НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичних методів захисту інформації

«До захисту допущено»
В.о. завідувача кафедри
_____ Михайло САВЧУК
«__» _____ 2021 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності: 113 Прикладна математика
на тему: **«Побудова квантових алгоритмів для задач на графах з використанням властивостей суперпозиції станів»**

Виконав: студент 4 курсу, групи ФІ-74
Слуцький Андрій Сергійович

Керівник: к.ф.-м.н., ст. викладач кафедри ММЗІ Фесенко А.В.

Консультант: _ _____

Рецензент: звання, степінь, посада Прізвище І.П. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ — 2021

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичних методів захисту інформації

Рівень вищої освіти — перший (бакалаврський)
Спеціальність (освітня програма) — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Михайло САВЧУК
«__» _____ 2021 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Слуцький Андрій Сергійович

1. Тема роботи: «*Побудова квантових алгоритмів для задач на графах з використанням властивостей суперпозиції станів*»,

керівник: к.ф.-м.н., ст. викладач кафедри ММЗІ Фесенко А.В.,

затверджені наказом по університету №__ від «__» _____ 2021р.

2. Термін подання студентом роботи: 04 червня 2021р.

3. Вихідні дані до роботи: квантовий алгоритм розфарбування двома кольорами графа, який представлений у вигляді матриці суміжностей, квантовий алгоритм Вігдерсона для графа, представленому як матриця суміжностей, аналіз побудованих алгоритмів.

4. Зміст роботи:

1) побудувано квантовий алгоритм розфарбування двома кольорами графа, який представлений у вигляді матриці суміжностей;

2) побудувано квантовий алгоритм Вігдерсона для графа, представленому як матриця суміжностей;

3) обчислено складність побудованих алгоритмів.

5. Перелік ілюстративного матеріалу : Презентація доповіді

6. Дата видачі завдання: 10 вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2020 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-жовтень 2020 р.	Виконано
3	Розбір побудови алгоритмів у квантовій моделі обчислення	Січень 2021 р.	Виконано
4	Побудова квантового алгоритму розфарбування двома кольорами графа, який представлений у вигляді матриці суміжностей	Лютий-березень 2021 р.	Виконано
5	Побудова квантового алгоритму Вігдерсона для графа, представленому як матриця суміжностей	Квітень-травень 2021 р.	Виконано
6	Аналіз побудованих алгоритмів	Травень 2021 р.	Виконано

Студент _____ Слуцький А.С.

Керівник _____ Фесенко А.В.

РЕФЕРАТ

Кваліфікаційна робота містить: 55 стор., 0 рисунки, 0 таблиць, 29 джерел.

Метою даної роботи є дослідити використання властивостей станів суперпозиції у квантових алгоритмах, а також побудувати квантові алгоритми на графах використовуючи ці властивості.

Було досліджено використання властивостей станів суперпозиції у квантових алгоритмах, а також побудований квантовий алгоритм розфарбування двома кольорами графу, який має представлення матрицею суміжності, та побудований квантовий алгоритм Вігдерсона для матриці суміжностей.

КВАНТОВІ АЛГОРИТМИ, АЛГОРИТМИ НА ГРАФАХ, РОЗФАРБУВАННЯ ГРАФУ, ВЛАСТИВОСТІ СТАНІВ СУПЕРПОЗИЦІЙ

ABSTRACT

Qualification work contains: 55 pages, 0 figures, 0 tables, 29 sources.

The aim of this work is to investigate the use of the properties of superposition states in quantum algorithms, and also to construct quantum algorithms on graphs by using these properties.

In this work was investigated the properties of superposition states in quantum algorithms and also built quantum algorithm for two colors graph coloring, which has representation adjacency matrix, and built a quantum algorithm for Wigderson algorithm for graph presented as adjacency matrix.

QUANTUM ALGORITHMS, GRAPH ALGORITHMS, GRAPH COLORING, PROPERTIES OF SUPERPOSITION STATES

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

BFT — Breadth First Traversal(Обхід в ширину)

$\oplus$ — операція побітового додавання

DFT — Depth First Traversal(Обхід в глубину)

ВСТУП

Актуальність дослідження. Одним із трендів сучасного світу являються квантові комп'ютери. Квантові комп'ютери кардинально змінюють поняття людей про інтернет, методи передачі даних, кібербезпеку, криптографію, машинне навчання та багато ще інших галузей.

Все почалося в 1980 роках, коли Пол Беніофф запропонував квантову модель машини Тюрінга. З того часу почалося більш детальне дослідження можливостей квантового комп'ютера. І на сьогодні зроблено уже багато квантових алгоритмів, які є набагато ефективніші за класичні алгоритми. Завдяки великим компаніям, кожна з яких намагається зробити свій прототип квантового комп'ютера, росте попит на квантові алгоритми для різних цілей.

Одні з задач, які можуть бути ефективно розв'язані у квантовій моделі, є задачі на графах[1]. Теорія графів покриває багато областей сучасного життя від транспортних та комп'ютерних мереж до будівельного проєктування та молекулярного моделювання. Одна з найвідоміших задач на графах, є задача розфарбування графу, яка використовується в багатьох різних сферах людської життєдіяльності. А також задачі розфарбування графу, використовуються для розв'язання інших задач на графах, наприклад задача розфарбування графу двома кольорами, може бути використана для визначення чи є граф двочастковим, чи ні.

Метою дослідження є побудувати квантовий алгоритм для задачі розфарбування графа з використанням властивостей суперпозицій станів. Для досягнення мети необхідно вирішити такі завдання:

1) детально розібрати особливості побудови алгоритмів у квантовій моделі;

2) дослідити наявні розв'язки в класичній моделі;

3) побудувати квантовий алгоритм розфарбування двома кольорами графа, який представлений у вигляді матриці суміжностей;

4) побудувати квантовий алгоритм Вігдерсона для графа, представленому як матриця суміжностей;

5) обчислити складність побудованих алгоритмів.

Об'єктом дослідження є процеси перетворення інформації в системах захисту

Предметом дослідження є складність задачі розфарбування графів у квантовій моделі обчислення

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: методи лінійної та абстрактної алгебри, теорії імовірностей, теорії складності алгоритмів, теорії графів та методи квантової моделі обчислень.

Наукова новизна отриманих результатів полягає у тому, що вперше розроблено квантовий алгоритм для розфарбування двома кольорами графа, представлений як матриці суміжностей, а також побудовано квантовий алгоритм Вігдерсона для графа у вигляді матриці суміжностей.

Практичне значення результатів полягає у тому, що можна використовувати розроблені алгоритми для подальших задач у квантових системах для графів, представлених як матриця суміжностей.

1 КВАНТОВІ АЛГОРИТМИ НА ГРАФАХ ОПИС ТА АНАЛІЗ

В цьому розділі розповідається про різні реалізації квантових алгоритмах на графах з використанням властивості суперпозицій станів. Це виконується задля того, щоб показати, що ця тема є доволі популярною та прогресивною. Також цей розділ показує основні переваги квантових алгоритмів з використанням властивості суперпозицій станів відносно класичних алгоритмів.

1.1 Квантовий алгоритм для перевірки на зв'язність графів та задачі оцінки формули

В цьому підрозділі розповідається про квантовий алгоритм для зв'язності графів та оцінки формули, який реалізований в роботі Стейсі Джеффрі та Шелбі Кіммелом[1].

Задача зв'язності графів – це задача вибору, яка полягає в тому, щоби сказати чи є досяжна вершина t з вершини s. Тобто якщо найдеться такий шлях який веде з вершини s до t, то говориться, що вершина t досяжна з вершини s, інакше вершина є недосяжною. Ця задача може бути використана для розв'язання різних задач на графах, а саме для виявлення певних підграфів[2], виявлені чи граф є лісом та чи є він двочастковим графом[3]. В статті Джеффрі та Кіммела розповідається про покращення квантового алгоритму для задачі зв'язності графів для деяких сімейств графів[1]. Покращення було зроблено для тих випадків коли вхідний граф є підграфом деякого планарного графа. Покращення впливає на ефективний опір основного графа та планарного. Також пошук таких сімейств графів відповідно дає експоненційне та поліноміальне покращення відносно основного алгоритму.

Також в статті показано як можна застосовувати квантовий алгоритм

для проблеми зв'язності графів на прикладі розв'язання проблеми оцінки формули.

Задача оцінки формули – це обчислювальна задача, яка дає вивід булевої схеми, але при умові що булева схема є деревом. Ця задача є NC складною. В статті Джеффрі та Кіммел показали, що використовуючи зведення задачі зв'язності графів до задачі оцінки формули можна вирахувати булеву схему з N вхідними змінами за $O(\sqrt{N})$ запитів[1]. Також для деяких сімейств графів можна отримати квадратичне та суперполіноміальне прискорення відносно класичного алгоритму цієї задачі.

Ця стаття показує можливості квантового комп'ютера для задачі зв'язності графів та можливість зведення квантового алгоритму. Також показується детальне зведення однієї задачі до іншої. При цьому отримуючи квадратичне та суперполіноміальне прискорення в деяких окремих випадках.

1.2 Квантовий алгоритм для задачі вивчення прихованого графа і не тільки

В цьому підрозділі розповідається про реалізацію квантового алгоритму для задачі вивчення прихованого графа, а також про схожі задачі розв'язані Ешлі Монтанаро та Чанпенг Шао[4].

Задача вивчення прихованого графа – це задача, яка говорить для набору вершин чи є там ребро з прихованого графа чи ні. В статті Ешлі Монтанаро та Чанпенг Шао розповідається про реалізації таких квантових алгоритмів, які розв'язують цю задачу[4]. Реалізовують вони цей алгоритм завдяки різним видам запитів до квантового оракула.

Перша модель запиту до оракула працює схожим чином до класичної моделі, тобто оракул говорить чи множина вершин має якісь ребра невідомого графу чи ні. Або іншими словами дано граф $G = (V,E)$

де V – це множина вершин, а E – множина ребер, ця модель бере підмножину множини вершин $S \subseteq V$ та говорить чи є множина $E \cap (S \times S)$ порожньою. Квантова реалізація цієї моделі використовує $O(m \log(\sqrt{m} \log n) + \sqrt{m} \log n)$ запитів, коли нижня межа класичного алгоритму має $\Omega(m \log(N^2/m))$. Нижня межа юудб-якого квантового алгоритму вивчення невідомого випадкового графа в цій моделі повинна мати $\Omega(n^2)$ запитів, тому складність цього алгоритму не може бути кращою. Також ця модель може квантово прискорити деякі задачі, наприклад пошук Гамільтонового циклу, задачі відповідності, пошук зірок та клік, а також в деяких задачах молекулярної біології.

