Застосування патернів проектування в сучасному C++ для реалізації операторів на графах

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Бублик В.В.

Виконав студент ПМ-4:
Нагорний Ю.А.

Мета роботи

Дослідження графових операторів на можливість реалізації через спільний обхід. Виділення класу операторів, для яких така реалізація можлива, і класу операторів, для яких це неефективно або неможливо.

Реалізація операторів на графах засобами метапрограмування сучасного C++ із застосуванням головним чином патерна відвідувача, а також компонувальника, фасаду і впровадження залежностей.

Актуальність

The Boost Graph Library (BGL) є стандартом реалізації графових алгоритмів за принципами узагальненого програмування. Але використання застарілих ідіом призвело до надмірної складності.

Застосовані у цій роботі концепти та діапазони C++20 дозволили спростити код графових операторів та покращити читабельність помилок компіляції.

У 2015 опубліковано ILIGRA – найшвидший* алгоритм побудови зворотного реберного графа, адаптація якого патерном відвідувачем забезпечила можливість спільного обходу в реалізації графових операторів.

*Найшвидший для розріджених реберних графів[3]

Статичний патерн відвідувач

Патерн відвідувач, який у роботі є базою операторів, реалізовано за допомогою класів, які:

- мають шаблонні параметри для вхідного графа та вихідних структур;
- мають методи, відповідні подіям обходу вершин та ребер;
- змінюють стан наданих у конструкторі вихідних структур в процесі обходу.

```
template <typename OutputType>
class ExampleGraphVisitor {
public:
    template <Graph G>
    void tree_edge(typename G::edge_descriptor e, const G& g) {...}

    ExampleGraphVisitor(OutputType&& out) ...
};
```

Оператор двочастковості

$$f : \mathcal{G} \to (\mathbb{B} \times \Phi) \cup \{(0, \emptyset)\}$$

для довільного графа $G = (V, E)$ визначає його двочастковість:

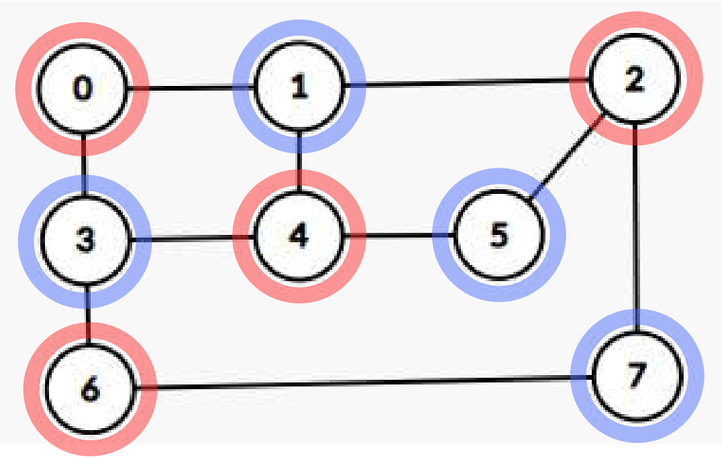
$$f(G) = \begin{cases} (1, \varphi), & \text{якщо } \forall \{u, v\} \in E : \varphi(u) \neq \varphi(v) \\ (0, \emptyset), & \text{в іншому випадку} \end{cases}$$

де $\mathbb{B} = \{0, 1\}$ — ознака двочастковості, а $\varphi : V \to C$ — відображення вершин у множину часток $C = \{c_1, c_2\}$.

Оператор двочастковості

Алгоритм виконується під час BFS обходу. Керований такими подіями:

- стартова вершина: задається стартова частка;
- деревне ребро: задається частка для недослідженого кінця ребра;
- не деревне ребро: перевіряється відмінність часток кінців ребра для виявлення недвочастковості.



