МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра програмування
(повна назва кафедри)

Магістерська робота

ПОРІВНЯННЯ АЛГОРИТМІВ ФАКТОРИЗАЦІЇ ДЛЯ КВАНТОВОГО ТА КЛАСИЧНОГО КОМП'ЮТЕРІВ

Виконала: студентка групи ПМіМ-21
спеціальності
122 – «Комп`ютерні науки»
(шифр і назва спеціальності)

(підпис) Лучинська Т.С. (прізвище та ініціали)

Керівник (підпис) Рикалюк Р.Є. (прізвище та ініціали)

Рецензент (підпис) Яцук Ю.О. (прізвище та ініціали)

Львів – 2022

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра програмування

Спеціальність 122 Комп'ютерні науки

«ЗАТВЕРДЖУЮ»
Завідувач кафедри Ярошко С.А.
" 13 " 09 2022 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Лучинській Тетяні Степанівні
(прізвище, ім'я, по батькові)

1. Тема роботи «Порівняння алгоритмів факторизації для квантового та класичного комп'ютерів»

керівник роботи Рикалюк Роман Євстахович, доцент, к. ф.-м. н.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені Вченою радою факультету від " 13 " вересня 2022 року № 15 .

2. Строк подання студентом роботи 12 грудня 2022 року

3. Вихідні дані до роботи літературні джерела, інтернет-ресурси, постановка задачі, наукові статті

4. Зміст дипломної роботи (перелік питань, які потрібно розробити)

1. Проаналізувати особливості квантового комп'ютера;
2. Скласти специфікацію вимог до програми та обрати технології розробки;
3. Дослідити класичні та квантові алгоритми факторизації;
4. Розробити та розгорнути застосунок для факторизації чисел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. UML – діаграма діяльності

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 5 вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення з предметною областю	05.09.2022 – 09.09.2022	
2	Аналіз сайтів-аналогів	09.09.2022 – 13.09.2022	
3	Написання специфікації вимог	13.09.2022 – 15.09.2022	
4	Вибір технологій	16.09.2022 – 18.09.2022	
5	Проектування архітектури ПЗ	19.09.2022 – 20.09.2022	
6	Розробка алгоритму Шора	21.09.2022 – 01.10.2022	
7	Розробка алгоритму з решетом Ератосфена	01.10.2022 – 07.10.2022	
8	Розробка ро-алгоритму Полларда	08.10.2022 – 15.10.2022	
9	Розробка веб-інтерфейсу	15.10-2022 – 25.10.2022	
10	Тестування програми	26.10.2022 – 27.10.2022	
11	Розгортання системи	28.10.2022 – 03.11.2022	
12	Оформлення магістерської роботи	04.11.2022 – 03.12.2022	
13	Подання магістерської роботи	12.12.2022	

Студент _____ Лучинська Т.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Рикалюк Р.Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Одним з цікавих напрямків розвитку науково-технічного потенціалу України є розв’язання задачі факторизації, тобто розкладання числа на множники. Існують очікування, що з використанням квантових технологій обчислення виконуватимуться за поліноміальний час. **Актуальність** дослідження алгоритмів факторизації для квантового та класичного комп’ютерів в тому, що це ставить під питання надійність найвідомішого алгоритму шифрування RSA.

Розроблено програмне забезпечення, яке надає такі можливості:

1. завантаження попередніх результатів з файлу;
2. розкладання числа на множники з використанням квантового алгоритму Шора;
3. розкладання числа на множники за допомогою класичного алгоритму з решетом Ератосфена;
4. розкладання числа на множники з використанням класичного ро-алгоритму Полларда;
5. графічне подання результатів;
6. табличне подання результатів;
7. збереження результатів у файл з власним розширенням «.qw_lnu».

Для розробки було використано такі технології: React, Redux, Python, QisKit. Для розгортання використано платформу Heroku.

У першому розділі магістерської роботи наведено особливості квантового комп’ютера та квантових технологій.

У другому розділі описано специфікацію вимог до розробленого веб-застосунку та обрані технології розробки.

У третьому розділі більш детально визначено проблему та алгоритми факторизації.

Четвертий розділ безпосередньо описує розроблений веб-застосунок з прикладами роботи.

Загальний обсяг роботи – 43 сторінки.

ЗМІСТ

ВСТУП 7

РОЗДІЛ 1. ОСОБЛИВОСТІ КВАНТОВОГО КОМП'ЮТЕРА 9

1.1 Історія квантових обчислень 9

1.2 Огляд квантових обчислень 10

1.3 Огляд фізичних реалізацій квантового комп'ютера 12

1.4 Порівняння квантового та класичного комп'ютерів 14

РОЗДІЛ 2. ПОСТАНОВКА ЗАВДАННЯ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО ВЕБ-ЗАСТОСУНКУ ДЛЯ ФАКТОРИЗАЦІЇ ЧИСЕЛ 15

2.1 Постановка завдання 15

2.2 Специфікація вимог до системи 16

2.3 Опис обраних технологій і засобів розробки 20

2.3.1 Серверна частина веб-застосунку 20

2.3.2 Клієнтська частина веб-застосунку 21

РОЗДІЛ 3. ПРОБЛЕМА ФАКТОРИЗАЦІЇ ТА ОГЛЯД АЛГОРИТМІВ 23

3.1 Проблема факторизації 23

3.2 Квантовий алгоритм Шора 24

3.2.1 Історія розвитку алгоритму 24

3.2.2 Огляд алгоритму 24

3.3 Аналіз класичних алгоритмів факторизації 27

3.3.1 Короткий огляд класичних алгоритмів 27

3.3.2 Алгоритм з використанням решета Ератосфена 27

3.3.3 Ро-алгоритм Полларда 28

РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ КВАНТОВИХ ТА КЛАСИЧНИХ АЛГОРИТМІВ 30

4.1 Проектування архітектури системи 30
4.2 Розгортання проекту 30
4.3 Опис роботи програми 32
4.4 Аналіз отриманих результатів 34
ВИСНОВКИ 38
СПИСОК ЛІТЕРАТУРИ 39
ДОДАТКИ 42

ВСТУП

Сьогодні, в час війни, особливо необхідно розвивати науково-технічний потенціал України, зокрема в напрямках, що пов'язані з криптографією. Одним з них є розв'язання задачі факторизації за поліноміальний час. Саме такий алгоритм було представлено Пітером Шором в 1994 році. Це завдання все ще залишається справою майбутнього, оскільки досі немає ефективної реалізації квантового комп'ютера. Проте дослідження алгоритмів факторизації для квантового та класичного комп'ютерів є **актуальними**, адже ставлять під питання надійність найвідомішого алгоритму шифрування RSA. В 2022 році Нобелівську премію з фізики отримали вчені, які досліджували природу квантової заплутаності, і очікується, що результати їх експериментів розпочнуть нову еру квантових технологій.

Перед початком написання магістерської роботи було поставлено таку **мету:** порівняти класичні та квантові алгоритми для вирішення задачі факторизації цілих чисел, дослідити особливості квантового комп'ютера, систематизувати матеріали, статті та наукові дослідження про вже існуючі алгоритми, розробити програмне забезпечення, що дозволить користувачам обирати метод для розкладання заданого числа на множники.

Для досягнення цієї мети необхідно виконати такі **основні завдання**:

1. проаналізувати відмінності між квантовим та класичним комп'ютерами;
2. дослідити особливості реалізації квантових алгоритмів на емуляторі;
3. спроектувати архітектуру майбутнього застосунку для вирішення проблеми факторизації та обрати програмні засоби;
4. розробити веб-застосунок, що дозволяє розкладати числа на множники як на класичному комп'ютері так і на квантовому емуляторі;
5. проаналізувати отримані результати за допомогою графіків та таблиць;
6. розгорнути розроблений веб-застосунок.

Практичне значення: розроблене програмне забезпечення може бути використане в навчальних цілях на лекціях з основ програмування, квантових обчислень, алгоритмів і структур даних. Створений програмний застосунок, звісно,

не зможе розшифрувати алгоритм RSA, проте дозволить вирішити **проблему** факторизації для невеликих чисел на квантовому емуляторі та класичному комп'ютері. Також розроблений веб-застосунок може бути корисним для подальших досліджень в галузі.

Об'єкт дослідження: задача розкладання на множники цілих чисел, алгоритми факторизації на квантовому комп'ютері, класичні алгоритми факторизації, існуючі веб-ресурси для розкладання числа на множники, особливості реалізації квантового комп'ютера.

Предмет дослідження: швидкодія алгоритму Шора на квантовому емуляторі IBM, порівняння класичного ро-алгоритму Полларда та методу з використанням решета Ератосфена.

Наукова новизна отриманих результатів:

1. розроблено зручний інтерактивний веб-застосунок для розкладання чисел на множники з можливістю вибору алгоритму;
2. проведено аналіз часу виконання квантових та класичних алгоритмів.

Апробація результатів роботи:

1. Лучинська Т. С. Порівняння алгоритмів факторизації для квантового та класичного комп'ютерів / Т. С. Лучинська, Р. Є. Рикалюк // Міжнародна наукова інтернет-конференція «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» – Тернопіль. – 2022. – Вип. 70 – С. 51-54.

РОЗДІЛ 1. ОСОБЛИВОСТІ КВАНТОВОГО КОМП'ЮТЕРА

1.1 Історія квантових обчислень

Сьогодні без доповідей дотичних до квантових обчислень не обходиться жодна конференція з інформаційних технологій. Але ж як все починалось?

На початку 1980-х фізик Пол Беніофф запропонував ідею квантово-механічної моделі машини Тьюрінга [2]. Звісно, вона була лише базовою, і сьогодні є набагато досконаліші моделі, проте ще тоді з'явився задум змоделювати квантові системи ефективним способом, адже класичні комп'ютери не справлялися з цією задачею. Саме Річард Фейнман та Юрій Манін першими зрозуміли, що за допомогою квантового комп'ютера можна буде імітувати обчислення, які не можна виконати, використовуючи класичний комп'ютер [15].

