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 № 5

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:

```
; Шаблони для збереження даних
(deftemplate number
   (slot value))

(deftemplate symbol-mapped
   (slot value)
   (slot symbol))

(deftemplate transition
   (slot from)
   (slot to)
   (slot pair-id))

(deftemplate status
   (slot stage))

(deftemplate sorted-list
   (multislot values))

(deftemplate distribution-params
   (slot lambda))

; Функції для роботи з даними
(deffunction factorial (?n)
   (if (<= ?n 1) then
      (return 1)
   else
      (return (* ?n (factorial (- ?n 1))))
   )
)

(deffunction poisson-pmf (?k ?lambda)
   (bind ?e 2.71828)
   (bind ?numerator (* (** ?lambda ?k) (** ?e (- 0 ?lambda))))
   (bind ?denominator (factorial ?k))
   (/ ?numerator ?denominator)
)

(deffunction poisson-cdf (?k ?lambda)
   (bind ?sum 0)
   (loop-for-count (?i 0 ?k)
      (bind ?sum (+ ?sum (poisson-pmf ?i ?lambda)))
   )
   ?sum
)

(deffunction insert-in-order (?val ?sorted)
```

```
  (bind ?result (create$))
  (bind ?inserted FALSE)
  (foreach ?x ?sorted
    (if (and (not ?inserted) (< ?val ?x)) then
      (bind ?result (create$ ?result ?val))
      (bind ?inserted TRUE))
    (bind ?result (create$ ?result ?x)))
  (if (not ?inserted) then
    (bind ?result (create$ ?result ?val)))
  ?result
)

; Правило для ініціалізації даних
(defrule initialize-data
  (initial-fact)
  =>
  (printout t "=== Перетворення чисельного ряду до лінгвістичного ланцюжка ===" crlf)
  (printout t "Розподіл ймовірностей: Пуассонівський" crlf)

  ; Вхідні дані
  (bind ?input-data (create$ 7.5 2.3 6.7 1.2 8.9 4.2 3.6 9.1 5.8 4.7 2.9 6.1 0.8 7.3 3.4))

  (printout t "Вхідні дані: ")
  (foreach ?val ?input-data
    (printout t ?val " ")
    (assert (number (value ?val)))
  )
  (printout t crlf)
  (printout t "Кількість чисел: " (length$ ?input-data) crlf)
)

; Правило для ініціалізації параметрів розподілу
(defrule initialize-params
  (initial-fact)
  (not (distribution-params))
  =>
  (bind ?lambda 2.0)
  (assert (distribution-params (lambda ?lambda)))
  (printout t "Параметр розподілу Пуассона (lambda): " ?lambda crlf)
)

; Правило для сортування чисел
(defrule sort-values
  (initial-fact)
  (not (sorted-list))
  =>
  (bind ?raw (create$))
  (do-for-all-facts ((?n number)) TRUE
```

```
    (bind ?raw (create$ ?raw ?n:value))
  )
  (bind ?sorted (create$))
  (foreach ?val ?raw
    (bind ?sorted (insert-in-order ?val ?sorted)))
  (assert (sorted-list (values ?sorted)))
  (assert (status (stage ready)))

  (printout t "Відсортований числовий ряд: ")
  (foreach ?val ?sorted
    (printout t ?val " "))
  (printout t crlf)
)

; Функція для відображення чисел на символи за розподілом Пуассона
(deffunction assign-symbols-poisson (?vals ?alphabet ?lambda)
  (bind ?count (length$ ?alphabet))
  (bind ?min (nth$ 1 ?vals))
  (bind ?max (nth$ (length$ ?vals) ?vals))
  (bind ?range (- ?max ?min))

  ; Обчислення інтервалів
  (bind ?interval-size (/ 1.0 ?count))
  (bind ?current-prob 0.0)
  (bind ?k 0)
  (bind ?max-k 20)  ; Обмежуємо максимальне значення k для уникнення переповнення
  (bind ?k-values (create$))

  (printout t "Розрахунок інтервалів:" crlf)

  (loop-for-count (?i 1 ?count)
    (bind ?target-prob (min 1.0 (* ?i ?interval-size)))