Друга модель повертає кількість ребер прихованого графа в множині вершин. Тобто запит до оракула повертає $|E \cap (S \times S)|$. Цю модель можливо представити також через першу модель, наприклад просто повертати розмір $E \cap (S \times S)$, ця модель також розглядалась в класичній моделі.Квантова реалізація цієї моделі для невідомого графу з максимальним степенем d розв'язується за $O(d \log m)$ запитів, коли нижня межа цього алгоритму в класичній моделі дорівнює $\Omega(nd \log(n/d))$. Також є квантовий алгоритм для такої моделі, коли граф має m, то потрібно буде лише $O(\sqrt{m \log m})$, коли нижня межа класичного алгоритму $\Omega(m \log(n^2/m))$. Для деяких випадків графів цей алгоритм навіть може виконуватись за сталий час, наприклад якщо граф має вигляд зірки або кліки то вивчити його можна завдяки цьому алгоритму всього лиш за $O(1)$ запитів. Це навіть краще ніж в першій моделі.

Третя модель дає копії станів графу відповідно до графа. В цьому випадку кожен граф G має свій стан $|G\rangle$. Та для того щоби розв'язати задачі розпізнавання невідомого графу потрібно знайти всі копії $|G\rangle$ графу G. Уже відомо, що потрібно $\Theta(n)$ копій для того, щоби розпізнати граф G, якщо граф G випадковий граф з n кількістю вершин. В статті Ешлі Монтанаро та Чанпенг Шао побудовано такий алгоритм, але з умовою що G має d кількість ребер, тоді можна розпізнати граф використовуючи $O(d \log m)$ копій. Також в деяких випадках можна

отримати покращення, наприклад коли відомо, що граф який потрібно розпізнати, можливо розпізнати з набору вершин довжиною L тоді потрібно $O(\log L)$ копій.

Завдяки цим трьом моделям вдалося побудувати три різні алгоритми розв'язку задачі вивчення прихованого графа. А також завдяки першій моделі в статті покращено квантовий алгоритм для розрахунку булевих функцій, оскільки той також працює на такій же моделі, але з гіршим часом виконання.

Порівнюючи з класичними алгоритмами квантова реалізація є набагато швидша, як показано в статті Ешлі Монтанаро та Чанпенга Шао[4]. В деяких випадках навіть можна отримати експоненційне прискорення. Також завдяки статті Ешлі Монтанаро та Чанпенга Шао можна побачити, що перетворюючи різні алгоритми запитів з класичної моделі у квантову, можна отримати не абияке покращення. Тим більше ці алгоритми можна використовувати в зовсім різних випадках і задачах, що охоплює великий клас задач для графів і не тільки.

1.3 Квантовий алгоритм для задачі відповідності та розфарбування вершин графу

Задача розфарбування вершин графу одна з найвідоміших задач на графах. Задача полягає в тому, щоб розфарбувати вершини графу так, щоб дві сусідні вершини не мали однакового кольору.В теорії графів[5] ця задача має вигляд функції $f : V(G) \to \mathbb{N}$ для таких : $\forall a_i,\ a_j \in V(G)$, $i \neq j \exists (e_i,e_j) \Rightarrow f(i) \neq f(j)$[4].

Ця задача для k=2 належить класу складності P, а якщо k>2, то належить класу складності NP. Також відомо, що задача при k=3 є NP-повною[6].

В статті Набіха Асгара[7] розглядається цікаве використання квантової суперпозиції станів до задачі розфарбування вершин графу.

Попри те, що ця задача може бути розв'язана за однаковий час як і у квантовій моделі, так і в класичній, розв'язок цієї задачі у квантовій моделі дає значну перевагу в тому, що використання квантової суперпозиції дає не одну відповідь, тобто не одне розфарбування графу, а одразу всі й це дійсно одне з незвичайних способів використання властивостей квантової суперпозиції. Цей підхід досягається тим, що вершини графа кодуються як кубіти, а кольори, в які хочемо розфарбувати граф, в стани кубітів. Наприклад для двох кольорів стани будуть такими: $|0\rangle$(перший колір) та $|1\rangle$(другий колір). Потім створюється суперпозиція всіх можливих розфарбувань та використовуються різні оператори для побудування розфарбувань. Таким чином отримується суперпозиція всіх можливих розфарбувань, обрахувавши яку можна отримати випадкове розфарбування, але, якщо працювати з цією суперпозицією далі, то можна використовувати її як всі можливі розфарбування в подальших квантових алгоритмах, що є дуже корисним.

Ще одну цікаву реалізацію задачі розфарбування графу виконали Кадзуя Шимідзу та Рюхей Морі[8]. Відомо, що найефективніший класичний алгоритм розв'язує цю задачу за $\Omega(2^n)$ коли $k \geqslant 5$, де k – це кількість вершин. Попри те, що покращити NP-повні задачі у квантовій моделі є все-таки дуже важкою задачею Кадзуї Шимідзу та Рюхею Морі вдалося отримати алгоритм розфарбування графу 20-кольорами з часом виконання $O(1.9575^n)$. Це досягалось завдяки застосуванню алгоритму Гровера. Також показано цікавий підхід використання квантового випадкового доступу до пам'яті(QRAM), завдяки цьому підходу використання пам'яті було отримано квантовий алгоритм для розрахунку хроматичного числа графу за час виконання $O(1.9140^n)$. Хроматичне число графу – це мінімальна кількість кольорів в які ми можемо розфарбувати граф.

Також в статті Набіха Асгара[7] розповідається про ще одну можливу реалізацію алгоритмів у квантовій моделі обчислень. Підхід для цієї реалізації – це один з найвідоміших підходів задля покращення

класичного алгоритму у квантовій моделі. В цьому випадку береться класичний алгоритм для задачі відповідності у двочасткових графах. Задача відповідності у двочасткових графах - це набір таких вершин, що ніякі дві вершини в цьому наборі не мають однакову кінцеву вершину. Ця задача поліноміально прискорюється завдяки алгоритму Гровера як підпрограми. Це показує як використання алгоритму Гровера для класичних алгоритмів покращує роботу алгоритму.

Однією з задач відповідності – є задача максимальної відповідності, яка полягає в тому, щоби знайти відповідність для найбільшої кількості ребер між набором несусідніх ребер в неорієнтованому графі. Шелбі Кіммел та Р.Тіл Віттер запропонували квантовий алгоритм для цієї задачі[9]. При тому цей квантовий алгоритм виконується для графа представленого як в матриці суміжності, так і в списку суміжності вершин. Такий квантовий алгоритм є виконаним завдяки комбінуванню класичного алгоритму Габова[10] та квантового методу вгадування дерева, який був запропонований Бейджіном та Тагаві[11]. Алгоритм робить $O(n^{7/4})$ запитів для графа представленого як матриця суміжності, та $O(n^{3/4}\sqrt{(m+n)})$ запитів для списку суміжності, де m – це кількість ребер, а n – це кількість вершин.

Цей підрозділ показує деякі з можливостей, що надає квантова модель обчислень. Можна реалізовувати класичні алгоритми у квантовій моделі повністю і працювати з суперпозиціями різним чином, як було наведено в прикладі отримуючи відповідь в вигляді суперпозиції, що відкриває різні види задач. Або використовувати квантову модель лише для підпрограм для класичних алгоритмів тим самим покращуючи швидкість виконання програми. Прискорення класичних алгоритмів, як можна побачити, виконується завдяки використанню алгоритму Гровера для деяких підалгоритмів.

1.4 Квантовий алгоритм для задач обходу графів

Задача обходу графу – це такий клас задач, основна ціль якого полягає в пошуку проходження графу по його вершинам з різними умовами. Самі відомі задачі цього класу, є: пошук ланцюга Ейлера, Гамільтонового циклу та мабуть, найвідоміша це є задача комівояжера. Цей клас задач є дуже популярний в теорії графів, оскільки він багато де використовується. Тому не є дивним, що реалізація цих задач також є і у квантовій моделі.

Задача пошуку ланцюга Ейлера полягає в тому, щоби знайти в даному орієнтованому графі шлях, який проходить кожну вершину графу один раз, тоді такий граф називається Ейлеровим.

Задача пошуку Гамільтонового циклу полягає в тому, щоби при даному орієнтованому графі знайти в ньому цикл який включає кожну вершину графу один раз. Якщо таке виконується в графі, то ми називаємо такий граф Гамільтоновий.

Задача комівояжера(TSP) – це задача проходження вершин найкоротшим шляхом. Тобто задача по суті є задачею пошуку Гамільтонового циклу, але такий цикл повинен бути найменшим по вазі, сума ваг всіх ребер в цьому циклу повинна бути найменша з усіх циклів. Ця задача є NP важкою.

У своїй статті Себастьян Дорн[12] реалізував ці задачі послідовно завдяки квантовому алгоритму пошуку Гровера, тим самим зменшивши час виконання. Через те, що алгоритм пошуку Гровера по суті можна використовувати як ефективний алгоритм для квантового обходу графу.

Також у статті Картіка Срінівасана, Сайпрія Сатьяджіта, Бікаша К. Бехера та Прасанта К. Паніграхі[13] запропоновано ефективний алгоритм для цієї задачі. Підхід, який використовується в цьому алгоритмі є не тривіальним, як в попередній статті та цим самим він доволі цікавий. Підхід полягає в закодування даних відстаней(ваг) між

вершинами як фази. Також там будуються унітарні оператори власні вектори яких є обчислювальними базовими станами, а власні значення – різні комбінації цих фаз. Потім вираховуються всі можливі ваги ребер, завдяки власним векторам і завдяки квантовим алгоритмам пошуку, алгоритм шукає мінімальну вагу по якій і буде будуватись шлях обходу. Цей підхід дозволяє квадратично покращити класичний метод перебору для великої кількості вершин. Також можна побачити в тій статті як показано симуляцію коду завдяки квантовому симулятору IBM для цієї задачі.

