за стратегією двонаправленого пошуку.

Намалюйте дерево пошуку за стратегією «в глибину» для задачі про комівояжера (див. рис. 1.5).

Знайдіть мінімальний шлях від вершини А до вершини К (див. рис. 1.5).

1.1.2 Контрольні запитання та завдання для самоперевірки

- Чим відрізняються системний та локальний підходи до подання інтелектуальних задач?
- Як реалізується пошук у просторі станів?
- Опишіть стратегію пошуку в глибину та в ширину.
- Які існують модифікації методу пошуку в глибину?
- У чому полягає відмінність евристичного пошуку від неін- формованого?
- Проаналізуйте прямий, зворотний та двонаправлений напрямки пошук.
- Як проводиться пошук рішення задачі на графі редукції?

Опишіть стратегії «сліпого» та евристичного пошуку для дерева типу «І-АБО

1.5 Вимоги до звіту

Звіт повинен містити:

- Титульну сторінку з даними про виконавця і перевіряючого, а також номер варіанта, тему і мету роботи.

-Лістинг програми.

-Результати застосування створеного ПМ до тестового графа.

-Висновки за результатами лабораторної роботи.

Звіт повинен бути оформлений відповідно до вимог СОКР.

2 Лабораторна робота №2

ПОШУК ОПТИМАЛНОГО РІШЕННЯ ЗА ДОПОМОГОЮ ГЕНЕТИЧНОГО АЛГОРИТМУ

Мета роботи: отримати навички розв'язання практичних задач за допомогою генетичних алгоритмів

2.1 Теоретичні відомості

Генетичні алгоритми (ГА) (Holland, 1969-1990) спрощено моделюють процеси природної еволюції і засновані на стохастических принципах.

Генетичні алгоритми зводяться до виконання наступних етапів:

1. Ініціалізувати популяцію.
2. Обчислити значення критерію якості для кожної особини популяції.
3. Виконати процес відтворення для кожної особини популяції.
4. Виконати схрещування і мутацію для кожної особини популяції.
5. Перевірити умову завершення. Якщо її не виконано, то повернутися до п. 2,.

Реалізація ГА зводиться до операцій з рядками: копіювання рядків, заміни фрагментів рядків і інверсії бітів.

Приклад.

Знайти

$$\max_{o \le x \le 255} f(x), \quad \text{де} \quad f(x) = \sin(\frac{\pi x}{256}),\ x \in Z\ . \ (1)$$

Функція залежить від однієї цілочисельної змінної. Особини популяції доцільно представити у вигляді бінарного рядка довжиною 1 байт.

Таблиця 7.1. Значення при ініціалізації

0	00000000
1	00000001
2	00000001
3	00000001
...	
255	11111111

Число особин однієї популяції в реальних задачах зазвичай складає 10–100. У даній задачі виберемо 8.

1. *Ініціалізація* — за допомогою генератора випадкових чисел у кожній з 8 позицій кожного рядка встановимо або 0 або 1.

Результати ініціалізації наведено в табл. 7.2.

Таблиця 7.2. Значення при ініціалізації

Особи	*x*	*fx*	*fnorm*
10111101	189	0.733	0.144
11011000	216	0.471	0.093
01100011	99	0.937	0.184
11101100	236	0.243	0.048
10101110	174	0.845	0.166
01001010	74	0.788	0.155
00100011	35	0.416	0.082
00110101	53	0.650	0.128

2. Обчислення значення критерію якості . В даномиу випадку це нормоване значення заданої функції

$$f_{norm}(x_i) = \frac{f(x_i)}{\left|\sum_{i=1}^{N} f(x_i)\right|}, \; i = \overline{1, N}.$$

1. Формування нової популяції з тим же числом особин. При формуванні нової популяції використовується принцип рулетки (рис. 7.1).

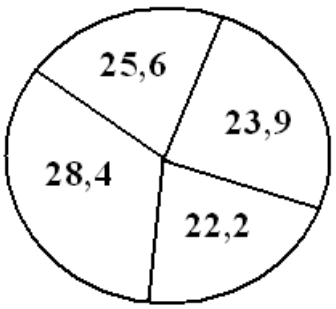

Рисунок. 7.1. Співвідношення ймовірності за критерієм якості

Результати застосування принципу рулетки показано в табл. 7.3.

Таблиця 7.3. Показники при формуванні покоління за принципом рулетки

N	Рядок	Значення	Крит. якості	% співвідн.
1	10111101	189	0,733	22,2
2	1100011	99	0,937	28,4
3	10101110	174	0,845	25,6
4	1001010	74	0,788	23,9
Разом:			3,303	100

Ймовірність влучення в кожний із сегментів пропорційна його величині.

Генеруються $N = 8$ випадкових значень з діапазону [0,1]:

$$r_i \in [0,1], i = \overline{1, N}.$$

Якщо

$$r_i \in \left[\sum_{j=1}^{i-1} f_{norm}(x_j), \sum_{j=1}^{i} f_{norm}(x_j) \right],$$

тоді

$$x \in G_{new}.$$

Наприклад, якщо $r_i \in [0, 0.144]$, то в нову популяцію включається x_1. Якщо $r_i \in [0.144, (0.144+0.093)=0.237]$, то x_2 включається в нову популяцію.

Таким чином максимальна імовірність включення в нову популяцію особин з максимальним значенням критерію якості.

Візьмемо набір з 8 випадкових чисел:

0.293, 0.971, 0.160, 0.469, 0.664, 0.568, 0.371, 0.109.

Індекси особин першої популяції, що ввійдуть у наступне покоління: 3, 8, 2, 5, 6, 5, 3, 1.

Після відтворення популяція матиме вигляд:

Таблиця 7.4. Значення особин-батьків, вибраних для схрещування

01100011
00110101
11011000
10101110
01001010
10101110
01100011
10111101

4. Схрещування — основна риса генетичного алгоритму полягає в обміні частин двох батьківських особин.

(а) Вибирається ймовірність (приблизно 0.65–0.80) того, що між двома батьками відбудеться схрещування (виберемо $p_c = 0.75$).

(б) Популяція випадковим чином розбивається на пари. Для будь-якої пари генерується випадкове число:

$$r_k \in [0,1], \quad k = \overline{0, \frac{N}{2}}.$$

Якщо

$$r_k < p_c < 0.75,$$

то пари піддаються схрещуванню.

