Міністерство освіти і науки України

Національний університет «Запорізька політехніка»

кафедра програмних засобів

ЗВІТ

з лабораторної роботи № 3

з дисципліни «Алгоритми та структури даних» на тему:

«ЖАДІБНІ АЛГОРИТМИ»

Варіант №8

Виконав:

ст. гр. КНТ-113сп Іван Щедровський

Прийняв:

Старший викладач Лариса ДЕЙНЕГА

2023

1 Мета роботи:

1.1 Вивчити основні принципи та особливості жадібних алгоритмів.

1.2 Навчитися використовувати жадібні алгоритми для розв’язання практичних завдань та обґрунтовувати прийняті рішення.

2 Завдання до лабораторної роботи:

2.1 Індивідуальне завдання. Варіант 3. Клас вхідних даних задачі має дозволяти:

– задавати початкові дані;

– вводити нові параметри;

– коригувати та видаляти існуючі;

– розв’язувати задачу з використанням жадібного алгоритму. 3.3.2.

Варіант № 3.

На дачі стоїть велика діжка, яка вміщує задану кількість рідини. Хазяїн використовує її для поливу рослин, але не маючи централізованого водопостачання, має принести воду з річки. У його розпорядженні є відра заданого обсягу. Визначити, яку мінімальну кількість відер води хазяїну потрібно принести з річки, щоб заповнити діжку. Вважати, що кожне відро може бути принесене тільки повністю заповненим, адже хазяїн не хоче носити зайвий вантаж. Результати виводити, демонструючи кількість відер кожного обсягу.

2.2 Розробити програмне забезпечення, яке реалізує використання алгоритму Хаффмана для стискання даних текстового файлу у вигляді класу. Клас повинен мати методи, які дозволяють задати файл з даними, виконати стискання даних, визначити параметри виконаного стискання та зворотне перетворення, записати результати кодування/декодування в файл.

3 Текст розробленого програмного забезпечення з коментарями:

src\encodeAndDecode.ts

```
import * as fs from 'fs'

class TreeNode {
  letter: string | null
  weight: number
  right: TreeNode
  left: TreeNode

  constructor(letter: string | null, weight: number) {
    this.letter = letter
    this.weight = weight
  }
}

class EncodeAndDecode {
  encodeFromFileToFiles(
    filePathForEncode: string,
    filePathBinary: string,
    filePathTree: string
  ) {
    const fileText = fs.readFileSync(filePathForEncode, 'utf-8')

    const encoded = this.encode(fileText)

    fs.writeFileSync(filePathBinary, encoded.string, 'binary')
    fs.writeFileSync(filePathTree, JSON.stringify(encoded.tree), 'utf-8')
  }

  decodeFromFilesToFile(
    filePathForDecode: string,
    filePathCodes: string,
    filePathForOutput: string
  ) {
    const fileText = fs.readFileSync(filePathForDecode, 'binary')
    const treeJson = JSON.parse(fs.readFileSync(filePathCodes, 'utf-8'))

    const treeRoot = this.treeFromObject(treeJson)

    const decoded = this.decode(fileText, treeRoot)

    fs.writeFileSync(filePathForOutput, decoded, 'utf-8')
  }

  encode(text: string) {
    const lettersCount = this.lettersCountInText(text)

    const list = [...lettersCount.entries()].map(([letter, weight]) => {
      return new TreeNode(letter, weight)
    })

    const huffmanBinaryTreeRoot = this.huffman(list)

    const codes = {}
    this.printCodesFromBinaryTree(huffmanBinaryTreeRoot, codes)
```

```
  const resultArray = Array.from(text).map(char => codes[char])

  const treeOutput = this.treeToJson(huffmanBinaryTreeRoot)

  return {
    tree: treeOutput,
    string: resultArray.join('')
  }
}

decode(encodedText: string, tree: TreeNode) {
  const result = []

  let node = tree
  for (let i = 0; i <= encodedText.length; i++) {
    node = encodedText.charAt(i) == '0' ? node.left : node.right

    if (node.letter !== null) {
      result.push(node.letter)
      node = tree
    }
  }

  return result.join('')
}

// eslint-disable-next-line
private treeFromObject(object: any) {
  const node = new TreeNode(object.letter, 0)

  if (object.letter === null) {
    node.right = this.treeFromObject(object.right)
    node.left = this.treeFromObject(object.left)
  }

  return node
}

private treeToJson(node: TreeNode) {
  const result = {
    letter: node.letter
  }

  if (node.letter === null) {
    result['left'] = this.treeToJson(node.left)
    result['right'] = this.treeToJson(node.right)
  }

  return result
}

private printCodesFromBinaryTree(
  node: TreeNode,
  result: Record<string, string>,
  c = ''
) {
  if (node.letter !== null) {
    result[node.letter] = c
```

```
      return
    }

    this.printCodesFromBinaryTree(node.right, result, c + '1')
    this.printCodesFromBinaryTree(node.left, result, c + '0')
  }

  private huffman(list: TreeNode[]) {
    while (list.length > 1) {
      list.sort((a, b) => {
        return b.weight - a.weight
      })

      const right = list.pop()
      const left = list.pop()

      const newTreeNode = new TreeNode(null, left.weight + right.weight)
      newTreeNode.left = left
      newTreeNode.right = right

      list.push(newTreeNode)
    }

    return list[0]
  }

  private lettersCountInText(text: string): Map<string, number> {
    const lettersCount = new Map<string, number>()

    for (const letter of text) {
      lettersCount.set(letter, (lettersCount.get(letter) ?? 0) + 1)
    }

    return lettersCount
  }
}

export { EncodeAndDecode }


src\main.ts
import * as readline from 'node:readline'
import * as nodeProcess from 'node:process'
import * as chalk from 'chalk'

const { stdin: input, stdout: output } = nodeProcess

import { BringWater } from './water'
import { EncodeAndDecode } from './encodeAndDecode'

const bringWater = new BringWater(0, [])
const encodeAndDecode = new EncodeAndDecode()

/*
  Menu 1:
  (0) - Individual task
  (1) - File task

  Menu individual:
```

```
  (1) - Set data
  (2) - Change barrelCapacity
  (3) - Change buckets
  (4) - Show result
  (0) - Go back

  File menu:
  (1) - Encode from file
  (2) - Decode from files
  (0) - Go back
*/

const rl = readline.createInterface({ input, output })

function menu() {
  const menuString =
    `\nChoice menu:\n` +
    `(1) - Individual task\n` +
    `(2) - File task\n` +
    `(0) - Close program\n` +
    `\nInput(default - 0): `

  rl.question(menuString, answer => {
    console.clear()
    try {
      const number = Number(answer)

      switch (number) {
        case 0: {
          rl.close()
          return
        }

        case 1: {
          individualMenu()
          break
        }

        case 2: {
          fileMenu()
          break
        }

        default: {
          throw new Error('Not valid input')
        }
      }
    } catch {
      console.log(chalk.red('Not valid input! Try again'))
    }

    menu()
  })
}

function consoleQuestion(
  questionString: string,
  callback: (answer: string) => void
): Promise<void> {
```

```
  return new Promise(resolve => {
    rl.question(questionString, async answer => {
      console.clear()

      try {
        callback(answer)
        resolve()
        return
      } catch {
        console.log(chalk.red('Not valid input! Try again'))
      }

      await consoleQuestion(questionString, callback)
      resolve()
    })
  })
}

function individualSetBarrelCapacityInput(): Promise<void> {
  const question = `\nInput new barrel capacity(current = ${bringWater.barrelCapacity}): `

  return consoleQuestion(question, (answer: string) => {
    const number = Number(answer)

    if (number < 0) {
      throw new Error('Not valid input')
    }

    bringWater.barrelCapacity = number
  })
}

function individualSetBuckets(): Promise<void> {
  const question = `\nInput new buckets sizes in line with "," as divider (current = [${bringWater.buckets.join(
    ','
  )}]): `

  return consoleQuestion(question, (answer: string) => {
    const buckets = answer.split(',').map(Number)

    buckets.forEach(bucket => {
      if (isNaN(bucket)) throw new Error()
    })

    bringWater.buckets = structuredClone(buckets)
  })
}

function individualMenu() {
  const menuString =
    `\nMenu individual:\n` +
    `(1) - Set data\n` +
    `(2) - Change barrelCapacity\n` +
    `(3) - Change buckets\n` +
    `(4) - Show result\n` +
    `(0) - Go back\n` +
    `\nInput(default - 0): `

  rl.question(menuString, async answer => {
```

```
    console.clear()

    try {
      const number = Number(answer)

      switch (number) {
        case 0: {
          menu()
          return
        }

        case 1: {
          await individualSetBarrelCapacityInput()
          await individualSetBuckets()
          break
        }

        case 2: {
          await individualSetBarrelCapacityInput()
          break
        }

        case 3: {
          await individualSetBuckets()
          break
        }

        case 4: {
          console.log(bringWater.result)
          break
        }

        default:
          throw new Error('Not valid input')
      }
    } catch {
      console.log(chalk.red('Not valid input! Try again'))
    }

    individualMenu()
  })
}

function fileMenu() {
  const menuString =
    `\nFile menu:\n` +
    `(1) - Encode file\n` +
    `(2) - Decode files\n` +
    `(0) - Close program\n` +
    `\nInput(default - 0): `

  rl.question(menuString, async answer => {
    console.clear()

    try {
      const number = Number(answer)

      switch (number) {
        case 1: {
```

```
  let filePathForEncode = ''

  await consoleQuestion(
    `\nInput file location (default = "/files/textForEncode"): `,
    (answer: string) => {
      if (answer.trim() === '') {
        filePathForEncode = './files/textForEncode'
        return
      }

      filePathForEncode = answer
    }
  )

  let filePathBinary = ''
  await consoleQuestion(
    `\nInput output file path (default = "/files/binary"): `,
    (answer: string) => {
      if (answer.trim() === '') {
        filePathBinary = './files/binary'
        return
      }

      filePathBinary = answer
    }
  )

  let filePathTree = ''
  await consoleQuestion(
    `\nInput tree file path (default = "/files/tree.json"): `,
    (answer: string) => {
      if (answer.trim() === '') {
        filePathTree = './files/tree.json'
        return
      }

      filePathTree = answer
    }
  )

  encodeAndDecode.encodeFromFileToFiles(
    filePathForEncode,
    filePathBinary,
    filePathTree
  )
  break
}

case 2: {
  let filePathForDecode = ''

  await consoleQuestion(
    `\nInput file location (default = "/files/binary"): `,
    (answer: string) => {
      if (answer.trim() === '') {
        filePathForDecode = './files/binary'
        return
      }
```

```
            filePathForDecode = answer
          }
        )

        let filePathTree = ''
        await consoleQuestion(
          `\nInput tree file path (default = "/files/tree.json"): `,
          (answer: string) => {
            if (answer.trim() === '') {
              filePathTree = './files/tree.json'
              return
            }

            filePathTree = answer
          }
        )

        let fileForOutput = ''
        await consoleQuestion(
          `\nInput output file path (default = "/files/decodeOutput"): `,
          (answer: string) => {
            if (answer.trim() === '') {
              fileForOutput = './files/decodeOutput'
              return
            }

            fileForOutput = answer
          }
        )

        encodeAndDecode.decodeFromFilesToFile(
          filePathForDecode,
          filePathTree,
          fileForOutput
        )
        break
      }

      case 0: {
        menu()
        return
      }

      default:
        throw new Error('Not valid input')
    }
  } catch (e) {
    console.log(e)
    console.log(chalk.red('Not valid input! Try again'))
  }

  fileMenu()
 })
}

console.clear()
menu()
```