1.5 Квантовий алгоритм для Гамільтонового I-HE дерева

I-HE дерево будується наступним чином: в кожний листок дерева задаються значення 1 або 0, а для кожного наступного вузла буде виконуватись операція I-HE, яке працює як заперечення до логічної операції I. Можна побачити, що для N кількості листків глибина цього дерева буде досягати $n = \log_n(N)$. Основна ціль цієї задачі полягає в тому, щоби обрахувати значення кореня дерева. В статті Фархі, Голдстона та Гутмана[14] розповідається про цікаву реалізацію цього алгоритму в квантовій моделі. Саме незвичайне є те, що там використовується не звичайний квантовий оракул[15], а Гамільтоновий[16, 17]. Сам гамільтоновий оракул виглядає наступним чином:

$$H_O = \sum_{i=1}^{N} H_j$$

де H_j – це оператор в ортогональному підпросторі. Кожний такий оператор може бути або оператором з 0, або з 1 відносно початкових значень листків.

Цей алгоритм є доволі цікавим як квантова реалізація тому, що не має схожих задач вирішених на такому оракулі тим більше нижня межа цього алгоритму є $\sqrt{N}$. Цей підхід не є тривіальним, та його можна застосувати до задач зв'язаних з деревами графів.

1.6 Алгоритм квантового запиту для задачі колізії графів

Задача колізії графів - це така задача, яка визначає чи в наборі промаркерованих вершин є дві сусідні вершини. Тобто, якщо у нас є граф $G(V,E)$, та дано рядок промаркерованих вершин $x \in \{0,1\}^V$, то задача полягає в тому щоби визначити чи існує таке ребро $(i,j) \in E$, що $x_i = x_j = 1$. Дмитро Гавінський та Цуйосі Ітo[18] продемонстрували свою реалізацію квантового алгоритму цієї задачі. Складність запиту якого досягла $O(\sqrt{n} + \sqrt{\alpha * (G)})$, де n є кількістю вершин, а $\alpha * (G)$ максимальному степеню графу G. Але, якщо G випадковий граф, в якому кожне ребро присутнє з відповідною ймовірністью, тоді алгоритм потребує $O(\sqrt{b \log n})$ запитів.

Для розв'язання цієї проблеми також було застосовано алгоритм Гровера і це можна зрозуміти чому. Алгоритм Гровера є першим прикладом такого алгоритму запит якого є набагато меншим ніж в класичній моделі. Наприклад, вираховуючи булеву функцію або для якоїсь кількості значень класичному алгоритму буде потрібно $\Omega(n)$ запитів, коли квантовому алгоритму буде потрібно лише $O(\sqrt{n})$ квантових запитів.

1.7 Алгоритм квантового блукання

Квантове блукання – це по суті є копією класичного випадкового блукання, але в класичному випадку стан випадкового блукання це ймовірносний розподіл, який описується стохастичною матрицею, яку можна отримати з матриці суміжності графу. Випадкове блукання виконується наступним чином, якщо в момент часу t оператор знаходиться на вершині v, то на сусідній вершині вершини v він повинен появитись в момент часу $t+1$. В квантовій моделі квантове блукання – це L_2 вектор в гільбертовому просторі, який динамічно описується

унітарною матрицею, яка піддається законам квантової механіки, а також така матриця повинна бути локальною[19]. Квантове блукання – це дуже цікавий алгоритм, який не просто використовується для покращення класичних алгоритмів, а також використовується задля того, щоб аналізувати складні фізичні системи[20, 21].

Перша реалізація квантового блукання була виконана у 2003 році Амбаїном[22]. На основі квантового блукання Амбаїн зробив власний алгоритм пошуку на спеціальних двочасткових графах, цей алгоритм розв'язував задачу досяжності вершини графу. Потім алгоритм Амбіїна був розширений Сегедіном[23], який отримав квадратичне покращення для цієї задачі.

В статті Стівена Піддока[24] наведена реалізація квантового алгоритму блукання для задачі пошуку промаркерованої вершини, коли починається пошук з випадкової вершини графу. Завдяки роботам Амбаїна[22] та Сегедіна[23], а також завдяки програмному каркасу Белова[25], який показав, що можна знайти промаркеровану вершину за $O(\sqrt{RW})$ кроків квантового блукання, де R – це ефективний опір, а W – це загальна вага. Тобто загальна вага – це сума всіх ваг ребер графу, а ефективний опір підраховується, коли граф представляється як електронна мережа, де кожна ребро e презентує провід з опором $1/w_e$, де w_e – це вага ребра e.

Можна побачити, що квантове блукання дуже сильно розвивається та використовується у квантовій моделі для пришвидшення класичних алгоритмів. Багато алгоритмів на графах можуть бути покращені завдяки квантовому блуканню, зазвичайу це алгоритми які розв'язують задачі обходу графів, але ці задачі на графах є дуже важливими для теорії графів.

1.8 Квантовий алгоритм для розрізу графа

Мінімальний розріз графу $G = (V,E)$, де V це множина вершин, а E множина ребер. Кожне ребро $e \in E$ визначається як набір значень (a,b,w_e), де a та b являються вершинами, а w_e вагою графу e. Тоді розріз $C(A,B)$ це підмножина множини ребер E така, що складається з вершин $a \in A$, $b \in B$ які належать до різних множин. Мінімальний розріз буде тоді таким розрізом який має мінімальну вагу, тобто: $min_{C \subset E}(\sum_{e \in C} w_e)$.

У статті Лізи Це, Пітера Маунтні, Пола Кляйн та Сімона Северіні[26] розповідається про реалізацію такої задачі в квантовій моделі завдяки алгоритму близької квантової оптимізації. Більш того було показано застосування такої реалізації для сегментування фотографій на сучасних квантових пристроях.

Висновки до розділу 1

Як висновок, можна побачити, що задачі на графах є дуже поширеними для квантової моделі. Всі задачі були вирішенні завдяки різним властивостям суперпозицій станів, що показує можливості квантової моделі. Також можна побачити, що задачі на графах, які представлені як матриця суміжностей ефективно розв'язуються квантовими алгоритмами, але не всі задачі є розв'язаними. Наприклад задача розфарбування графу двома кольорами, де граф представлений матрицею суміжностей не була ще розв'язана у квантовій моделі, а також квантовий алгоритм Вігдерсона, також не був ще реалізованим у квантовій моделі обчислення. Також по першому розділу можна побачити, що всі методи реалізацій квантових алгоритмів є варіативними. Хоча зазвичай квантові алгоритми будують завдяки прискоренню класичного алгоритму методом швидкого пошуку Горнера, але це не є єдиним методом. Ще одним важливим методом є метод випадкового

блукання у квантовій моделі числення. Такий метод також є ефективним у використані.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ ТА ПІДХОДИ ДЛЯ РЕАЛІЗАЦІЙ АЛГОРИТМІВ В КВАНТОВІЙ МОДЕЛІ ОБЧИСЛЕННЯ

В цьому розділі розповідається про основні механізми квантової моделі, а також алгоритми розв'язання задач на графах, які будуть реалізовані у квантовій моделі завдяки розібраним реалізаціям. А також більш детальне пояснення їх роботи.

2.1 Теоретичні відомості

Перед тим як розглянути саму реалізацію квантового алгоритму, потрібно ввести деякі відомості про квантове обчислення.

Квантова система в цьому розділі буде позначатися в нотації бра-кет $|\psi\rangle$, яка запроваджена Полем Діраком[27], для зручності використання. Система з n кубітів буде описана суперпозицією з 2^n символів. Тобто кубіт можна представити в стані $|1\rangle$ або $|0\rangle$, якщо у нас всього два можливих станів. Також можна представити кубіт у вигляді суперпозиції станів. Це такий стан який визначається як лінійна комбінація всіх можливих станів. Наприклад для двох можливих станів суперпозиція буде мати вигляд:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Тут α та β – це комплексні числа. До суперпозиції станів можна виконати обчислення, яке дасть одне значення, в випадку двох станів обчислення видасть або $|1\rangle$, або $|0\rangle$. Також можна вирахувати ймовірність стану, наприклад $|\alpha|^2$ – це ймовірність, що після обчислення у нас буде стан $|0\rangle$, а $|\beta|^2$ – що буде стан $|1\rangle$. Очевидне є те, що у сумі коефіцієнтів при станах повинна дорівнювати 1. Тобто в випадку з двома станами повинно виконуватись наступне $|\alpha|^2 + |\beta|^2 = 1$.

Кожну суперпозицію станів можна представити у вигляді вектора. Тобто, наприклад для $|\psi\rangle$:

$$|\psi\rangle = \alpha \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \\ \beta \end{bmatrix} \tag{2.1}$$

$|\psi\rangle$ – називається кет вектором у квантовій моделі. А $\langle\psi|$ – називається бра вектором в квантовій моделі. Вони зв'язані тим що, один являється матрицею з один рядком, а другий такою самою матрицею тільки з одним стовпчиком, тобто $|\psi\rangle^T = \langle\psi|$.

Оператори у квантовій моделі працюють наступним чином. Наприклад є оператор A, він перетворює один кет вектор в інший. Тобто:

$$A|\psi\rangle = |\phi\rangle$$

Оператор таким же чином працює і на бра вектор:

$$\langle\psi|A = \langle\phi|$$

Одиним з найвідоміших операторів у квантовому обчисленні є оператор Паулі, якій працює як оператор заперечення. Зазвичай він позначається як X і працює наступним чином:

$$X|0\rangle = |1\rangle,\ X|1\rangle = |0\rangle$$

Для $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ він буде мати вигляд матриці, а саме:

$$X = \begin{bmatrix} 0 & 1 \\ \\ 1 & 0 \end{bmatrix}$$

тоді $X|\psi\rangle = \beta|0\rangle + \alpha|1\rangle$

2.2 Розв'язання задачі розфарбування графу двома кольорами у квантовій моделі

В цьому підрозділі більш детально розповідається про розв'язання проблеми розфарбування вершин графу двома кольорами, а також про метод який буде використовуватись в третьому розділі. Для початку, щоб реалізувати алгоритм у квантовій моделі потрібно більше детально ознайомитись з класичним алгоритмом.

Для початку розглянемо класичний алгоритм описаний Набіхом Асгаром[7]. Цей алгоритм написаний для списку суміжності графу, але так, як задача полягає в тому, щоб показати для матриці суміжності графу, то алгоритм буде перероблений.