Ще більшої популярності квантові обчислення здобули після 1994 року, коли Пітер Шор запропонував квантовий алгоритм для розкладання великих цілих чисел на множники і сміливо заявив, що це буде експоненціально швидше, ніж будь-який інший алгоритм, який виконуватиметься на найкращому класичному комп'ютері. Адже існують підрахунки: для розкладання 300-значних чисел класичному комп'ютеру знадобляться мільйони років [19]. Таким чином вже у 1995 році навіть Армія Сполучених Штатів США виявила неабиякий інтерес у квантових обчисленнях.

У 1996 році з'являється квантовий алгоритм для пошуку в базі даних. Індійсько-американський науковець Лов Ґровер запропонував ідею квадратичного прискорення. З його допомогою можна виконувати завдання пошуку в 4 рази швидше [18].

Незважаючи на такий ажіотаж і захопленість, вчені тільки у 1998 році створили 2-кубітний комп'ютер з використанням перших квантових алгоритмів. А в 2000 році з'явилися ще 2 робочі квантові комп'ютери. Це 5-кубітний комп'ютер в університеті Мюнхена та 7-кубітний в лабораторії Лос-Аламоса. А вже наступного року вдалося реалізувати алгоритм Шора у Стенфордському університеті. Також продовжують з'являтися нові алгоритми для оптимізації

процесів у квантовому комп’ютері, вивчається явище квантової заплутаності, квантовий байт. Пропонуються методи, що дозволяють вимірювати стан кубітів без їх руйнування.

Зацікавленість у квантових технологіях весь час зростає. У 2007 році створено комп’ютер кластерного стану, який реалізовує алгоритм Дойча. Крім цього в 2007 році було вперше представлено фотонний квантовий комп’ютер.

Великим кроком стало відкриття «Quantum Experience» у 2016 році. Це інтернет-інтерфейс до квантових систем компанії IBM. З’явилися нові протоколи інтерпретації квантової інформації. Також компанією IBM створено 50-кубітний квантовий комп’ютер, який 90 мікросекунд підтримує стан [16].

В 2017 році було випущено систему Qiskit. Це програмне забезпечення, яке допомагає користувачам знайомитися з квантовими обчисленнями, спрощує роботу з кодом і надає можливості тестування програмного забезпечення на квантовому емуляторі, або навіть процесорі. Це значно полегшило знайомство з квантовими обчисленнями для людей, які просто зацікавлені в нових провідних технологіях та дозволило ефективно використовувати квантові емулятори в навчальному процесі університетів.

В 2020 році кілька різних команд вчених представили методи розв’язання нелінійних диференціальних рівнянь, використовуючи квантовий комп’ютер [17].

Також цікаво, що зовсім нещодавно в 2022 році Нобелівську премію у галузі фізики отримали Ален Аспект, Джон Клаузер та Антон Цайлінгтер «за експерименти із заплутаними фотонами, встановлення порушення нерівностей Белла та новаторство квантової інформаційної науки» [22]. Вони досліджували заплутані квантові стани, коли дві частинки починають поводитися однаково навіть при розділенні. Є очікування, що з результатами їх експериментів полегшиться розробка нових квантових технологій.

1.2 Огляд квантових обчислень

Для зберігання інформації класичний комп’ютер використовує послідовності одиниць та нулів. Всі прилади (комп’ютери, смартфони, розумні годинники, консолі, телевізори тощо) вміють приймати та обробляти таку інформацію.

Користувач може грати онлайн-гру, проходити навчальний курс, переглядати фотографії з важливої події, спілкуватися в чатах з сім'єю, виконувати складні математичні розрахунки, слідкувати за станом свого здоров'я під час бігу тощо, і навіть не здогадуватися, що насправді все це відбувається у двійковій системі, де є тільки «0» та «1» і більше нічого. Це дуже схоже на звичні «ні» і «так».

Можна здогадатися, що такий підхід є не надто ефективним у сьогоднішньому світі. Це стає помітно, коли система наближається до своїх меж. Наприклад, вчені працюють над розробкою математичної моделі поширення пандемії, або на новенькому телефоні стає недостатньо пам'яті для відео. В обох випадках корінь проблеми в тому, що нулів і одиниць стало занадто багато, а ресурсів і потужності для обчислення інформації надто мало. З цієї причини при виборі нової техніки ми завжди звертаємо увагу на характеристики пам'яті, процесора, можливості додавання карти пам'яті чи використання сховищ для зберігання важливих даних.

Дану проблему вирішує кубіт. Термін вперше запропоновано американським фізиком-теоретиком Бенджаміном Шумахером. Це квантовий біт, тобто основна одиниця інформації, яку використовують квантові комп'ютери. На відміну від класичного аналога, кубіт, завдяки квантовій механіці, може одночасно перебувати в стані суперпозиції «0» та «1». У теорії це явище дозволяє покращити швидкість обчислень та збільшити об'єми пам'яті [21]. На рис. 1.1. зображено схему кубіта:

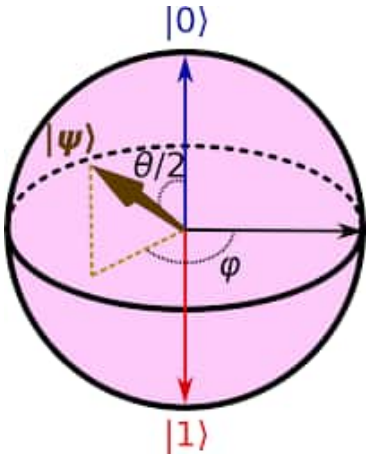


Рис.1.1. Схематичне зображення кубіта

Важливою особливістю в квантових обчисленнях є квантова заплутаність. Два кубіти є поєднаним таким чином: коли ми вивчаємо один з них, то інший знаходиться в ідентичному стані. З'являється кореляція між фізичними властивостями об'єктів, за якими ведуться спостереження. Явище заплутаності дозволяє групування кубітів в ефективні одиниці, що в свою чергу покращує показники швидкості запису та обробки інформації [13].

Виходить, що квантові пристрої використовують властивості окремих фотонів, атомів та інших елементарних частинок для інтерпретації інформації. Персональні чи будь-які інші класичні комп'ютери не здатні обчислювати в такий спосіб [9, с. 18]. Проте ще впродовж 1990-2000 років теоретичні дослідження підтвердили, що квантова механіка не має обґрунтованих тез для заборони розробки квантових комп'ютерів.

Квантове обчислення проходить три важливі етапи:

1. Ініціалізацію квантового регістра (приведення його в початковий стан);
2. Виконання дії квантових схем, складених із квантових вентилів (логічних елементів);
3. Прочитання (вимірювання) кінцевого стану [6].

1.3 Огляд фізичних реалізацій квантового комп'ютера

Створення фізичного квантового комп'ютера є нетривіальною задачею, адже потрібно не тільки підібрати оптимальну фізичну реалізацію кубітів, але й розробити контрольовану систему. Крім цього важливо, як організовано вимірювальну підсистему та первісний стан. У даному етапі й полягає технічна складність розробки. Трапляється й таке, що квантові стани добре розрізняються в системі, але швидко руйнуються через вплив зовнішніх факторів. У таких випадках є необхідність залучати додаткові ресурси для підтримки стану. А іноді бувають протилежні ситуації: квантовий стан існує достатньо довго (макроскопічний час), але його практично неможливо виміряти.

Провідні вчені в областях квантових обчислень зараз можуть запропонувати кілька головних ідей для реалізації комп'ютерів: на фотонах, на оптичних

резисторах, йони у пастках, гармонічний осцилятор та ядерний магнітний дисонанс. Всі ці моделі мають ряд переваг та недоліків.

Зокрема схема з гармонічним осцилятором представляє кубіти у формі рівнів енергії у багаторівневій системі квантів. Цей підхід не можна використовувати з деякими квантовими операторами через невизначений спектр.

Дуже цікавою є система, в основі якої є ідея реалізації квантового біта оптичним фотоном. Фотони – це нейтральні частинки. Вони проявляють слабку взаємодію між собою та середовищем. Загалом оптичні комп'ютери вже давно подавали як альтернативу класичним і навіть розроблено експериментальні зразки оптичних процесорів та комп'ютерів [11]. Проте багато надій в розробці поки не виправдали себе. Переваги у швидкості досить незначні, незважаючи на можливість паралельних обчислень. Ще не вдалося отримати матеріали з високим ступенем нелінійності й існує багато невизначеностей у налаштуваннях.

З поданих вище прикладів можна зрозуміти, що реалізація квантового комп'ютера є справді складною задачею, над якою працюють десятиліттями команди вчених по всьому світу, при цьому враховуючи багато майже протилежних вимог. Поки із запропонованих варіантів важко обрати оптимальний, адже всі ідеї потребують вдосконалення для того, щоб в найближчому майбутньому стала можливою побудова багатокубітного квантового комп'ютера. Однак не можна казати, що фізична реалізація квантового комп'ютера є неможливою, адже дослідження активно проводяться і є ще багато варіантів [12]. На рис. 1.2 можемо побачити фото одного з квантових комп'ютерів компанії IBM.

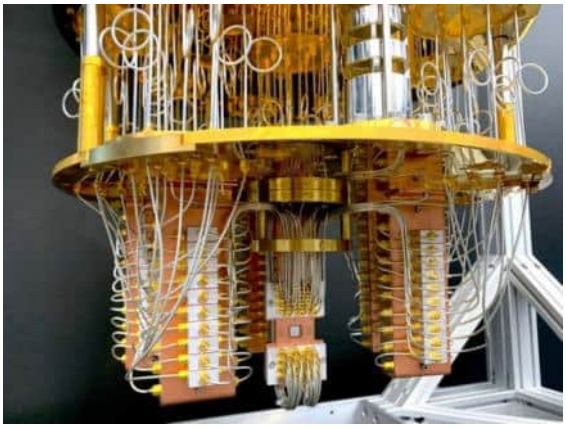


Рис. 1.2. Квантовий комп'ютер IBM

1.4 Порівняння квантового та класичного комп’ютерів

Якщо потрібно порівняти класичний і квантовий комп’ютер, то часто чуємо приклад, що це як свічка і лампочка. Обидва прилади проводять розрахунки, але шляхи обчислень кардинально відрізняються. Не можемо сказати, що лампочка – це просто краща, більша чи оптимальна свічка. Так само і квантовий комп’ютер – це повністю новий пристрій, нова технологія.

