

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Дата сдачи на проверку:

«__» _____ 2025 г.

Проверено:

«__» _____ 2025 г.

ГРАФЫ. ОПЕРАЦИИ НАД ГРАФАМИ.

Отчёт по лабораторной работе №4

по дисциплине

«Дискретная математика»

Разработал студент гр. ИВТб-1301-05-00 _____ /Черкасов А. А./
(подпись)

Проверила преподаватель _____ /Пахарева И. В./
(подпись)

Работа защищена «__» _____ 2025 г.

Киров

2025

Цель

Цель работы: Изучить операцию построения дополнения для неориентированного графа на основе представления в виде матрицы смежности и разработать программу, которая по заданной в файле `input.txt` матрице смежности формирует матрицу смежности дополненного графа. Это позволяет закрепить навыки работы с представлением графов в памяти компьютера и понять принципы формирования нового графа на основании исходного.

Задание

- Неориентированный граф G_1 задается матрицей смежности, которая записана в файле `input.txt`. Размерность графа задается в программе (вершин ≥ 5 , дуг ≥ 7)
- Сформировать дополнение графа.

Решение

Схема алгоритма решения представлена на рисунке 1.1 и 1.2. Примеры работы программы представлены на рисунках 2.1 и 2.4, графы построенные по матрицам из примеров представлены на рисунках 2.2, 2.3, 2.5, 2.6. Исходный код решения представлен в Приложении А1.

