Денис Дацко
Комп’ютерний проект з дискретної математики

Тема: “Дано списки ребер двох простих графів з не більше ніж шістьома вершинами. Визначити, чи ізоморфні ці графи.”

Загальний опис програми

Відомо, що для точного вирішення цієї задачі потрібно перебрати всі можливі відповідності вершин, і перевірити на відповідність усі ребра графа, тобто асимптотика цього алгоритму O(n!*m), де n і m - кількість вершин та ребер графа відповідно. Проте, я зробив деякі покращення, а саме:

- Спочатку графи перевіряються на кількість вершин, ребер, степені вершин, що часто дозволяє визначити неізоморфні графи відразу
- У процесі генерування перестановок, якщо степінь якоїсь вершини не відповідає степеню відповідної за цією перестановкою, то пропускаються всі перестановки з цією вершиною на даній позиції, що дає змогу іноді в декілька разів пришвидшити алгоритм

Моя програма може приймати дані як з терміналу так і з файлу, якщо вони будуть у форматі [(v1, v2), (v3, v4)], проте допускається і опускання усіх дужок(і квадратних і круглих), адже вони все рівно будуть стерті. Дані про 2 графи мають міститися в 2-х різних рядках.

Присутня також функція для візуалізації графів за бажанням користувача. Вона відкриває граф у стандартному переглядачі зображень і створює файл типу .png для збереження графа. Кожна вершина має випадковий колір, а розмір залежить від степеня вершини.**Важливо**: якщо не закрити вікно переглядача після перегляду першого графа, то це може унеможливити перегляд другого графа

Примітка

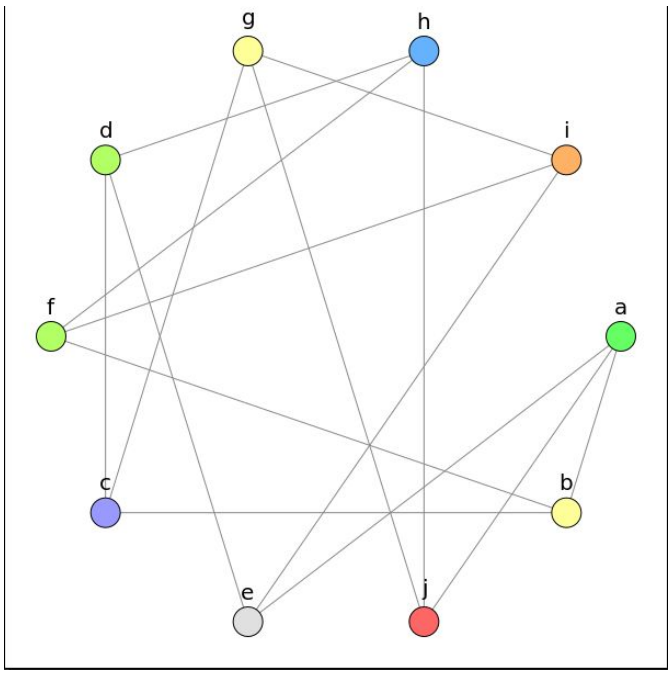
Функція для візуалізації використовує бібліотеку **igraph** (igraph.org) (У мене спочатку виникли проблеми з нею, адже бібліотека igraph вже була встановлена у мене на комп’ютері (допомогло встановлення бібліотеки python-igraph)

У програмі використовується команда терміналу “`clear`”, тому візуально все буде краще працювати в терміналах, де вона присутня.