Обробка інформації у квантовому комп’ютері здійснюється незвичним для нас сьогодні шляхом. Тут немає транзисторів з «0» та «1», зовсім немає двійкової системи. Запропоновано цілком новий спосіб подання інформації через кубіти, які можуть одночасно бути як «0», так і «1».

Відповідно, якщо потужність класичного комп’ютера покращується прямо пропорційно до кількості транзисторів, то потужність квантового комп’ютера збільшується в геометричній прогресії до кількості кубітів, які пов’язані між собою. Завдяки цьому очікується, що в майбутньому квантові комп’ютери будуть дуже ефективними для деяких типів обчислень. У цьому полягає надефективність квантового процесора. Головна мета – це розв’язати за поліноміальний час задачі, які класичний комп’ютер може обробити лише за експонентний час [7].

Є припущення, що квантові комп’ютери матимуть перевагу у таких завданнях як оптимізація маршрутів чи моделювання хімічної реакції. Однак вже існує багато задач, з якими класичний комп’ютер сьогодні справляється дуже добре. Тому, ймовірно, що значна частина повсякденної обробки інформації так і не перейде на квантові обчислення, навіть коли з’явиться більш потужне квантове апаратне забезпечення.

Можна зробити висновок, що ключовою відмінністю квантового й класичного комп’ютерів є те, що програми в квантовому комп’ютері є ймовірнісними, а в класичному – детермінованими. Тобто під час квантових обчислень всі результати мають амплітуду ймовірності і після проведення вимірювань результатом є лише один з можливих станів. Тоді як під час звичних нам обчислень, результат завжди подається бітами, а біт завжди детермінований – «0» або «1».

РОЗДІЛ 2. ПОСТАНОВКА ЗАВДАННЯ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО ВЕБ-ЗАСТОСУНКУ ДЛЯ ФАКТОРИЗАЦІЇ ЧИСЕЛ

2.1 Постановка завдання

У результаті виконання магістерської роботи повинен бути розгорнутий веб-застосунок, який надаватиме користувачеві можливість знаходити дільники числа. У застосунку буде доступний вибір алгоритму. Тобто користувач зможе обрати, яким саме способом факторизувати число. У програмі буде реалізовано кілька алгоритмів для класичного комп’ютера та алгоритм Шора, який можна запустити на квантовому емуляторі.

Звісно, з урахуванням технічних можливостей на веб-сторінці всі алгоритми мають працювати до 30 секунд, тому класичні алгоритми не зможуть знайти дільники 100-цифрових чисел, а квантовий алгоритм Шора буде виконуватися на квантовому емуляторі, який не надає ресурсів для факторизації великих чисел.

Користувачам веб-застосунку для факторизації чисел не потрібно буде реєструватися чи проходити процес авторизації. Система одразу дозволяє запускати та аналізувати алгоритми.

Для аналізу отриманих результатів веб-сторінка автоматично представлятиме всі дані в графічному та табличному вигляді.

Щоб не втратити результати зроблених досліджень, користувач може зберегти їх на своєму комп’ютері у файл з власним розширенням «.qw_lnu». Створення файлу з власним розширенням, допоможе ідентифікувати, які саме файли можна завантажити в систему на наступній сесії для подальшого аналізу.

Отже, перед написанням магістерської роботи було поставлено такі завдання:

1. ознайомитися з алгоритмами факторизації для квантового та класичного комп’ютерів;
2. порівняти сайти аналоги;
3. написати специфікацію вимог для програмного забезпечення;
4. реалізувати обрані алгоритми факторизації;
5. спроектувати архітектуру системи;

6. розробити веб-застосунок для факторизації чисел;
7. розгорнути веб-застосунок;
8. протестувати розроблений функціонал і виправити знайдені помилки.

2.2 Специфікація вимог до системи

2.2.1 Вступ

2.2.1.1 Призначення

Веб-застосунок призначений для порівняння методів знаходження дільників числа. Програма може бути корисною, як для учнів та студентів, які тільки починають свій шлях в програмуванні або математиці, так і для проведення досліджень більш досвідченими спеціалістами.

2.2.1.2 Продукти-аналоги

Оскільки не було знайдено платформи, яка поєднує квантові і класичні алгоритми, візьмемо для порівняння такі програми: «Shor's Algorithm simulator» (розроблено в дослідницькому університеті штату Колорадо), Prime Factorization Calculator на платформі Calculator.net та Prime Factorization у системі Wolfram Alpha. Результати порівняння наведено у таблиці 2.1.

Таблиця 2.1.

Порівняння сайтів-аналогів

Функціонал	**Shor's Al. sim.**	**Calculation.net**	**Wolfram Alpha**
Вибір алгоритму	+/-	-	-
Реєстрація та авторизація	-	-	+
Зручність користування	-	-	-
Зберігання результату	-	-	-
Покроковий вивід	-	+	+
Ділення чисел з 10 цифрами	-	+	+
Графіки	+	+/-	-

2.2.2 Загальний опис

2.2.2.1 Перспективи продукту

Зараз дуже важливо вдосконалювати науково-технічний потенціал України. Одним з найважливіших є напрямок кібербезпеки. Перспективним є розв'язок задачі факторизації. Над створенням квантового комп'ютера для реалізації квантового алгоритму Шора працюють вчені по всьому світу. Розроблений веб-застосунок дозволить користувачам ознайомитися із задачею факторизації, а також порівнювати результати роботи різних алгоритмів знаходження дільників числа.

2.2.2.2 Характеристики продукту

Веб-сторінка надаватиме користувачам такі функціональні можливості:

8. завантаження результатів з файлу;
9. знаходження дільників числа з використанням квантового алгоритму Шора;
10. знаходження дільників числа за допомогою класичного алгоритму з решетом Ератосфена;
11. знаходження дільників числа з використанням класичного ро-алгоритму Полларда;
12. графічне подання результатів;
13. табличне подання результатів;
14. збереження результатів у файл з власним розширенням.

2.2.2.3 Класи користувачів

У веб-застосунку не має реєстрації та авторизації, тому немає потреби в адміністраторі чи додаткових класах користувачів. Кожен користувач платформи зможе завантажувати та зберігати файли, використовувати алгоритми та аналізувати дані за допомогою графіків та таблиці результатів.

2.2.2.4 Обмеження проектування та реалізації

Під час розробки варто пам'ятати про такі обмеження:

1. всі алгоритми повинні працювати не довше ніж 30 секунд;
2. квантові емулятори не дають достатньо пам'яті для опрацювання великих чисел;

3. термін розробки програмного забезпечення та розгортання на сервері – 3 місяці;

2.2.2.5 Середовище функціонування

Веб-сайт можна відкривати з будь-якої операційної системи (Windows, Ubuntu, Android тощо). Для запуску на локальному комп'ютері рекомендується мати не менше 2 гігабайт оперативної пам'яті та встановлені React, Node, Python, Qiskit.

2.2.3 Характеристики системи

2.2.3.1 Завантаження файлу

2.2.3.1.1 Опис і пріоритет

Користувач веб-застосунку може завантажувати файли з розширенням «.qw_lnu».

Пріоритет – середній.

2.2.3.1.2 Функціональні вимоги

REQ1: перевірка формату файлу;

REQ2: обробка інформації з файлу;

REQ3: повідомлення про помилку, якщо система не може опрацювати вміст файлу;

REQ4: представлення даних з файлу у коректному вигляді на графіку;

REQ5: представлення даних з файлу у коректному вигляді у таблиці.

2.2.3.2 Зберігання результатів у файл

2.2.3.2.1 Опис і пріоритет

Користувач веб-застосунку може зберігати результати своїх досліджень у файл з розширенням «.qw_lnu».

Пріоритет – середній.

2.2.3.2.2 Функціональні вимоги

REQ1: автоматична генерація імені файлу з відповідною датою;

REQ2: формування вмісту файлу у JSON форматі;

REQ3: вивід всієї інформації у файл;

REQ5: збереження файлу;

2.2.3.3 Знаходження дільників числа

2.2.3.3.1 Опис і пріоритет

Користувач веб-застосунку може вводити число, для якого необхідно знайти дільники та обирати алгоритм.

Пріоритет – високий.

2.2.3.3.2 Функціональні вимоги

REQ1: введення числа, для якого потрібно знайти дільники;

REQ2: можливість вибору квантового алгоритму Шора;

REQ3: можливість вибору класичного алгоритму з решетом Ератосфена;

REQ4: можливість вибору класичного ро-алгоритму Полларда;

2.2.3.4 Відображення результатів

2.2.3.4.1 Опис і пріоритет

Результат роботи алгоритмів одразу додається до графіку та таблиці.

Пріоритет – високий.

2.2.3.4.2 Функціональні вимоги

REQ1: побудова графіку на основі результатів алгоритму;

REQ2: відображення на одному графіку коректних даних про роботу всіх алгоритмів, де по осі X буде номер запуску алгоритму, а по осі Y – час роботи алгоритму;

REQ3: представлення даних в таблиці, посортованій за назвою алгоритму;

2.2.4 Вимоги зовнішніх інтерфейсів

2.2.4.1 Користувацькі інтерфейси

Система повинна передбачити, що дії користувача можуть бути некоректними. Наприклад, завантаження файлу, який неможливо прочитати. Для покращення користувацького досвіду користувача потрібно розробити інтуїтивний дизайн зі зрозумілими шрифтами та кольоровою схемою.

2.2.4.2 Апаратні інтерфейси

Немає додаткових умов до апаратних інтерфейсів. Програма може відкриватися у будь-якому браузері комп’ютерів, телефонів чи інших пристроїв з доступом до мережі Інтернет.

2.2.4.3 Програмні інтерфейси

Для запуску на локальному комп’ютері рекомендується мати не менше 2 гігабайт оперативної пам’яті та встановлені React, Node, Python, QisKit.

2.2.5 Нефункціональні вимоги

2.2.5.1 Вимоги продуктивності

Запит повинен тривати не довше ніж 30 секунд. Користувач повинен бачити результати на графіку одразу після успішного проходження алгоритму.

2.2.5.2 Вимоги надійності

Система не повинна ламатися після некоректних дій користувача. Користувач може продовжити роботу після повідомлення про помилку.

2.2.5.3 Вимоги безпеки

Запити між серверною частиною та користувацьким інтерфейсом повинні використовувати захищений протокол HTTPS.