Оператор точок зчленування

$$a : \mathcal{G} \to \mathcal{P}(V)$$

знаходить множину точок зчленування графа $G = (V, E)$

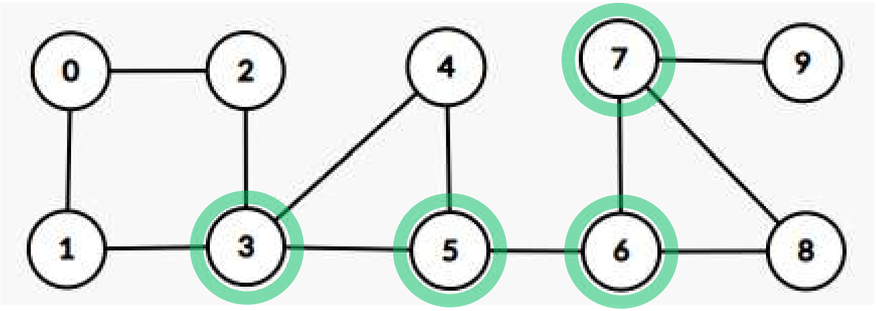
$$a(G) = \{v \in V \mid \omega(G \setminus \{v\}) > \omega(G)\}$$

де $\mathcal{P}(V)$ — множина всіх підмножин V, а $\omega(G)$ — кількість компонент зв'язності графа.

Оператор точок зчленування

У роботі архітектурно переосмислена адаптація алгоритму Тар’яна на DFS обхід. Застосування C++20 концептів та діапазонів дозволило:

- відфільтрувати некоректні з точки зору типізації виклики на етапі компіляції, замінивши громіздкі помилки BGL чіткими вимогами концепцій;
- застосувати сучасний механізм впровадження залежностей з нульовими накладним витратами.



Оператор реберного графа

$$L : \mathcal{G} \to \mathcal{G}$$

довільному графу $G = (V, E)$ ставить у відповідність граф $L(G) = (E, E')$, де множина ребер E' визначається як:

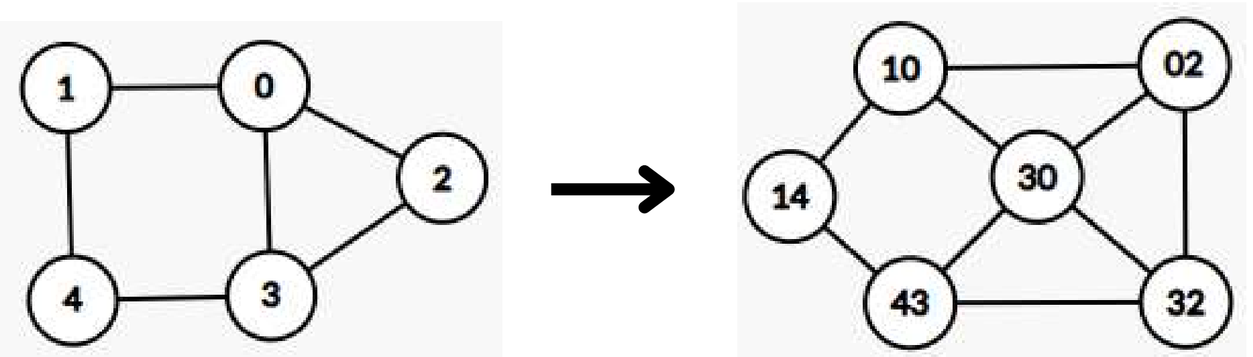
$$E' = \{\{e_1, e_2\} \mid e_1, e_2 \in E, e_1 \cap e_2 \neq \emptyset\}$$

де кожна вершина графа $L(G)$ відповідає ребру графа G, а суміжність вершин в $L(G)$ визначається наявністю спільної інцидентної вершини у відповідних ребер в G.

Оператор реберного графа

Створений у цій роботі алгоритм виконується під час BFS обходу. Керований подіями:

- першовідкриття ребра: створюється вершина реберного графа, і додається у два масиви ребер-прообразів зі спільною вершиною;
- завершення вершини: у реберному графі формуються кліки з вершин з вищезгаданих масивів .