src\water.ts

```
import * as chalk from 'chalk'

/*
Варіант No 3.
На дачі стоїть велика діжка, яка вміщує задану кількість рідини.
Хазяїн використовує її для поливу рослин, але не маючи
централізованого водопостачання, має принести воду з річки. У його
розпорядженні є відра заданого обсягу. Визначити, яку мінімальну
кількість відер води хазяїну потрібно принести з річки, щоб заповнити
діжку. Вважати, що кожне відро може бути принесене тільки повністю
заповненим, адже хазяїн не хоче носити зайвий вантаж. Результати
виводити, демонструючи кількість відер кожного обсягу.
*/

class BringWater {
  barrelCapacity = 0
  private _buckets: number[] = []

  constructor(barrelCapacity: number, buckets: number[]) {
    this.barrelCapacity = barrelCapacity
    this.buckets = buckets
  }

  get buckets() {
    return this._buckets
  }

  set buckets(newBuckets) {
    this._buckets = newBuckets.sort((a, b) => a - b)
  }

  get result() {
    let notFilledCapacity = this.barrelCapacity

    const usedBuckets = {}

    if (this.buckets.length === 0) {
      return chalk.red('Buckets must be!')
    }

    while (notFilledCapacity > 0) {
      for (let i = 0; i < this.buckets.length; i++) {
        const bucket = this.buckets[i]

        if (notFilledCapacity < bucket || i === this.buckets.length - 1) {
          if (!(bucket in usedBuckets)) {
            usedBuckets[bucket] = 0
          }

          usedBuckets[bucket]++
          notFilledCapacity -= bucket
          break
        }
      }
    }

    return usedBuckets
  }
```

```
}

export { BringWater }
```