2.2.5.4 Атрибути якості програмного продукту

Користувацький інтерфейс веб-платформи мусить бути зрозумілим користувачам. Крім цього треба не забувати про надійність та швидкодію, тобто алгоритми повинні працювати миттєво для невеликих чисел. Проте допускається, що для більших чисел результати повернуться в межах до 30 секунд.

2.3 Опис обраних технологій і засобів розробки

2.3.1 Серверна частина веб-застосунку

Для реалізації серверної частини було використано мову програмування Python. Це відома строго типізована об’єктно-орієнтована мова програмування, яка підтримує багато додаткових бібліотек для полегшення написання коду. Зокрема, має пакети для роботи з квантовими технологіями.

Для опрацювання REST запитів було використано мікрофреймворк Flask. Обрана технологія називається мікрофреймворком, оскільки не вимагає ніяких додаткових засобів чи спеціальних бібліотек для використання. Запуск сервера і клієнта на одному комп’ютері також можливий, оскільки було використано надбудову «Flask-Cors».

Для розробки квантового алгоритму Шора було використано QisKit. Це один з найпопулярніших фреймворків для полегшення процесу розробки коду, який необхідно запустити на квантових комп'ютерах або емуляторах. QisKit має добре написану документацію, і відео-лекції, в яких розповідають, як працювати з фізичним квантовим комп'ютером доступним від IBM.

QisKit пропонує кілька бібліотек:

1. QisKit Aer – пакет, який дозволяє запускати код на квантовому емуляторі;
2. QisKit Aqua – оптимізує квантові обчислення для моделювання хімічних чи фінансових процесів, та розробки штучного інтелекту;
3. QisKit Terra – один з найпопулярніших пакетів QisKit, який надає розробниками можливість писати програми на рівні схем та імпульсів. Такий підхід оптимізує роботу з фізичним квантовим процесором, навіть при віддаленому доступі.

2.3.2 Клієнтська частина веб-застосунку

При розробці клієнтської частини веб-додатку для факторизації числа використано відкриту JavaScript бібліотеку React. Це декларативна бібліотека для створення інтерактивних веб-інтерфейсів на основі компонентів. Під час розробки необхідно правильно розбити сторінку на невеликі компоненти. Кожен компонент вміє самостійно контролювати свій стан та взаємодіяти з іншими компонентами.

Компоненти React спрощують написання інтерфейсу та логіки сторінки, адже поєднують і HTML, і JavaScript. Вся інформація про відображення даних, а також логіка їх обробки для невеликої частини сторінки знаходиться саме у файлах компонентів. Компоненти можуть використовуватися в інших і складати деревовидну структуру [23]. На рис. 2.1. зображено структуру компонентів:

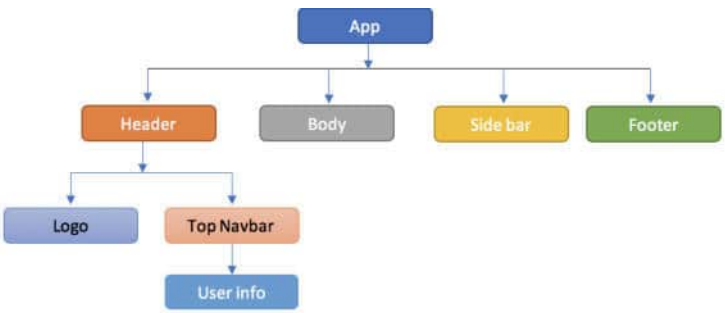


Рис. 2.1. Дерево компонентів з використанням React

Для зв’язку з сервером і зручного зберігання даних використано бібліотеку Redux. Це менеджер стану системи, який доступний на всіх рівнях компонентів. Щоб змінити стан, необхідно викликати відповідну дію. Компоненти слідкують за змінами стану в Redux.

Наприклад, під час асинхронного запиту на сервер можна контролювати стан таким чином:

1. якщо запит ще не отримав відповідь, ми можемо показати індикатор завантаження;
2. якщо отримано помилку, показуємо повідомлення про помилку;
3. якщо запит успішний – обробляємо відповідь і показуємо коректні дані.

На рис. 2.2. зображено модель використану для реалізації програми з використання Redux, React Components і серверу з Python:

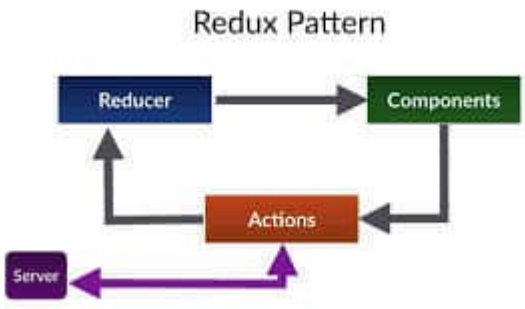


Рис. 2.2. Redux модель

Для побудови графіків обрано бібліотеку Recharts. Одразу після обробки числа на сервері, графік оновлюється з часом виконання останнього запиту.

Для відображення даних в таблиці, а також числових полів і кнопок використовується бібліотека BlueprintJS. Ця бібліотека надає пакети з необхідними інструментами для розробки користувацького інтерфейсу. У додатку А знаходиться UML-діаграма діяльності для розробленого застосунку, яка показує послідовність переходів від однієї активності до іншої.

РОЗДІЛ 3. ПРОБЛЕМА ФАКТОРИЗАЦІЇ ТА ОГЛЯД АЛГОРИТМІВ

3.1 Проблема факторизації

У наш час всі сфери життя використовують інформаційні технології. Комп'ютери допомагають розв'язувати різні види задач із застосуванням різноманітних розрахункових методів. Деякі з цих задач обчислювально прості, а деякі надзвичайно складні і досі потребують покращення алгоритмів чи потужнішого апаратного забезпечення. Всі ці завдання класифікують за складністю їх розв’язку. При цьому враховується наскільки сильно збільшуються обчислювальні ресурси під час збільшення вхідних даних до алгоритму. Метриками є час виконання та вимоги до пам’яті. Прикладом складної в обчисленнях задачі є факторизація, тобто знаходження дільників числа.

Проблема розкладання на множники є актуальною в сфері криптографії. Множення чисел зараз є легкою задачею, і у сучасних комп’ютерах є достатньо ресурсів для роботи навіть з дуже великими числами. А ось обернена до множення операція, розкладання числа на множники, потребує значно більших витрат. Це і є основоположною ідеєю багатьох методів шифрування та дешифрування. На вирішення завдання факторизації числа n найкращими класичними алгоритмами необхідно стільки часу:

$$\exp\{c\sqrt{\ln n \ln\ln n}\} \tag{3.1}$$

де в найкращому випадку c = 1 [14].

Сьогодні вже реально розкласти на множники числа з 150 цифрами, й існують очікування про розкладання чисел з 200 цифрами в майбутньому [3]. Також існує багато досліджень, про те, що на квантовому комп’ютері завдання факторизації буде виконуватися за поліноміальний час.

Існує кілька причин, чому розробка квантового алгоритму для розкладання числа на множники є такою важливою:

1. доведення теорії, що квантові комп’ютери є ефективнішими за класичні;
2. над завданням факторизації працює багато дослідників і поява нових алгоритмів завжди викликає зацікавлення;

3. ефективний алгоритм факторизації дозволить зламати методи, які використовуються в криптографії. Зокрема відомий криптографічний алгоритм RSA з відкритим ключем, який є дуже поширеним у мережі Інтернет [4].

3.2 Квантовий алгоритм Шора

3.2.1 Історія розвитку алгоритму

Вже давно ведуться дискусії про недосконаліть класичних алгоритмів розкладання на множники для великих цілих чисел. Багато сучасних алгоритмів шифрування користуються цим фактом і часто множать прості числа. Саме це стало причиною інтересу до квантових технологій після представлення алгоритму Шора більше ніж 25 років тому. Передбачається, що розробка ефективного квантового комп'ютера і виконання на ньому алгоритму значно вплине на розвиток криптографії та машинного навчання.

У 1994 році американський вчений Пітер Шор вперше запропонував поліноміальний алгоритм, який дозволяє знаходити дільники числа значно швидше ніж всі відомі сьогодні класичні алгоритми. Вже у 2001 році цю теорію було перевірено на практиці. Група дослідників Стенфордського університету у співпраці з компанією IBM змогли розкласти на множники число 15. Для цього було створено фізичний квантовий комп'ютер із сімома кубітами.

Ще через 6 років незалежні спеціалісти із США та Китаю змогли виконати алгоритм, використовуючи фотонний квантовий комп'ютер.

Саме завдяки алгоритму Шора для факторизації чисел, квантові технології набули такого розголосу в кінці минулого століття, і досі залишаються актуальними. Нобелівську премію з фізики у 2022 році здобули вчені, які працюють з квантовими інформаційними технологіями. Тому в найближчому майбутньому можна очікувати ще більше новин про розвиток квантових обчислень.

3.2.2 Огляд алгоритму

Алгоритм Шора умовно можна поділити на дві частини: класичну і квантову. Класична частина робить необхідні перетворення, щоб привести завдання до

знаходження періоду функції. Квантова частина знаходить період функції, використовуючи перетворення Фур'є.

Нехай задано ціле число *N*. Тоді необхідно знайти число *p,* яке є дільником числа *N*. Відомо, що $1 < p < N$.

Покроковий алгоритм класичної частини:

1. обрати псевдовипадкове число *a* менше за *N*;
2. обчислити найбільший спільний дільник для *a* і *N*. Наприклад, використовуючи алгоритм Евкліда. Нехай *K* – обчислений дільник;
3. якщо $K \neq 1$, то існує нетривіальний множник *N*, можна зупинятися на цьому кроці;
4. якщо $K = 1$, необхідно викликати функцію пошуку періоду, щоб знайти *r*, період наступної функції: $f(x) = a^x \bmod N$, тому шукаємо найменше ціле число *r* , для якого $f(x + r) = f(x)$;
5. якщо *r* непарне або $r/2 \equiv -1 \pmod N$, повертаємося до кроку 1;
6. фактором *N* є найбільший спільний дільник а $r/2 \pm 1$ та *N* .

