Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №2
з дисципліни
Мультипарадигменне програмування

Виконала:
Студентка групи ІА-21
Палінчак І.В.

Перевірив
доц. каф. ІПІ
Баклан І.В.

Київ 2025

ЛАБОРАТОРНА РОБОТА №2

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 дані: чисельний ряд, вид розподілу ймовірностей, потужність алфавіту.

Вихідні дані: лінгвістичний ряд та матриця передування.

Мова програмування: Racket, Common Lisp, Clojure..

	A	B	...	Z
A	$a_{1,1}$	$a_{1,2}$	...	$a_{1,26}$
B	$a_{2,1}$	$a_{2,2}$	...	$a_{2,26}$
...	...	...	...	...
Z	$a_{26,1}$	$a_{26,2}$	...	$a_{26,26}$

де □$_{1,2}$ – кількість в лінгвістичному ряду випадків літери B після літери A.

№ варіанту	Розподіл ймовірностей
11	Розподіл Райса

Виконання лабораторної

Розв’язання задачі (варіант 11 — розподіл Райса)

1. **Ввід даних**
   - **Запитується потужність алфавіту *n* (ціле число, ≥ 2).**
   - **Зчитується CSV-файл, обробляється стовпець "Price":**

- **Видаляються лапки та розділювачі розрядів (,) — наприклад, "1,203.30" → 1203.30.**
- **Перетворюються на числове значення.**

- **Отримується числовий ряд numbers довжиною *m*.**

2. **Обчислення діапазону**
   - **numbers сортується в зростаючому порядку.**
   - **Визначаються:**
     - **min_val = numbers[0] — мінімальне значення.**
     - **max_val = numbers[m-1] — максимальне значення.**
3. **Створення інтервалів (розподіл Райса)**
   - **Якщо min_val == max_val, то всі значення відображаються на перший символ алфавіту.**
   - **Інакше:**

1. **Параметри розподілу**

   Обчислюємо динамічні параметри:

$$\nu = \frac{max\_val - min\_val}{4}, \quad \sigma = \frac{max\_val - min\_val}{6}.$$

2. **Квантилі Rice**

   Для ймовірностей :

$$p_i = \frac{i}{n}, \quad i = 0, 1, \ldots, n$$

чисельно обчислюємо зворотну функцію розподілу (inverse CDF або “quantile”)

$$q_i = F^{-1}(p_i; \nu, \sigma)$$

методом бінарного пошуку по чисельно інтегрованій CDF.

3. Масштабування в інтервал даних

Вважаємо qmax=qnq_{\max}=q_nqmax=qn.

Кожен квантиль масштабуємо лінійно:

$$b_i = min\_val + \frac{q_i}{q_{\max}} (max\_val - min\_val), \quad i = 0, \ldots, n.$$

Отримуємо межі інтервалів $\{b_0, b_1, \ldots, b_n\}$.

Перетворення числового ряду в лінгвістичний

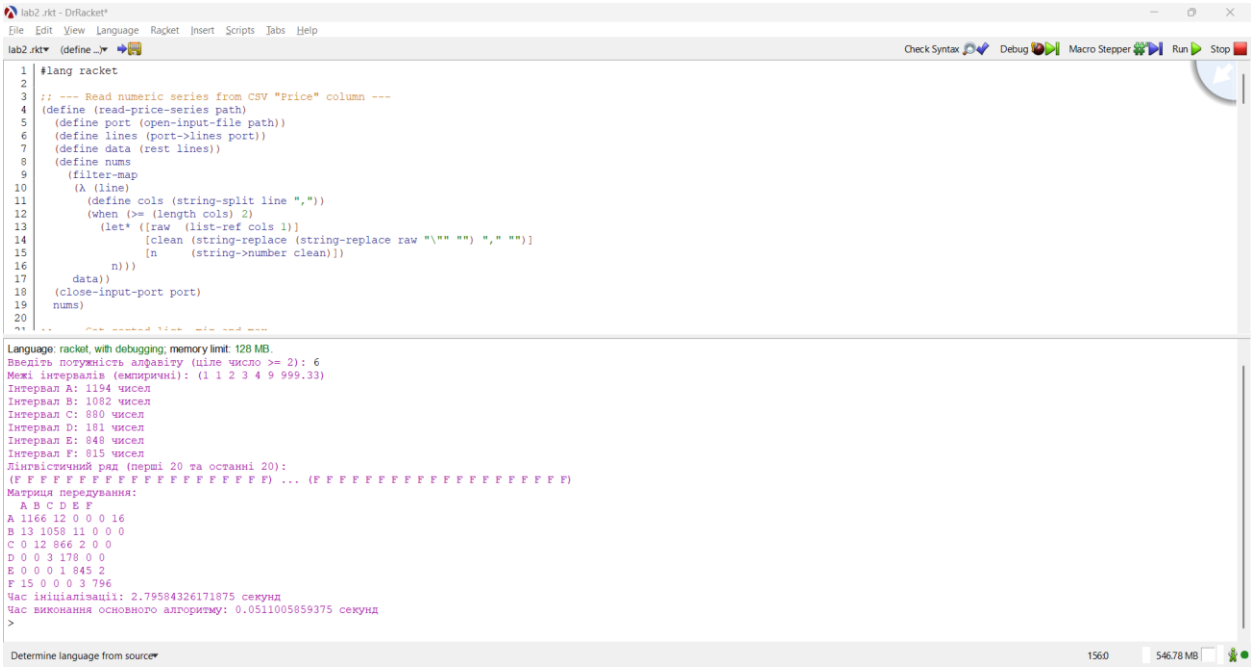
- **Будується алфавіт із *n* символів: {A,B,… }**
- **Для кожного x∈numbers знаходимо індекс *i* такий, що** $b_i \le x \le b_{i+1}$**,**
  і прив’язуємо до нього символ alphabet[i].
- **Формується лінгвістичний ряд ling_seq довжиною *m*.**

