Міністерство освіти і науки України КПІ ім. Ігоря Сікорського Факультет Інформатики та Обчислювальної Технiки

**ЗВІТ**

**Лабораторна робота №2 з дисципліни**

**«Мультипарадигменне програмування»**

|  |  |
| --- | --- |
| Перевірив:  Викладач кафедри ІПІ, ОТ  ФІОТ  Очеретяний О.К | Виконав:  Климентьєв Нікіта  гр. ІК-21 |

2025

**ЛАБОРАТОРНА РОБОТА №2**

**Завдання:** на процедурній мові 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.

**Хід розв’язання задачі:**

1. Чисельний ряд сортується від найменшого значення до найбільшого. Тим самим отримуємо діапазон (область) припустимих значень.

2. ОПЗ розбивbаємо на інтервали (кількість залежить від потужності обраного алфавіту) у відповідності до розподілу ймовірностей, при цьому враховуємо, що ймовірність потрапляння до інтервалу P[a,b]=F(b)-F(a).

3. Кожному числовому значенню ставиться у відповідність знак з алфавіту, який має той самий індекс, що і інтервал.

4. Після заміни отриманий ряд літер видається на вихід.

5. На основі лінгвістичного ряду будується матриця

**Обраний варіант:**

|  |  |
| --- | --- |
| **№ варіанту** | **Розподіл ймовірностей** |
| **20** | Розбиття на рівні інтервали |

**Розв`язання задачі:**

*Вхідні дані:*

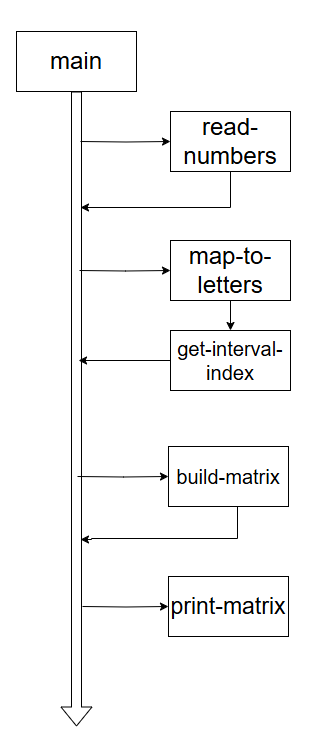
* Потужність алфавіту (наприклад, 3 для літер A, B, C).
* Один рядок чисел, розділених пробілами (наприклад, "78.07 77.11 77.39 ...").

*Процес:*

* Зчитує числа з одного рядка без вказівки їх кількості.
* Визначає діапазон (мін. і макс. значення) числового ряду.
* Ділить діапазон на рівні інтервали залежно від потужності алфавіту.
* Відображає кожне число на літеру (A, B, C тощо) відповідно до інтервалу.
* Формує лінгвістичний ланцюжок із цих літер.
* Будує матрицю передування, підраховуючи, як часто одна літера слідує за іншою.

*Вихідні дані:*

* Лінгвістичний ланцюжок (наприклад, "BCBBC...").
* Матриця передування (наприклад, скільки разів 'A' слідує за 'B').

Рис. 1 – Функціональна схема

*Пояснення схеми*

* **main**: Головна функція, яка координує роботу програми. Вона зчитує потужність алфавіту, викликає функції для зчитування чисел, їхнього перетворення на літери, побудови матриці передування та її виведення.
* **read-numbers**: Зчитує числа з вхідних даних (кожне число з нового рядка).
* **map-to-letters**: Перетворює числа на літери (A, B, C, ...) на основі рівних інтервалів. Для цього використовує допоміжну функцію get-interval-index.
* **get-interval-index**: Обчислює індекс інтервалу для кожного числа залежно від мінімального значення, ширини інтервалу та потужності алфавіту.
* **build-matrix**: Будує матрицю передування, аналізуючи послідовність літер у лінгвістичному ланцюжку.
* **print-matrix**: Виводить матрицю передування з позначками літер.

### *Взаємозв’язок функцій*

* **main** є центральною функцією, яка:
  + Викликає **read-numbers** для отримання списку чисел.
  + Викликає **map-to-letters** для перетворення чисел у лінгвістичний ланцюжок (ця функція використовує **get-interval-index**).
  + Викликає **build-matrix** для створення матриці передування.
  + Викликає **print-matrix** для виведення результату.