Зворотний реберний оператор

$$L^{-1} : \mathcal{G} \to (\mathbb{B} \times \mathcal{G}) \cup \{(0, \emptyset)\}$$

для довільного графа H визначає його приналежність до класу реберних графів та знаходить його прообраз, якщо можливо:

$$L^{-1}(H) = \begin{cases} (1, G), & \text{якщо } \exists G \in \mathcal{G} : L(G) \cong H \\ (0, \emptyset), & \text{в іншому випадку} \end{cases}$$

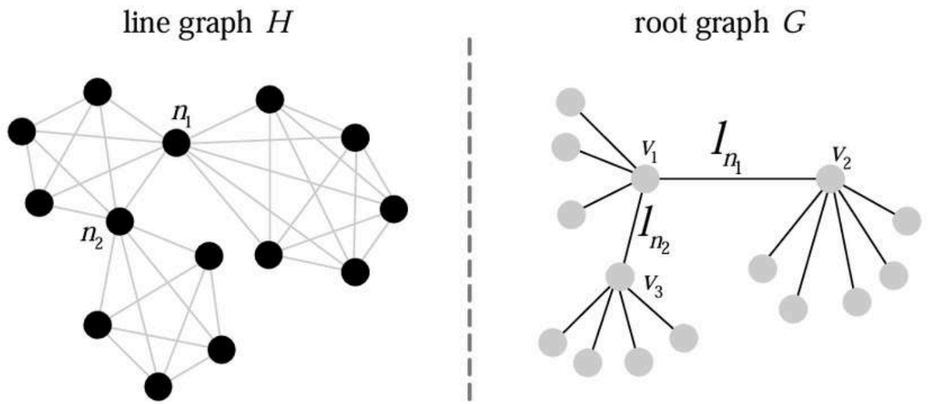
де $\mathbb{B} = \{0, 1\}$ — ознака реберності графа, а $L(G)$ — оператор побудови реберного графа.

Зворотний реберний оператор

Алгоритм ILIGRA *O(V + E)[3]*

Базова ідея: поступова побудова графу-прообразу на основі властивостей реберних графів, а не теореми Крауса.

- Визначає сусідів обох кінців першого ребра-прообразу.
- Інтерується по напіввизначених ребрах-прообразах. Довизначає їх та напіввизначає їх сусідів.

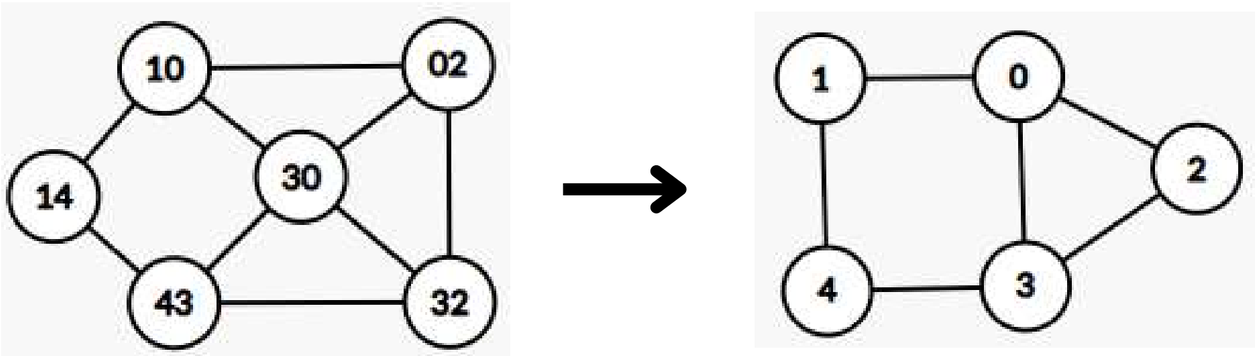


Зворотний реберний оператор

Алгоритм ILIGRA, був адаптований до виконання під час BFS обходу.

Керований подіями:

- стартова вершина: визначаються сусіди першого ребра-прообразу;
- розгляд вершини: закінчується побудова напіввизначеного ребра-прообразу;
- деревне ребро: напіввизначається нове ребро-прообраз;
- перехресне ребро: з’єднуються два напіввизначених ребра-прообрази;
- закінчення вершини: перевірка чи утворюють сусіди в реберному графі кліку.



Виділений клас операторів

Оскільки BFS/DFS проходять кожну вершину лише раз,
для адаптації під спільний обхід графовий оператор:

- має базуватися на алгоритмі лінійної складності;
- може бути реалізований через послідовний обхід;
- не вимагає на кожному кроці даних поза межами сусідів.

Невиконання умов призводить до неповноти спільного обходу.