Квантова частина є більш складною для опису і включає в себе квантовий паралелізм, перетворення Фур'є, знаходження періоду та дільника числа *N*:

1. починаємо з пари вхідних і вихідних регістрів кубітів з $\log 2\, N$ кожен й ініціалізуємо їх

$$N^{-1/2} \sum_x |x\rangle |0\rangle \tag{3.2}$$

де $0 < x < (N - 1)$;

2. побудуємо та застосуємо $f(x)$ як квантову функцію;
3. використаємо квантове перетворення Фур'є до вхідного регістра для *N* точок:

$$U_{QFT} |x\rangle = N^{1/2} \sum_y e^{2\pi i x \frac{y}{N}} |y\rangle \tag{3.3}$$

Тоді ми знаходимося у такому стані:

$$N^{-1} \sum_x \sum_y e^{2\pi i x y/N} |y\rangle \ |f(x)\rangle \tag{3.4}$$

4. проводимо вимірювання й вираховуємо *y* та $f(x_0)$. Функція $f(x)$ є періодичною, тому визначаємо ймовірність вимірювання *y*;

5. перетворюємо *y/N* як нескоротний дріб і дістаємо можливе *r;*
6. перевіряємо, чи $f(x) = f(x + r)$. Якщо так, то розв'язок знайдено.
7. шукаємо інші можливі значення r, використовуючи значення поблизу y або кратні попередньому r і перевіряємо крок 6.
8. повертаємося до кроку 1 [1].

На рис. 3.1 зображено квантову схему алгоритму Шора:

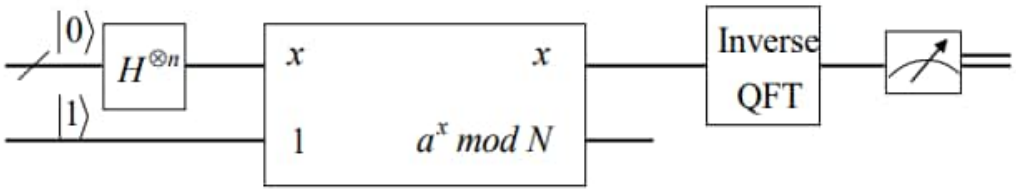


Рис. 3.1. Квантова схема алгоритму Шора

Отже, описаний алгоритм розкладання на множники числа використовує можливості квантових технологій перебувати одночасно в декількох станах. Використовуючи суперпозицію, отримуємо можливість обчислити період функції *f* одразу у всіх точках. Проте через обмеження квантової фізики, ми можемо отримати лише одне можливе значення, тому і використовуємо квантове перетворення Фур'є.

З часу презентації алгоритму у 1994 році існує багато досліджень і дискусій про цей метод. Зокрема, запропоновано кілька модифікацій для оптимізації ресурсів. Незважаючи на те, що поки через обмеження апаратного забезпечення не можливо використати алгоритм для розкладання великих чисел, вже є розрахунки прогнозованого часу виконання факторизації числа з 100 десяткових чисел, тобто 375 біт (рис. 3.2). У першій колонці тут тактова частота процесора, а в другій прогнозований час роботи алгоритму.

Clock Speed	**Approximate Time**
1 GHz	< 1 *second*
1 MHz	~120 *seconds*
1 KHz	~15 *days*
1 Hz	~10 *years*

Рис. 3.2. Прогнозований час виконання факторизації числа з 375 біт

Вже з цих прогнозів можна зробити висновок, що після фізичної реалізації квантового комп'ютера алгоритми криптографічних систем типу RSA, тобто ті, які використовують множення великих простих чисел, будуть під загрозою [19].

3.3 Аналіз класичних алгоритмів факторизації

3.3.1 Короткий огляд класичних алгоритмів

Проблема розкладання цілих чисел на множники не є новою, тому існує багато класичних методів її розв'язання. Кожен підхід має право на існування і вибір найкращого з них досить суб'єктивний, адже бувають різні вимоги. У сфері інформаційних технологій ефективним можна вважати алгоритм, який не потребує значних ресурсів пам'яті комп'ютера та зусиль на програмну реалізацію.

Очевидним є підхід зі створенням списку простих чисел. Після цього відбувається перевірка, чи ділиться задане число на кожен елемент списку без остачі. Однак цей процес займає багато часу, й потребує вдосконалення.

3.3.2 Алгоритм з використанням решета Ератосфена

Сьогодні є відомим алгоритм з використанням решета давньогрецького математика Ератосфена. Він дозволяє швидко генерувати велику кількість простих чисел. На основі цієї ідеї запропоновано кілька методів для ефективного розкладання цілих чисел на множники. Одним з найпростіших варіантів є наступний підхід: після створення достатньо великого списку простих чисел, тобто від 2 до заданого числа, перевіряється кожне число з числом, яке факторизується.

Принцип дії полягає в тому, що ми вже знаємо число, яке необхідно розкласти для множники. Тоді ми можемо скласти список простих чисел, які менші за задане число. Простіше пояснити метод складання списку на прикладі. Нехай задано число 15. Тоді ми маємо список цілих чисел від 2 до 14. Почнімо з простого числа 2 і видалимо зі списку всі числа кратні 2 (4, 6, 8, 10, 12, 14). Наступним є число 3. Видаляємо числа кратні 3 (9). Оскільки 4 видалено, наступним є число 5. Кратних йому вже немає в списку. Так само з 7, 11 і 13. Отже, отримуємо такий список: 2, 3, 5, 7, 11, 13. Тепер по черзі пробуємо задане число 15 розділити на кожен елемент списку і отримуємо результат 3 і 5.

Покрокове створення списку можна описати так:

1. починаємо зі списку цілих чисел від 2 до заданої межі;
2. нехай $p = 2$, тоді позначаємо всі кратні числа до p у списку, крім самого p;
3. переходимо до наступної ітерації: нехай p буде рівним наступному не позначеному числу в списку. Тоді ще раз позначаємо всі кратні до p;
4. повторюйте, доки не дійдемо до кінця списку.

На рис. 3.3. зображено приклад знаходження простих чисел на етапі 67, кольорами показано ітерації.

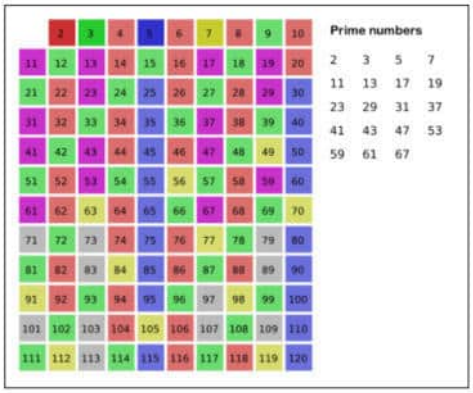


Рис. 3.3. Приклад списку простих чисел з використанням решета Ератосфена

3.3.3 Ро-алгоритм Полларда

Алгоритми для класичного комп'ютера вже вміють швидко справлятися з факторизацією чисел, що складаються зі 100 цифр. Прикладом може бути ро-алгоритм Полларда, заснований на ідеях модульної арифметики. Він використовує алгоритм пошуку циклу Флойда.

Часто цей алгоритм також називаються методом черепахи та зайця. Оскільки в ньому присутні кілька циклів, один з яких рухається повільніше за інший (рис. 3.4).

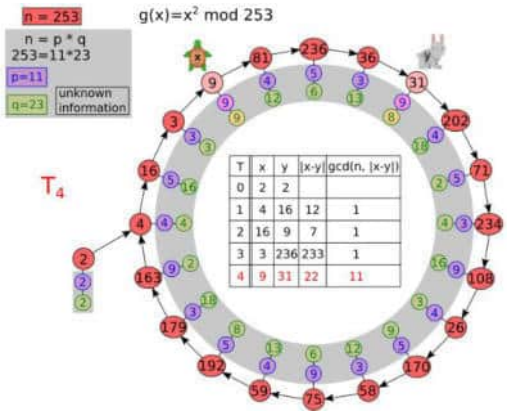


Рис. 3.4. Приклад факторизації числа 253

Алгоритм пропонує побудову числової послідовності, яка утворює цикл, схожий на літеру ρ. Складність алгоритму $O(N^{1/4})$ [5].

Складається ро-алгоритм Полларда з наступних кроків:

1. нехай необхідно розкласти на множники число *N*. І задано поліном *f(x)=F(xn)(mod N);*
2. тепер оберемо число x_o і побудуємо послідовність $n\{xn\}$, де n = 0, 1, 2…;
3. знаходимо наступні значення x використовуючи функцію *f(x);*
4. знаходимо найменший спільний дільник між N і $|x_i - x_j|$, де i – будь-який крок ітерації, й $j < i$;
5. якщо найменший спільний дільник *d* більший за 1, то ми отримали відповідь;
6. в іншому випадку повертаємося до попередніх кроків і беремо замість *N* значення *N/d.*

Для реалізації алгоритму у магістерській роботі визначимо *f(x)* наступним чином:

$$(x^2 - 1) \% n \tag{3.5}$$

Ро-Алгоритм Полларда також є ймовірнісним алгоритмом, в якому число кроків не можна точно визначити, адже це випадкова величина, яка залежить від конкретної реалізації алгоритму і використаного генератора псевдовипадкових чисел [20].

РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ КВАНТОВИХ ТА КЛАСИЧНИХ АЛГОРИТМІВ

4.1 Проектування архітектури системи

Веб-застосунок для факторизації чисел було розроблено із застосуванням клієнт-серверної архітектури. На рис. 4.1 подано схему роботи системи.

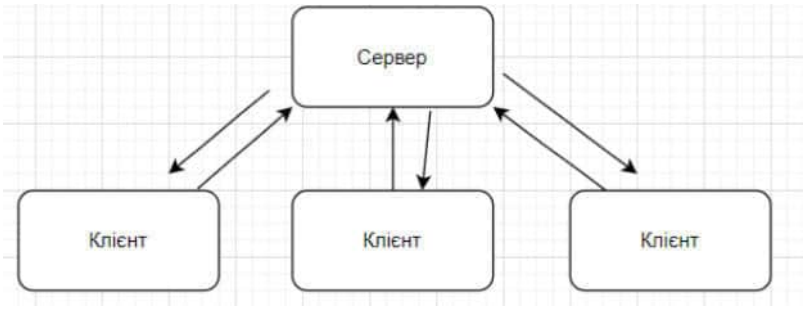


Рис. 4.1. Клієнт-серверна архітектура

Клієнтом є веб-сторінка з графічним інтерфейсом, яка відправляє HTTPS запити серверу, відповідно до того, які дані ввів користувач. У розробленому застосунку клієнт відповідає за те, щоб правильно сформувати URL для сервера, тобто взяти до уваги обраний користувачем алгоритм та число, яке необхідно розкласти на множники. Після натиснення кнопки «Запустити» на сервер відправляється запит GET з цими параметрами.

