**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Системного проектування
(повна назва кафедри)

«На правах рукопису»
УДК 004:004.453

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"___" _____2018 р.

# Магістерська дисертація

зі спеціальності (спеціалізації) 122 – комп'ютерні науки та інформаційні
(код і назва спеціальності)

технології (Системне проектування сервісів)

на тему: Адаптивна індексна структура бази даних для точкового пошуку за допомогою підходів машинного навчання

Виконав (-ла): студент (-ка) 6 курсу, групи ДА-62м
(шифр групи)

Михалько Віталій Геннадійович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник зав. кафедри, д.т.н., проф., Петренко А.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Розробка стартап-проекту д.т.н., проф., Петренко А.І. _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

**Національний технічний університет України**
**«Київський політехнічний інститут**
**імені Ігоря Сікорського»**

Інститут/факультет ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 122 – комп'ютерні науки та інформаційні технології (Системне проектування сервісів)
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2018 р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**
Михальку Віталію Геннадійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Адаптивна індексна структура бази даних для точкового пошуку за допомогою підходів машинного навчання

науковий керівник дисертації Петренко Анатолій Іванович, д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження індексні структури в базах даних

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за освітньо-професійною програмою) методи машинного навчання для побудови адаптивних індексних структур для точкового пошуку

5. Перелік завдань, які потрібно розробити _____

1. Дослідження існуючих рішень, їх аналіз і пошук шляхів покращення.
2. Розробка адаптивної індексної структур.
3. Тестування розробленої індексної структури та порівняльний аналіз із існуючими.
4. Розробка стартап-проекту.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу _____ __
презентація на тему «Адаптивна індексна структура бази даних для точкового пошуку за допомогою підходів машинного навчання»

7. Орієнтовний перелік публікацій:
1. Круш І. В., Михалько В. Г. Адаптивний точковий пошук з використанням методів машинного навчання / Круш І. В., Михалько В. Г. // System analysis and information technology: 20-th International conference SAIT 2018. – 2018. – No20. – С. 177-178.

8. Консультанти розділів дисертації*

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Розробка стартап-проекту | Петренко А.І., проф. | | |
| | | | |

9. Дата видачі завдання 01.02.2018

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Строк виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Отримання завдання | 01.02.2018 | |
| 2 | Огляд стану предметної області | 15.02.2018 | |
| 3 | Дослідження теоретичних основ побудови індексних структур | 05.03.2018 | |
| 4 | Аналіз існуючих рішень | 30.03.2018 | |
| 5 | Розробка загальної архітектури адаптивної індексної структури | 10.04.2018 | |
| 6 | Реалізація запропонованої індексної структури | 17.04.2018 | |
| 7 | Тестування адаптивної індексної структури | 30.04.2018 | |
| 8 | Отримання допуску до захисту та подача роботи в ДЕК | 10.05.2018 | |

Студент _____ В.Г. Михалько
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ А.І. Петренко
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника

# РЕФЕРАТ МАГІСТЕРСЬКОЇ ДИСЕРТАЦІЇ

виконаної на тему: Адаптивна індексна структура бази даних для точкового пошуку за допомогою підходів машинного навчання

студентом: Михальком Віталієм Геннадійовичем

Загальний обсяг роботи: 82 сторінки, 16 ілюстрацій, 30 таблиць, перелік посилань із 31 найменування.

**Актуальність теми**

Внаслідок стрімкого зростання обсягів даних, використання традиційних підходів для пошуку інформації у цих даних стає неефективним. Більшість індексних структур, які використовуються для пошуку, були розроблені доволі давно та не враховують реальні розподіли даних.

Темою дослідження є застосування методів машинного навчання для побудови нового класу індексних структур для точкового пошуку, що будуть враховувати особливості даних. Зважаючи на те, що такі структури є адаптивними, вони можуть бути значно ефективнішими з точки зору використання пам’яті.

**Мета та задачі дослідження**

Метою даної роботи є пошук шляхів побудови більш ефективних індексних структур за допомогою підходів машинного навчання. Задачею дослідження є реалізація адаптивної індексної структури для точкового пошуку, що враховує розподіл даних та показує кращі результати по використанню пам’яті в порівнянні з класичними індексними структурами.

**Вирішення поставлених завдань та досягнуті результати**

Було запропоновано два типи адаптивних індексних структур для точкового пошуку з використанням підходів машинного навчання. Одна з них працює на основі лінійної регресії, а інша на основі нейронної мережі з одним прихованим шаром.

Роботу зазначених адаптивних індексних структур було апробовано на двох наборах даних. На одному з них вони показали значно кращі результати по ефективності використання пам’яті в порівнянні з класичними альтернативами, а другому наборі результати виявились дещо гіршими.

**Об’єкт дослідження**

Індексні структури в базах даних

**Предмет дослідження**

Методи машинного навчання для побудови адаптивних індексних структур для точкового пошуку

**Методи дослідження**

Досліджується використання лінійної регресії та нейронних мереж для вирішення задачі вивчення розподілу даних. Розроблене рішення використовує сучасні підходи машинного навчання, методи покращення точності, а також бібліотеки для тренування і застосування моделей.

