Graph Subgraph Isomorphism Program

Build and Run Instructions

Overview

The program is implemented in the C language and provides a command-line interface for finding subgraph isomorphisms and computing minimal graph extensions. It relies only on the standard C library and does not require additional third-party dependencies.

The application is compiled using a GNU-compatible gcc compiler with C11 standard support. The project contains a Makefile with predefined build rules.

The program expects an input file containing two graphs. Each graph must have the number of its vertices, followed by the adjacency matrix. The program supports both exact and heuristic algorithms for subgraph isomorphism detection and minimal edge extension computation.

Building and Running

1. Compile the code

Navigate to the Source directory and compile:

```
cd Source
make
```

The executable `aac` will be created in the Source directory.

3. Run the program

The program accepts command-line arguments in the following format:

```
aac <command> <graph_file> [n]
```

Available commands:

- `iso_exact` – Find n subgraph isomorphisms using exact algorithm
- `iso_approx` – Find n subgraph isomorphisms using heuristic algorithm
- `ext_exact` – Find minimal extension for n isomorphisms using exact algorithm
- `ext_approx` – Find minimal extension for n isomorphisms using heuristic algorithm

Example usage:

```
aac iso_exact ..\Examples\01_identical_triangles.txt 3
aac iso_approx ..\Examples\09_star_graph.txt 1
aac ext_exact ..\Examples\03_perfect_subgraph.txt 1
aac ext_approx ..\Examples\11_medium_sparse.txt 2
```

4. Clean the build

To remove the compiled binary:

```
make clean
```

Input File Format

The input file must contain two graphs in the following format:

```
number_of_vertices_in_graph_g
adjacency_matrix_of_graph_g
number_of_vertices_in_graph_h
adjacency_matrix_of_graph_h
```

Matrix entries represent edge counts (0 for no edge, positive integers for edge multiplicities in multigraphs).

Example Input File

`02_triangle_in_square.txt`:

```
3
0 1 0
0 0 1
1 0 0
4
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0
```

This file contains:

- Graph G: 3 vertices forming a triangle cycle
- Graph H: 4 vertices forming a square cycle

Example Execution

Example 1: Exact Isomorphism Algorithm

Running the exact algorithm to find isomorphisms:

```
aac iso_exact ..\Examples\02_triangle_in_square.txt 3
```

Output:

```
Loaded G: 3 vertices, H: 4 vertices
G adjacency matrix:
  0 1 0
  0 0 1
  1 0 0
H adjacency matrix:
  0 1 0 0
  0 0 1 0
  0 0 0 1
  1 0 0 0
```

```
=== Finding 3 isomorphism(s) [EXACT] ===
Found 4 isomorphism(s):
Isomorphism 1: 0->0, 1->1, 2->2
Isomorphism 2: 0->1, 1->2, 2->3
Isomorphism 3: 0->2, 1->3, 2->0
Isomorphism 4: 0->3, 1->0, 2->1
```

The program found 4 distinct ways to map the triangle into the square cycle.

Example 2: Heuristic Isomorphism Algorithm

Running the heuristic algorithm for faster computation:

```
aac iso_approx ..\Examples\09_star_graph.txt 1
```

Output:

```
Loaded G: 4 vertices, H: 5 vertices
G adjacency matrix:
  0 1 1 1
  0 0 0 0
  0 0 0 0
  0 0 0 0
H adjacency matrix:
  0 1 1 1 1
  0 0 0 0 0
  0 0 0 0 0
  0 0 0 0 0
  0 0 0 0 0

=== Finding 1 isomorphism(s) [HEURISTIC] ===
Found 1 isomorphism(s):
Isomorphism 1: 0->0, 1->1, 2->2, 3->3
```

The heuristic algorithm quickly found a mapping of the smaller star graph into the larger one.

Example 3: Minimal Extension Exact Algorithm

Computing minimal edges to extend graph G:

```
aac ext_exact ..\Examples\03_perfect_subgraph.txt 1
```

Output:

```
Loaded G: 3 vertices, H: 5 vertices
G adjacency matrix:
  0 1 0
  0 0 1
  0 0 0
H adjacency matrix:
  0 1 1 0 0
  0 0 1 0 0
  0 0 0 1 0
  0 0 0 0 1
  1 0 0 0 0

=== Finding minimal extension for 1 isomorphism(s) [EXACT] ===
Extension found with 0 edges added
G is already a subgraph of H
```

```
Isomorphism: 0->0, 1->1, 2->2
```

No extension needed as G is already a subgraph of H.

Example 4: Minimal Extension Heuristic Algorithm

Running the heuristic extension algorithm:

```
aac ext_approx ..\Examples\11_medium_sparse.txt 2
```

Output:

```
Loaded G: 5 vertices, H: 7 vertices

=== Finding minimal extension for 2 isomorphism(s) [HEURISTIC] ===
Extension found with 2 edges added:
  Edge: 1 -> 4
  Edge: 2 -> 5

Extended G adjacency matrix:
  0 1 0 0 0
  0 0 1 1 1
  0 0 0 1 0
  0 0 0 0 1
  1 0 0 0 0

Found 2 isomorphism(s) after extension
```

The algorithm added 2 edges to make G isomorphic to a subgraph of H.

Example 5: No Isomorphism Case

Testing a case where no isomorphism exists:

```
aac iso_exact ..\Examples\08_no_isomorphism_possible.txt 1
```

Output:

```
Loaded G: 5 vertices, H: 3 vertices

=== Finding 1 isomorphism(s) [EXACT] ===
No isomorphisms found.
G cannot be embedded in H (G is larger than H).
```

Test Examples

The `Examples` directory contains 20 test cases covering various scenarios:

- **Simple cases**: 01 (identical triangles), 04 (single vertex), 09 (star graphs)
- **Negative cases**: 08 (impossible embedding), 20 (empty graph)
- **Special features**: 05 (self-loops), 06/15 (multigraphs), 16 (disconnected graphs)
- **Complex cases**: 10/17 (complete graphs), 18 (larger graphs), 19 (worst-case greedy)

Refer to `Examples/example_descriptions.txt` for detailed descriptions of each test case.