Для того, щоб надіслати запит на сервер, необхідно знати його URL-адресу. І клієнт, і сервер можуть працювати на одному комп’ютері та localhost, проте тоді необхідно налаштувати проксі конфігурацію.

4.2 Розгортання проекту

Для розгортання системи було обрано сервіс Heroku. Це хмарна платформа, яка має можливість самостійно обрати, які саме сервіси необхідні для ефективного розгортання розробленого програмного забезпечення. Невеликі додатки можна перенести на Heroku майже без додаткової конфігурації.

Щоб розгорнути розроблений веб-застосунок необхідно 2 сервіси:

1. клієнтська частина розроблена за допомогою бібліотеки React.
2. серверна частина з кодом мовою Python, на якій безпосередньо виконуються алгоритми факторизації;

Для розгортання серверної частини було створено файли «Procfile» та «requirements.txt», які вказують Heroku, які саме залежності потрібні для роботи програми та як запустити код. Після виконання необхідних команд і створення проекту Heroku, серверну частину було розгорнуто за таким посиланням: *//quantum-world-app.herokuapp.com/*.

На рис. 4.2 зображено приклад роботи запитів до сервера (факторизація числа 15 ро-алгоритмом Полларда):

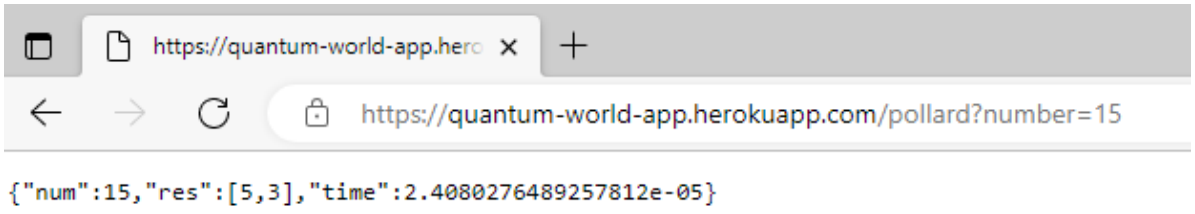


Рис. 4.2. Приклад роботи серверної частини на Heroku

Оскільки бібліотека React є дуже популярною, то вже існують пакети, які необхідно просто додати до проекту, щоб запустити його на Heroku. Для розгортання клієнта було вказано, який саме пакет використовувати, за допомогою команди: *heroku buildpacks:set mars/create-react-app*. На рис. 4.3. зображено приклад роботи клієнтської частини розробленого веб-додатку:

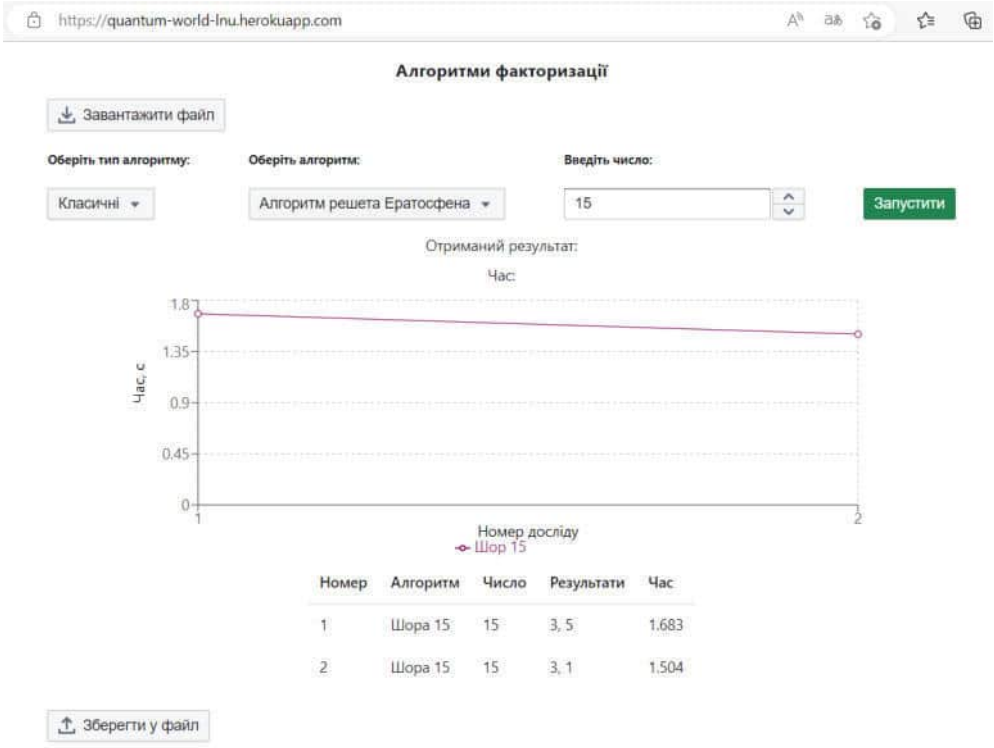


Рис. 4.3. Приклад роботи клієнтської частини на Heroku

Клієнтська частина доступна за посиланням: *//quantum-world-lnu.herokuapp.com.* Вона відправляє запити на серверну частину, і вміє обробляти отримані результати. Цікаво, що на Heroku швидкість роботи алгоритмів зросла, порівняно з локальним комп'ютером, який використовувався для розробки.

4.3 Опис роботи програми

При вході на веб-сторінку ми бачимо пустий екран без даних про попередні дослідження (рис. 4.4).

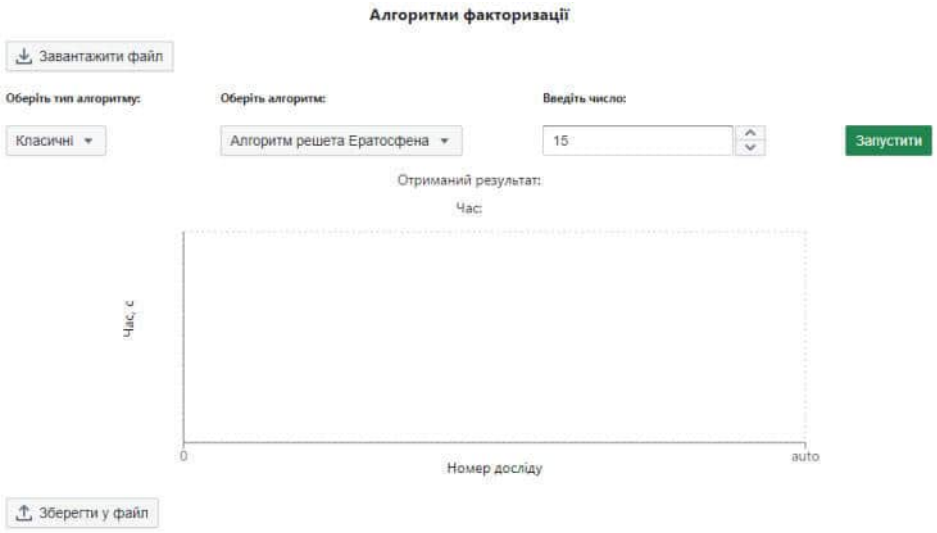
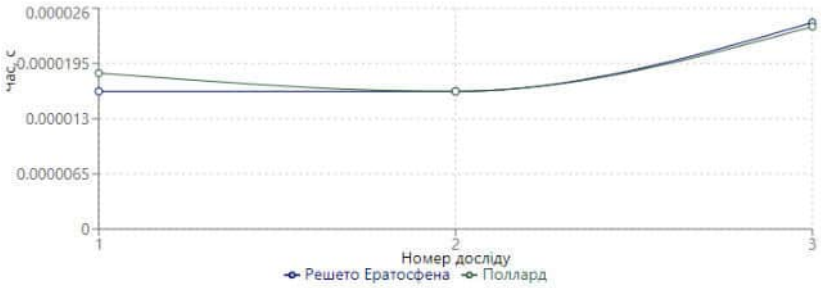


Рис. 4.4. Початковий інтерфейс програми

Користувач може обрати тип алгоритму (квантовий або класичний), сам алгоритм, ввести число, яке необхідно розкласти на множники та натиснути на кнопку «Запустити». Після цього почекати, поки отримані результати з'являться на графіку та у таблиці. У системи є кілька обмежень. Тільки класичні алгоритми можуть ділити великі числа. Реалізований квантовий алгоритм Шора дозволяє ділити тільки число 15, а алгоритм Шора з пакету QisKit дозволяє ділити числа 15 і 21, проте обчислення на емуляторі тривають довше ніж 30 секунд і запуск на сервері Heroku повертає помилку.

Для того, щоб швидко завантажити дані минулих досліджень, необхідно натиснути на кнопку «Завантажити файл». Тоді відкриється вікно файлової системи, в якому дозволено обирати файли тільки з розширенням «.qw_lnu». Такі файли генеруються системою після натискання на кнопку «Зберегти у файл».

Після того, як файл успішно завантажено, дані з файлу обробляються, відповідно до алгоритму і подаються у табличному та графічному вигляді, як на рис. 4.5:



Номер	Алгоритм	Число	Результати	Час
1	Поллард	15	5, 3	0.000
2	Поллард	15	5, 3	0.000
3	Поллард	21	3, 7	0.000
1	Ератосфен	15	3, 5	0.000
2	Ератосфен	15	3, 5	0.000

Рис. 4.5. Дані прочитані з файлу

Користувачам веб-сторінки для факторизації чисел важливо зберегти результати своїх досліджень. Адже деякі алгоритми досить довго працюють (зокрема квантовий алгоритм Шора, або класичні алгоритми для великих чисел). Наприклад, іноді алгоритм Шора може ділити число 15 близько десяти секунд. Також, може бути цікаво порівняти результати з отриманими раніше, або ж поділитися ними з іншими користувачами системи.