4. Побудова матриці передування

- **Ініціалізується нульова матриця розміром *n* × *n*.**
- **Для кожної суміжної пари символів (si,si+1) у ling_seq збільшуємо елемент** $matrix[\text{index}(s_i)][\text{index}(s_{i+1})]$ **на 1.**

5. **Вивід результатів**
    - **Ме́жі інтервалів b0,…,bn.**
    - **Розподіл чисел по інтервалах (кількість у кожному).**
    - **Лінгвістичний ряд (перші 20 і останні 20 символів).**
    - **Матриця передування.**
    - **Час ініціалізації (зчитування даних, побудова інтервалів).**
    - **Час виконання (побудова лінгвістичного ряду та матриці).**

Скріншоти виконання



Лістинг коду

Для реалізації завдання було обрано мову Racket

```
#lang racket

;; --- Read numeric series from CSV "Price" column ---
(define (read-price-series path)
  (define port (open-input-file path))
  (define lines (port->lines port))
  (define data (rest lines))
  (define nums
    (filter-map
     (λ (line)
       (define cols (string-split line ","))
       (when (>= (length cols) 2)
         (let* ([raw  (list-ref cols 1)]
                [clean (string-replace (string-replace raw "\"" "") "," "")]
                [n     (string->number clean)])
           n)))
     data))
  (close-input-port port)
  nums)

;; --- Get sorted list, min and max ---
```

```
(define (get-sorted-range nums)
  (define sorted (sort nums <))
  (values sorted (first sorted) (last sorted)))

;; --- Empirical intervals by data quantiles ---
(define (create-empirical-intervals sorted n)
  (define m (length sorted))
  (define boundaries
    (for/list ([i (in-range (add1 n))])
      (define idx
        (inexact->exact
         (min (sub1 m)
              (floor (* i m (/ 1.0 n))))))
      (list-ref sorted idx)))
  boundaries)

;; --- Alphabet of size n ---
(define (build-alphabet n)
  (if (<= n 26)
      (for/list ([i (in-range n)])
        (string (integer->char (+ 65 i))))
      (for/list ([i (in-range n)])
        (format "A~a" i))))

;; --- Count numbers in each interval ---
(define (count-intervals nums intervals alpha)
  (define counts (make-vector (length alpha) 0))
  (for ([x nums])
    (define idx
      (or (for/first ([i (in-range (sub1 (length intervals)))]
                      #:when (<= x (list-ref intervals (add1 i))))
            i)
          (sub1 (length alpha))))
    (vector-set! counts idx (add1 (vector-ref counts idx))))
  counts)

;; --- Map number to symbol ---
(define (number->letter x intervals alpha)
  (or (for/first ([i (in-range (sub1 (length intervals)))]
                  #:when (<= x (list-ref intervals (add1 i))))
        (list-ref alpha i))
      (last alpha)))

;; --- Build linguistic sequence ---
(define (numeric->ling nums intervals alpha)
  (map (λ (x) (number->letter x intervals alpha)) nums))

;; --- Adjacent pairs for transitions ---
(define (in-adjacent-pairs seq)
```

```
  (for/list ([i (in-range (sub1 (length seq)))])
    (list (list-ref seq i) (list-ref seq (add1 i)))))

;; --- Build transition matrix ---
(define (build-matrix seq alpha)
  (define n (length alpha))
  (define mat (make-vector (* n n) 0))
  (for ([pr (in-adjacent-pairs seq)])
    (define i (index-of alpha (first pr)))
    (define j (index-of alpha (second pr)))
    (when (and i j)
      (vector-set! mat (+ (* i n) j)
                   (add1 (vector-ref mat (+ (* i n) j))))))
  mat)

;; --- Print matrix with headers ---
(define (print-matrix mat alpha)
  (define n (length alpha))
  (printf "Матриця передування:\n")
  (printf "  ~a\n" (string-join alpha " "))