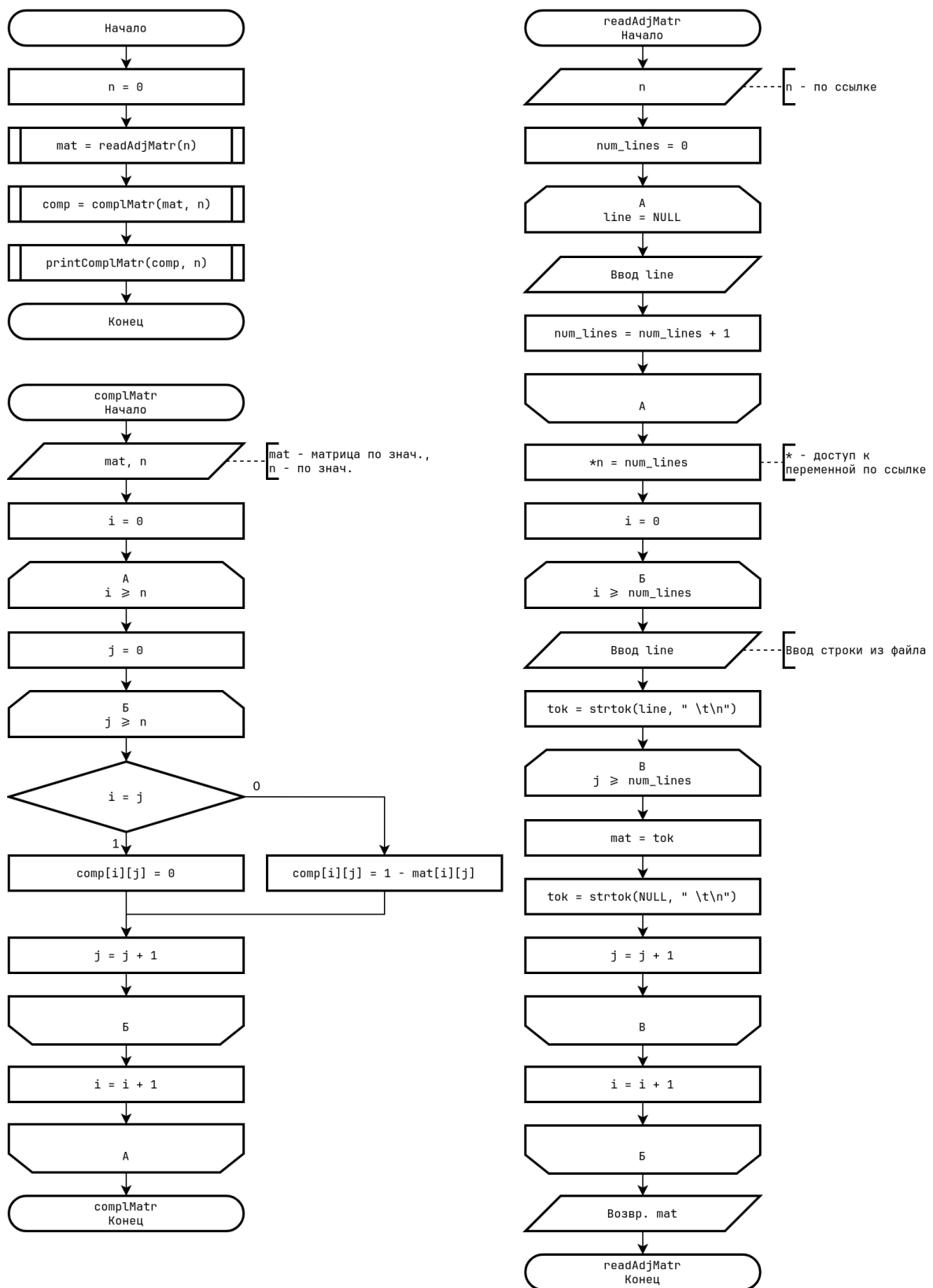


Рисунок 1.1 - Схема алгоритма основной программы, подпрограммы чтения матрицы из файла, подпрограмма формирования дополненной матрицы.