4 Результати роботи програмного забезпечення:

На рисунках 4.1, 4.2, 4.3, 4.5, 4.6, 4.7 показано виконання програми:

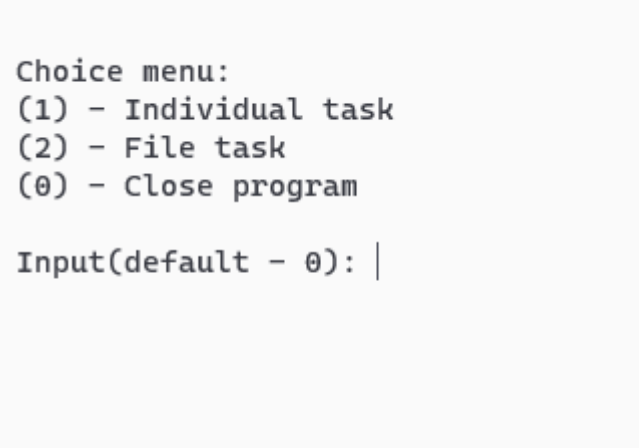

Рисунок 4.1 – Вигляд головного меню

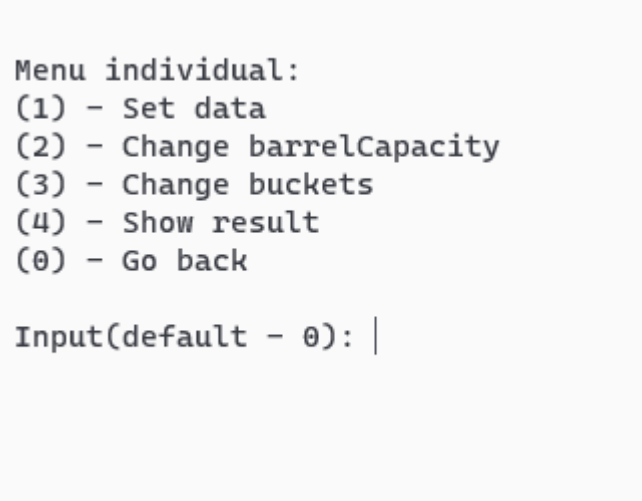

Рисунок 4.2 – Вигляд меню індивідуального завдання

Рисунок 4.3 – Вигляд меню задання з кодування файлу

Рисунок 4.4 – Вигляд встановлення значення наповненості бочки в індивідуальному завданні

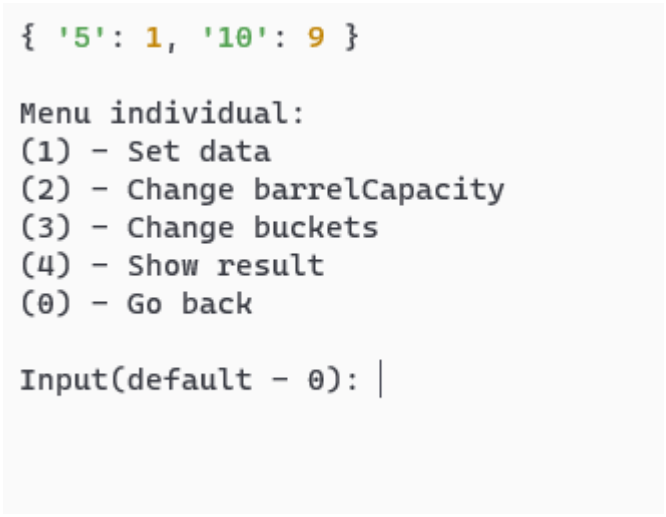

Рисунок 4.5 – Вигляд результату індивідуального завдання при значені бочки = 94, та відрах 2, 4, 5 та 10

```
{ '2': 1, '10': 9 }

Menu individual:
(1) - Set data
(2) - Change barrelCapacity
(3) - Change buckets
(4) - Show result
(0) - Go back

Input(default - 0):
```

Рисунок 4.6 – Вигляд результату індивідуального завдання при значені бочки = 92, та відрах 2, 4, 5 та 10

```
Not valid input! Try again

Menu individual:
(1) - Set data
(2) - Change barrelCapacity
(3) - Change buckets
(4) - Show result
(0) - Go back

Input(default - 0):
```