    (bind ?found FALSE)

    (while (and (not ?found) (< ?k ?max-k))
      (bind ?cdf (poisson-cdf ?k ?lambda))
      (if (>= ?cdf ?target-prob) then
        (bind ?found TRUE)
      else
        (bind ?k (+ ?k 1))
      )
    )

    (bind ?k-values (create$ ?k-values ?k))
    (bind ?current-prob ?target-prob)

    (printout t "Інтервал " ?i ": k = " ?k ", p = " ?target-prob crlf)
  )
```

```
  ; Нормалізація інтервалів
  (bind ?intervals (create$))
  (foreach ?k ?k-values
    (bind ?normalized-val (+ ?min (* (/ ?k ?max-k) ?range)))
    (bind ?intervals (create$ ?intervals ?normalized-val))
  )
  (bind ?intervals (create$ ?intervals ?max))

  (printout t "Границі інтервалів: ")
  (foreach ?val ?intervals
    (printout t ?val " "))
  (printout t crlf)

  ; Створення відображення
  (bind ?mapping-intervals (create$))
  (bind ?i 1)
  (while (< ?i ?count)
    (bind ?start (nth$ ?i ?intervals))
    (bind ?end (nth$ (+ ?i 1) ?intervals))
    (bind ?sym (nth$ ?i ?alphabet))
    (bind ?mapping-intervals (create$ ?mapping-intervals ?start ?end ?sym))
    (bind ?i (+ ?i 1))
  )

  ; Відображення чисел на символи
  (do-for-all-facts ((?n number)) TRUE
    (bind ?v ?n:value)
    (bind ?j 0)
    (while (< ?j (* (- ?count 1) 3))
      (bind ?a (nth$ (+ ?j 1) ?mapping-intervals))
      (bind ?b (nth$ (+ ?j 2) ?mapping-intervals))
      (bind ?s (nth$ (+ ?j 3) ?mapping-intervals))
      (if (and (>= ?v ?a) (< ?v ?b)) then
        (assert (symbol-mapped (value ?v) (symbol ?s)))
        (bind ?j (* ?count 3))
      else
        (bind ?j (+ ?j 3))
      )
    )

    ; Перевірка для максимального значення
    (if (= ?v ?max) then
      (bind ?last-sym (nth$ ?count ?alphabet))
      (assert (symbol-mapped (value ?v) (symbol ?last-sym)))
    )
  )
)
```

```
; Правило для відображення чисел на символи
(defrule map-values-to-symbols
   ?sorted <- (sorted-list (values $?vals))
   ?status <- (status (stage ready))
   ?params <- (distribution-params (lambda ?lambda))
   =>
   (bind ?alphabet (create$ a b c d e f g h i j))
   (printout t "Використовуваний алфавіт: ")
   (foreach ?sym ?alphabet
      (printout t ?sym " "))
   (printout t crlf)

   (assign-symbols-poisson ?vals ?alphabet ?lambda)
   (modify ?status (stage mapped))

   (printout t "Відображення значень на символи:" crlf)
   (do-for-all-facts ((?s symbol-mapped)) TRUE
      (printout t "Значення " ?s:value " -> символ " ?s:symbol crlf)
   )
)

; Правило для побудови матриці передування
(defrule build-transitions
   ?status <- (status (stage mapped))   ; Fixed: properly define ?status variable
   =>
   (bind ?all-facts (find-all-facts ((?s symbol-mapped)) TRUE))
   (bind ?sorted-facts (create$))

   ; Сортування фактів за значеннями
   (foreach ?fact ?all-facts
      (bind ?val (fact-slot-value ?fact value))
      (bind ?idx 1)
      (bind ?inserted FALSE)
      (while (and (<= ?idx (length$ ?sorted-facts)) (not ?inserted))
         (bind ?curr-val (fact-slot-value (nth$ ?idx ?sorted-facts) value))
         (if (< ?val ?curr-val) then
            (bind ?sorted-facts (insert$ ?sorted-facts ?idx ?fact))
            (bind ?inserted TRUE)
         )
         (bind ?idx (+ ?idx 1))
      )
      (if (not ?inserted) then
         (bind ?sorted-facts (create$ ?sorted-facts ?fact))