(в) Для кожної з пар, що підлягають схрещуванню, випадковим чином задаються два числа (або одне число для одноточкового схрещування), що визначають границі рядка для обміну (табл. 7.3).

Таблиця 7.5. Значення при схрещуванні

Батьківська популяція	Нове покоління	x	$f(x)$
011-000-11	1110111	119	0.999
011-101-01	100001	33	0.394
1-1011-000	10101000	168	0.882
1-0101-110	11011110	222	0.405
01-00101-0	010001010	138	0.998
10-10111-0	1101110	110	0.976
01100011	01100011	99	0.937
10111101	10111101	189	0.733
Оптимальне значення		10000000	128

(г) Мутація — інвертування випадково обраних бітів (зазвичай з постійною імовірністю для кожного біта популяції, приблизно рівною 0.001–0.01).

Таким чином будь-який біт інвертується з ймовірністю 0.1%–1%. Оскільки в наведеному прикладі число бітів у популяції складає 64, то при ймовірності мутації p_m=0.001 або 0.01 швидше за все жоден біт не змінить значення.

Тепер двом особинам нового покоління відповідає значення критерію якості >0.99.

5. Перехід до нової ітерації.

2.2 Порядок виконання роботи

1. Ознайомитися з геометричним методом вирішення завдань лінійного програмування.

2. Письмово розрахуйте одну ітерацію із зазначенням всіх параметрів і проміжних результатів для задачи, що відповідає вашому варіанту.

3. Реалізувати генетичний алгоритм, пристосований для розв‘язання задачи що відповідає вашому варіанту. Доповніть створену програму графічними засобами контролю коректності роботи алгоритму.

4. За допомогою створеної програми розв’язати задачу згідно з номером вашого варіанту (розділ 7.3).

5. Результати роботи оформити звітом, який має містити: Варіанти завдань.

2.3 Варіанти завдань до виконання роботи

Враховуючи , що n – кількість особин популяції, pc – ймовірність схрещування, pm – ймовірність мутації Реалізуйте генетичний алгоритм для розв'язання задачі максимізації функції:

Варіант	Функція критерія якості	n	pc	pm
1	$f(x)=-(x-1)^2/256$, $x \in [0, 255]$	8	0.61	0.010
2	$f(x)=-(x^2-3x+2)/256$, $x \in [0, 255]$	10	0.62	0.025
3	$f(x)=-(x^2-4x+15)^2/256$, $x \in [0, 255]$	12	0.75	0.001
4	$f(x)=-(x^2-4x+3)/256$, $x \in [0, 255]$	14	0.68	0.005
5	$f(x)=-(x^2-6x+19)/256$, $x \in [0, 255]$	16	0.72	0.010
6	$f(x)=-(2x^2-5x+13)^2/256$, $x \in [0, 255]$	18	0.64	0.025
7	$f(x)=-(6x^2-5x-1)^2/256$, $x \in [0, 255]$	20	0.60	0.001
8	$f(x)=-(4x^2-4x+2)^2/256$, $x \in [0, 255]$	22	0.75	0.001
9	$f(x)=-(3x^2-15)^2/256$, $x \in [0, 255]$	24	0.62	0.005
10	$f(x)=-(17x^2-14x+15)^2/256$, $x \in [0, 255]$	26	0.75	0.010
11	$f(x)=-(x^2-5x+13)^2/256$, $x \in [0, 255]$	28	0.62	0.015
12	$f(x)=-(6x^2-4x-1)^2/256$, $x \in [0, 255]$	30	0.75	0.025
13	$f(x)=-(2x^2-4x+2)^2/256$, $x \in [0, 255]$	32	0.62	0.001
14	$f(x)=-(3x^2-12)^2/256$, $x \in [0, 255]$	34	0.75	0.001
15	$f(x)=-(x^2-4x+2)/256$, $x \in [0, 255]$	36	0.62	0.005
16	$f(x)=-(x^2-4x+14)^2/256$, $x \in [0, 255]$	38	0.75	0.010

2.4 Вимоги до звіту

Звіт повинен містити:

- Титульну сторінку з даними про виконавця і перевіряючого.
- Порядковий номер, номер варіанта, тему і мету роботи.
-Постановку задачі та опис послідовності дій при виконанні генетичного алгоритму.
- Письмовий варіант розрахунку однієї ітерації із зазначенням всіх параметрів і проміжних результатів для задачи, що відповідає вашому варіанту.
- Лістинг і інтерфейс програми з промізними (початкова популяція, таблиця розрахунків для функії пристосованості, особині для кросоверу, результиати кросоверу, результати мутації. Трекба представити 5

поколінь.) та кінцевими результатами її роботи для задачи, що відповідає вашому варіанту.

- Висновки про виконання роботи.

Звіт повинен бути оформлений відповідно до вимог ДСТУ

3 Лабораторна робота №3

Знайомство з інструментальними засобами для створення експертних систем.

Мета - ознайомлення з оболонкою "GURU" для створення експертних систем з використанням діалогового режиму роботи і коректування бази знань існуючої експертної системи.

3.1 Теоретичні відомості

Під експертною системою розуміється система, що об'єднує можливості комп'ютера зі знаннями і досвідом експерта так, що система може запропонувати розумну пораду або здійснити розумне рішення поставленої задачі. Додатковою можливістю системи є здатність пояснити хід своїх міркувань у зрозумілій для запитувача формі.

При створенні своїх користувальницьких експертних систем на якій-небудь мові високого рівня програміст стикається з тим, що розробка інтерфейсу програми, реалізація її системних функцій вимагають більших витрат часу, ніж створення самого набору правил експертної системи (ЕС). Для того щоб розвантажити розробника ЕС від такої роботи, існують спеціальні інструментальні засоби (оболонки) експертних систем. Такі інструментальні засоби є в ЕС MYCIN, GURU, LEONARDO, DENDRAL та ін.

Цей лабораторний практикум пов'язаний з освоєнням оболонки "GURU".

1.1.1 Режими роботи

"GURU" має три режими роботи:

- Діалоговий: у ході діалогу типу "запит-відповідь" за допомогою розвиненої системи меню, не вдаючись до написання власних програм, користувач створює експертну систему;

- Природна мова: користувач на запит системи вводить фрази природною мовою і отримує результати. Наприклад, система запитує "Ваш запит?". Написавши в командному рядку фразу "Знайти всіх працюючих 1967 народження", користувач отримує від системи розумний відповідь;

- Командний: як в мовах високого рівня (МВР), пишеться програма, компілюється і працює відповідно до ваших вимог.

