

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №1**

**з дисципліни «Інженерія якості програмних засобів»**

**на тему:**

**«** Верифікація програм»

Виконав:

студент гр. ПЗ2421

Кулик С. В.

Прийняв:

Шинкаренко В.І.

Дніпро, 2025

**Тема:** Верифікація програм.

**Мета:** Отримати навички верифікації програм.

**Постановка задачі**

Розробити алгоритм для реалізації індивідуального завдання лабораторної роботи №1 та відповідає розробленій RSL-специфікації.

Виконати верифікацію програми, довівши часткову коректність та завершуваність програми. При верифікації розглядати лише ті частини коду, що реалізують логіку індивідуального завдання.

**Хід роботи**

1. **Постановка задачі згідно загального та індивідуального завдання**

**Вхідні дані:**

* Два орієнтовані графи **G1** та **G2**, представлені у вигляді матриць суміжності.
* Кожен граф має вершини з навантаженням.
* Навантаження на дуги відсутнє.

**Вихідні дані:**

* Новий граф **G**, який формується шляхом об'єднання вершин та ребер з двох графів. При конфлікті вершин перевага надається вершинам з **G1**, а при конфлікті ребер — ребрам з **G2**.

1. **Реалізація алгоритму на мові RSL**

Використаємо специфікацію mergeGraphs, представлену у першій роботі, для об'єднання графів.

Опис функцій:

* mergeGraphs: Об'єднує два графи, враховуючи пріоритети вершин і ребер.

module GraphMerge

types

VertexId = Text

Weight = Int

Vertex :: id : VertexId, weight : Weight

Edge :: from : VertexId, to : VertexId

Graph :: vertices : set of Vertex, matrix : VertexId × VertexId → Bool

values

mergeGraphs: Graph × Graph → Graph

mergeGraphs(g1, g2) as mergedGraph post

exists v, w in mergedGraph.vertices • mergedGraph.matrix(v.id, w.id) = true ==

let

mergedVertices: set of Vertex = g1.vertices ∪ { v2 in g2.vertices | not exists v1 in g1.vertices • v1.id = v2.id }

mergedMatrix: VertexId × VertexId → Bool = λ (id1, id2).

if (id1 ∈ { v.id | v ∈ mergedVertices } ∧ id2 ∈ { v.id | v ∈ mergedVertices })

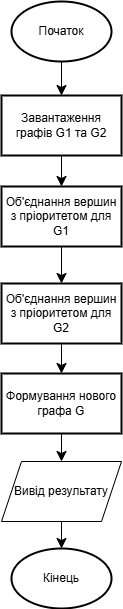
then (if g2.matrix(id1, id2) then true else if g1.matrix(id1, id2) then true else false)

else false

in

Graph(mergedVertices, mergedMatrix)

1. **Блок-схема алгоритму**

****

1. **Часткова коректність**

Специфікація вказує, що функція mergeGraphs повинна повертати новий граф **G**, який:

* Містить усі вершини з **G1** та вершини з **G2**, які відсутні в **G1**.
* Містить ребра з **G2**, а якщо вони відсутні, то з **G1**.
* Задовольняє умову, що матриця суміжності нового графа містить хоча б одне значення true.

**Доказ:**

* Перевірка вершин: вершини з **G1** мають пріоритет, тому вони завжди будуть включені до нового графа.
* Перевірка ребер: ребра з **G2** мають пріоритет, тому вони замінюють ребра з **G1** у разі конфлікту.
* Матриця суміжності гарантує, що новий граф містить хоча б одне ребро.

1. **Завершуваність**

Алгоритм завершується, оскільки:

* Кількість вершин і ребер обмежена.
* Об'єднання вершин і ребер виконується за скінченну кількість кроків.
* Рекурсія відсутня, тому глибина викликів не обмежена.

**Висновок**

У цій лабораторній роботі було реалізовано алгоритм об'єднання графів, який відповідає специфікаціям, розробленим у лабораторній роботі №1. Проведено верифікацію програми, що підтвердило часткову коректність і завершуваність алгоритму.

**Переваги:**

* Чітка специфікація дозволяє уникнути двозначностей у задачі.
* Алгоритм забезпечує коректне об'єднання графів з урахуванням пріоритетів.

**Недоліки:**

* Висока обчислювальна складність для великих графів через використання матриць суміжності.
* Можливі труднощі з розумінням формалізованої специфікації для новачків.