      )
   )

   ; Створення переходів
```

```
  (bind ?len (length$ ?sorted-facts))
  (loop-for-count (?i 1 (- ?len 1))
    (bind ?from (fact-slot-value (nth$ ?i ?sorted-facts) symbol))
    (bind ?to (fact-slot-value (nth$ (+ ?i 1) ?sorted-facts) symbol))
    (assert (transition (from ?from) (to ?to) (pair-id ?i)))
  )
  (modify ?status (stage transitions))
)

; Правило для виведення лінгвістичного ряду
(defrule print-symbol-sequence
  (status (stage transitions))
  =>
  (printout t crlf "Лінгвістичний ряд: ")
  (bind ?sorted-facts (create$))
  (bind ?all-facts (find-all-facts ((?s symbol-mapped)) TRUE))

  ; Сортування фактів за значеннями
  (foreach ?fact ?all-facts
    (bind ?val (fact-slot-value ?fact value))
    (bind ?idx 1)
    (bind ?inserted FALSE)
    (while (and (<= ?idx (length$ ?sorted-facts)) (not ?inserted))
      (bind ?curr-val (fact-slot-value (nth$ ?idx ?sorted-facts) value))
      (if (< ?val ?curr-val) then
        (bind ?sorted-facts (insert$ ?sorted-facts ?idx ?fact))
        (bind ?inserted TRUE)
      )
      (bind ?idx (+ ?idx 1))
    )
    (if (not ?inserted) then
      (bind ?sorted-facts (create$ ?sorted-facts ?fact))
    )
  )

  ; Виведення лінгвістичного ряду
  (foreach ?fact ?sorted-facts
    (printout t (fact-slot-value ?fact symbol) " ")
  )
  (printout t crlf)
)

; Правило для виведення матриці передування
(defrule print-transition-matrix
  (status (stage transitions))
  =>
  (bind ?alphabet (create$ a b c d e f g h i j))
  (printout t crlf "Матриця передування:" crlf)
```

```
  (printout t "    ")
  (foreach ?col ?alphabet
    (printout t ?col " "))
  (printout t crlf)

  (foreach ?row ?alphabet
    (printout t ?row ": ")
    (foreach ?col ?alphabet
      (bind ?count (length$ (find-all-facts ((?t transition))
        (and (eq ?t:from ?row) (eq ?t:to ?col)))))
      (printout t ?count " "))
    (printout t crlf))
)
```

Результати:

```
CLIPS> (run)
=== Перетворення чисельного ряду до лінгвістичного ланцюжка ===
Розподіл ймовірностей: Пуассонівський
Вхідні дані: 7.5 2.3 6.7 1.2 8.9 4.2 3.6 9.1 5.8 4.7 2.9 6.1 0.8 7.3 3.4
Кількість чисел: 15
Параметр розподілу Пуассона (lambda): 2.0
Відсортований числовий ряд: 0.8 1.2 2.3 2.9 3.4 3.6 4.2 4.7 5.8 6.1 6.7 7.3 7.5 8.9 9.1
Використовуваний алфавіт: a b c d e f g h i j
Розрахунок інтервалів:
Інтервал 1: k = 0, p = 0.1
Інтервал 2: k = 1, p = 0.2
Інтервал 3: k = 1, p = 0.3
Інтервал 4: k = 1, p = 0.4
Інтервал 5: k = 2, p = 0.5
Інтервал 6: k = 2, p = 0.6
Інтервал 7: k = 3, p = 0.7
Інтервал 8: k = 3, p = 0.8
Інтервал 9: k = 4, p = 0.9
Інтервал 10: k = 12, p = 1.0
Границі інтервалів: 0.8 1.215 1.215 1.215 1.63 1.63 2.045 2.045 2.46 5.78 9.1
Відображення значень на символи:
Значення 2.3 -> символ h
Значення 1.2 -> символ a
Значення 4.2 -> символ i
Значення 3.6 -> символ i
Значення 9.1 -> символ j
Значення 4.7 -> символ i
Значення 2.9 -> символ i
Значення 0.8 -> символ a
Значення 3.4 -> символ i

Лінгвістичний ряд: a a h i i i i i j

Матриця передування:
   a b c d e f g h i j
a: 1 0 0 0 0 0 0 1 0 0
b: 0 0 0 0 0 0 0 0 0 0
c: 0 0 0 0 0 0 0 0 0 0
d: 0 0 0 0 0 0 0 0 0 0
e: 0 0 0 0 0 0 0 0 0 0
f: 0 0 0 0 0 0 0 0 0 0
g: 0 0 0 0 0 0 0 0 0 0
h: 0 0 0 0 0 0 0 1 0
i: 0 0 0 0 0 0 0 4 1
j: 0 0 0 0 0 0 0 0 0 0
CLIPS>
```