Звичайно застосовуються змішані режими.

3.1.1 Характеристики ЕС

Основними характеристиками є: інтерфейс користувача, машина логічних висновків і збережені експертизи.

Інтерфейс користувача описує відносини між користувачем і системою. Користувач ставить задачу, а машина должка її виконати або пояснити, чому не можна її виконати.

Машина логічних висновків - це програмне забезпечення (ПЗ), яке можна використовувати в рішенні задач шляхів аргументації.

Збережені експертизи - це набір правил, що відображають знання. У кожному правилі є посилка (IF) і висновок (THEN).

Якщо машина логічних висновків визнає посилку вірної, ТО і висновок буде вірним.

1.1.2 Оперативна допомога

Перебуваючи в будь-якому меню, можна отримати підказку по діям, допустимим в цьому меню. Для цього викликається допомога одночасним натисканням <Ctrl-L>.

1.1.3 Правила "GURU"

Система "GURU" базується на правилах. Правило складається з посилки (IF) і висновку (THEN). Посилка може включати:

- □ різні типи і види змінних, підтримуваних "GURU";
- □ -логічні оператори (EQ, NE, GT, GE, LT, LE, IN, AND, OR, XOR, NOT);
- □ числові оператори (+, -, /, *, **);
- □ числові функції (SIN, COS і т.д.);
- □ символьні функції.

Висновок може включати команди:

- □ присвоєння значення різним змінним;
- □ дозвол проконсультуватися з іншим набором правил;
- □ різні команди "GURU" і т.д.

Правила зберігаються в звичайному текстовому файлі.

приклад:

```
RULESET: EASYCALC
GOAL: INTRATE
RULE: R1
IF: MONTHPAY <50
THEN: PERIOD = 120
RULE: R2
IF: PERIOD> 90
THEN: INTRATE = 12.5
```

Тут EASYCALC - ім'я набору правил (RULESET вказувати не обов'язково);

INTRATE - ім'я змінної цілі;

R1, R2 - імена правил;

PERIOD, INTRATE, MONTHPAY - змінні.

3.1.2 Прямой висновок

Одне з важливих питань для ЕС - яке правило розглядати наступним. Цим процесом керує машина логічних висновків (МЛВ).

При виборі правила потрібно користуватися двома основними стратегіями управління: прямим і зворотним висновками.

Даний метод діє від посилки до дії до тих пір, поки змінної не буде присвоєно значення. Машина логічних висновків починає переглядати набір правил спочатку і проводить перегляд його до тих пір, поки змінній не буде присвоєно значення. Спочатку шукається перше правило, в якому й визначено справжне значення посилки. Це правило буде виконано, і отриманий результат можна буде використовувати для тестування інших правил. Далі система шукає наступне правило з певним і справжнім значенням посилки. Це продовжується до тих пір, поки не буде виконано правило для змінної цілі.

приклад:

RULESET: EASYCALC
GOAL: INTRATE
RULE: R1
IF: RERIOD> 90
THEN: INTRATE = 12.5
RULE: R2
IF: MOMTHPAY <50
THEN: PERIOD = 120
RULE: R3
IF: MOHTHPAY> 50
THEN: PERIOD = 60
RULE: R4
IF: PERIOD <90
THEN: INTRATE = 11.0

Нехай спочатку змінної MONTHPAY присвоєно значення 42. МЛВ шукає в наборі правил то правило, де визначено і справжне значення посилки (це R2). Тоді змінної PERIOD присвоюється значення 120. Слідом за тим, починаючи знову з першого правила, шукається правило, в якому визначено й справжнє значення посилки (це R1). Змінній цілі присвоюється значення 12.5. Мета досягнута, система закінчила роботу.

Спробуйте пояснити, що вийде, якщо MONTHPAY = 70.

3.1.3 Зворотний висновок

Зворотній висновок - найбільш часто використовуваний метод управління. При цьому МЛВ починає з цілі і, переглядаючи набір правил, знаходить перше правило, за допомогою якого можна досягти цілі. Якщо посилка цього правила визначена і вірна, то система виконує відповідні дії. Якщо посилка невизначена, то МЛВ тимчасово змінює мету - встановлює в якості цілі змінну, яка дозволить визначити істинність першої знайденої посилки і шукає перше правило, визначних і вірне для нової поставленої цілі.

Скористаємося прикладом з попереднього прикладу. У цій ЕС мета (GOAL) -знайти INTRATE. Шукаємо перше правило, в якому обчислювалася б змінна цілі (це R1). Але його не можна виконати, поки невідома PERIOD. Шукаємо правило, де знаходиться PERIOD (це R2). Припустимо, що MONTHPAY задано і одно 42. Тоді виконується R2 і потім R1. Мета досягнута.

Але тепер припустимо, що MONTHPAY = 70. Тоді ланцюжок R2 -R1 не приводить до знаходження цілі (PERIOD НЕ визначна як I, отже, не визначна в цьому ланцюжку і INTRATE).

Починаємо спочатку і шукаємо наступне правило, де знаходиться мета INTRATE (це R4). Тепер необхідно визначити PERIOD (нову змінну цілі). PERIOD знаходиться в правилі R3. Т.к. MONTHPAY = 70, то РЗ - вірно, тоді PERIOD = 60. Далі перевіряється R4. Воно вірно. Отже, INTRATE = 11.0.

3.1.4 Робочі змінні

Робоча змінна (РЗ) - це звичайна змінна, аналогічна змінним в МВУ.

Спочатку все РЗ мають значення UNKNOWN. Їм можна привласнити значення будь-якого типу.

A = 12.5 - приклад числової змінної;

B = "це строкова змінна" - приклад строкової змінної;

C = TRUE,

D - FALSE - логічні змінні.

1.1.8 Попередньо певні змінні

Існує два типи попередньо визначених змінних (ПВЗ): середовища і утиліти. Середовище "GURU" визначається змінними середовища. Вони визначають різні Функціональні характеристики середовища "GURU". Ім'я цієї змінної завжди починається з букви E. Наприклад:

E.HELP = TRUE

Завдання ПВЗ змушує "GURU" автоматично реагувати на помилку в команді. Наприклад:

E.LSTR = 80

Максимальна довжина символьного рядка дорівнює 80.

