Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 6

з дисципліни

«Мультипарадигменне програмування»

ВИКОНАВ:
студент III курсу ФІОТ
групи ІО-23
Бичкар Н. В.
Залікова № 2302

ПЕРЕВІРИВ:
ас. Очеретяний О. К.

Київ – 2025

Лістинг:

```
/* === ІНІЦІАЛІЗАЦІЯ ВХІДНИХ ДАНИХ === */
data : [3.2, 7.8, 1.5, 9.0, 4.6]$
alphabet : ["A", "B", "C", "D"]$

/* === СОРТУВАННЯ ЧИСЕЛЬНОГО РЯДУ === */
sorted_data : sort(data)$
min_val : lmin(data)$
max_val : lmax(data)$
range_val : max_val - min_val$

/* === РОЗРАХУНОК ІНТЕРВАЛІВ ЗА ПУАССОНІВСЬКИМ РОЗПОДІЛОМ === */
n : length(alphabet)$
/* Вибираємо lambda так, щоб відповідало характеру наших даних */
lambda : float(3.0)$  /* Параметр розподілу Пуассона */

/* Функція густоти ймовірності Пуассона */
poisson_pmf(k, lambda) := float((lambda^k * exp(-lambda)) / factorial(k))$

/* Додаємо функцію для відлагодження */
debug_print(msg, val) := (
    print("DEBUG:", msg, "=", val),
    val
)$

/* Розрахунок інтервалів за Пуассонівським розподілом */
intervals : block([probs, cum_prob, interval_bounds, result],
    /* Генеруємо ймовірності для значень k від 0 до n-1 за розподілом Пуассона */
    probs : makelist(poisson_pmf(k, lambda), k, 0, n-1),
    print("DEBUG: Poisson probabilities =", probs),

    /* Нормалізуємо ймовірності, щоб їх сума = 1 */
    total_prob : apply("+", probs),
    norm_probs : makelist(p/total_prob, p, probs),
    print("DEBUG: Normalized probabilities =", norm_probs),

    /* Розраховуємо межі інтервалів на основі нормалізованих ймовірностей */
    interval_bounds : [min_val],
    cum_prob : 0,

    for i : 1 thru n - 1 do (
        cum_prob : cum_prob + norm_probs[i],
        interval_bounds : endcons(min_val + cum_prob * range_val, interval_bounds)
    ),
    interval_bounds : endcons(max_val, interval_bounds),
    print("DEBUG: Interval bounds =", interval_bounds),

    /* Формуємо остаточні інтервали */
```

```
    result : [],
    for i : 1 thru n do (
        result : endcons([interval_bounds[i], interval_bounds[i+1]], result)
    ),

    debug_print("Final intervals", result)
)$

/* === ВИПРАВЛЕНЕ ВІДОБРАЖЕННЯ ЧИСЕЛ В ЛІНГВІСТИЧНІ СИМВОЛИ === */
map_number_to_symbol(x) := block([i, result],
    result : false,
    print("DEBUG: Mapping number", x),

    /* Для кожного числа знаходимо відповідний інтервал */
    for i : 1 thru n do (
        print("DEBUG: Testing interval", i, ":", intervals[i]),
        if float(x) >= float(intervals[i][1]) and float(x) <= float(intervals[i][2]) then (
            print("DEBUG: Number", x, "is in interval", i, "-> symbol", alphabet[i]),
            result : alphabet[i],
            return(result)
        )
    ),

    /* Якщо число не потрапляє в жоден інтервал (що малоймовірно),
       повертаємо символ з найближчого інтервалу */
    if result = false then (
        print("DEBUG: Number", x, "didn't match any interval!"),
        if float(x) < float(intervals[1][1]) then (
            print("DEBUG: Using first symbol as fallback"),
            result : alphabet[1]
        ),
        if float(x) > float(intervals[n][2]) then (
            print("DEBUG: Using last symbol as fallback"),
            result : alphabet[n]
        )
    ),

    if result = false then (
        print("DEBUG: Using default symbol as fallback"),
        result : alphabet[1]
    ),

    print("DEBUG: Final result for", x, "is", result),
    result
)$

/* Перетворюємо числовий ряд на лінгвістичний */
linguistic_sequence : map(map_number_to_symbol, data)$
```

```
/* === ПОБУДОВА МАТРИЦІ ПЕРЕХОДІВ === */
/* Функція для знаходження індексу елемента в списку */
find_index(element, lst) := block([i],
   for i : 1 thru length(lst) do (
      if is(equal(lst[i], element)) then return(i)
   ),
   return(1)  /* Повертаємо 1 за замовчуванням, якщо елемент не знайдено */
)$

/* Функція для побудови матриці переходів */
transition_matrix : block([matrix, prev, current, i],
   /* Ініціалізуємо матрицю нулями */
   matrix : zeromatrix(n, n),

   /* Якщо лінгвістичний ряд містить хоча б два елементи */
   if length(linguistic_sequence) > 1 then (
      /* Знаходимо індекс першого елемента */
      prev : find_index(linguistic_sequence[1], alphabet),

      /* Для кожної пари послідовних елементів */
      for i : 2 thru length(linguistic_sequence) do (
         /* Знаходимо індекс поточного елемента */
         current : find_index(linguistic_sequence[i], alphabet),

         /* Збільшуємо відповідний елемент матриці */
         matrix[prev, current] : matrix[prev, current] + 1,

         /* Поточний елемент стає попереднім для наступної ітерації */
         prev : current
      )
   ),

   /* Повертаємо матрицю */
   matrix
)$

/* === ВИВІД РЕЗУЛЬТАТІВ === */
print("Вхідні дані:")$
print(data)$

print("Діапазон значень:", min_val, "-", max_val)$

print("Параметр розподілу Пуассона (lambda):", lambda)$

print("Розраховані інтервали за Пуассонівським розподілом:")$
for i : 1 thru n do
   print("Інтервал", i, ":", float(intervals[i][1]), "-", float(intervals[i][2]), "->", alphabet[i])$
```

```
print("Лінгвістичний ряд:", linguistic_sequence)$
print("Кількість елементів:", length(linguistic_sequence))$

/* Спробуємо вручну побудувати матрицю переходів */
manual_matrix : zeromatrix(n, n)$

/* Якщо лінгвістичний ряд правильний, він має бути як ["B","D","A","D","C"] */
/* Встановимо відповідні переходи */
if length(linguistic_sequence) > 1 then (
    /* Знаходимо індекси кожного елемента в алфавіті */
    indices : [],
    for i : 1 thru length(linguistic_sequence) do (
        for j : 1 thru length(alphabet) do (
            if is(equal(linguistic_sequence[i], alphabet[j])) then (
                indices : append(indices, [j])
            )
        )
    ),

    print("Індекси елементів у алфавіті:", indices),

    /* Будуємо матрицю переходів вручну */
    for i : 1 thru length(indices) - 1 do (
        from_idx : indices[i],
        to_idx : indices[i+1],
        manual_matrix[from_idx, to_idx] : manual_matrix[from_idx, to_idx] + 1
    )
)$

/* Виводимо результат вручну збудованої матриці */
print("Вручну побудована матриця переходів:")$
print("   | ", alphabet)$
print("---+", makelist("---", i, 1, n))$
for i : 1 thru n do (
    printf(true, " ~a | ", alphabet[i]),
    for j : 1 thru n do
        printf(true, "~2d ", manual_matrix[i,j]),
    print("")
)$

/* Функція для перевірки, чи є в алфавіті елемент */
contains(element, lst) := block([result, i],
    result : false,
    for i : 1 thru length(lst) do (
        if is(equal(lst[i], element)) then result : true
    ),
    result
```