Після проведення необхідних досліджень користувач може легко зберегти дані у своїй файловій системі, натиснувши кнопку «Зберегти у файл». Результати всіх досліджень зберігаються у форматі JSON під власним розширенням «.qw_lnu», щоб потім можна було легко ідентифікувати файли, які створені розробленим веб-сервісом. Назва формується автоматично – дата збереження. У додатку Б наведено приклад сформованого файлу з заповненими результатами для всіх типів алгоритмів.

Оскільки всі результати зберігаються на клієнтській частині проекту, було обрано бібліотеку «FileSaver.js», яка вміє швидко генерувати файли у потрібному форматі.

4.4 Аналіз отриманих результатів

Після розробки програмного забезпечення було проведено тестування для перевірки результатів. Для полегшення аналізу програма одразу будує графік з отриманими результатами. По вертикальній осі тут час виконання алгоритму в секундах. По горизонтальній осі послідовний номер запуску алгоритму. Кожен алгоритм має інший колір на графіку.

На рис. 4.6 бачимо результати роботи алгоритмів для числа 15:

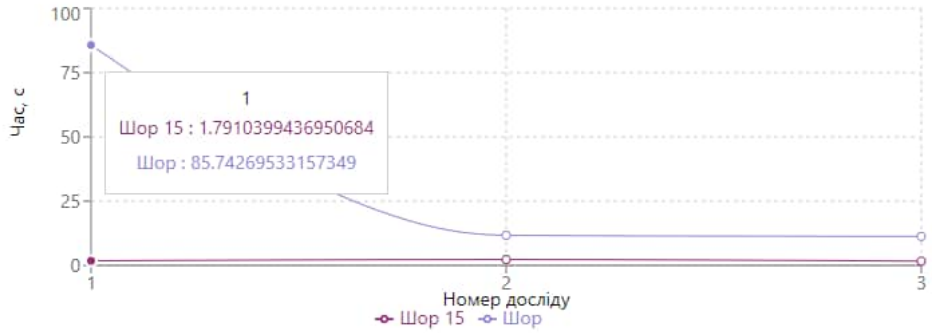
Номер	Алгоритм	Число	Результати	Час
1	Поллард	15	5, 3	0.000
2	Поллард	15	5, 3	0.000
1	Ератосфен	15	3, 5	0.000
2	Ератосфен	15	3, 5	0.000
1	Шора 15	15	3, 5	2.888

Рис. 4.6. Розкладання на множники числа 15

Всі алгоритми правильно розклали число 15 на множники, проте одразу помітно, що навіть для такого невеликого числа, як 15, класичні алгоритми працюють набагато швидше. На жаль, квантовий емулятор має обмеження в кількості кубітів, тому провести дослідження квантового алгоритму Шора на більших числах не вийде.

Спробуємо проаналізувати власноруч написаний алгоритм Шора для числа 15 з алгоритмом, який пропонується пакетом Qiskit. Власноруч написаний алгоритм працює швидше, ніж алгоритм QisKit, проте він створений для числа 15 і не зможе опрацювати більші числа. Алгоритм Qiskit успішно розклав на множники число 21, проте це зайняло аж 85 секунд. Тестування проводилося локально, адже розгорнута система має обмеження запиту в 30 секунд. При наведенні на графік можемо побачити більш детальну інформацію про кожен запуск. На рис. 4.7

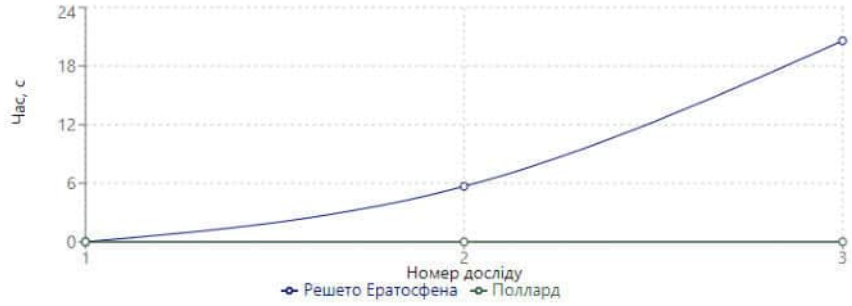
зображено тестування квантових алгоритмів, тут в легенді графіку «Шор 15» – це власноруч розроблений алгоритм Шора, а «Шор» - алгоритм з пакету QisKit:



Номер	Алгоритм	Число	Результати	Час
1	Шора	21	3, 7	85.743
2	Шора	15	3, 5	11.715
3	Шора	15	3, 5	11.282
1	Шора 15	15	3, 5	1.791
2	Шора 15	15	3, 1	2.308

Рис. 4.7. Результат тестування квантових алгоритмів

Розглянемо детальніше класичні алгоритми. Для цього використаймо добутки більших простих чисел і запустимо класичні алгоритми для них. На рис. 4.8 наведено результати роботи програми з класичними алгоритмами:



1	Поллард	15	5, 3	0.000
2	Поллард	12616679	3547, 3557	0.001
3	Поллард	48867617	23, 2124679	0.008
1	Ератосфен	15	3, 5	0.000
2	Ератосфен	12616679	3547, 3557	5.679
3	Ератосфен	48867617	23, 2124679	20.586

Рис. 4.8. Результати роботи класичних алгоритмів

Тут розглянуто просте число Вольстенхольма 2 124 679 помножене на 23, а також добуток простих чисел 3547 і 3557. З результатів зображених на рисунку 4.8

помітно, що ро-алгоритм Полларда працює дуже швидко навіть для таких великих чисел. А от час роботи алгоритму з використанням решета Ератосфена значно зріс. Число 48 867 617 алгоритми факторизували за 0.008 і 20.568 секунд. З графіку також помітно, як зростає час роботи алгоритму з решетом Ератосфена.

На рис. 4.9 бачимо, що час роботи ро-алгоритму Полларда зріс аж в числі з 17 десятковими цифрами:

Номер	Алгоритм	Число	Результати	Час
1	Поллард	15	5, 3	0.000
2	Поллард	12616679	3547, 3557	0.001
3	Поллард	48867617	23, 2124679	0.008
4	Поллард	4886761748867617	17, 23, 5882353, 2124679	0.017
5	Поллард	48867617488676180	5, 4, 2443380874433809	36.580

Рис. 4.9. Результати виконання ро-алгоритму Полларда

Проте це залежить від числа. На рис. 4.10 бачимо, що з іншим числом алгоритм справився за 5 секунд:

5	Поллард	48867617488676180	5, 4, 2443380874433809	36.580
6	Поллард	15143343213312720	5, 72, 2, 7, 3004631589943	5.048

Рис. 4.10. Ро-алгоритм Полларда для чисел з 17 цифрами

Варто зазначити, що час виконання тут вимірюється безпосередньо для самого алгоритму. Час отримання відповіді від сервера не враховано. Тому швидкість інтернету не створює похибки вимірювань.

Під час розробки функціоналу враховано евристики Нільсена. Зокрема «Попередження помилок» [8]. Під час завантаження файлу дозволено обирати тільки файли з розширенням «.qw_lnu». Також всі елементи інтерфейсу, з якими взаємодіє користувач мають певні обмеження. Користувачеві доступний вибір тільки реалізованих у програмі алгоритмів. Для вибору потрібно лише обрати один з алгоритмів в меню, а не друкувати вручну. Крім цього в поле для введення числа можна заносити тільки цифри.

Також враховано евристику, яка говорить про те, що користувачі повинні діагностувати помилку та відновитися після помилок. Тобто повідомлення, яке бачить користувач після виникнення помилки, є зрозумілим і добре описує проблему. Наприклад, якщо користувач завантажує пошкоджений файл, і веб-застосунок не може його правильно прочитати, з’являється червоне повідомлення з інформацією про те, що не вдалося завантажити файл і посиланням на місце, де саме у файлі є проблема. Після цього користувач може спробувати завантажити інший файл.

Крім цього використано евристику «Послідовність та стандарти». Всі дії, які може виконати користувач розташовані в правильній черговості: тобто спочатку завантажуємо старі дані, якщо вони є, тоді обираємо тип алгоритму, сам алгоритм, вводимо число і натискаємо на зелену кнопку «Запустити». Нижче наведено отримані результати і в самому кінці кнопка для збереження результатів. Для завантаження та зберігання файлів використано стандартні іконки.

ВИСНОВКИ

Розв’язок задачі розкладання на множники великих цілих чисел є серйозним викликом, який зумовив глибокі наукові дослідження, що залежать і від фізичної реалізації квантового комп’ютера. На жаль, існуючі квантові системи ще не здатні забезпечити надійні обчислення, оскільки є недостатньо контрольованими і залежать від багатьох чинників.

У магістерській роботі було реалізовано квантовий алгоритм Шора для числа 15, та проведено порівняння часу його роботи з алгоритмом Шора з бібліотеки QisKit та класичними алгоритмами з використанням решета Ератосфена та ро-алгоритмом Полларда. Час отримання результатів на квантовому емуляторі значно перевищив час роботи алгоритмів для класичного комп’ютера. Також квантові емулятори мають обмеження в кількості кубітів, і не дозволяють розкладати на множники числа більші за 21.

Було розгорнуто веб-застосунок, на якому кожен користувач може запустити обраний алгоритм факторизації та проаналізувати отримані результати. Тези роботи з описом програмної реалізації опубліковано на науковій конференції [10].

У розробленому веб-застосунку класичні алгоритми працюють швидше, проте оскільки сьогодні неможливо перевірити роботу алгоритмів на квантовому комп’ютері, то складно реалізувати чітке порівняння з тестами виконаними на класичному комп’ютері.

Можна зробити висновок: хоч алгоритм квантової факторизації Шора теоретично і перевершує будь-який класичний алгоритм, проте залежить від архітектурних особливостей, зокрема реалізації квантового комп’ютера. І лише тоді, коли це стане реальністю, криптографічні системи з відкритим ключем, такі як RSA, можуть опинитися під загрозою.