Змінні типу утиліти служать для різних допоміжних цілей. Вони починаються зі знака #. наприклад:

#GOAL = INTRATE
COAL визначає мету ЕС.

3.1.5 Вирази зі змінними

Числові:
2 + 4
DEPTH = 5
6 + DEPTH
2 ** 2
SQRT (4) 60/5
5.67 * PI
Рядкові:
NAME = "Іванов" + "Іван" - зчеплення NAME стає рівною "Іванов Іван".
Логічні:
A = B
A <> B
A <= B
A> = B
IN - вводиться для позначення рівносильности одного елемента іншому (перевірки того, чи відповідають один одному права і ліва частини виразу); допустимо треба знайти службовця, чиє прізвище Мінев або Манев або Монєва, тоді вводимо логічне вираз:

NAME IN [M $ HEB].
Складові логічні вирази, наприклад:
15> 9 AND 20 <100 - справжнийє вираз;
9> 15 AND 20 <100 - помилковий вираз.

3.1.6 Пояснення аргументації

Важливою характеристикою "GURU" є можливість пояснити весь хід дій при консультації з набором правил. Це робиться за допомогою команд HOW і WHY. Як це робити, буде пояснено в подальшому.

3.1.7 Синтаксис правил "GURU"

Запишемо приклад згідно синтаксичним правилам "GURU" і докладно пояснимо.

Припустимо, ми проводимо технічний огляд автомобіля і хочемо знати, поїде він чи ні. Ми перевіряємо акумулятор і стартер і на основі результатів огляду приймаємо рішення. Ось наші правила в цій експертній системі:

Правило 1.

Якщо акумулятор сів або несправний стартер, то двигун не заведеться.

Правило 2.

Якщо акумулятор заряджений, справний стартер, то двигун заведеться.

Правило 3.

Якщо двигун заведеться, то автомобіль поїде.

Правило 4.

Якщо двигун не заведеться, то автомобіль не поїде.

А от як запишуться ці правила експертної системи з урахуванням синтаксису "GURU".

```
RULESET: CAR
GOAL: MOVE

INITIAL: CLEAR
E.LSTR = 80
MOVE = UNKNOWN
AKKUM = UNKNOWN
STARTER = UNKNOWN
MOTOR = UNKNOWN
OUTPUT "Система діагностики автомобіля"

VARIABLE: MOVE
LABEL: Чи буде рухатися автомобіль?
VARIABLE: AKKUM
LABEL: чи заряджена батарея?
FIND: input akkum using "u" with "Заряджений у Вас акумулятор (Y / N)?"
VARIABLE: STARTER
LABEL: Справний чи стартер?
FIND: input starter using "u" with "Справний у Вас стартер (Y / N)?"
VARIABLE: MOTOR
LABEL: Чи працює мотор?

DO:
CLEAR
OUTPUT "Автомобіль", MOVE
RULE: R1
IF: AKKUM <> "Y" OR STARTER <> "Y"
THEN: MOTOR = 0
REASON: Якщо акумулятор сів або стартер не працює, то мотор не заведеться.
RULE: R2
IF: AKKUM = "Y" AND STARTER = "Y"
THEN: MOTOR = 1
```

REASOH: ЯКЩО акумулятор заряджений і стартер працює, то мотор заведеться

RULE: R3

IF: MOTOP = 1

THEN: MOVE = "ПОЇДЕ"

REASON: Якщо мотор працює, то автомобіль поїде

RULE: R4

IF: MOTOR = 0

THEM: MOVE = "НЕ ПОЇДЕ"

Пояснюючи все більш докладно.

RULESET: CAR - це ім'я набору правил (необов'язково вказувати);

INITIAL: - це розділ ініціалізації. Сюди входять ті команди, які повинні виконатися до консультації з наборів правил.

CLEAR - команда для очищення екрана.

E.LSTR = 80 - ця змінна встановлює максимальну довжину символьних рядків.

MOVE = UNKNOWN,

MOTOR = UNKNOWN - ініціалізація змінних, при якій їм присвоюється значення UNKNOWN.

ODTPUT "Система діагностики автомобіля" - виводить на екран символьний рядок.

VARIABLE: MOVE - визначається змінна, використовувана в наборі правил. Всі змінні повинні бути визначені.

LABEL: - Пояснення на "природній мові, навіщо потрібна дана змінна.

FIND: - якщо в посилці зустрічається змінна з невизначеним значенням (UNKNOWN), яка не присутня в укладенні будь-якого правила, то виводяться ті команди, які знаходяться після FIND. Тут знаходиться команда вводу: input akkum using "u" with "Заряджений у Вас акумулятор?" Ця команда чекає введення з екрану в змінну akkum символів "Y" або "N". У цій команді вводу:

using "u" - шаблон вводу;

with "..." - виводяться на екран у вигляді запиту для підказки.

RULE: R1 - ім'я правила;

IF: - посилка правила;

THEN: - укладення правила;

REASOH -пояснення на природній мові, що робить правило.

DO - розділ завершення. Виконуються команди, які необхідні для виконання консультації з ЕС.

OUTPUT "Автомобіль", MOVE - виводиться рядок "Автомобіль" і слідом за нею змінна MOVE.

Система працює таким чином: послідовним перебором правил, починаючи з першого, знаходиться правило R3, що містить в укладенні

змінну цілі - MOVE. МЛВ визначає, що вона UNKNOWN, отже необхідно знайти умову для її знаходження. У правилі R3 це умова задається змінною MOTOR. MOTOR - змінна з невідомим значенням, що міститься в посилці правила R3, яка стає новою змінною цілі. Ця змінна при послідовному переборі правил, починаючи з першого, вперше зустрічається в укладенні правила R1. В посилці цього правила - дві змінних. Вони теж невідомі, причому їх не можна виявити в укладенні будь-якого правила, але вони описані в FIND. Отже, вводиться запит на введення цих змінних. Коли AKKUM і STARTER введені, то визначається MOTOR. Потім перевіряємо MOTOR і визначаємо MOVE. Мета досягнута.

3.2 Порядок виконання роботи

1. Запустіть "GURU" з командного рядка, для цього введіть GURU.EXE.

Система виводить на екран: ІМ'Я НОВОГО СЕАНСУ ... Введіть ім'я вашого сеансу роботи. Це ім'я буде у подальшій використовуватися для завантаження сеансу вашої роботи з "GURU".