Рисунок 4.7 – Вигляд меню при помилці

На рисунку 4.8 показана файлова структура файлів для виконання кодування та декодування даних файлу

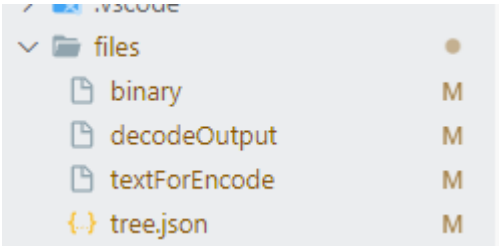

Рисунок 4.8 – Вигляд файлів для кодування

Текст перед кодуванням:

Aenean lobortis mi ut pulvinar fermentum. Nunc sit amet velit ante. Maecenas ac tristique erat. Donec dignissim tortor non feugiat ornare. Morbi rhoncus ligula urna, sed maximus diam fermentum non.

Закодований текст:

000101101110000111010100000011001110100001001101001110110010011010
011011010000110000110001000101111000100110001110010000000101011100110010
011101111011111000001101000101111000001000101001000000011001001110101000
110001010110111110110001000111011110011010001100010101000001101111100000
100011110101111110011110000010111010010101110010010110011101001101011001
000001010110001110011111011101010110110000010001101010100000111110010011 0
010100000100000000100110111010100101100101101010011101101010011100100 0
010100000001100100111100000010000100010101100011010011100000101011111111
000001000111110100111000100101000010111000110111010000011001100011010011
001101000001000100010011010100110000111000001010001100000111011111001010
011011010100011001010010111000110100110010101000101101100110010011101111
011111000001101000101100100001010000011000001

Бінарне дерево:

{"letter":null,"left":{"letter":null,"left":{"letter":null,"left":{"letter":null,"left":{"letter":null,"left":{"let
ter":"n"},"right":{"letter":null,"left":{"letter":null,"left":{"letter":null,"left":{"letter":"g"},"
right":{"letter":"b"}},"right":{"letter":null,"left":{"letter":null,"left":{"letter":"N"},"right":
{"letter":"q"}},"right":{"letter":null,"left":{"letter":"A"},"right":{"letter":"p"}}}},"right":
{"letter":null,"left":{"letter":null,"left":{"letter":null,"left":{"letter":","},"right":{"letter":"
x"}},"right":{"letter":null,"left":{"letter":"D"},"right":{"letter":"h"}}},"right":{"letter":nul
l,"left":{"letter":"v"},"right":{"letter":"M"}}}}},"right":{"letter":"
"}},"right":{"letter":null,"left":{"letter":null,"left":{"letter":"i"},"right":{"letter":"a"}},"rig
ht":{"letter":null,"left":{"letter":"t"},"right":{"letter":"r"}}}},"right":{"letter":null,"left":{"
letter":null,"left":{"letter":null,"left":{"letter":"u"},"right":{"letter":null,"left":{"letter":nul

l,"left":{"letter":"f"},"right":{"letter":"d"}},"right":{"letter":"l"}}},"right":{"letter":null,"left":{"letter":"o"},"right":{"letter":"m"}}},"right":{"letter":null,"left":{"letter":null,"left":{"letter":null,"left":{"letter":"."},"right":{"letter":"c"}},"right":{"letter":"s"}},"right":{"letter":"e"}}}}

Розкодований текст:

Aenean lobortis mi ut pulvinar fermentum. Nunc sit amet velit ante. Maecenas ac tristique erat. Donec dignissim tortor non feugiat ornare. Morbi rhoncus ligula urna, sed maximus diam fermentum non.

5 Висновки:

У цій лабораторній роботі я вивчив основні принципи та особливості жадібних алгоритмів. Навчився використовувати їх для розв’язання практичних завдань та обґрунтовувати прийняті рішення.

У варіанті № 3, я вирішував задачу про те, скільки відер води потрібно принести з річки, щоб заповнити діжку. Розробив програмне забезпечення, що демонструє оптимальний спосіб носити відра, мінімізуючи зайвий вантаж.

Також реалізував алгоритм Хаффмана для стискання даних текстового файлу. Створив клас, який дозволяє здійснити стискання та розпакування даних, а також зберегти результати у файл. Це надає ефективність при обробці та передачі великих обсягів інформації.

У результаті цієї лабораторної роботи я набув важливий досвід в розробці та застосуванні алгоритмів для оптимізації процесів та ефективної роботи з даними.