Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №4

з дисципліни:

«Мультипарадигменне програмування»

Виконав:

студент групи ІК-21

Бераудо Лоренцо Раффаелович

Київ 2025

ЛАБОРАТОРНА РОБОТА №4

Завдання: за допомогою логічного програмування реалізувати перетворення чисельного ряду до лінгвістичного ланцюжка за певним розподілом ймовірностей потрапляння значень до інтервалів.

Вхідні данні: чисельний ряд, вид розподілу ймовірностей, потужність алфавіту.

Вихідні дані: лінгвістичний ряд та матриця передування.

Мова програмування: Prolog і його клони.

Варіант 1: Дискретний рівномірний розподіл (рівноймовірний)

ХІД ВИКОНАННЯ ЗАВДАННЯ

1. **Генерація числового ряду.**

Вхідні числові дані задаються у вигляді списку (факт numeric_series). Можуть бути отримані з попередніх лабораторних або створені вручну.

2. **Сортування числового ряду.**

Для подальшого розбиття на інтервали ряд сортується у порядку зростання. Це дозволяє точно визначити мінімальне та максимальне значення, які задають межі діапазону значень.

3. **Розбиття діапазону значень на інтервали.**

Діапазон розбивається на N рівних інтервалів, де N — потужність алфавіту. Для дискретного рівномірного розподілу всі інтервали мають однакову довжину. Розрахунок проводиться за формулою:

`Step = (max - min + 1) / N`, де Step — довжина одного інтервалу.

Межі інтервалів формуються як список пар [a,b].

4. **Перетворення чисел у символи алфавіту.**

Для кожного числового значення визначається інтервал, у який воно

потрапляє, після чого числу присвоюється відповідний символ алфавіту за індексом цього інтервалу.

5. **Формування лінгвістичного ряду.**

Усі числа замінюються на відповідні символи, і формується лінгвістичний ряд.

6. **Побудова матриці передування.**

Із сформованого лінгвістичного ряду будується матриця переходів між символами.

Вона має вигляд квадратної матриці розміром N x N, де N — потужність алфавіту.

Кожен елемент a(i,j) вказує кількість випадків появи символа j після символа i.

ПРИКЛАД ВИВОДУ

Лінгвістичний ряд:

```
Лінгвістичний ряд: [A,J,E,B,H,I,D,A,C,G]
```

Матриця передування (фрагмент):

```
Матриця передування:
A -> A: 0
A -> B: 0
A -> C: 1
A -> D: 0
A -> E: 0
A -> F: 0
A -> G: 0
A -> H: 0
A -> I: 0
A -> J: 1
B -> A: 0
B -> B: 0
B -> C: 0
B -> D: 0
B -> E: 0
B -> F: 0
B -> G: 0
B -> H: 1
B -> I: 0
B -> J: 0
C -> A: 0
C -> B: 0
C -> C: 0
C -> D: 0
C -> E: 0
C -> F: 0
C -> G: 1
```

ВИСНОВОК

У ході виконання лабораторної роботи було реалізовано програму на мові логічного програмування Prolog для перетворення чисельного ряду у лінгвістичний ланцюжок на основі дискретного рівномірного розподілу.

Програма дозволяє:

- автоматично будувати інтервали на основі вхідного числового ряду та заданої потужності алфавіту;
- відображати кожне числове значення на відповідний символ алфавіту згідно з розподілом;
- формувати лінгвістичний ряд;
- будувати матрицю передування символів, яка показує частоту переходів між літерами.

Особливістю реалізації є гнучкість у виборі алфавіту, що дозволяє змінювати як його кількість, так і зміст без потреби модифікації логіки програми.

Усі компоненти були реалізовані у вигляді фактів і правил, що повністю відповідає парадигмі логічного програмування. Отримані результати збігаються з результатами, отриманими у попередніх лабораторних роботах, виконаних на інших мовах (R, Clojure тощо), що підтверджує правильність обраного підходу.

Таким чином, у процесі виконання лабораторної роботи було:

- поглиблено знання з логічного програмування;
- здобуто практичні навички роботи з базами фактів, рекурсією та обробкою списків у Prolog;
- продемонстровано, що логічне програмування є ефективним інструментом для розв’язання задач класифікації та аналізу даних.

Код програми

```
alphabet(['A','B','C','D','E','F','G','H','I','J']).
alphabet_power(10).
numeric_series([134, 987, 543, 234, 765, 876, 453, 100, 321,
678]).

min_max(List, Min, Max) :-
    min_list(List, Min),
    max_list(List, Max).

build_intervals(Min, Max, Power, Intervals) :-
    Step is (Max - Min + 1) / Power,
    build_intervals_rec(Min, Step, Power, Intervals).

build_intervals_rec(_, _, 0, []) :- !.
build_intervals_rec(Start, Step, N, [[Start, End]|T]) :-
    End is Start + Step,
    N1 is N - 1,
    build_intervals_rec(End, Step, N1, T).

find_interval(Number, [[Low, High]|_], 0) :-
    Number >= Low, Number < High, !.
find_interval(Number, [_|Rest], Index) :-
    find_interval(Number, Rest, I),
    Index is I + 1.

convert_to_letters([], _, _, []).
convert_to_letters([N|Rest], Intervals, Alphabet, [L|Converted])
:-
    find_interval(N, Intervals, Index),
    nth0(Index, Alphabet, L),
    convert_to_letters(Rest, Intervals, Alphabet, Converted).

build_matrix([], []).
build_matrix([_], []).
build_matrix([A,B|T], [[A,B]|Pairs]) :-
    build_matrix([B|T], Pairs).

count_transitions(Pairs, Alphabet, Matrix) :-
    findall([X,Y,Count],
        (member(X, Alphabet), member(Y, Alphabet),
         include(=([X,Y]), Pairs, Filtered),
         length(Filtered, Count)),
        Matrix).

run :-
    numeric_series(Nums),
    sort(Nums, Sorted),
    min_max(Sorted, Min, Max),
    alphabet_power(Power),
    build_intervals(Min, Max, Power, Intervals),
    alphabet(Alpha),
```

```
    convert_to_letters(Nums, Intervals, Alpha, Letters),
    format("Лінгвістичний ряд: ~w~n", [Letters]),
    build_matrix(Letters, Pairs),
    count_transitions(Pairs, Alpha, Matrix),
    writeln('Матриця передування:'),
    forall(member([A,B,C], Matrix), format('~w -> ~w: ~w~n',
[A,B,C])).
```