2. Виберіть в основному меню рядок "експертні системи". У новому меню оберіть рядок "Створення експертної системи".

3. Ви будете редагувати наявний набір правил. Виберіть рядок "Існуюча база знань" (БЗ).

3.Виберіте існуючу БЗ "Animal", яка зберігається у файлі з розширенням * .rss і натисніть "Enter". Ознайомтеся зі структурою даної БЗ. Проведіть консультацію. Побудуйте дерево цієї БЗ. Доповніть дерево гілками правил, що дозволяють визначити живу істоту, згідно вашого варіанту (см. Додаток 3).

4. Виберіть рядок меню "Правила". Виберіть рядок меню "Створення". Введіть ім'я правила і натисніть «Enter». Ви потрапляєте в середовище створення правила. Перехід від одного до іншого правила - за допомогою клавіш PgDn і PgUp.

У полі IF введіть посилку. У полі THEN введіть висновок. У полі ВИСНОВОК введіть пояснення. За допомогою клавіші «ESC» вийдіть із середовища створення правила.

5. Виберіть рядок Маню "Редагування". Перегляньте введене правило. Виправте, помилки, якщо вони є.

6. Аналогічно введіть всі правила. Виберіть "Перегляд". Перегляньте всі введені правила. Виберіть "Попереднє меню" і ще раз "Попереднє меню".

7. Виберіть рядок меню "Змінні". Виберіть рядок "Створення". Ви потрапите в середу для створення змінних. Перехід між вікнами - за допомогою клавіш PgDown і PgUp.

Введіть у вікно "LABEL" пояснення.

Введіть у вікно "FIND" інформацію (якщо необхідно). Натисніть «ESC».

8. Виберіть рядок меню "Редагування". Перегляньте створену змінну. Введіть, якщо потрібно, виправлення.

9. Аналогічно створіть всі необхідні змінні. Виберіть рядок меню "Перегляд". Перегляньте всі створені змінні.

10. Виберіть двічі рядок "Попереднє меню".

11. Виберіть рядок меню "Завершення". На екрані з'являється текстовий редактор бази знань. Тут вводяться команди, які виконуються після того, як виконаний сеанс роботи з експертною системою (див. Розділ 8 цього опису лабораторної роботи). Найміть «ESC».

Виберіть рядок меню "Вихід".

Виберіть рядок меню "Збереження". Ваш набір правил зберігається у файлі з розширенням * .rss.

Виберіть "Компілювання". Відкомпілюйте ваш набір правил. Перегляньте результати компіляції. Виправте базу знань у відповідності з цими результатами і знову відкомпілюйте (якщо це буде потрібно). Для цього поверніться в попереднє меню і повторіть всі операції по пунктах 4-15, виключаючи ті, які Вам не будуть потрібні. Виберіть рядок меню "Кінець".

16. Виберіть рядок меню "Попереднє меню". Виберіть рядок: "Консультація з експертна системою". Перевірте працездатність вашої експертної системи. Якщо вона не працює, то виправте набір правил та виправте помилки.

3.3 Варіанти завдань

ВАРІАНТ №1

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «курки».

ВАРІАНТ №2

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «крокодила».

ВАРІАНТ №3

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «домашньої мухи».

ВАРІАНТ №4

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «зозулі».

ВАРІАНТ №5

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «баракуди».

ВАРІАНТ №6

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «мурени».

ВАРІАНТ №7

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «бджоли».

ВАРІАНТ №8

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «їжака».

ВАРІАНТ №9

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «лелеки».

ВАРІАНТ №10

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «тарантула».

ВАРІАНТ №211

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «рака».

ВАРІАНТ №12

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «анаконди».

ВАРІАНТ №13

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «зебри».

ВАРІАНТ №14

Доповніть Експертну систему "Animal" правилами і змінними, необхідними для визначення «криветки».

3.4 Вимоги до звіту

Звіт про роботу повинен містити:
1. Короткі теоретичні відомості з теми роботи
2. Дерево доповненої ЕС з виділеними гілками ваших доповнень.
3. Доповнений варіант експертної системи;
4. Висновки по роботі.

3.5 Додаток 1 Опис змінних середовища

Ім'я	Тип Опис	Значення за замовчуванням
E.BELL	Логічний Лунає дзвінок, якщо вводиться недійсне значення	TRUE
E.DECI	Числовий Встановлює кількість цифр праворуч від десяткової точки	2
E.HRES	Числовий Встановлює ступінь відповіді на команду HOW від 0 (немає відповіді) до 6 (найбільш докладну відповідь)	4
E.LLOG	Числовий Задає довжину логічного шаблону за замовчуванням (максимум - 5)	5
E.LSTR	(максимум - 255)	15
E.LNUM	Числовий Задає довжину для числового шаблону (максимум - 14)	14
E.OCOH	Логічний Виведення даних на дисплей	TRUE
E.OPRN	Логічний Виведення всіх вихідних даних на принтер	FALSE
E.PDEP	Числовий Довжина друкованої сторінки	60
E.WHN	Символьний Вказує, коли з'являються команди FIND при знаходженні невідомої змінної:	N - ніколи; L - тільки як останній засіб; F - перш ніж буде зроблена спроба оцінити значення невідомих змінних L

3.6 Додаток 2 Вирази та Функції GURU

Ім'я	Призначення
арифцілічн	
+	Додавання;
-	Віднімання;
*	Множення;
/	Ділення;
**	Зведення в ступінь;
MOD	ділення по модулю.
порівняння:	
EQ =	рівно;
NE, <>	не дорівнює;
GT,>	більше ніж;
LT, <	менше ніж;
GE, <=	менше або дорівнює;
LE,> =	більше або дорівнює.
логічні:	
NOT	ні;
AND, &	і;
OR	або;
XOR	що виключає "або";
=	Привласнення;
()	Індекси масиву.
строкові:	
+	Зчеплення рядків;
'	Лапка;
$	Символ відповідності символу;
*	Символ відповідності рядка.
Числові Функцій:	
ABS	абсолютне значення;
ARCSIN	арксинус;
EXP	е в ступені;
INIT	инициализирует масив;
LEN	визначає довжину рядка;
LN	обчислює натуральний логарифм;
LOG	обчислює логарифм з основою 10;
MAX	найбільше з двох чисел;
MENU	створює меню;

MIN	менше з двох чисел;
RAND	випадкове число;
SIN	синус;
SQRT	квадратний корінь.
Символьні Функції:	
CHR	перетворить код ASCII в його символьний еквівалент;
VAL	перетворить символ в його код ASCII;
INIT	инициализирует масив;
SUBSTR	виділяє підрядок з рядка;
TIME	повертає поточний час;
TOSTR	перетворить числа в символи;
TONUM	перетворить рядок в число;
TRIM	відсікає кінцеві прогалини;
TYPI	тип змінної.
Логічні Функції:	
ALPHASTR	чи вся рядок складається з букв;
INIT	инициализирует масив.