Компонувальник відвідувачів

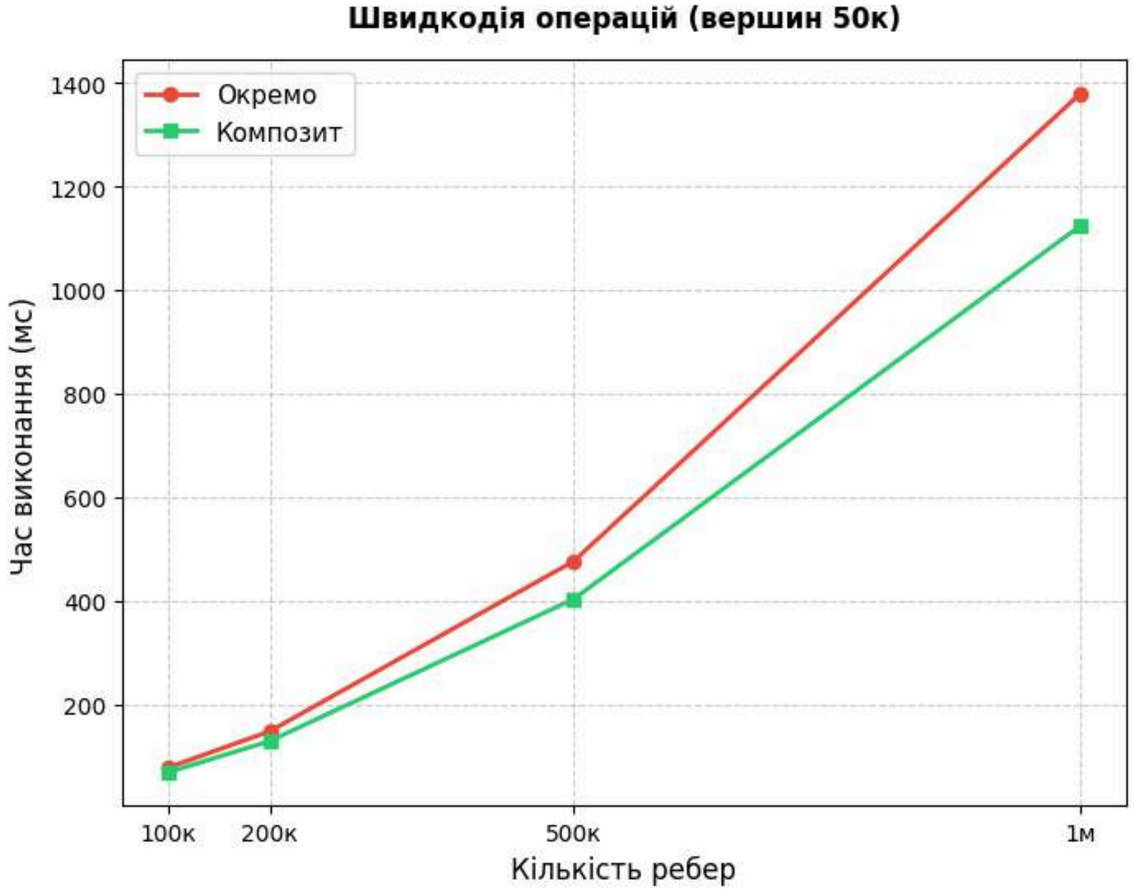
Спільний обхід під час виконання операторів забезпечує реалізований засобами метапрограмування сучасного C++ патерн компонувальник.