**Результат розрахунку:**

Результат розрахунку з потужностю алфавіта 10 і першими 100 числами з числового ряду:

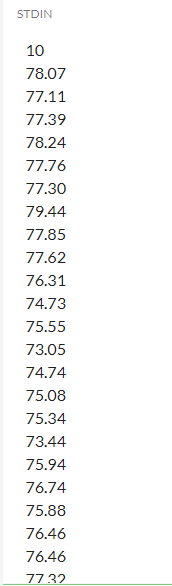


Рис. 2 – Вхідні дані

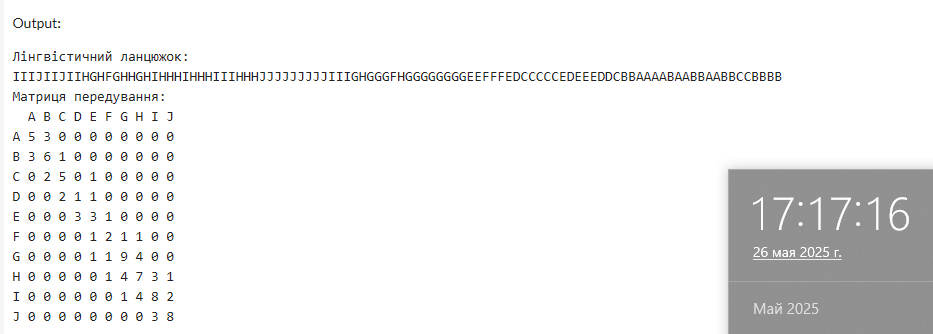


Рис. 3 – Результат розрахунку

Результат розрахунку з потужностю алфавіта 26 і всіма(5000) числами з числового ряду:

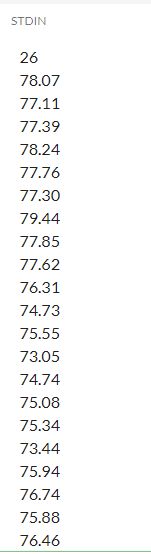


Рис. 4 – Вхідні дані

Лінгвістичний ланцюжок:

NMMNMMNMMMMMMMMMMMMMMMMMMMMMMMMMNNNNNNNNNMMMMMMMMMMMMMMMMMMLLLLLLLKLLLLLLLLLLLKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKLLLLLLLLLLLLLLLLLLLLLLLLKKKKKKKKKKKKKKKKJJKKKKKKKKKKKJKJJJKKKKKKJJJJJJJJJJJJJJJJIIIIIIIIIJJJJJIIIIIIIIIIIIIIIIIHIHIIIHHIHHHHIHIIIIIIHIIHHHHHHHHHHHHHHHHHHHHHHGGGGGGHHHHHHHHHHHHHHHIIHIIIIIIHHHHHHHHHHHHHHHIHIIIIIIIIIIIIIIIIIHHHHHHHHHHHHHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIHGHHHHHHHGGHGGGGGGGGHHHHHHHHHHIHHIIHIIIIHHHHHHGHGHHGGGGGHHHHHHHHGHHHHHHHHHHHHHHGGGGGGGGGGGGGGGGHGHHHGHGHGHHHHHHHHHHHHHHHHHHHIIHHHHHHHHHHHHHHHHHHHHGGGGGGGHHHGGGGGGGGGGGGGFFFFFFFFFFFGGFGFFFFFFFFFFFFEEEEEEEEEEEEDDDEEEEEEEEEEDEDDDDDEDDEEEEFFFFFFFFEFFFFFFFFFFFGGGGGGGGGGGGGGGGGGHHHHHHHHHHHHHHHHHHHHHHIIHIHHHHHHHHHHHHHHGGHHHHHHHHHIHHGGGGGHHHHHHHHHHHHHIIIIIIIIIIIJIJJJJIIIJJJKJKKKKKKKKKKKKKKKKJKKKKKJKKKKKKKKKKKKKKLKKKKKKKKKKJKKKJJJJIIJJIIIIIJIIIIIIIIIIJIJJJJJJKJJJJJJJKJJIIIJJIIJIIHHHHHHHHHHHHGHHHHHIIJJJJJJJJJJJJJKKKKLLLLLLLMNNNNNNNNMNNNONNNOOOOOOOOOOOOOOOPPPPPPPPQQQQQQQQQQQRQQQQRRRRRRRRRRRRRRRRRRRSRSSSSSSSSSSSSSSSSSSSSSSSSSTTTTTTTTTTTUUTTTTTTTTSSSSSSTTTTTTTTSTTTSSSSSSSSSSSSTTSSTTTSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSTSSTTTTTTTSSSSSSTSSSSSSSSSSSSSSSSSSSSSSSSSSTTTTTTTTTTSTSSTSSTTTTTTTTTTTTSSSSSSSSSRSSSSSTSTSSSTTTSTTTTTSTTTSSSSSSSSSSSTSTTTTTTUUUUTTUUTTTTTTTTTTTSSSSSSSTSSSSSSSSSSSSSSSSSSSSSRRRRRRRRRSSSSSRRRSRRRRRRRSRRRRRSSRRRRRSSSSRRRRRRRRRRRRQRRRSSSSRSSTTTTSSSSSSTTSSTTTTTTTTTTTTTTUUUUUUUUUUUUUUUUTTTTTTTTTTTTTTTTTTTTTTTTSSTTSSSSTSSSSSTTTTTTTTTTTTSTTSSSSSTSSSSSSTSSSSTTTUUTUTTTTTSTTTTTTTTTSTTUUUUUTTTTUTTTTTTUUTTTUUTTTTTTTSSSSSSSSRRSSSRRRRRQRQRRRQQPPPPPPPQQQQQQQRRRQQQRRSSSSSSSSSTTTTTTTTTUUUUUVUUUUUUUUVVVUVVVVWVVVWWVVVVWWVVWWWWVVVVWVVVWVVVVUVUUUUUUUUTTTTTTTTTTTTTTTTTTTTTTTTSSSSSSSSRRSSTSSSTTTTSTTSSSSSSSSTTTTTTUTTTSTTTTSTTTTSTTTTTTSSSRRRRRRSRRSTTSTUTTTTTUTTTTUTTTTTSSSSTTTSSSRRSSTUUUUUUUUUUUUUUUUUUUTTTTTTSSSSTTTTTUVUUUUUTUUTUTUUUTTTTTTTTTTUUSTVVVWVVVVVVVVVVVVVWVVVVUUUUUUUUUUTUTSTTUUTUUUUUTTTTSSRRRRRRRRRRRRRRRRQQQQQQQQQQQQQQPQQPQQPQQQQQPPPPPPPPPPPPPPPPOOOOONOOONOOOPPOOOOOOONONOONNONONOOOOOOOONONNNNNNMNNNNNNNNMNMMMMMMMMMMLMMMMMMMMMNNNNNNNNMMMMMMMMMMMMMMMMMLLLLMMMNMMMNNNNMMMMMLLLMMLMMMLLLLLMMMMNNNNNNNOPOOOOOOOOOOOOOOOOOOOOONNNNNNNNNNNNMNNNNNNNNNMMMNMNNMMMLMMLLLLLMMMLLLMMLMMMMMMNNNNNNNNMMMMMMMMMMMLLLLLMMMNMNNMMNMMMMNNNMMMMMMMNNMMMMMMNNNMMMMMLLLLKLLLLLKKKKLLLLLLKKLLLLKKKLLLLLLLMMMMLLLMMLMMMMMMLLKLLLLKKKKKKJJJJJKKKKLLLLLLLKLLLLLLLLLLLLKLLKKKJJJJJJJIIJJJJIIIIIHHHHHHHHHIIHIIIHHIIIHHHIIHIIHHHHGGGFGGGGGGGGGGGFGGFFGGGGGGGGGGGGGGGHHGGGGGGGGGGGGHHHGFFFFFGGGGGGGHGGGFGGGHIIIHIHHHHIIIIIJJJJKJKKKJJJKKLLLKLMMMNOOOPPQQQRSRRSRQQPPQQQRRRSSSSTTUTTTVTTTTTTTTTUUUVVVWVWVWWWXWXYYZZZYYZZZZYYYYXYYXXYXXXYXXXYWVVWWWXWXXXXWVVVVVVWVVVUTTTTUUTUUTTTTTTSSSSSSRRRRRSRRRRRSRSSSSRRRRQRRRQRQQQQQQQQPPPPOOPPPPPPPPPOOOPPPPPPPQQQQQQQQQQPPPPPPPPQPOPPOPPOPPPQQQQQPPPPPPPPPPPPPPOPPOONNOONONNNNMMNNMMMNNMMNNNNMMMMMMMMMMMMLLLLLLLLLLLLLLLLLLLLMMMMMMMMMMMMMMMMMMMMMMMMLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLKKKKKKKKKKLLLLKLKKKKKLLLKKLLLLLLKKKKKJJJJJJJJJJKJJJJJJJJJJJJJJJIIJJJJJJIJIIIIIIIIHIIIIHIIIIIJJJJJJKKKKJKJJJJJKKKKKKKJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJKKJJJJJJJKKJKKKKKKLLLLLLLLLLMMLLMMMMMMMNMMMMMMMMMMMMMMMMMMMMMMMLMMMLLLLLLLLLKKKLLLLLLLLLLLLLLLLLLLLLLLLLLLMMMLLLLMMMMLLLLLLKLKKKKKKKKKKJJJKKKKJJJJJJKKKJJKJJJJJJJJJJJJJKKKKKKKKKKKKKKKKJKJJJKJJJJJJIIIIIIIJJJJJJIJJIIIIIIIIIIIIIIIIIIJJJJJJJJJJJJJJJJJJJJJJJJJJKKKKKKKKKKKJKKJJKKKKKKLKKKKKKKKKKKKKKKJKJJJJJJJJJJJIIJIJJJJJJJJJJJIIJJJJIJJJIIIIIIIIIIIIHHHHHHHHHHHHHHHHHHHHHIIIIIIIIHHHHIIIIIIIIIIIIIIIIIIIIIIIIIHIHHHHHHHHGGGGGGGGGGGGGGGGGGHGGGGGGGGGGGGGFFFFFFFFGGGGGFFFFFFFFFGGGGGGGGGGGGFGGGGGGGHGHHHHHHHHHHHHHHHHHHHHGGGGGGGGGGGGFGGFFGFFFFGGFFFFGGGGGGGGGGFGFFFFFFFFFFFFFFFFFFFFFFFFFEEEEEEEEEEFFEEEEEEEEEFFFFFFFFFFFFFFFFFEFFFEEEEEEEEEEEEEEEEEEDDEEEEEEEEEEEEEEEEEEEEEEEEEEEDDDDEDDDDDDDDDDDDDDDDDDEDEDDEEEEEEDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDEDDDDDDDDDDDDDDDDCCCCCDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDCDDDDCCDDDDCCCCCCCCCCCCCCDCCCCCCCCCCCDDDDCCDCCDDDEEEEEEEEEEEEEEEEEEEEEEEEEEEEEDEEEDDDDDDDDEEDDDDDDDDDDDDDDDDDDDDDDDDCDCDCCCCCCCCCCCCCCCCCCCCCCCCCCCDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDCCCCCCCCCCCCCCCDDDDDDDDCCCCDDCCCCCCCCCCCCCCCCCCCCCCCCCCCCDDDDDDDDDCDCDDDDDDDDDDCCCCCDDDDDDDDCCCCCCCCCCCCCCCCCCCCCCBBBBCBCCCBCBBBBBBBBBBBBBBBBBBBBCCCCCCCCBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBCBCBBBBBBBCCCCCCCCCCCCCCCCCCCCCCCCCCCCDDDDDDDDDDDDDDDDDDDCCCCCCCDDDCCDDCDCCCCCCCCCCCCCCCCDDDDDCCDCCDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDCCCCCCCCCDCCCCCCCCCCDDDDDDDDCDDDDDDDDDDDDDDDDDDDDDDDDDDDDCCDDDCCCCCCCCCCCCCCCDDCCDDDDDDDDEEEEEEEEEEEEEEEEEEEDDEDEDEEEEEDEEEEEEEDDDDEEDDDDDEEEEEEEEEEEEEEEEEEEDDDDDDDEEEDDDDDDDDDDDDDDDDDDDDDDDDDDDDDEDDDDDDDDDDDEEEEDDDDDDDDDDDDDDDDDDDDDDDDCCCCCCCCCCCCCCCCCCCCCCCCCCCDCCCCDDDDDDDDEDDDDDDDDDDDDDDDDDDDDDDDDDDCCDDDDDDDCCCCCCCCCCCCCCCCCCDCCCCCCCDCCCCCCCCCDCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCBCCCCCCCCBCBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBAAAABBBABAAABBBBBBBBBBBBBBBBBBBBBAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