4 Лабораторна робота №4

Створення пробної експертної системи.

Мета - самостійне програмування в повному обсязі найпростішої експертної системи.

Реалізація названої цілі лабораторної роботи передбачає грунтовну проробку наведеного в додатку I цієї лабораторної роботи прикладу реалізація експертної системи. Для полегшення такого пророоблення це додаток забезпечено докладним коментарем.

4.1 Теоретичні відомості

Основні команди GURU

BUILD <ім'я набору правил> - використовується для створення, модифікації і компіляції набору правил.

COMPILE <ім'я набору правил> - створює файл відкомпільованого набору правил.

CONSULT <ім'я експертної системи> - звернення до ЕС за консультацією.

WHY <вираз> - пояснює, чому "GURU" використовувала конкретне правило.

HOW <вираз> - пояснює, як "GURU" знайшла значення змінної.

TEXT <ім'я файлу> - запускається текстовий редактор "GURU".

CLEAR - очистити екран.

INPUT <змінна> <USING шаблон> <WITH вираз> - введення даних в змінну з використанням шаблону (USING) і з підказкою (WITH). шаблони:

a - для буквених символів (латинський шрифт);

c - для букви або числа;

d - для цифри, знака (+ чи -) або десяткового дробу;

e - перетворення в символи нижнього регістра;

n - символ шаблону, який сприймає тільки цифри в займаній ним позицій;

r - для символів ASCII;

u - перетворить в символи верхнього регістру. наприклад:

INPUT num USING "dddd" with "Введіть номер"

На екрані з'являється текст:

Введіть номер __

Розглянемо ще одну команду "GURU".

OUTPUT <ім'я змінної> <USING шаблон> - виводить на екран змінну або рядок.

Шаблони У цій команді аналогічні шаблонам для команди INPUT.

наприклад:

OUTPUT " Лабораторна робота N1 "

Виводить на екран:

Лабораторна робота N1

Інший приклад:

OUTPUT num

Виводить на екран значення змінної num. Наведемо також перелік наступних команд "GURU":

HELP - виводить довідкову інформацію;

RUN - виконує зовнішню програму;

DIR - переглядає директорію;

BYE - виходить із режиму;

RELEASE - звільняє пам'ять, видаляючи дані і програми "GURU";

PEPFORM - виконує процедуру;

WAIT - призупиняє обробку до натискання будь-якої клавіші.

1. Перелічіть

4.2 Підготовка до роботи

1. Ознайомтеся з матеріалом лекцій та даними методичними вказівками.

2. Дайте відповідь на контрольні питання.

3. Запустіть DOSBOX та пропишить конфігураційному файли для монтування папки з оболонкою VC та GORU та запуску оболлнки VC.

```
MOUNT c E:\DOS22\
c:
CD VC
VC
```

4. Запустsть оболонку VC, потім - утиліту cyrilik.com, потім - GORU. Ознайомтеся з готовим варіантом ЕС (додаток 1 до цієї лабораторної роботи).

5. Відповідно до заданих варіантом (додаток 2 до цієї лабораторної роботи) напишіть програму, що реалізовує невелику експертну систему. Практично будь-який з заданий варіанту може бути реалізований невеликим набором правил.

4.3 Порядок виконання роботи

1. Пред'явіть викладачеві текст програмної реалізації заданого варіанту ЕС, написаний в ході домашньої підготовки.

2. Отладьте ЕС.

3. За допомогою HOW і WHY перевірьте правильність роботи системи.

4. Покажіть результати роботи ЗС викладачеві.

4.4 Приклад експертної системи

```
/ * INVESTORS.RSS * /
/ * * /

GOAL: advice

INITIAL:
e.tryp = "e"
e.lstr = 80 / * Максимальна довжина рядка 80 * /
e.deci = 0 / * нема цифр після десяткової коми * /
e.lnum = 8 / * Довжина числа * /
savperdep = 5000 / * Допустимий мінімум заощаджень * /
/ * У розрахунку на одного утриманця, при якому загальні сбере- * /
/ * ження сім'ї можна назвати 'хорошими' (true) * /
incperdep = 4000 / * Мінімально допустимий дохід на * /
/ * Одного утриманця * /
basincome = 15000 / * Мінімально допустимий дохід * /
  / * Глави сім'ї * /
/ * Загальний хороший дохід сім'ї визначається об'єднанням * /
/ * Incperdep і basincome * /
advice = unknown
goodsave = unknown
goodincome = unknown
income = unknown
savings = unknown
steady = unknown
needincome = unknown
dependents = unknown
newcash = unknown
clear
output "КАПІТАЛОВКЛАДЕННЯ"
input newcash num
```

```
DO: / * Цей розділ виконується після того, * /
/ * Як оброблені правила * /

output "НА ОСНОВІ ЦІЄЇ ІНФОРМАЦІЇ:"
test advice
case "АКЦІЇ":
output "ВАМ СЛІД ВКЛАСТИ ВРЮ СУМУ В АКЦІЇ."
break
case "ЗАОЩАДЖЕННЯ":
output "ВАМ СЛІД помістити всю СУМУ В ЗАОЩАДЖЕННЯ."
break
case "КОМПРОМІС":
tosave = min (newcash, (savperdep * dependents) - savings)
tostock = max (0, newcash - tosave)
output "ВАМ слід помістити в ЗАОЩАДЖЕННЯ"
tosave using "$ ff, fff, fff"
if tostock> 0 then
output "І ВКЛАСТИ", tostock using "$ ff, fff, fff", "В АКЦІЇ."
endif
break
endtest
e.deci = w
e.lnum = 14

RULE: R1
IF: goodincome and goodsave
THEN: advice = "АКЦІЇ"
NEEDS: goodincome goodsave
REASON: ВКЛАДАТИ В АКЦІЇ, якщо клієнт
НАДІЙНИЙ У фінансовому відношенні

RULE: R2
IF: not goodincome
THEN: advice = "ЗАОЩАДЖЕННЯ"
NEEDS: goodincome
REASON: НЕ вкладаються в акції, ЯКЩО ВАШ ДОХІД
НИНІ нестійкий

RULE: R3
IF: not goodsave and goodincome
THEN: advice = "КОМПРОМІС"
REASON: ЯКЩО ЗАОЩАДЖЕННЯ НЕВЕЛИКІ, ТОДІ ВОНИ ПОВИННІ
Бути збільшена до ТОГО, ЯК ЇХ ВКЛАДАТИ
```