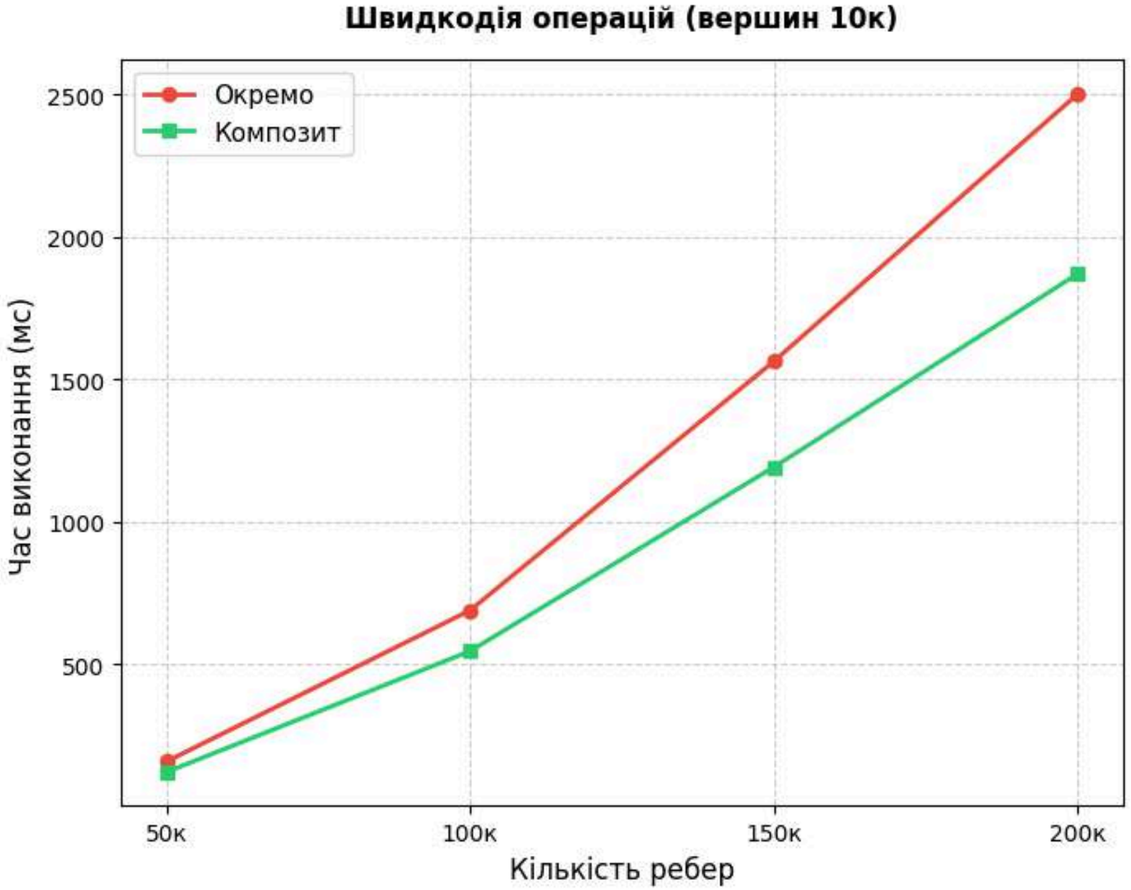
Він дозволяє передати в алгоритм обходу графа декілька відвідувачів як композит, який задовольняє ті самі концепції, що і звичайний відвідувач обходу.

```
template <typename... Visitors>
auto make_bfs_composite(Visitors&&... vis) {...}
```

Ефективність

Порівняння швидкодії без/з композитом для 2-х операторів L(G)





Результати

- Реалізовано узагальнені графові оператори на основі патерна відвідувач та BFS/DFS обходів засобами C++20;
- Створено алгоритми побудови реберного графа та адаптовано ILIGRA для реалізацій операторів побудови реберного графа та зворотнього ребреного графа на основі BFS відвідувачів;
- Виділено клас графових операторів, які підтримують ефективну реалізацію через спільний обхід.

Джерела

[1] Siek J., Lee L.-Q., Lumsdaine A. The Boost Graph Library: User’s Guide and Reference Manual. — Boston: Addison-Wesley, 2002.

[2] Josuttis N. M. C++20: The Complete Guide. — Braunschweig: Nicolai Josuttis, 2020.

[3] Liu D., Trajanovski S., Van Mieghem P. ILIGRA: An Efficient Inverse Line Graph Algorithm. Journal of Mathematical Modelling and Algorithms in Operations Research. — 2015. — Vol. 14, no. 1. — P. 13–33.

[4] Harary F. Graph Theory. — Reading: Addison-Wesley, 1969.

[5] Tarjan R. Depth-First Search and Linear Graph Algorithms. SIAM Journal on Computing. — 1972. — Vol. 1, no. 2. — P. 146–160.