

Міністерство освіти і науки України
Національний технічний університет України «КПІ»
імені Ігоря Сікорського

ЗВІТ
з лабораторної роботи №2
з дисципліни «Мультипарадигмне програмування»

Виконав:
Студент 3 курсу кафедри ОТ ФІОТ,
Навчальної групи ІО-23
Прохоренко Артем

Київ 2025

Завдання: на мові функціонального програмування реалізувати перетворення чисельного ряду до лінгвістичного ланцюжка за певним розподілом ймовірностей потрапляння значень до інтервалів з подальшою побудовою матриці передування.

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

Вихідні дані: лінгвістичний ряд та матриця передування.

Мова програмування: Racket.

Варіант: 20

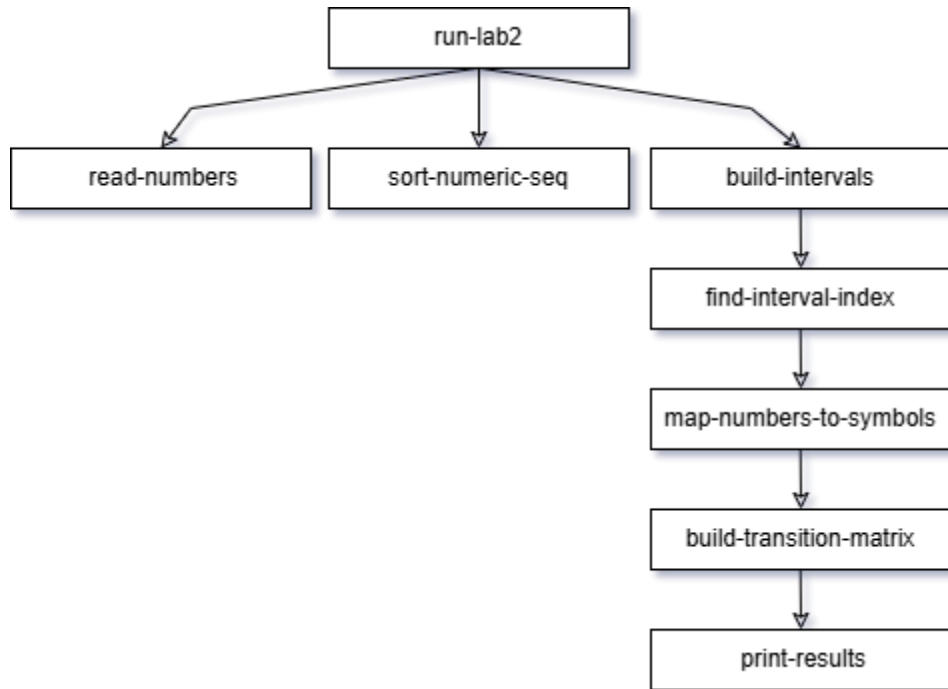
Розподіл ймовірностей: Розбиття на рівні інтервали

Хід розв'язання задачі:


Основні етапи роботи програми:

№	Етап	Опис
1	Зчитування даних	Функція <code>read-numbers-from-file</code> читає числа з текстового файлу та формує список.
2	Сортування	Функція <code>sort-numeric-sequence</code> сортує ряд для визначення діапазону значень.
3	Побудова інтервалів	<code>build-intervals</code> формує рівномірні інтервали згідно з потужністю алфавіту.
4	Відображення чисел	<code>find-interval-index</code> та <code>map-numbers-to-symbols</code> перетворюють числа у символи.
5	Матриця передування	<code>build-transition-matrix</code> підраховує переходи між символами.
6	Виведення результатів	<code>print-results</code> виводить лінгвістичний ряд і матрицю переходів.

Функціональна схема



Перший числовий ряд (5 значень – 3 символи): 3.2, 7.8, 1.5, 9.0, 4.6

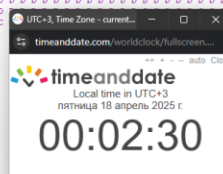


Другий числовий ряд (5000 значень - 5 символів): B-C-D-E-Gold Futures Historical Data (Price)

[illegible]

	A	B	C	D	E
A:	1813		5	0	0
B:	6	631		14	0
C:	0	15	937		30
D:	0	0	30	1072	
E:	0	0	0	9	428

Третій числовий ряд (5000 значень - 10 символів): B-C-D-E-S&P 500 Historical Data (Price)

[illegible]

Матриця передування:

```

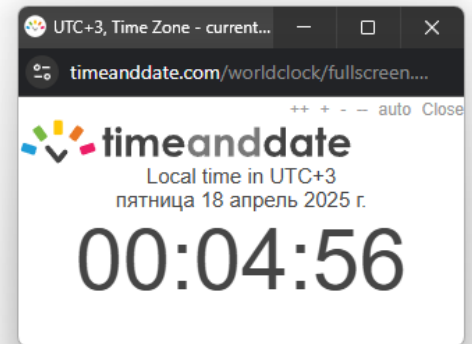
      A  B  C  D  E  F  G  H  I  J
A: 241  26  0  0  0  0  0  0  0  0
B: 26   752  33  0  0  0  0  0  0  0
C: 0   33  1497  38  0  0  0  0  0  0
D: 0   0   39  907  2  0  0  0  0  0
E: 0   0   0   3  169  4  0  0  0  0
F: 0   0   0   0   5  314  12  0  0  0
G: 0   0   0   0   0  13  401  2  0  0
H: 0   0   0   0   0  0  3  179  1  0
I: 0   0   0   0   0  0  0  2  119  10
J: 0   0   0   0   0  0  0  0  11  157
>