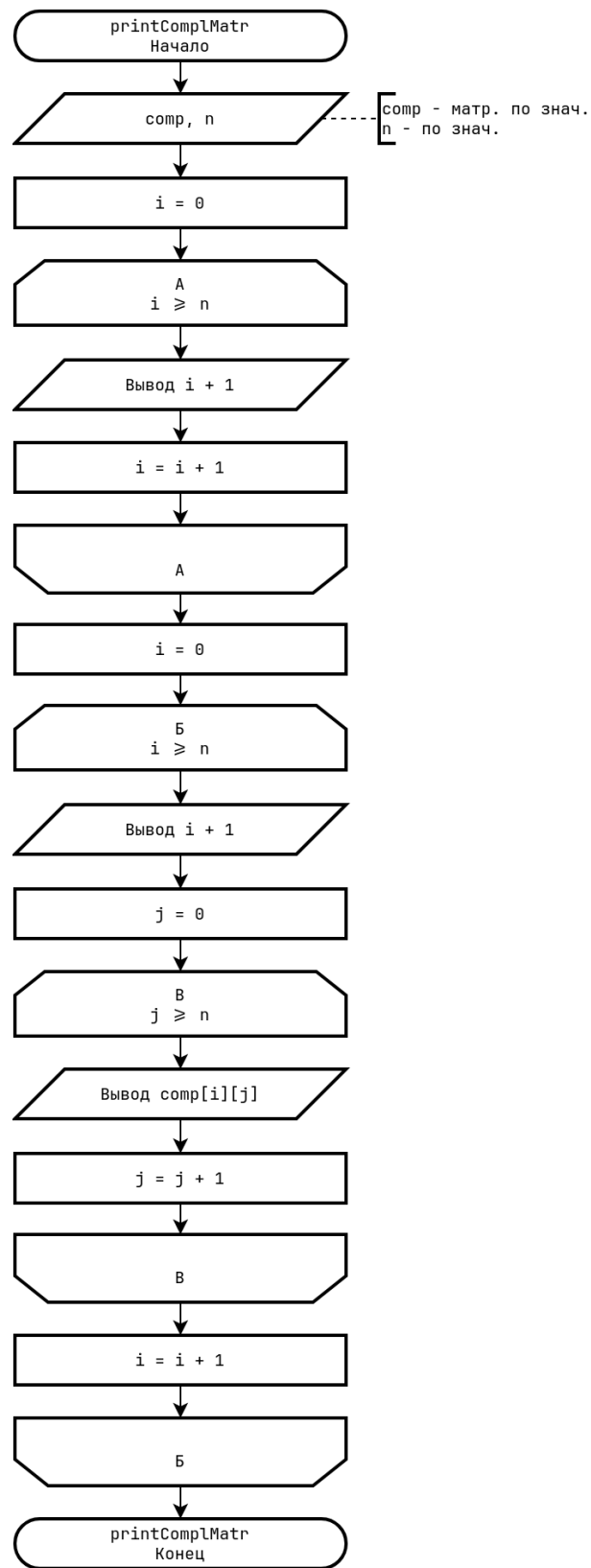


Рисунок 1.1 - Схема алгоритма подпрограммы вывода итоговой матрицы.

```
> go run  $\frac{1}{2}$ 
```

	ОРИГИНАЛЬНАЯ МАТРИЦА											ДОПОЛНЕНИЕ									
#	1	2	3	4	5	6	7	8	9	10		1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	0	0	0	0	0	1		0	0	0	0	1	1	1	1	1	0
2	1	0	1	1	0	0	1	0	0	0		0	0	0	0	1	1	0	1	1	1
3	1	1	0	1	0	0	0	0	1	0		0	0	0	0	1	1	1	1	0	1
4	1	1	1	0	1	0	0	0	0	0		0	0	0	0	0	1	1	1	1	1
5	0	0	0	1	0	1	0	1	0	0		1	1	1	0	0	0	1	0	1	1
6	0	0	0	0	1	0	1	0	0	0		1	1	1	1	0	0	0	1	1	1
7	0	1	0	0	0	1	0	1	0	0		1	0	1	1	1	0	0	0	1	1
8	0	0	0	0	1	0	1	0	1	0		1	1	1	1	0	1	0	0	0	1
9	0	0	1	0	0	0	0	1	0	1		1	1	0	1	1	1	1	0	0	0
10	1	0	0	0	0	0	0	0	1	0		0	1	1	1	1	1	1	1	0	0

Рисунок 2.1 - Пример работы программы 1.

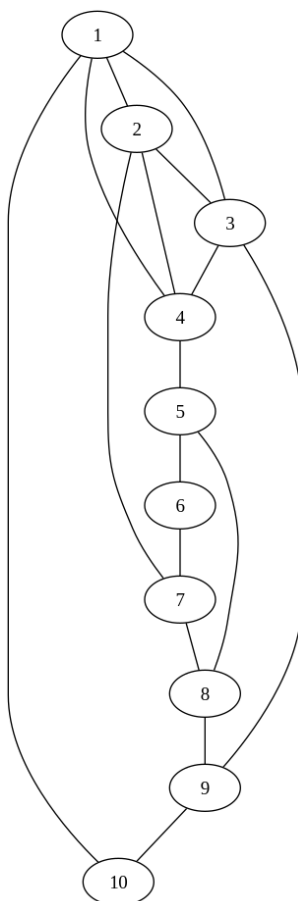


Рисунок 2.2 - Оригинальный граф построенный по входной матрице 1 примера.