Для того, щоб розв'язати задачу розфарбування графу спочатку потрібно представити метод обходу графу. Одним зі способів обійти граф є метод обходу в глибину(DFT). DFT метод працює з графом, як з деревом, навіть якщо він не є деревом. Тобто спочатку обирається вершина яка буде коренем, а потім алгоритм йде в глибину, поки не вертається до вершини що була. В наступному алгоритмі використовуються означення деревоподібне ребро, а також повертаюче ребро.

Означення 2.1. Деревоподібне ребро – це таке ребро, в якому кінцева вершина не є відвіданою.

Означення 2.2. Повертаюче ребро – це таке ребро, що введе в вершину, яка була вже відвідана.

Тобто по деревоподібним ребрам можна побудувати дерево, А повертаючі ребра утворюють цикл в такому дереві.

Алгоритм 2.1. DFT алгоритм для списку суміжності графу
Дано неорієнтований граф G = (V, E)(працюємо з графом як зі списком):
1: Обираємо випадкову вершину $r \in V$ і позначаємо її як відвідану.

2: Для кожного ребра e який є в списку ребер вершини r виконуємо наступне:

3: Якщо w кінцева вершина ребра e не була ще відвідана, то:

4: Позначаємо e як деревоподібне ребро.

5: Рекурсивно виконуємо алгоритм DFT для вершини w як кореня.

6: Інакше:

7: Позначаємо e як повертаюче ребро.

Можна побачити, що кожна вершина була оброблена не більше одного разу, так як і ребро.

Твердження 2.1. *([7]) Тобто, якщо у нас граф має n вершин, та m ребер, то складність алгоритму 2.1 дорівнює* $O(n + m)$.

Використання DFT алгоритму, як методу обходу, дає перевагу в тому, що можна визначити чи є цикли в графі. Відомо що, якщо в графі є непарний цикл, тобто цикл в якому є непарна кількість вершин, то такий граф неможливо розфарбувати двома кольорами. Алгоритм розфарбування графу двома кольорами за допомогою алгоритму DFT працює наступним чином:

Алгоритм 2.2. Алгоритм для розфарбування графа двома кольорами на принципі алгоритму DFT

Дано неорієнтований граф G = (V, E)(працюємо з графом як зі списком):

0: Створюємо глобальну зміну і називаємо її L. Ініціалізуємо її значенням '1'.

1: Обираємо випадкову вершину $r \in V$ і присвоюємо їй значення L.

2: Для кожного ребра e який є в списку ребер вершини r виконуємо наступне:

3: Якщо w кінцева вершина ребра e не була ще позначена, то:

4: Позначаємо w як $L + 1(mod 2)$.

5: Позначаємо L як $L + 1(mod 2)$ і рекурсивно виконуємо цей

алгоритм для вершини w як кореня.

6: Інакше:

7: Перевіряємо значення вершини w. Якщо це така сама вершина як і r то зупиняємо алгоритм і говоримо, що ця задача не є виконувана для такого графу.

Твердження 2.2. *([7]) Складність алгоритму 2.2 дорівнює* $O(n+m)$.

Можна побачити, що алгоритм має по суті таку саму кількість кроків як і DFT алгоритм. Тобто кожна вершина, та кожне ребро оброблюється не більше одного разу. Це значить, що складність алгоритму така сама як в DFT алгоритмі $O(n+m)$.

Тепер, щодо реалізації такого алгоритму для матриці суміжності графу. Для початку, як і для випадку зі списком суміжності, потрібно задати порядок обробки вершин. Буде використовуватись такий самий порядок як і для випадку зі списком суміжності, але алгоритм дещо буде відрізнятись. Відрізнятися вони будуть тим, що у випадку матриці суміжності немає прямого доступу до ребер, а лише є до вершин. Тому для початку потрібно задати основний каркас алгоритму типу стеку, а потім поверх нього описати алгоритм DFT для матриці суміжності. Каркас алгоритму пошуку в глибину буде рекурсивним алгоритмом, та виглядати буде наступним чином:

Алгоритм 2.3. Каркас алгоритму DFT

На вхід дається вершина v:

1: Позначаємо вершину v як відвідану.

2: Для кожної вершини u, що є сусідом вершини v виконується наступне:

3: Якщо u не була ще відвідана, то:

4: Рекурсивно запускаємо цей алгоритм для вершини u.

5: Інакше закінчуємо роботу алгоритму

Тепер можна зовсім легко побудувати алгоритм для DFT для матриці суміжності незваженого неорієнтованого графа. Далі потрібно вважати, що вершина не може бути зв'язана сама з собою ребром.

Алгоритм 2.4. DFT алгоритм для матриці суміжності графу

Дано матриця суміжності графу G:

1: Обираємо випадкову вершину $v \in G$ і позначаємо її як відвідану.

2: Для кожної вершини графу $u \in G$, крім вершини v виконуємо наступне:

3: Якщо $G(v,u) = 1$, а також вершина u не є ще відвіданою то:

4: Рекурсивно виконуємо алгоритм DFT для вершини u як кореня.

Твердження 2.3. *Складність алгоритму 2.4 дорівнює* $O(n(n-1))$.

Доведення. Працюючи з графом через матрицю суміжності, можна побачити, що алгоритм 2.4 оброблює для кожної вершини $(n-1)$ вершин. Тому складність алгоритму буде $O(n(n-1))$. □

Хоча може показатися, що працювати з представленням графу в списку суміжності є вигідніше, але це не є завжди так. Тому що, маючи велику кількість ребер, алгоритм побудований на списку суміжностей буде виконуватись повільніше.

Далі буде приведений класичний алгоритм для розфарбування графу двома кольорами на принципі алгоритму DFT з використанням графу, як матриці суміжностей.

Алгоритм 2.5. Алгоритм для розфарбування графа двома кольорами на принципі алгоритму DFT

Дано неорієнтований граф G (працюємо з графом як з матрицею):

0: Створюємо глобальну зміну і називаємо її L. Ініціалізуємо її значенням '1'.

1: Обираємо випадкову вершину $v \in V$ і присвоюємо їй значення L.

2: Для кожної вершини u, крім вершини u, виконуємо наступне:

3: Якщо $G(v,u) = 1$, а також не була ще позначена, то:

4: Позначаємо u як $L + 1(mod 2)$.

5: Позначаємо L як $L + 1(mod 2)$ і рекурсивно виконуємо цей алгоритм для вершини u як кореня.

6: Інакше:

7: Перевіряємо значення вершини u. Якщо це така сама вершина як і u то зупиняємо алгоритм і говоримо, що ця задача не є виконувана для такого графу.

Твердження 2.4. *Складність алгоритму 2.5 дорівнює* $O(n(n-1))$.

Доведення. Працюючи з графом через матрицю суміжності, можна побачити, що алгоритм 2.5 оброблює для кожної вершини $(n-1)$ вершин, тобто так само як просто алгоритм обходу DFT. Тому складність алгоритму буде $O(n(n-1))$. □

Тепер можна приступити до квантової реалізації. Квантова реалізація такого алгоритму є доволі цікавою, оскільки алгоритм є повністю квантовий, а також працює з вершинами графу як з кубітами, а кольори представляються в вигляді суперпозиції станів. Також основна ідея полягає в тому, щоб спочатку згенерувати набір всіх можливих розфарбувань графу ігноруючи ребра, а потім поступово додавати кожне ребро. Генерація суперпозиції всіх можливих розфарбувань виконується завдяки матриці Адамара, яка має вигляд:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.2}$$

Тоді оператор $H^{\oplus n}$ створює рівнозважену суперпозицію всіх розфарбувань. Для випадку з двома кольорами генерація суперпозицій буде мати наступний вигляд:

$$H^{\oplus n}|0\rangle^{\oplus n} = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle$$

де i записано як бітову послідовність. Для двох вершин буде виглядати як $|00\rangle + |01\rangle + |10\rangle + |11\rangle$ з коефіцієнтом нормалізації $1/2$.

Але для розфарбування двома кольорами, зрозуміло, що неможливо щоб дві сусідні вершини мали стани $|00\rangle$ та $|11\rangle$. Для розв'язання цієї проблеми потрібний оператор, який буде позбуватися від таких значень, а потрібні значення залишати. Такий оператор буде мати вигляд:

$$E_{ab} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{2.3}$$

Такий оператор є унітарним, а також він буде працювати наступним чином:

$|00\rangle_{ab} \to |01\rangle_{ab} + |00\rangle_{ab}$

$|01\rangle_{ab} \to |01\rangle_{ab} - |00\rangle_{ab}$

$|10\rangle_{ab} \to |10\rangle_{ab} + |11\rangle_{ab}$

$|11\rangle_{ab} \to |10\rangle_{ab} - |11\rangle_{ab}$

з коефіцієнтом нормалізації $1/\sqrt{2}$. Такий оператор справедливий для деревоподібних ребер, але є ще повертаючі ребра, які визначають чи можливо розфарбувати граф двома кольорами. Такі ребра значать, що кінцева вершина є вже відвіданою і тоді потрібно перевірити, чи зазначений колір не співпадає. Для цього потрібно обрахувати проєктивне вимірювання M_{ab}, яке складається з проектуючих оператора O_{ij} та з $I - O_{ij}$ де

$O_{ij} = |01\rangle_{ij}\langle 01| + |10\rangle_{ij}\langle 10|$

$I - O_{ij} = |00\rangle_{ij}\langle 00| + |11\rangle_{ij}\langle 11|$

Таке вимірювання є детермінованим. Результат вимірювання 0 буде відповідати проєкції O_{ij}, а результат вимірювання 1 проєкції $I - O_{ij}$.

Тобто якщо вимірювання дорівнює 1, то можна припинити виконувати алгоритм, так, як дві сусідні вершини розфарбовані одним кольором, в інакшому випадку продовжується виконання. Тоді квантовий алгоритм розфарбування двома кольорами графу буде мати вигляд:

Алгоритм 2.6. Квантовий алгоритм для розфарбування двома кольорами графу

Дано неорієнтований граф G = (V, E):

1: Генеруємо суперпозицію всіх можливих розфарбувань.

2: Для кожного ребра $e_{ab} \in E$ в DFT порядку виконуємо наступне:

3: Якщо e_{ab} є деревоподібним ребром, то:

4: Виконуємо E_{ab}

5: Інакше:

6: Виконуємо M_{ab}

7: Якщо $M_{ab} = 1$ то зупиняємося і вертаємо що "Не існує розфарбування двома кольорами для такого графу"

8: Обраховуємо кінцеву суперпозицію для отримання випадкового розфарбованого двома кольорами графу, або використовуємо суперпозицію всіх розфарбувань графу G для подальших процесів.