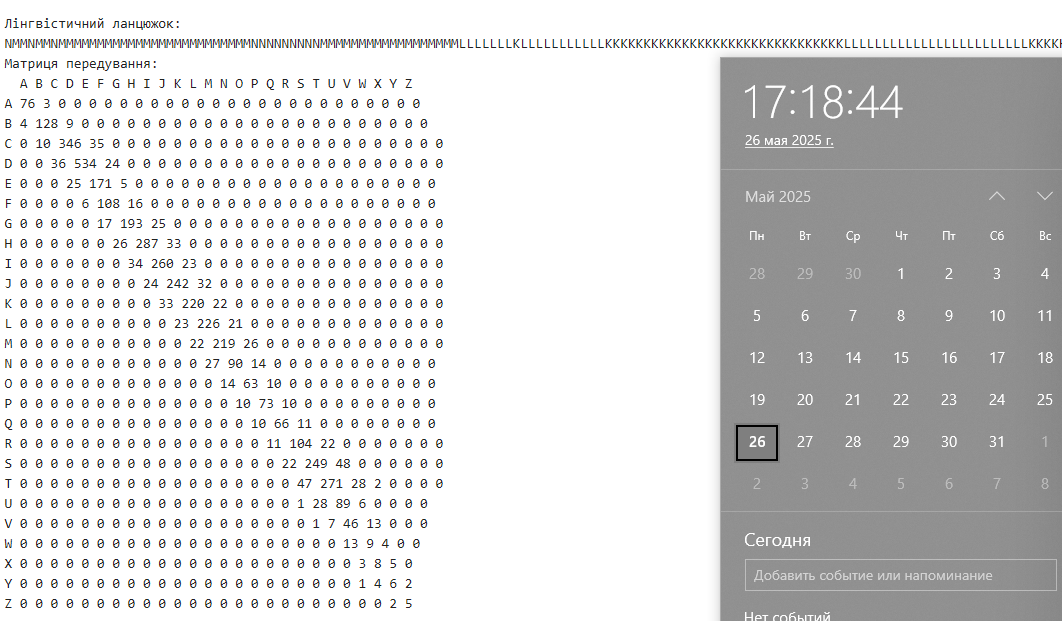


Рис. 5 – Результат розрахунку

**Лістінг програмного тексту:**

program.rkt:

#lang racket

;; Функція для зчитування чисел з окремих рядків

(define (read-numbers)

(let loop ([numbers '()])

(define line (read-line))

(cond [(or (eof-object? line) (string=? line ""))

(reverse numbers)]

[else

(define num (string->number line))

(if num

(loop (cons num numbers))

(error "Некоректне число у введенні"))])))

;; Функція для визначення індексу інтервалу для числа

(define (get-interval-index x min-val width alphabet-size)

(if (zero? width)

0

(min (floor (/ (- x min-val) width)) (sub1 alphabet-size))))

;; Функція для відображення чисел на літери за рівними інтервалами

(define (map-to-letters numbers alphabet-size)

(define min-val (apply min numbers))

(define max-val (apply max numbers))

(define width (if (= alphabet-size 1)

0

(/ (- max-val min-val) alphabet-size)))

(for/list ([x numbers])

(define index (inexact->exact (get-interval-index x min-val width alphabet-size)))

(integer->char (+ 65 index)))) ; 65 — ASCII-код для 'A'

;; Функція для побудови матриці передування

(define (build-matrix chain alphabet-size)

;; Ініціалізація матриці з окремими векторами для кожного рядка

(define matrix (for/vector ([i (in-range alphabet-size)])

(make-vector alphabet-size 0)))

;; Побудова матриці передування

(for ([i (in-range (sub1 (string-length chain)))])

(define prev (char->integer (string-ref chain i)))

(define next (char->integer (string-ref chain (add1 i))))

(define row (- prev 65)) ; 'A'=65 -> 0, 'B'=66 -> 1, 'C'=67 -> 2

(define col (- next 65))

(vector-set! (vector-ref matrix row) col

(add1 (vector-ref (vector-ref matrix row) col))))

matrix)

;; Функція для виведення матриці з позначками літер

(define (print-matrix matrix alphabet-size)

(printf " ")

(for ([i (in-range alphabet-size)])

(printf "~a " (integer->char (+ 65 i))))

(newline)

(for ([i (in-range alphabet-size)])

(printf "~a " (integer->char (+ 65 i)))

(for ([j (in-range alphabet-size)])

(printf "~a " (vector-ref (vector-ref matrix i) j)))

(newline)))

;; Головна функція для координації програми

(define (main)

(define alphabet-size (string->number (read-line)))

(when (or (< alphabet-size 1) (> alphabet-size 26))

(error "Потужність алфавіту має бути від 1 до 26"))

(define numbers (read-numbers))

(when (null? numbers)

(error "Числа не введено"))

(define chain (list->string (map-to-letters numbers alphabet-size)))

(define matrix (build-matrix chain alphabet-size))

(printf "Лінгвістичний ланцюжок:\n~a\n" chain)

(printf "Матриця передування:\n")

(print-matrix matrix alphabet-size))

;; Запуск програми

(main)