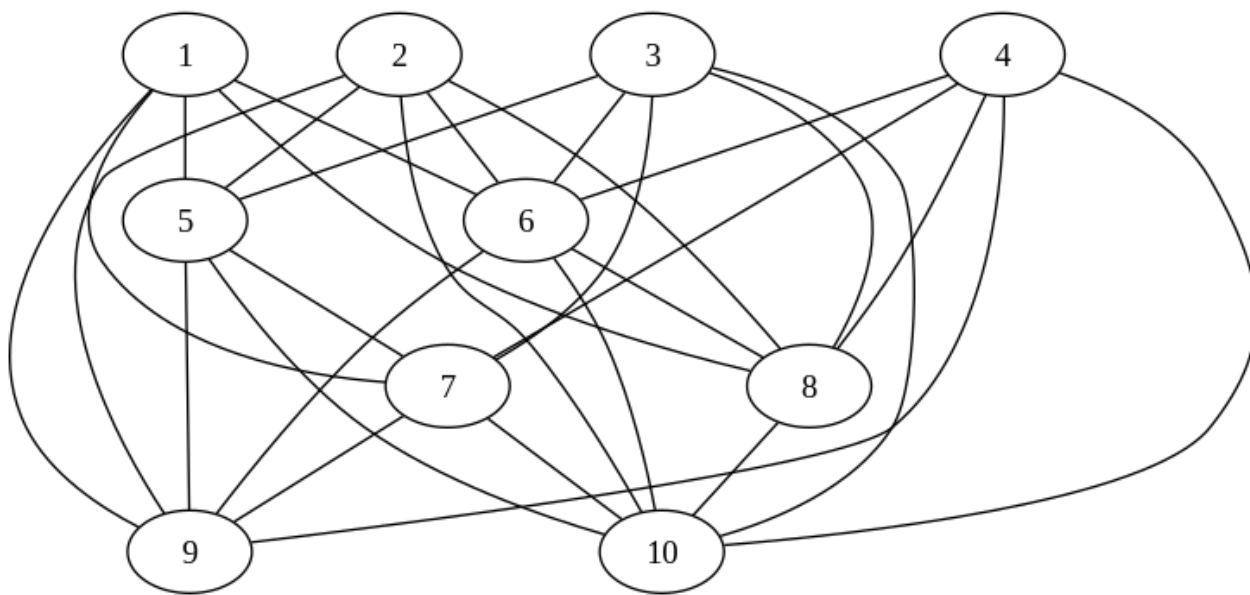


Рисунок 2.3 - Дополненный граф построенный по итоговой матрице 1 примера.

```
> go run .
```

	ОРИГИНАЛЬНАЯ МАТРИЦА										ДОПОЛНЕНИЕ	
#	1	2	3	4	5		1	2	3	4	5	
1	0	1	0	0	0		0	0	1	1	1	
2	1	0	1	0	0		0	0	0	1	1	
3	0	1	0	1	0		1	0	0	0	1	
4	0	0	1	0	1		1	1	0	0	0	
5	0	0	0	1	0		1	1	1	0	0	

Рисунок 2.4 - Пример работы программы 2.

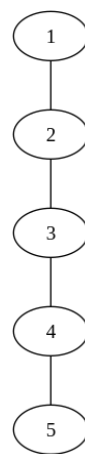


Рисунок 2.5 - Оригинальный граф построенный по входной матрице 2 примера.

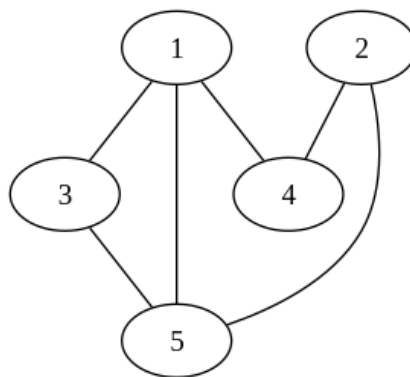


Рисунок 2.6 - Дополненный граф построенный по итоговой матрице 2 примера.

Вывод

В ходе выполнения лабораторной работы были изучены основы представления неориентированных графов с помощью матрицы смежности, а также алгоритм построения дополнения графа. Реализованная программа на языке Go, которая осуществляет чтение исходной матрицы из файла, проверяет её корректность, формирует матрицу дополненного графа и выводит результат. Работа позволила закрепить навыки работы с графами, улучшить понимание операций над графами и применение их в программировании.