)$

/* Перевірка лінгвістичного ряду */
print("Перевірка кожного елемента лінгвістичного ряду:")$
for i : 1 thru length(linguistic_sequence) do (
    elem : linguistic_sequence[i],
    in_alphabet : contains(elem, alphabet),
    print("Елемент", elem, "є в алфавіті:", in_alphabet)
)$

Результати:

```
Вхідні дані:
[3.2,7.8,1.5,9.0,4.6]
Діапазон значень:  1.5  –  9.0
Параметр розподілу Пуассона (lambda):  3.0
Розраховані інтервали за Пуассонівським розподілом:
Інтервал  1  :  1.5  –  2.076923076923077  →  A
Інтервал  2  :  2.076923076923077  –  3.807692307692308  →  B

Інтервал  3  :  3.807692307692308  –  6.403846153846154  →  C

Інтервал  4  :  6.403846153846154  –  9.0  →  D
Лінгвістичний ряд:  [B,D,A,D,C]
Кількість елементів:  5
Індекси елементів у алфавіті:  [2,4,1,4,3]
Вручну побудована матриця переходів:
   |   [A,B,C,D]
---+  [---,---,---,---]
 A |   0   0   0   1

 B |   0   0   0   1

 C |   0   0   0   0

 D |   1   0   1   0
```