СПИСОК ЛІТЕРАТУРИ

1. Алгоритм розкладання чисел на множники Шора [Електронний ресурс]. – Режим доступу: https://www.quantiki.org/wiki/shors-factoring-algorithm (дата звернення: 21.05.2022).
2. Беніофф П. Комп'ютер як фізична система: мікроскопічна квантово-механічна гамільтонова модель комп'ютерів, представлена машинами Тьюрінга / П. Беніофф // Журнал статистичної фізики. – 1980. – Вип. 5. – С. 563–591.
3. Вербіцький О. В. Вступ до криптології / О. В. Вербіцькій. – Львів: ВНТЛ, 1998. – 246 с.
4. Карлаш Г. Ю. Квантові інформаційні системи. Навчальний посібник для спеціальності «Прикладна фізика та наноматеріали» / Г. Ю. Карлаш. – Київ: факультет радіофізики, електроніки та комп’ютерних систем Київського національного університету імені Тараса Шевченка, 2018. – 77 с.
5. Козюкалов М. Дослідження та аналіз алгоритмів факторизації чисел / М. Козюкалов, М. Бойко // Мистецтво наукової думки. – 2020. – Вип. 10. – С. 64-68.
6. Крохмальський Т. Є. Вступ до квантових обчислень: навчальний посібник / Т. Є. Крохмальський. – Львів: ЛНУ імені Івана Франка, 2018. – 204 с.
7. Крохмальський Т. Є. Квантові комп’ютери: основи й алгоритми (короткий огляд) / Т. Є. Крохмальський // Журнал фізичних досліджень. – 2004. – Вип. 1. – С. 1–15.
8. Левус. Є. Вступ до інженерії програмного забезпечення / Є. Левус, Н. Мельник. – Львів: Львівська Політехніка, 2018. – 236 с.
9. Ллойд С. Програмуючи Всесвіт. Космос — квантовий комп’ютер / С. Ллойд. – Львів: Книжковий Клуб «Клуб Сімейного Дозвілля», 2006. – 256 с.
10. Лучинська Т. С. Порівняння алгоритмів факторизації для квантового та класичного комп’ютерів / Т. С. Лучинська, Р. Є. Рикалюк // Міжнародна наукова інтернет-конференція «Інформаційне суспільство: технологічні,

економічні та технічні аспекти становлення» – Тернопіль. – 2022. – Вип. 70 – С. 51-54.

11. Оптичний комп'ютер. Вільна енциклопедія [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Optical_computing (дата звернення: 23.05.2022).
12. Остапов С.Е. Квантова інформатика та квантові обчислення / С. Е. Остапов, Ю. Г. Добровольський. – Чернівці: ЧНУ, 2021. – 99 с.
13. Світлик Ю. Про квантові комп’ютери простими словами [Електронний ресурс]. – Режим доступу: https://root-nation.com/ua/articles-ua/tech-ua/ua-pro-kvantovi-kompyuteri (дата звернення: 23.05.2022).
14. Стасюк М. Ф. Деякі підходи до проблем факторизації / М.Ф. Стасюк // Збірник наукових праць п’ятої Міжнародної науково-практичної конференції «Інформаційно-комунікаційні технології в сучасній освіті: досвід, проблеми, перспективи». – Львів, 2017. – С. 84-88.
15. Фейнман Р. Імітація фізики за допомогою комп’ютерів / Р. Фейнман // Міжнародний журнал теоретичної фізики. – 1982. – Вип. 21. – С. 467–488.
16. Хронологія квантових обчислень [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Хронологія_квантових_обчислень (дата звернення 13.11.2022).
17. Хронологія квантових обчислень та комунікацій [Електронний ресурс]. – Режим доступу: https://uk.upwiki.one/wiki/Timeline_of_quantum_computing_and_communication (дата звернення: 22.05.2022).
18. A brief history of quantum computing [Електронний ресурс]. – Режим доступу: https://medium.com/@markus.c.braun/a-brief-history-of-quantum-computing-a5babea5d0bd (дата звернення: 11.11.2022).
19. Hamdi Sh. A Compare between A Compare between Shor's quantum factoring algorithm and General Number Field / Sh. Hamdi. // Computer Science 2014 International Conference on Electrical Engineering and Information & Communication Technology. – 2014.

20. Pollards`s Rho Algorithm for Prime Factorization [Електронний ресурс]. – Режим доступу: https://medium.com/smith-hcv/pollards-rho-algorithm-for-prime-factorization-9002bf71d281 (дата звернення 15.10.2022).
21. Schumacher B. Quantum Coding / B. Schumacher // Phisical Review A (Atomic, Molecular, and Optical Physics). Volume 51, Issue 4. – 1995. – С. 2738-2747.
22. The Nobel Prize In Physics 2022. Press release [Електронний ресурс]. – Режим доступу: https://www.nobelprize.org/prizes/physics/2022/press-release/ (дата звернення 26.11.2022).
23. What is React? W3Schools. Web Development [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/whatis/whatis_react.asp (дата звернення 29.11.2022).

ДОДАТКИ

Додаток А. UML-діаграма діяльності для розробленого веб-застосунку

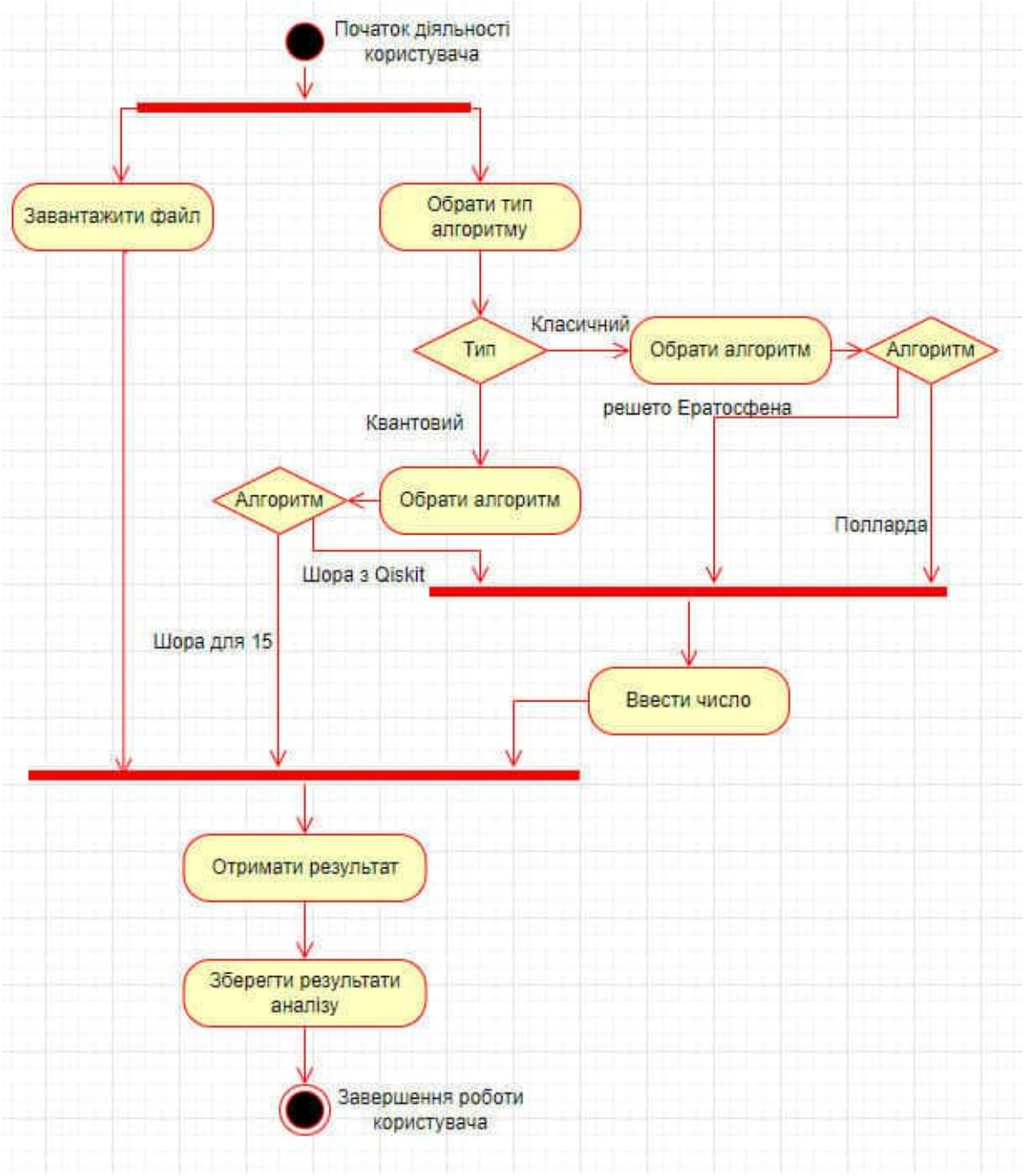


Рис. А.1. Діаграма діяльності

Додаток Б. Приклад сформованого файлу


```
{
  "pollard": [
    {
      "num": 48867617,
      "res": [
        23,
        2124679
      ],
      "time": 0.011964559555053711
    },
    {
      "num": 12616679,
      "res": [
        3547,
        3557
      ],
      "time":
0.0010330677032470703
    }
  ],
  "eratosfen": [
    {
      "num": 48867617,
      "res": [
        23,
        2124679
      ],
      "time": 23.535144805908203
    },
    {
      "num": 12616679,
      "res": [
        3547,
        3557
      ],
      "time": 3.5782878398895264
    },
    {
      "num": 12616679,
      "res": [
        3547,
        3557
      ],
      "time": 3.937382698059082
    }
  ],
  "shor": [
    {
      "num": 21,
      "res": [
        3,
        7
      ],
      "time": 85.74269533157349
    },
    {
      "num": 15,
      "res": [
        3,
        5
      ],
      "time": 11.714815378189087
    },
    {
      "num": 15,
      "res": [
        3,
        5
      ],
      "time": 11.282181978225708
    }
  ],
  "shor15": [
    {
      "num": 15,
      "res": [
        3,
        5
      ],
      "time": 1.7910399436950684
    },
    {
      "num": 15,
      "res": [
        3,
        1
      ],
      "time": 2.308281660079956
    },
    {
      "num": 15,
      "res": [
        3,
        1
      ],
      "time": 1.6610124111175537
    }
  ]
}
```