Твердження 2.5. *([7]) Складність алгоритму 2.6 дорівнює* $O(n + m)$*, де* n *кількість вершин, а* m *кількість ребер.*

2.3 Алгоритм Вігдерсона в класичній моделі

В цьому підрозділі надається алгоритм Вігдерсона, а також алгоритми які використовуються в алгоритмі Вігдерсона.

Класичний алгоритм Вігдерсона – це алгоритм, який розв'язує задачу розфарбування трьох-хроматичного графа $1 + \lceil\sqrt{n}\rceil$ кольорами. Реалізація цього алгоритму виконана для варіанту коли граф представляється як список суміжності. Цей алгоритм має в собі два підалгоритми, а саме жадібне розфарбування графу, а також

розфарбування графу двома кольорами. Алгоритм Вігдерсона взятий у статті Тора Хасфельда[28], перероблений для зручності розуміння та наведено нижче.

Алгоритм 2.7. Алгоритм Вігдерсона

Дано трьох-хроматичний граф G = (V, E), цей алгоритм знаходить розфарбування вершин графу $O(\sqrt{n})$ кольорами.

1: [Ініціалізація].Нехай $c = 1$.

2: [$\Delta(G) \geqslant \lceil\sqrt{n}\rceil$].Шукаємо вершину в G зі $degv \geqslant \lceil\sqrt{n}\rceil$. Якщо такої не існує то переходимо до кроку W3. Інакше застосовуємо алгоритм розфарбування двома кольорами графу для того, щоб розфарбувати сусідів G[N(v)] кольорами c та $c + 1$. Видаляємо N(v) з G і збільшуємо c на $\chi(G[N(v)])$. Повторюємо крок 2.

3: [$\Delta(G) < \lceil\sqrt{n}\rceil$].Використовуємо жадібний алгоритм розфарбування графу, щоби розфарбувати вершини які залишились c,$c + 1$,...,$c + \lceil\sqrt{n}\rceil$кольорами.

Твердження 2.6. *([28]) Складність алгоритму 2.7 є $O(n + m)$, де n кількість вершин, а m кількість ребер.*

Для розфарбування графу двома кольорами в алгоритмі Вігдерсона використовується зовсім інакший порядок обробки вершин. В алгоритмі розібраному раніше використовувався порядок пошуку в глибину, який працював за схемою стеку, тобто перший зайшов - останній вийшов. Але в алгоритмі Вігдерсона можна побачити, що на другому кроці, обробляються не кінцеві вершини, а сусідні. Тому для цього тут краще підійде алгоритм який буде працювати по схемі черги, тобто перший зайшов - перший вийшов. Саме так працює метод обходу вершин обхід в ширину(BFT). Та на цьому методі і побудований алгоритм розфарбування графу двома кольорами в алгоритмі Вігдерсона, який є наведений нижче.

Алгоритм 2.8. Алгоритм B

Дано зв'язний граф G = (V, E), цей алгоритм знаходить розфарбування

графу двома кольорами, якщо такий існує, або виводить непарний цикл.
1: [Ініціалізація].Ініціалізуємо $f(v_1) = 1)$ і нехай Q(черга) буде пустою послідовністю. Для найближчого сусіда w вершини v_1, визначаємо $p(w) = v_1$(предок вершини w) та додаємо w в Q.
2: [Наступна вершина].Якщо Q пуста, то переходимо до кроку 3. Інакше видаляємо першу вершину v з Q і позначаємо $f(v)$ до кольору який ще не було зазначено в $p(v)$. Для кожного сусіда w вершини v, якщо вершина w не розфарбована і не є в послідовності Q, тоді зазначаємо що $p(w) = v$ і додаємо вершину w в кінець послідовності Q. Повторюємо крок 2.
3: [Перевірка на подвійне розфарбування].Проходимо через всі ребра перевіряючи чи $f(v) \neq f(w)$ для кожного ребра vw. Якщо виконується, то виводимо f, як відповідь.
4: [Утворюємо непарний цикл].Нехай vw буде ребром з кольором $f(v) = f(w)$ і нехай u буде найближчий предок вершини v та w. Тоді виводимо шлях w,$p(w)$,$p(p(w))$,...,u, і так само v,$p(v)$,$p(p(v))$,...,u, які з'єднані ребром vw.

Твердження 2.7. *([28]) Складність алгоритму 2.8 є $O(n + m)$, де n кількість вершин, а m кількість ребер.*

Тепер, щодо жадібного алгоритму, який застосований в алгоритмі Вігдерсона, можна сказати, що він працює наступним чином.Він проходить вершини одну за іншою, та задає вершинам перший доступний колір. Нехай буде існувати така функція $f(v)$, що буде приймати на вхід вершину графу, а вертати колір в який та вершина розфарбована. Тоді алгоритм розфарбування двома кольорами буде мати вигляд:

Алгоритм 2.9. Алгоритм G(жадібний алгоритм)
Дано граф G = (V, E) з максимальним степенем Δ та порядком v_1,v_2,...,v_n вершин, цей алгоритм знаходить розфарбування вершин графу з $max_i|\{j < v : v_j v_i \in E\}| + 1 \leqslant \Delta + 1$ кольорами.

1: [Ініціалізація].Ініціалізуємо $i = 0$.

2: [Наступна вершина].Збільшуємо i.Якщо $i = n + 1$, закінчуємо з f як з відповіддю.

3: [Знаходження $N(v_i)$].Вираховуємо набір $C = \cup_{j<i} f(v_j)$ кольорів які уже застосовані до сусідів v_i.

4: [Застосовуємо перший доступний колір до вершини v_i].Збільшуючи значення c = 1,2,..., перевіряти чи $c \in C$. якщо ні, то призначаємо $f(v_i) = c$ і вертаємось до кроку 2.

Твердження 2.8. *([28]) Складність алгоритму 2.9 є $O(n + m)$, де n кількість вершин, а m кількість ребер.*

Можна побачити, що кроки G3 та G4 робить в найгіршому випадку $O(1 + degv_i)$ операцій. Сумуючи це все по i отримуємо $n + (degv_1 + degv_2 + ... + degv_n) = n + 2m$. Таким чином алгоритм G виконується за $O(n + m)$ часу. Така складність часу є лише у випадку коли використовується список суміжності графу.

2.4 Квантова реалізація алгоритму жадібного розфарбування графу

В цьому підрозділі розповідається про квантову реалізацію алгоритму жадібного розфарбування графа побудований Джеком Моррісом та Фанг Сонгом[29]. Цей алгоритм жадібного розфарбування працює для графу представленого як матриця суміжності. Він розв'язує проблему розфарбування графу $O(1 + \epsilon)\Delta$ кольорами. При тому, як було доведено в статті, представлений ними алгоритм в класичній моделі складність алгоритму є $O(\epsilon^{-1/2} n\sqrt{n} \log n)$, а в квантовій моделі $O(\epsilon^{-1} n^{4/3})$

Для початку, щоб представити алгоритм, потрібно пояснити роботу деяких функцій. А саме, нехай існує такий запит до матриці суміжності $M[v,u]$, який повертає 1 коли існує ребро між вершинами v та u, інакше

вертає 0. А також нехай $\chi(c)$ буде визначати набір вершин з кольором c після того.

Алгоритм 2.10. Жадібний алгоритм розфарбування графу $(1+\epsilon)\Delta$ кольорами.

На вхід дається граф у вигляді матриці суміжності, та значення ϵ:

1: Для кожної вершини графу v виконується:

2: Допоки вершина v не розфарбована, виконується наступне:

3: Вибирається випадковий колір $c \in [(1+\epsilon)\Delta]$

4: Для кожної вершини $u \in \chi(c)$:

5: Виконується запит $M[u,v]$

6: Якщо $M[u,v] = 1$, то:

7: Повертаємося до кроку 3.

8: Задаємо вершині v колір c, а також додаємо цю вершину в $\chi(c)$.

9: Повертає список $\{\chi(1),...\chi((1+\epsilon)\Delta)\}$

Твердження 2.9. *([29]) Складність алгоритму 2.10 є* $O(\epsilon^{-1/2} n\sqrt{n}\log n)$*, де* n *кількість вершин.*

У квантовій реалізації жадібного алгоритму розфарбування графу використовується підалгоритм пошуку конфлікту, який приймає на вхід вершину v, а також множину вершин S, а вертає відповідь чи є вершина v сусідньою до вершини $s \in S$, такий алгоритм розписаний в статті[29]. Тоді квантовий алгоритм жадібного розфарбування буде виглядати наступним чином:

Алгоритм 2.11. Квантова реалізація жадібного алгоритму розфарбування графу $(1+\epsilon)\Delta$ кольорами.

На вхід дається граф у вигляді матриці суміжності, та значення ϵ:

1: $t \leftarrow 0$:

2: Допоки $t \leqslant T$ виконується наступне:

3: Для кожної вершини графу v, виконується наступне:

4: Вибирається випадковий колір $k \in [(1+\epsilon)\Delta]$

5: Якщо $|\chi(k)| > \frac{2n}{\epsilon\Delta}$

6: Повертаємося до кроку 4.

7: Інакше:

8: Якщо знаходиться конфлікт між вершиною v та $\chi(k)$, тоді:

9: $t \leftarrow t + 2\sqrt{\frac{n}{\epsilon\Delta}} \log n$

10: Повертаємося до кроку 4.

11: Інакше:

12: Задаємо вершині v колір c, а також додаємо цю вершину в $\chi(c)$.

13: Повертає список $\{\chi(1),...\chi((1+\epsilon)\Delta)\}$

Твердження 2.10. *([29]) Складність алгоритму 2.11 є* $O(\epsilon^{-1}n^{4/3})$, *де* n *кількість вершин.*

В цьому алгоритмі $T = 9\epsilon^{-3/2}\log^2 n\sqrt{\frac{n^2}{\Delta}}$. Це потрібно для того, щоб обмежити похибку алгоритму. Тому, що алгоритм пошуку конфлікту має похибку $\frac{1}{n^2}$. Тобто такий алгоритм вертає розфарбування з можливістю в найгіршому випадку $1-\Theta(n^{-2})$, для значно великого n похибка може бути покращена до $1-\Theta(n^{-k})$, де $k > 0$ константа.