RULE: R4
IF: not steady
THEN: goodincome = false
REASON: ДЛЯ ГАРНОГО ДОХОДУ НЕОБХІДНА ПОСТІЙНА РОБОТА

RULE: R5
IF: not (income> needincome)
THEN: goodincome = false
REASON: ДОХІД НЕ ЗАЛЕЖИТЬ ВІД ВАС І ВІД утриманців

RULE: R6
IF: not (income> needincome)
THEN: goodincome = true
REASON: ЩОБ ДОХІД БУВ ХОРОШИЙ, КЛІЄНТ ПОВИНЕН МАТИ
ПОСТІЙНУ РОБОТУ

RULE: R7
IF: known ("income") and known ("dependents")
THEN: needincome = baseincome + (dependents * incperdep)
REASON: НЕОБХІДНИЙ ДОХІД - ЦЕ ДОХІД, ЯКИЙ ВАМ
НЕОБХІДНИЙ ПЛЮС ЗАГАЛЬНИЙ ДОХІД ВСІХ ВАШИХ утриманців

RULE: R8
IF: savings> (saveperdep * dependents)
THEN: goodsave = true
REASON: ЗАОЩАДЖЕННЯ КЛІЄНТІВ повинні залежати від НИХ
САМИХ І ВІД утриманців

RULE: R9
IF: savings <= (saveperdep * dependents)
THEN: goodsave = false
REASON: ЗАОЩАДЖЕННЯ КЛІЄНТІВ повинні залежати від НИХ
САМИХ І ВІД утриманців

NEEDS: savings dependents
REASON: ЗАОЩАДЖЕННЯ КЛІЄНТІВ НЕ ЗАЛЕЖАТЬ ВІД НИХ САМИХ І ВІД утриманців.
/ * Визначення змінних * /
VAR: NEWCASH
LABEL: суми готівки ДЛЯ ВКЛАДУ

```
VAR: ADVICE
LABEL: ДАНИЙ РАДА
VAR: GOODINCOME
LABEL: поточнідоходи - ХОРОШИЙ
VaR: GOODSAVE
LABEL: поточні заощадження - ХОРОШІ
VAR: NEEDINCOME
LABEL: НЕОБХІДНА СУМА ДОХОДУ
VAR: INCOME
FIND: input income num with "ЯКОЮ ВАШ РІЧНИЙ ДОХІД СІМ'Ї?"
LABEL: поточнідоходи
VAR: SAVINGS
FIND: input savings num with "СКІЛЬКИ у вас ЗАОЩАДЖЕНЬ?"
LABEL: поточні заощадження
VAR: STEADY
FIND:
output "МОЖЕТЕ ВИ чекати стабільного ДОХІД НА"
output "СЛІД. РІК? (y / n)"
input steady str using "u"
steady = (steady = "Y")
LABEL: НАДІЙНИЙ ДОХІД
VAR: DEPENDENTS
FIND:
output "СКІЛЬКИ У ВАС утриманців? "
input dependents num using "dd"
LABEL: кількість утриманців
END:
```

Опишемо детально роботу набору правил. Він призначений для ілюстрація зворотного аргументації.

В INITIAL йде ініціалізація змінних. Розглянемо її окремі рядки.

e.tryp = 'e' - задає стратегію оцінки посилки (частини "if" правила), що містить невідомі змінні. Істинність посилки оцінюється відразу ж після того, як чергова невідома змінна стає відомою. Тестування посилки припиняється (незважаючи на те, що всі змінні в ній ще не визначені), якщо тільки вдається точно встановити її істинність або хибність.

e.lstr = 80 - максимальна довжина символьного рядка, яка може виводитися на екран.

e.lnum = максимальна довжина числа.

В VAR описуються користувача змінні (см. Списання лабораторної роботи 1).

Частина DO - закінчення роботи експертної системи. Конструкція test ... case ... endtest перевіряє змінну advice і залежно від її значення виконує ті чи інші дії.

Розглянемо, як може працювати ця система. Після запуску відбувається ініціалізація змінних. Консультація з ЕС йде методом зворотної аргументації. Система "GURU" в цьому випадку починає е кінця. Визначається мета ADVICE. Т.к. вона невідома, то проглядаються ті правила, які визначають ADVICE. У нашому випадку першим правилом, де визначається ADVICE буде R1. Тут невідомі GQODINCOME і GOODSAVE.

GQODINCOME спочатку визначається в R4. GOODSAVE - в R8. В R4 перевіряється змінна STEADY. Її значення запитується за допомогою оператора FIND в описі змінної. Якщо R4 не може бути виконано (посилка помилкова), то тестується R5, якщо R4 істинно (посилка вірна), змінної GOODINCOME присвоюється значення "false", "GURU" визначає, що посилка правила R1 невірна і переходить до оцінки посилки правила R2. В R2 посилка вірна, тому з цього правила визначається advice - мета системи. На цій висновок завершується. При тестуванні R5 нам доведеться визначити значення змінних INCOME і NEEDINCOME і т.д.

Аналогічно знаходиться змінна GOODSAVE (правила R8 і R9).

Перше питання, яке задасть ЕС: "Яку суму готівкових грошей Ви хотіли б вкласти?"

Введіть: 12000 <Enter>

Наступне питання: "Чи можете Ви очікувати стабільний дохід на наступний рік?"

Введіть: "Y"

"Який Ваш річний дохід сім'ї?"

Введіть: 35000 <Enter>

"Скільки у Вас утриманців?"

Введіть: 4

"Скільки у Вас заощаджень?"

Введіть: 10000 <Enter>

Експертна система виводять суму, яку слід відкласти в заощадження і суму, яку необхідно вкласти в акція.