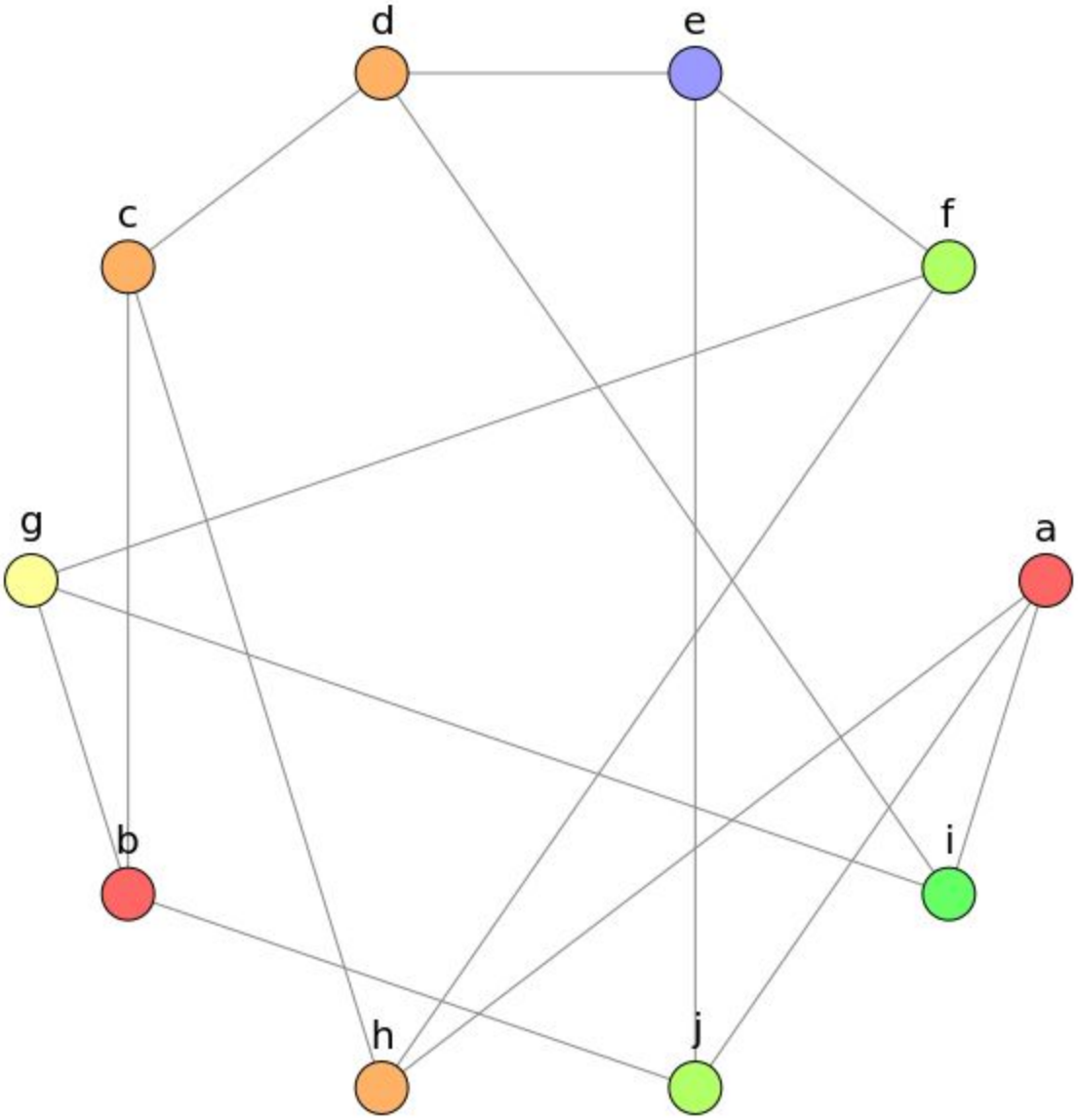
Приклад виконання програми

```
This program checks whether 2 graphs are isomorphic
You can input data from file or terminal but in both cases
it should be in the format [(v1, v2), (v3, v4), (v5, v6)] or
v1, v2, v3, v4, understanding that it means that there is an edge between
v1 and v3, v2 and v4 ... (while reading all the symbols except "," are deleted)
You can use any names for vertices.

If you choose to visualize a graph - it will be shown to you via
default image viewer and there will be created an image (.png) of it.

NOTE
The program can work very long you enter graphs with more than 10 vertices).

Do you want to get graph from file [y] or enter it here [n]? [y/n] ds
Enter either 'y' or 'n':
Enter either 'y' or 'n': n
[(a, b), (a, j), (a, e), (b, c), (b, f), (c, d), (c, g), (d, e), (d, h), (e, i), (f, h), (f, i), (g, j), (g, i), (h, j)]
[(a, i), (a, j), (a, h), (b, g), (b, j), (b, c), (c, d), (c, h), (d, i), (d, e), (e, j), (e, f), (f, g), (f, h), (g, i)]
```

```
Do you want to be shown the visualisation of the 1-st graph [y/N]: y
```



```
Do you want to be shown the visualisation of the 2-nd graph [y/N]: y
```



```
Hooray!!! Your graphs are isomorphic. Here`s one of possible correspondences:
a --> a
b --> h
j --> i
e --> j
c --> c
f --> f
d --> b
g --> d
h --> g
i --> e
Do you want to check more graphs? [y/N]:
```