Висновки до розділу 2

У цьому розділі розглянуто основні теоретичні відомості квантової моделі, а також методи та підходи реалізацій квантових алгоритмів для задач розфарбування графа, а саме: квантовий алгоритм для розфарбування двома кольорами графа представлений як список суміжностей, класичний алгоритм жадібного розфарбування, квантовий алгоритм жадібного розфарбування, класичний алгоритм Вігдерсона для графу який представлений списком суміжностей. Ці методи та підходи

разом з алгоритмами будуть використовуватись для побудування квантового алгоритму Вігдерсона, а також квантового алгоритму для розв'язання задачі розфарбування двома кольорами графу, представленого матрицею суміжностей.

3 ПОБУДОВА КВАНТОВИХ АЛГОРИТМІВ

В цьому розділі будується квантова реалізація алгоритму розфарбування графу двома кольорами представленому як матриця суміжності, який працює завдяки методу обходу в ширину. А також будується квантовий алгоритм Вігдерсона також для графа представленого як матриця суміжності.

3.1 Розфарбування графу двома кольорами представленому як матриця суміжності

В цьому підрозділі реалізовано алгоритм розфарбування двома кольорами графу представленому як матриця суміжності з використанням методу обходу в ширину. Для початку реалізовано класичний алгоритм з всіма підалгоритмами послідовно, а потім уже сам квантовий алгоритм.

Для того, щоб зробити класичний алгоритм потрібно спочатку побудувати алгоритм обходу в ширину(BFT), який працює по схемі черги, тобто перший зайшов - перший вийшов і цей алгоритм повинен працювати для матриці суміжності. Сам алгоритм обходу графу в ширину працює наступним чином:

Алгоритм 3.1. Каркас алгоритму BFT

На вхід дається вершина v:

1: Позначаємо вершину v як відвідану, та додаємо її в чергу Q.

2: Допоки $|Q| > 0$ виконується наступне:

3: Дістається вершина $u \in Q$.

4: Для кожного сусіда x вершини u виконується наступне:

5: Якщо вершина x не відвідана, то:

6: Позначаємо вершину x як відвідану і додаємо її в чергу

Q.

Тепер коли є алгоритм обходу вершин в ширину потрібно зробити такий алгоритм для матриці суміжності.

Алгоритм 3.2. Алгоритм BFT для графа представленого як матриця суміжності

На вхід дається матриця суміжності графу G:

1:Вибираємо випадкову вершину v графу, позначаємо її як відвідану та додаємо її в чергу Q.

2:Допоки $|Q| > 0$ виконується наступне:

3: Дістається вершина $u \in Q$.

4: Для кожної вершини графу $x \in G$, крім вершини v виконується:

5: Якщо $M[x,v] = 1$, а також вершина x не відвідана, то:

6: Позначаємо вершину x як відвідану і додаємо її в чергу Q.

Твердження 3.1. *Складність алгоритму 3.2 дорівнює* $O(n(n-1))$, *де* n *кількість вершин.*

Доведення. Можна побачити, що кожна вершина обходиться 1 раз, а також для кожної вершини перевіряються всі інші на суміжність. Тобто можна побачити, що складність такого алгоритму буде $O(n(n-1))$. □

Тепер можна побудувати алгоритм розфарбування графу двома кольорами представленому як матриці суміжності. Алгоритм буде мати вигляд:

Алгоритм 3.3. Алгоритм для розфарбування графа двома кольорами на принципі алгоритму BFT

Дано неорієнтований граф G (працюємо з графом як з матрицею):

0: Створюємо глобальну зміну і називаємо її L. Ініціалізуємо її значенням '1'.

1: Обираємо випадкову вершину $v \in V$ і присвоюємо їй значення L, а

також додаємо в чергу Q.

2: Допоки $|Q| > 0$ виконується наступне:

3: Дістається вершина $u \in Q$.

4: Для кожної вершини графу $x \in G$, крім вершини v виконується:

5: Якщо $G(x,u) = 1$, а також не була ще позначена, то:

6: Позначаємо u як $L + 1(mod 2)$.

7: Позначаємо L як $L + 1(mod 2)$

8: Інакше, якщо $G(x,u) = 1$, то:

9: Перевіряємо значення вершини x. Якщо це така сама вершина як і u, то зупиняємо алгоритм і говоримо, що ця задача не є виконувана для такого графу.

Твердження 3.2. *Складність алгоритму 3.3 дорівнює* $O(n(n-1))$, *де* n *кількість вершин.*

Доведення. Такий алгоритм має таку саму складність як і просто алгоритм обходу BFT, оскільки вершини обробляються один раз, а потім перевіряються на суміжність, тобто складність буде $O(n(n-1))$. □

Реалізація квантового алгоритму для задачі розфарбування графу який представлений як матриця суміжності буде виконуватись по тому же принципу, якій був показаний в попередньому розділі для списку суміжності. Тобто вершини будуть зображуватись як кубіти, а кольори як стани цих кубіт. Спочатку буде генеруватись суперпозиція всіх розфарбувань, а потім обходячи вершини будуть видалятись непотрібні стани, а також перевірятись на можливість такого розфарбування.

Генеруватись суперпозиція буде завдяки Адамаровому оператору, наступним чином:

$$H^{\oplus n}|0\rangle^{\oplus n} = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle$$

Для випадку видалення непотрібних станів буде використовуватись оператор E_{ab} коли вершини є сусідніми, але одна з них не є ще відвіданою.

Такий оператор для двох вершин буде виглядати наступним чином:

$$E_{ab} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{3.1}$$

А також буде робити такі операції:

$|00\rangle_{ab} \to |01\rangle_{ab} + |00\rangle_{ab}$

$|01\rangle_{ab} \to |01\rangle_{ab} - |00\rangle_{ab}$

$|10\rangle_{ab} \to |10\rangle_{ab} + |11\rangle_{ab}$

$|11\rangle_{ab} \to |10\rangle_{ab} - |11\rangle_{ab}$

Для випадку, коли дві вершини є сусідніми, але також дві є відвідані, буде виконуватись вимірювання цих вершин, щоби перевірити чи є такі вершини одного кольору. Для цього буде використовуватись вимірювання M_{ab}, яке складається з двох операторів O_{ij} та з $I - O_{ij}$, де

$O_{ij} = |01\rangle_{ij}\langle 01| + |10\rangle_{ij}\langle 10|$

$I - O_{ij} = |00\rangle_{ij}\langle 00| + |11\rangle_{ij}\langle 11|$

В цьому випадку O_{ij} буде відповідати нулю вимірювання M_{ab}, а $I - O_{ij}$ одиниці. Тобто отримавши одиницю з цього вимірювання можна сказати, що такий граф не можна розфарбувати двома кольорами.

Тепер щодо самого алгоритму розфарбування графу двома кольорами представленого як матриця суміжності у порядку обходу BFT. Такий алгоритм буде мати вигляд:

Алгоритм 3.4. Квантовий алгоритм для розфарбування двома кольорами графу представленого як матриця суміжності у порядку обходу BFT

Дано неорієнтований, суміжний граф G представлений як матриця суміжності:

1: Генеруємо суперпозицію всіх можливих розфарбувань.

2: Вибираємо випадкову вершину $v \in G$ позначаємо її відвіданою, а також додаємо її в чергу Q.

3: Допоки $|Q| > 0$:

4: Витягуємо вершину $a \in Q$:

5: Перебираємо всі вершини $b \in G$,$b \neq a$:

6: Якщо $G[a,b] = 1$, а також вершина b не була ще позначена, то:

7: Виконуємо E_{ab}.

8: Позначаємо вершину b та додаємо її в чергу Q.

9: Інакше, якщо $G[a,b] = 1$:

10: Виконуємо M_{ab}.

11: Якщо $M_{ab} = 1$ то зупиняємося і вертаємо що "Не існує розфарбування двома кольорами для такого графу"

12: Обраховуємо кінцеву суперпозицію для отримання випадкового розфарбованого двома кольорами графу, або використовуємо суперпозицію всіх розфарбувань графу G для подальших процесів.

Твердження 3.3. *Складність алгоритму 3.4 дорівнює* $O(n(n-1))$, *де n кількість вершин.*

Доведення. Такий алгоритм має таку саму складність як і просто алгоритм обходу BFT, оскільки вершини обробляються один раз, а потім перевіряються на суміжність, тобто складність буде $O(n(n-1))$. □

Ефективність такого алгоритму є в тому, щоб використовувати такий алгоритм в подальших квантових алгоритмах, як підалгоритм. Діло в тому, що такий алгоритм дає суперпозицію всіх можливих розфарбувань, що може бути використана для подальшої обробки. Також використання в цьому алгоритмі матриці суміжності, може навіть покращити виконання цього алгоритму, в зрівнянні з класичним, це можна зробити, завдяки деяким модифікаціям алгоритму пошуку Гровера, для знаходження таких вершин, що $G[a,b] = 1$. Тому можна сказати, що цей алгоритм навіть можна покращити, отримавши при

цьому додаткову ефективність.

Твердження 3.4. *Алгоритм 3.4 – є коректним.*

Доведення. Для того, щоб довести коректність алгоритму потрбіно пройти по деяким крокам алгоритму, тим самим показавши, що немає такого випадку коли алгоритм буде працювати неправильно. Перший крок алгоритму генерує суперпозицію всіх можливих розфарбувань, наприклад для випадку з двома вершинами суперпозиція буде виглядати наступним чином: $|00\rangle + |01\rangle + |10\rangle + |11\rangle$ з кофіцієнтом нормалізації $1/2$. Зразу можна побачити, що наприклад стани $|00\rangle$ $|11\rangle$ не можуть існувати в цій суперпозиції задля коректного розфарбування. Для цього і працює оператор E_{ab}, він позбувається таких станів.

Тепер щодо обходу графу, так як граф є суміжним, то можна побачити, що кожна вершина графу буде додана до черги Q на 2, а також на 8 кроці. Тим самим кожна вершина графу буде оброблена, оскільки цикл на 3 кроці виконується допоки $|Q| > 0$.

Тепер щодо обробки вершин. Очевидно, що перевірки на умови, які виконуються на 6 та на 9 кроці, покривають всі потрібні нам варіанти суміжності вершин. А саме випадок, коли дві вершини є суміжними і одна з них не є позначеною, а також випадок, коли дві вершини є суміжними і двоє є позначеними. В першому випадку виконується оператор E_{ab}, є зрозумілим тим, що він виконається для кожної пари суміжних вершин рівно один раз. А в другому випадку виконується оператор M_{ab}, який оброблює вершини які утворюють цикл, тим самим перевіряючи на можливе розфарбування.