  (for ([i (in-range n)])
    (printf "~a ~a\n"
            (list-ref alpha i)
            (string-join
             (for/list ([j (in-range n)])
               (number->string (vector-ref mat (+ (* i n) j))))
             " "))))

;; --- Read alphabet size from user ---
(define (get-alphabet-size)
  (printf "Введіть потужність алфавіту (ціле число >= 2): ")
  (let ([inp (string->number (read-line))])
    (if (and (integer? inp) (>= inp 2))
        inp
        (begin
          (printf "Помилка: введіть ціле число >= 2\n")
          (get-alphabet-size)))))

;; --- Main solver with timing ---
(define (solve-task csv-file)
  ;; Ініціалізація
  (define init-start (current-inexact-milliseconds))
  (define n     (get-alphabet-size))
  (define alpha (build-alphabet n))
  (define nums  (read-price-series csv-file))
  (define-values (sorted mn mx) (get-sorted-range nums))
  (define intervals (create-empirical-intervals sorted n))
  (define counts (count-intervals nums intervals alpha))
  (define init-end (current-inexact-milliseconds))
```

```
  (printf "Межі інтервалів (емпиричні): ~a\n" intervals)
  (for ([i (in-range (length alpha))])
    (printf "Інтервал ~a: ~a чисел\n"
            (list-ref alpha i)
            (vector-ref counts i)))

  ;; Основне виконання
  (define exec-start (current-inexact-milliseconds))
  (define seq (numeric->ling nums intervals alpha))
  (define mat (build-matrix seq alpha))
  (define exec-end (current-inexact-milliseconds))

  ;; Повертаємо всі дані та часи
  (values alpha seq mat
          (/ (- init-end init-start) 1000.0)
          (/ (- exec-end exec-start) 1000.0)))

;; --- Execute ---
(define csv-file "B-C-D-E-F-Dow Jones Industrial Average Historical
Data.csv")
(define-values (alpha seq mat t-init t-exec)
  (solve-task csv-file))

;; Вивід лінгвістичного ряду
(define k 20)
(printf "Лінгвістичний ряд (перші ~a та останні ~a):\n~a ... ~a\n"
        k k
        (take seq (min k (length seq)))
        (take-right seq (min k (length seq))))

;; Вивід матриці
(print-matrix mat alpha)

;; Вивід часу
(printf "Час ініціалізації: ~a секунд\n" t-init)
(printf "Час виконання основного алгоритму: ~a секунд\n" t-exec)
```

Висновок

В данній лабораторній роботі реалізовано метод обробки числового ряду з файлу "B-C-D-E-F-Dow Jones Industrial Average Historical Data.csv" та перетворення його в лінгвістичний ряд за допомогою розподілу Райса. Було:

1. Розроблено і протестовано власну реалізацію щільності (PDF), функції розподілу (CDF) та оберненого розподілу (quantile) розподілу Райса з чисельною інтеграцією й бінарним пошуком.
2. Динамічно підібрано параметри ν\nuν та σ\sigmaσ на основі розмаху даних, що забезпечило адекватне масштабування квантилів.
3. Побудовано межі інтервалів {bi} шляхом лінійного масштабування квантилів у діапазон [minf_0,maxf_0].
4. Здійснено відображення кожного значення ряду на символ з алфавіту розмірності *n* та побудовано лінгвістичний ряд.
5. Побудовано матрицю передування, що відображає частоти переходів між символами.
6. Виміряно час ініціалізації (завантаження та підготовка інтервалів) і час основного алгоритму (перетворення та обчислення матриці).

Отримане рішення демонструє застосування статистичного підходу (розподіл Райса) до дискретизації числового ряду та забезпечує інформаційне представлення даних у вигляді лінгвістичної послідовності й матриці переходів. Це дозволяє досліджувати частоти й стани системи на основі аналізу часових рядів у функціональному стилі програмування на Racket.