Приложение А1. Исходный код

```
package main

import (
    "bufio"
    "flag"
    "fmt"
    "os"
    "strconv"
    "strings"
)

func readAdjacencyMatrix(filename string) ([] []int, error) {
    f, err := os.Open(filename)
    if err != nil {
        return nil, fmt.Errorf("не открыть файл %s: %v", filename, err)
    }
    defer f.Close()

    sc := bufio.NewScanner(f)
    var lines []string
    for sc.Scan() {
        line := strings.TrimSpace(sc.Text())
        if line != "" {
            lines = append(lines, line)
        }
    }
    if err := sc.Err(); err != nil {
        return nil, fmt.Errorf("ошибка чтения %s: %v", filename, err)
    }

    n := len(lines)
    if n == 0 {
```

```

        return nil, fmt.Errorf("пустой файл %s", filename)
    }

    matrix := make([] []int, n)
    for i, line := range lines {
        parts := strings.Fields(line)
        if len(parts) != n {
            return nil, fmt.Errorf(
                "строка %d: %d элементов, нужно %d", i+1, len(parts), n)
        }
        row := make([]int, n)
        for j, tok := range parts {
            val, err := strconv.Atoi(tok)
            if err != nil || (val != 0 && val != 1) {
                return nil, fmt.Errorf("элемент [%d,%d] = %v", i, j, tok)
            }
            row[j] = val
        }
        matrix[i] = row
    }

    for i := 0; i < n; i++ {
        if matrix[i][i] != 0 {
            return nil, fmt.Errorf("диагональ [%d,%d] должна быть 0", i, i)
        }
        for j := i + 1; j < n; j++ {
            if matrix[i][j] != matrix[j][i] {
                return nil, fmt.Errorf(
                    "несимметрично: [%d,%d]=%d, [%d,%d]=%d",
                    i, j, matrix[i][j], j, i, matrix[j][i],
                )
            }
        }
    }
}

```

```

        return matrix, nil
    }

func complementMatrix(orig [][]int) [][]int {
    n := len(orig)
    comp := make([][]int, n)
    for i := 0; i < n; i++ {
        comp[i] = make([]int, n)
        for j := 0; j < n; j++ {
            if i == j {
                comp[i][j] = 0
            } else {
                comp[i][j] = 1 - orig[i][j]
            }
        }
    }
    return comp
}

func printMatrices(orig, comp [][]int) {
    n := len(orig)
    const (
        reset = "\033[0m"
        blue  = "\033[1;34m"
        green = "\033[1;32m"
    )

    fmt.Println("          ОРИГИНАЛЬНАЯ МАТРИЦА          |          ДОПОЛНЕНИЕ")
    fmt.Print("  #  ", blue)
    for i := 1; i <= n; i++ {
        fmt.Printf("%2d ", i)
    }
    fmt.Print(reset, " | ", blue)
    for i := 1; i <= n; i++ {

```

```

        fmt.Printf("%2d ", i)
    }
    fmt.Println(reset)

    for i := 0; i < n; i++ {
        fmt.Print(green, fmt.Sprintf("%3d ", i+1), reset)
        for j := 0; j < n; j++ {
            fmt.Printf("%2d ", orig[i][j])
        }
        fmt.Print(" | ")
        for j := 0; j < n; j++ {
            fmt.Printf("%2d ", comp[i][j])
        }
        fmt.Println()
    }
}

func main() {
    viz := flag.Bool("viz", false, "визуализировать граф")
    vizLong := flag.Bool("visualize", false, "визуализировать граф")
    flag.Parse()
    visualize := *viz || *vizLong

    mat, err := readAdjacencyMatrix("input.txt")
    if err != nil {
        fmt.Fprintf(os.Stderr, "Ошибка: %v\n", err)
        os.Exit(1)
    }

    comp := complementMatrix(mat)
    printMatrices(mat, comp)

    if visualize {
        if err := WriteDOT(mat, "original_graph.dot", "OriginalGraph"); err != nil

```

```

        fmt.Fprintln(os.Stderr, "Не создать original_graph.dot:", err)
    } else {
        fmt.Fprintln(os.Stderr, "Создан: original_graph.dot")
    }

    if err := WriteDOT(comp, "complement_graph.dot", "ComplementGraph"); err != nil {
        fmt.Fprintln(os.Stderr, "Не создать complement_graph.dot:", err)
    } else {
        fmt.Fprintln(os.Stderr, "Создан: complement_graph.dot")
    }
}
}

```