Таким чином, алгоритм або виконає розфарбування графу та верне суперпозицію на 12 кроці. Або зупиниться на 11 і скаже, що такого графу не існує. □

3.2 Квантова реалізація алгоритму Вігдерсона

В цьому підрозділі будується алгоритм Вігдерсона в квантовій моделі обчислення, коли граф представлений матрицею суміжності. Для цього спочатку будується класичний алгоритм Вігдерсона з матрицею суміжності, а потім перетворений в квантовий алгоритм.

Через те, що алгоритм розфарбування двома кольорами графу представленому як матриця суміжності уже представлений, то саме він буде використовуватись в алгоритмі Вігдерсона. А щодо жадібного алгоритму, то можливо переробити алгоритм який напряму даний до алгоритму Вігдерсона, а можливо використати кращий алгоритм який був наданий Джеком Моррісоном та Фанг Сонгом[29]. Але в випадку кращого алгоритму там є деякі обмеження щодо ефективності, які в класичній моделі є доволі неприємними для алгоритму Вігдерсона. Тому все-таки буде показано перетворення жадібного алгоритму, який використовувався в алгоритмі Вігдерсона для списку суміжностей. Алгоритм буде мати вигляд:

Алгоритм 3.5. Жадібне розфарбування графу представленого як матриця суміжності

Дано неорієнтований граф G представлений як матриця суміжності:

1: Ініціалізуємо $i = 1$.

2: Поки не $i = n + 1$, виконуємо наступне:

3: Ініціалізується C = {}.

4: Для кожної вершини $u_j \neq u_i$ виконується:

5: Якщо $G[u_i,u_j] = 1$, то:

6: C = C $\cup f(u_j)$.

7: Збільшуючи значення $c = 1.2...$, виконується:

8: Якщо $c \notin$C, то:

9: $f(u_i) = c$.

10: $i = i + 1$.

Твердження 3.5. *Складність алгоритму 3.5 дорівнює* $O(2n(n-1))$, *де* n *кількість вершин.*

Доведення. Можна побачити, що цикл утворений на 4 кроці робить $n-1$ кроків, а також цикл утворений на 7 кроці в найгіршому випадку утворює також $n-1$ кроків. Тобто разом вони утворюють $2(n-1)$ кроків, та сумуючи їх по i бачимо, що алгоритм робить $2n(n-1)$ кроків. Тобто складність алгоритму буде $O(n^2)$. □

Тепер можна утворити алгоритм Вігдерсона, для матриці суміжності.

Алгоритм 3.6. Алгоритм Вігдерсона для матриці суміжності

Дано неорієнтований граф G представлений як матриця суміжності:

1: Нехай $c = 1$.

2: Для кожної вершини $v_i \in G$, виконуємо наступне:

3: Ініціалізується $d = 0$.

4: Ініціалізується $m = \{\}$.(матриця суміжності сусідів вершини u_i)

5: Для кожної вершини $u_j \neq u_i$ виконується:

6: Якщо $G[u_i,u_j] = 1$, то:

7: $d = d + 1$.

8: Додаємо вершину u_j в матрицю, а також додаємо зв'язки з присутніми уже вершинами в матриці m.

9: Якщо $d \geqslant \lceil\sqrt{n}\rceil$, то:

10: Виконується розфарбування m кольорами c, $c+1$.

11: $c = c + 1$.

12: з множини G видаляються вершини, які є в m.

13: Для вершин які ще не розфарбовані виконуємо жадібний алгоритм розфарбування.

Твердження 3.6. *Складність алгоритму 3.6 дорівнює* $O(3n(n-1))$, *де* n *кількість вершин.*

Доведення. Можна побачити, що найгірший випадок тут буде,

якщо не знайдеться такої вершини для якої $degv \geqslant \lceil\sqrt{n}\rceil$. Саме тому, що якщо найдеться, то на 13 кроці виконується видалення вершин і тоді зменшується обхід вершин на 2 кроці, а також в жадібному алгоритмі зменшується кроків. Тому найгірший випадок, коли виконається цикл на 2 кроці, який займає $n(n-1)$ кроків, а потім ще виконається алгоритм на 13 кроці який також буде займати $2n(n-1)$ кроків. Тому можна сказати, що складність алгоритму буде $O(3n(n-1))$ □

Квантова реалізація алгоритму буде будуватись, так само як і класичний алгоритм, але з використанням квантових підалгоритмів, саме квантового розфарбування графу двома кольорами для випадку коли граф має вигляд матриці суміжностей, тобто алгоритм 3.4. А також алгоритм жадібного розфарбування алгоритм 2.11. Тоді квантовий алгоритм Вігдерсона буде виглядати:

Алгоритм 3.7. Алгоритм Вігдерсона для матриці суміжності

Дано неорієнтований граф G представлений як матриця суміжності:

1: Нехай $c = 1$.

2: Для кожної вершини $v_i \in G$, виконуємо наступне:

3: Ініціалізується $d = 0$.

4: Ініціалізується $m = \{\}$.(матриця суміжності сусідів вершини u_i)

5: Для кожної вершини $u_j \neq u_i$ виконується:

6: Якщо $G[u_i,u_j] = 1$, то:

7: $d = d + 1$.

8: Додаємо вершину u_j в матрицю, а також додаємо зв'язки з присутніми уже вершинами в матриці m.

9: Якщо $d \geqslant \lceil\sqrt{n}\rceil$, то:

10: Виконується розфарбування m кольорами c, $c + 1$, завдяки алгоритму 3.4.

11: $c = c + 1$.

12: з множини G видаляються вершини, які є в m.

13: Для вершин які ще не розфарбовані виконуємо жадібний алгоритм

розфарбування 2.11 з вхідним параметром ϵ.

Значення ϵ повинен бути більший нуля, та не дорівнювати $\frac{1}{\Delta}$. Це все через те, що в алгоритмі 2.11 не розв'язано задачі розфарбування $\Delta + 1$ кольорами, а розв'язано $(1 + \epsilon)\Delta$. Сам ϵ вираховується наступним чином

$$\epsilon = \frac{\lceil\sqrt{n}\rceil+1}{\Delta} - 1$$

Можна побачити, що так, як $\epsilon > 0$, то повинна бути $\Delta \leqslant \lceil\sqrt{n}\rceil$. Таким чином можна сказати, що чим менше ϵ, тим більше Δ можливе.

Твердження 3.7. *Складність алгоритму 3.7 буде дорівнювати* $O(n(n-1) + \epsilon^{-1}n^{4/3})$.

Доведення. Використовувавши той найгірший випадок, що використовувався у доведені складності для класичної реалізації алгоритму Вігдерсона 3.6. Тобто випадок коли в алгоритмі Вігдерсона використовується лише жадібний алгоритм, можна побачити, що цикл на 3 кроці буде утворювати $n(n-1)$ кроків, а складність жадібного алгоритму буде $O(\epsilon^{-1}n^{4/3})$. Тоді складність квантового алгоритму Вігдерсона буде $O(n(n-1) + \epsilon^{-1}n^{4/3})$. □

Твердження 3.8. *Алгоритм 3.7 є коректним.*

Доведення. Коректність алгоритму справедлива лише в тому випадку, якщо коректний класичний алгоритм Вігдерсона[28]. Побудований квантовий алгоритм полягається на коректність класичного алгоритма через те, що там використовується такий самий обхід як і в класичному алгоритмі. Тобто спочатку шукається вершина a для якої $d \geqslant \lceil\sqrt{n}\rceil$, де d – степінь вершини a. А потім всі сусіди вершини a розфарбовуються двома наданими кольорами алгоритмом 3.4. Після того як всі вершини для яких $d \geqslant \lceil\sqrt{n}\rceil$ були оброблені виконується жадібний алгоритм 2.11, коректність якого доведена[29]. □

3.3 Аналіз побудованих квантових алгоритмів та порівняння з класичними аналогами

В цьому підрозділі дається порівняння утворених квантових алгоритмів з наявними класичними аналогами. Для початку буде розглянуто алгоритм розфарбування двома кольорами графу, а потім алгоритм Вігдерсона.

Алгоритмів розфарбування двома кольорами графів є велика кількість, кожна з яких є ефективна для деяких окремих випадків. Наприклад алгоритм розфарбування двома кольорами графу представленого як список суміжностей ефективно використовувати для випадків коли граф є розрідженим. Побудований квантовий алгоритм 3.4 є ефективний у випадку коли граф є щільним, тобто коли граф має велику кількість ребер близьку до максимальної. Крім того, використаний обхід графу BFT у квантовому алгоритмі 3.4, дозволяє цей алгоритм розпаралелювати в тих випадках, коли, наприклад потрібно обробляти лише сусідні вершини графу, не йдучи в глибину графу.

Тепер щодо класичних аналогів. В класичній моделі можливо побудувати схожий алгоритм, який був показаний як алгоритм 3.3. Через те, що там використовуються однакові методи обходу графу, то по складності квантовий та класичні алгоритми зовсім не будуть відрізнятись, тобто складність як класичного, так і квантового алгоритму буде дорівнювати $O(n(n-1))$. Дивлячись на це можна подумати, що тоді не має переваг у використані саме квантової моделі, але це не так. Основні переваги квантової моделі з'являються у вигляді вихідних даних алгоритму. Діло в тому, що працюючи з суперпозицією в квантовій моделі, алгоритм працює з усіма можливими розфарбуваннями. Тобто за ту саму кількість кроків, що робить класичний алгоритм, квантовий може подати на вихід не одне розфарбування графу, а всім можливі у вигляді суперпозиції, коли класичний видає лише одне. Більш того, це

може бути незвичайно ефективним для використання цього квантового алгоритму, як алгоритму для інших квантових алгоритмів.

Також можна побачити по першому розділу, що використання матриці суміжності, як представлення графу, є набагато ефективнішим у квантових алгоритмах порівняно з класичними аналогами. Це все полягає в тому, що в класичній моделі пошук ребра між вершинами виконується простим перебором всіх вершин, коли у квантовій моделі пошук може бути покращеним завдяки модифікаціям пошуку Гровера. Алгоритм розфарбування двома кольорами графу представленого як матриця суміжності у порядку обходу BFT не обмеженнями, його також можна покращити таким способом, що збільшить його ефективність порівняно з класичним аналогом в рази.