```

```

> (define test-nums (read-numbers-from-file "1.txt"))
> (displayln test-nums)
(3.2 7.8 1.5 9.0 4.6)
> (define test-sorted (sort-numeric-sequence test-nums))
> (displayln test-sorted)
(1.5 3.2 4.6 7.8 9.0)
> (define test-alphabet '(A B C))
> (displayln test-alphabet)
(A B C)
> (define test-intervals (build-intervals test-sorted test-alphabet))
> (displayln test-intervals)
((1.5 4.0) (4.0 6.5) (6.5 9.0))
> (define test-symbols (map-numbers-to-symbols test-nums test-intervals test-alphabet))
> (displayln test-symbols)
(A C A C B)
> (define test-matrix (build-transition-matrix test-symbols test-alphabet))
> (for-each (lambda (row) (displayln (vector->list row)))
  (vector->list test-matrix))
(0 0 2)
(0 0 0)
(1 1 0)
>

```



Лістинг програмного тексту

```
#lang racket

;; Зчитування чисел із текстового файлу
(define (read-numbers-from-file path)
  ;; Зчитуємо вміст файлу як рядок
  (define content (file->string path))
  ;; Розбиваємо рядок по пробілах
  (define parts (string-split content))
  ;; Перетворюємо кожен частину в число
  (map string->number parts))

;; Сортуювання числового ряду
(define (sort-numeric-sequence seq)
  (sort seq <))

;; Побудова інтервалів за рівномірним розподілом
(define (build-intervals sorted-seq alphabet)
  (define min-val (first sorted-seq))
  (define max-val (last sorted-seq))
  (define range (- max-val min-val))
  (define step (/ range (length alphabet)))
  ;; Повертаємо список інтервалів у вигляді пар [a, b)
  (for/list ([i (in-range (length alphabet))])
    (list (+ min-val (* i step)) (+ min-val (* (+ i 1) step)))))

;; Знаходження індексу інтервалу, в який потрапляє значення
(define (find-interval-index value intervals)
  (let loop ([i 0] [ints intervals])
    (cond
      [(null? ints) (- (length intervals) 1)]
      [(and (<= (first (first ints)) value)
            (< value (second (first ints)))) i]
      ;; Додаткова перевірка на крайній правий кінець
      [(and (= i (- (length intervals) 1))
            (= value (second (first ints)))) i]
      [else (loop (+ i 1) (rest ints))]))

;; Відображення чисел на символи алфавіту
(define (map-numbers-to-symbols seq intervals alphabet)
  (map (lambda (x)
        (list-ref alphabet (find-interval-index x intervals)))
    seq))
```

```

;; Побудова матриці передування
(define (build-transition-matrix symbol-seq alphabet)
  (define size (length alphabet))
  ;; Створюємо матрицю як вектор векторів, заповнений нулями
  (define table
    (build-vector size
      (lambda (_)
        (make-vector size 0)))))
  ;; Проходимо по всіх парах символів у ряді
  (for ([i (in-range (- (length symbol-seq) 1))])
    (let* ([a (index-of alphabet (list-ref symbol-seq i))]
           [b (index-of alphabet (list-ref symbol-seq (+ i 1)))]])
      (vector-set! (vector-ref table a) b
        (+ 1 (vector-ref (vector-ref table a) b)))))
  table)

;; Основна функція
(define (run-lab2 filepath alphabet)
  (define numeric-seq (read-numbers-from-file filepath))
  (define sorted (sort-numeric-sequence numeric-seq))
  (define intervals (build-intervals sorted alphabet))
  (define symbols (map-numbers-to-symbols numeric-seq intervals
alphabet))
  (define matrix (build-transition-matrix symbols alphabet))
  (values symbols matrix))

;; Функція для виводу результатів
(define (print-results symbols matrix alphabet)
  (displayln "Лінгвістичний ряд:")
  (displayln symbols)
  (newline)
  (displayln "Матриця передування:")
  ;; Виводимо заголовки стовпців
  (display " ")
  (for-each (lambda (c) (display (format "~a " c))) alphabet)
  (newline)
  ;; Виводимо кожен рядок з матриці
  (for ([i (in-range (length alphabet))])
    (display (format "~a: " (list-ref alphabet i)))
    (for ([j (in-range (length alphabet))])
      (define val (vector-ref (vector-ref matrix i) j))
      (display (format "~a " val)))
    (newline)))

;; === Параметри запуску ===

```



```
(define alphabet '(A B C)) ;; Можна змінювати розмір і склад
(define filepath "1.txt")  ;; Ім'я вхідного файлу

;; === Запуск ===
(define-values (symbols matrix) (run-lab2 filepath alphabet))
(print-results symbols matrix alphabet)
```