Розберіть докладно роботу цієї ЕС для того, щоб створити свою.

Пріложеніе.2 Варіанти завдань

1.2 Варіанти завдань

варіант 1

Розробіть ЕС, яка повторює хід ваших думок при переході через дорогу. Спочатку ви визначаєте, який колір на світлофорі. Якщо червоний - чекаєте, якщо зелений - дивіться, чи немає якого-небудь "божевільного"

водія, якій міг би їхати на червоний колір. Якщо є, то чекайте, поки він проїде. Якщо ні, то переходите через дорогу.

варіант 2

Створіть ЕС, що визначає несправність магнітофона. Ви включаєте магнітофон, і він працює, але звучання погане. Ви перевіряєте чи, забруднена головка. Якщо так, то необхідно протерти головку. Якщо ні, то перевіряйте, чи правильно встановлена касета. Якщо ні, то поправляєте, а якщо правильно, то кажіть, що необхідно викликати майстра.

варіант 3

Створіть ЕС, що визначає приймати чи ні людини на роботу. Якщо людина не має вищої освіти, то відмовити. Якщо має, то скільки років пропрацював претендент за фахом. Якщо менше року, то відмовити, якщо більше - то прийняти. Якщо претендент має вчене звання, то запропонувати посаду наукового співробітника, якщо ні, то посаду інженера-конструктора.

варіант 4

Розробіть ЕС. яка дає поради при створенні та налагодженні програми .. Якщо ви написали програму на МВУ і при компіляції виявили синтаксичні помилки, то необхідно виправити програму. Якщо помилок немає, то необхідно запустити редактор зв'язків. Якщо редактор зв'язків видає помилки, то необхідно перевірити наявність всіх вихідних модулів. Якщо помилок при Лінкування немає, то програма готова до роботи.

варіант 5

Розробіть ЕС, яка визначає, чи буде сьогодні дощ. Спочатку ви визначаете, чи ясне небо. Якщо небо ясне, то дощу не буде. Якщо небо похмуре, то ви дивитеся, чи є на небі чорні грозові хмари. Якщо ні, то дощу не буде. Якщо є, то дивіться, в який бік вони рухаються. Якщо в вашу сторону, то дощ буде. Якщо ж ні, то не буде.

варіант 6

Розробіть ЕС, визначальну чи є у дитини труднощі при вивченні арифціліки. ЕС дитині пропонує по одному прикладу на додавання, віднімання, множення і ділення. Якщо він правильно відповідає на всі питання, то у нього немає проблем з арифцілікою. При неправильних відповідях система вказує на труднощі з виконанням конкретних операцій.

варіант 7

Розробіть ЕС, яка буде прогнозувати на біржі рівень цін. Якщо валютний курс долара падає, і процентні ставки ростуть то рівень цін на біржі падає. Якщо валютний курс долара зростає, і процентні ставки падають., то рівень цін на біржі падає. В інших випадках ситуація оцінюється як нестабільна.

варіант 8

Розробіть ЕС, яка буде прогнозувати, чи буде в результаті весняного паводку повінь чи ні. Якщо рівень води в річці в межах міста високий і йдуть сильні дощі, тепла погода і багато снігу в горах, то очікується

повінь. Якщо ж хоча б один з цих факторів не виконується, то повені не буде.

варіант 9

Необхідно визначити, даний об'єкт є танком або автомобілем. У танка є гармата і люк. У автомобіля є дверцята і колеса. У танка і автомобіля є кузов. Уточнюючи всі ці характеристики, ЕС повинна визначити об'єкт.

варіант 10

Ви вмикаєте телевізор, а він не працює. Ви хочете визначити, чому це сталося. Якщо запобіжник згорів, то його необхідно замінити. Якщо запобіжник цілий, то перевіряєте кабель живленнія. Якщо він розірваний в якомусь місці, то необхідно його замінити. Якщо кабель живлення цілий, і ви самі розбираєтеся в радіоелектроніці, то чиніие телевізор. Якщо не розбираєтеся, то викликаєте майстра.

варіант 11

Розробіть ЕС, яка повторює хід ваших думок при переході через залізницю. Спочатку ви визначаєте чи є світлофор, якщр є то який колір на світлофорі. Якщо червоний - чекаєте, якщо зелений - то переходите через дорогу. Якщо нема свііглофора дивіться, чи немає якого-небудь потяга якій міг би їхати. Якщо є, то чекайте, поки він проїде. Якщо ні, то переходите через дорогу.

варіант 12

Розробіть ЕС, що визначає несправність CD-плеєра. Ви включаєте його, і він працює, Але не звучить. Ви перевіряєте, чи забруднений диск. Якщо так, то необхідно його протерти . Якщо ні, то перевіряйте, чи правильно диск встановлений. Якщо ні, то поправляєте, а якщо правильно, то необхідно викликати майстра.

варіант 13

Створіть ЕС, визначальну приймати чи ні людини роботу водія. Якщо людина не має прав, то відмовити. Якщо має, то скільки років пропрацював претендент за фахом. Якщо менше року, то відмовити, якщо більше - то прийняти. Якщо претендент має категорію «Д», то запропонувати посаду водія автобуса, якщо ні, то посаду водія вантажівки.

варіант 14

Розробіть ЕС, яка повторює хід думок водія при переїзді через залізницю. Спочатку водій визначає чи є занк «стоп» та чи є світлофор, якщо є то який колір горить. Якщо червоний, то водій чекає, якщо ні - то можна перетинати залізницю. Якщо нема світлофора дивіться, чи немає якого-небудь потяга якій міг би їхати. Якщо є, то чекайте, поки він проїде. Якщо ні, то можна їхати.

4.5 Вимоги, до звіту

Звіт про роботу повинен містити

1) тема роботи та короткі теоретичні відомості.

2) граф міркувань вашого варіанту ЕС виконаний природньою мовою;

3} пояснення, чому в написаній вами ЕС обрані ті чи інші змінні

4) скан дерева ЕС вашого варіанту.

5) роздруківку rss-файлу варіант експертної системи;

6) висновки по роботі.

РЕКОМЕНДОВАНА ЛИТЕРАТУРА

1. Методи Системи штучного інтелекту : Навч. посіб. для студ. / В. М. Заяць, Р. М. Камінський; Нац. ун-т “Львів. політехніка”. – Л., 2004. – 173 с. – Бібліогр.: 21 назв.