Алгоритм Вігдерсона не має такої великої кількості різних аналогів. Було знайдено лише алгоритм 2.7 який виконується дла графу представленого як список суміжності. Тому для утворення квантового алгоритму Вігдерсона для матриці суміжності спочатку знадобилось побудувати класичний алгоритм Вігдерсона для матриці суміжностей. В класичному алгоритмі використовувались стандартні алгоритми для жадібного розфарбування, а також для розфарбування двома кольорами графу, тому складність класичного алгоритму Вігдерсона для матриці суміжностей згідн з твердженням 3.6 дорівнює $O(3n(n-1))$. Щодо побудованого квантового алгоритму, то можна побачити, що у нього менша складність, а саме $O(n(n-1)+\epsilon^{-1}n^{4/3})$. Можна тепер побачити складність квантового алгоритму буде менша від класичного на $2n(n-1)-\epsilon^{-1}n^{4/3}$ разів. Крім того, квантовий алгоритм Вігдерсона також будується для графа представленого як матриця суміжностей, а це значить, що він може бути покращеним модифікаціями пошуку Гровера.

Висновки до розділу 3

Як результат, вдалося побудувати квантовий алгоритм Вігдерсона, а також квантовий алгоритм який розв'язує задачу розфарбування графу двома кольорами, де граф представлений як матриця суміжності. От щодо квантового алгоритму розфарбування двома кольорами, можна сказати що отриманий алгоритм і так має переваги не дивлячись на те що час виконання є такою самою як і класичний аналог, переваги полягають як раз в тому, що в кінці буде отримано всі можливі розфарбування графу і це можливо використовувати в інших квантових алгоритмах, де використовується розфарбування двома кольорами графу. Більш того, оскільки він працює з матрицею суміжностей, він доволі ефективний в випадках коли граф має велику кількість ребер. Також, через те що там використовується матриця суміжностей, складність алгоритму можна покращити, використовуючи деякі модифікації прискорення Горнера для пошуку сусідів вершини, таке покращення може бути дуже ефективним.

Щодо квантового алгоритму Вігдерсона, то там також можна покращити виконання алгоритму і не тільки через те що покращується сам алгоритм розфарбування двома кольорами графу, а також можливо покращити алгоритм жадібного розфарбування, так щоби алгоритм розфарбовував не $(1+\epsilon)\Delta$ кольорів, а $\Delta+1$ кольорів.

ВИСНОВКИ

Як результат оглядової частини було розглянуто різноманітні квантові реалізації алгоритмів, які розв'язували задачі на графах. Всі розібрані задачі вирішенні завдяки різним властивостям суперпозицій станів, що показує можливості квантової моделі. Також можна побачити, що задачі на графах, які представлені як матриця суміжностей, можуть бути розв'язані ефективним чином.

У другому розділі було розглянуто деякі особливі реалізації, які були використані для побудови квантових алгоритмів. Такі реалізації дали чітке пояснення як працювати у квантовій моделі, щоб побудувати квантовий алгоритм. Саме ті методи і підходи, які були розглянуті, використовувались для побудови квантового алгоритму розфарбування графу двома кольорами, коли граф представляється як матриця суміжностей. А також був розібраний квантовий алгоритм жадібного розфарбування, який був використаний для побудови алгоритму Вігдерсона.

Як результат, виконаної вище роботи, було отримано:

1) Побудовано квантовий алгоритм Вігдерсона для графа, представленого як матриця суміжності.

2) Побудовано квантовий алгоритм розфарбування двома кольорами графу, який представлений як матриця суміжності.

3) Проведений аналіз побудованих алгоритмів та їх порівняння з класичними аналогами.

Щодо отриманих результатів, то побудований квантовий алгоритм розфарбування двома кольорами, складність якого дорівнює $O(n(n-1))$, є доволі цікавим. Він є цікавим тим, що у квантовій моделі використання графу, як матриці суміжностей є ефективним. Побудовано багато алгоритмів у квантовій моделі саме для такого випадку, тому що у квантовій моделі для матриці суміжностей можна використовувати

алгоритм пошуку Гровера, який покращує результат в зрівнянні з класичними аналогами. Побудований алгоритм розфарбування двома кольорами також є коректним.

Побудований алгоритм Вігдерсона також є доволі ефективним, складність побудованого алгоритму дорівнює $O(n(n-1)+\epsilon^{-1}n^{4/3})$, коли у класичного алгоритму складність дорівнює $O(3n(n-1))$. Таким чином побудований алгоритм вийшов ефективнішим на $O(n(n-1)+\epsilon^{-1}n^{4/3})$ разів.

Також можна покращити алгоритм розфарбування двома кольорами завдяки деяким модифікаціям алгоритму Гровера. Щодо квантового алгоритму Вігдерсона, то там також можна покращити виконання алгоритму, і не тільки через те, що покращується сам алгоритм розфарбування двома кольорами графу, а також можливо покращити алгоритм жадібного розфарбування, так щоби алгоритм розфарбовував не $(1+\epsilon)\Delta$ кольорів, а $\Delta+1$ кольорів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Quantum Algorithms for Graph Connectivity and Formula Evaluation [Електроний ресурс] / S. Jeffery, S. Kimmel. — 2019. — Режим доступу: https://arxiv.org/pdf/1704.00765.pdf

2. A. Belovs, B. W. Reichardt. *Span programs and quantum algorithms for stconnectivity and claw detection*. 2012.

3. Time and space efficient quantum algorithms for detecting cycles and testing bipartiteness. [Електроний ресурс] / C. Cade, A. Montanaro, A. Belovs. — 2016. — Режим доступу: https://arxiv.org/pdf/1610.00581.pdf

4. Quantum algorithms for learning a hidden graph and beyond [Електроний ресурс] / A. Montanaro, C Shao. — 2021. — Режим доступу: https://arxiv.org/pdf/2011.08611.pdf

5. Diestel, Reinard. *Graph Theory, Graduate Texts in Mathematics, Springer*. 2005.

6. Holyer, I. *The NP-completeness of edge-coloring SIAM Journal on Computing*. 1981.

7. Quantum Algorithms for Matchings and Vertex-Colorings in Graphs [Електроний ресурс] / Nabiha Asghar. Режим доступу: https://nabihach.github.io/co681.pdf

8. Exponential-time Quantum Algorithms for Graph Coloring Problems [Електроний ресурс] /K. Shimizu, R. Mori. — 2019. — Режим доступу: https://arxiv.org/abs/1907.00529

9. A Query-Efficient Quantum Algorithm for Maximum Matching on General Graphs [Електроний ресурс] / S. Kimmel, R. Teal Witter. — 2021. — Режим доступу: https://arxiv.org/pdf/2010.02324.pdf

10. The Weighted Matching Approach to Maximum Cardinality Matching [Електроний ресурс] / Gabow, H.N. — 2017. — Режим доступу: https://arxiv.org/pdf/1703.03998.pdf

11. Quantum Speedup Based on Classical Decision Trees [Електроний

ресурс] / Beigi, S., Taghavi, L. — 2020. — Режим доступу: https://quantum-journal.org/papers/q-2020-03-02-241/pdf/

12. Quantum Algorithms for Graph Traversals and Related Problems [Електроний ресурс] / S. Doern. — 2007. — Режим доступу: https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.190/Mitarbeiter/doern/GT.pdf

13. Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience [Електроний ресурс] / K.Srinivasan, S.Satyajit, Bikash K. Behera, Prasanta K. Panigrah. — 2018. — Режим доступу: https://arxiv.org/pdf/1805.10928.pdf

14. A Quantum Algorithm for the Hamiltonian NAND Tree [Електроний ресурс] / E. Farhi, J. Goldstone, S. Gutmann. — 2007. — Режим доступу: https://arxiv.org/pdf/quant-ph/0702144.pdf

15. Lower Bounds on Quantum Query Complexity [Електроний ресурс] / P. Hoyer, R. Spalek. — 2005. — Режим доступу: https://arxiv.org/pdf/quant-ph/0509153.pdf

16. An Analog Analogue of a Digital Quantum Computation [Електроний ресурс] / E. Farhi and S. Gutmann. — 1998. — Режим доступу: https://arxiv.org/pdf/quant-ph/9612026.pdf

17. Hamiltonian Oracles [Електроний ресурс] / Carlos Mochon. — 2007. — Режим доступу: https://arxiv.org/pdf/quant-ph/0602032.pdf

18. A quantum query algorithm for the graph collision problem [Електроний ресурс] / D.Gavinsky and T. Ito. — 2012. — Режим доступу: https://arxiv.org/pdf/1204.1527.pdf

19. Quantum walk-based search algorithms with multiple marked vertices [Електроний ресурс] / G. A. Bezerra, P. H. G. Lugao, R. Portugal. — 2021. — Режим доступу: https://arxiv.org/pdf/2103.12878.pdf

20. M. Mohseni, P. Rebentrost, S. Lloyd, A. Aspuru-Guzik. *Environment-assisted quantum walks in photosynthetic energy transfer*. 2008.

21. S. E. Venegas-Andraca. *Quantum walks: a comprehensive review. Quantum Inf. Process*. 2012.

22. A. Ambainis. *Quantum walk algorithm for element distinctness.* 2007.

23. M. Szegedy. *Quantum speed-up of Markov chain based algorithms.* 2004.

24. Quantum walk search algorithms and effective resistance [Електроний ресурс] / Stephen Piddock. — 2019. — Режим доступу: https://arxiv.org/pdf/1912.04196.pdf

25. Quantum walks and electric networks [Електроний ресурс] / A. Belovs. — 2013. — Режим доступу: https://arxiv.org/pdf/1302.3143.pdf

26. Graph Cut Segmentation Methods Revisited with a Quantum Algorithm [Електроний ресурс] / Lisa Tse, Peter Mountney, Paul Klein, Simone Severini. — 2019. — Режим доступу: https://arxiv.org/pdf/1812.03050.pdf

27. Dirac, P. A. M. *A new notation for quantum mechanics* 1939.

28. Graph colouring algorithms [Електроний ресурс] / Thore Husfeldt. — 2015. — Режим доступу: https://arxiv.org/pdf/1505.05825.pdf

29. Simple vertex coloring algorithms [Електроний ресурс] / Jackson Morris and Fang Song. — 2021. — Режим доступу: https://arxiv.org/pdf/2102.07089.pdf