**Наукова новизна**

В даній роботі, на відміну від попередніх, для побудови адаптивних індексних структур були використані інші моделі машинного навчання, які є більш простими та ефективними в плані обчислень. Зокрема, були використані лінійна регресія та нейронна мережа з одним прихованим шаром. В цілому, адаптивні індексні структури, побудовані на базі цих моделей моделей, є більш ефективними внаслідок меншого ступеня складності. Але, з іншого боку, вони гарно працюють не на всіх розподілах даних.

**Практичне значення одержаних результатів**

Розроблені індексні структури показують більш ефективне використання пам’яті на деяких наборах даних, тому, за умови подальшого вдосконалення, вони можуть бути інтегровані у сучасні бази даних.

Крім цього, отримані результати підтверджують загальну ідею доцільності використання машинного навчання для заміни класичних індексних структур.

**Апробація результатів дисертації**

Результати дослідження оприлюднені на 20-й Міжнародній науково-технічній конференції SAIT 2018.

**Публікації**

Круш І. В., Михалько В. Г. Адаптивний точковий пошук з використанням методів машинного навчання / Круш І. В., Михалько В. Г. // System analysis and

information technology: 20-th International conference SAIT 2018. – 2018. – No20. – С. 177-178.

**Ключові слова**

Індексні структури, хеш-таблиці, машинне навчання, лінійна регресія, нейронні мережі, функція розподілу

# ABSTRACT FOR MASTER'S THESIS

on: Adaptive database index structure for point queries using machine learning approaches

by: Mykhalko Vitaly Hennadiyovych

The thesis contains 82 pages, 16 figures, 30 tables, and 31 references.

**Relevance**

Due to the rapid growth of data volumes, usage of traditional approaches for retrieving information in this data becomes inefficient. Most of the existing index structures were developed a long time ago and they do not take advantage of real world data patterns.

The subject of this work is usage of machine learning techniques for creating a new class of index structures for point queries, which will take into account data patterns. Given that such structures are adaptive, they can be much more efficient in terms of memory usage.

**Purpose**

This work aims to find new ways of building more efficient index structures using machine learning approaches. The research objective is to implement an adaptive index structure for point search that takes into account the distribution of data and shows more efficient memory usage comparing to traditional index structures.

**Results**

Two types of adaptive index structures for point queries were proposed. Both of them use machine learning techniques. One of them works using linear regression, and the other is based on a neural network with one hidden layer.

Proposed adaptive index structures were tested against two different datasets. Both index structures showed more efficient memory usage on first dataset comparing to traditional alternatives. On the other hand, proposed index structures showed less efficient memory usage on second dataset.

**Object of research**

Database index structures

**Subject of research**

Machine learning techniques for building adaptive index structures for point queries

**Research methods**

Linear regression and neural network models are studied and applied for learning cumulative distribution function of data. The developed solution uses modern machine learning techniques, methods for improving accuracy and common libraries for model trainings and applications.

**Scientific novelty**

Comparing to previous works, this one uses another machine learning models to build adaptive index structures. Proposed models are simpler and more efficient in terms of computing. In particular, linear regression and neural network with one hidden layer were used in this work. In general, adaptive index structures based on these models are more efficient due to their simplicity. On the other hand, proposed models do not work well on all datasets.

**Practical value**

The developed index structures show more efficient memory usage on some datasets. In case of further improvements, they can be integrated into modern database systems.

In addition, the results of this work confirm feasibility of using machine learning techniques to replace traditional index structures.

**Approbation of research results**

Research results presented at the 20th International Conference on System Analysis and Information Technology SAIT 2018.

**Publications**

Круш І. В., Михалько В. Г. Адаптивний точковий пошук з використанням методів машинного навчання / Круш І. В., Михалько В. Г. // System analysis and information technology: 20-th International conference SAIT 2018. – 2018. – No20. – С. 177-178.

**Keywords**

Index structures, hash tables, machine learning, linear regression, neural networks, cumulative distribution function.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

СКБД – система керування базами даних

ACID – atomicity, consistency, isolation, durability, атомарність, узгодженість, ізольованість, довговічність

CDF – cumulative distribution function, функція розподілу

ML – machine learning, машинне навчення

ReLU – rectified linear unit, усічене лінійне перетворення

API – application programming interface, прикладний програмний інтерфейс

GPU – graphics processing unit, графічний процесор, відеокарта

PCA – principal component analysis, метод головних компонент

# ВСТУП

З появою обчислювальної техніки та інтернету обсяги даних, з якими доводиться працювати людині, збільшуються експоненційними темпами. Внаслідок зростання популярності технологій Веб 2.0 та соціальних медіа, які заохочують користувачів мережі генерувати власний контент, темпи зростання кількості даних ще більше посилились. За деякими дослідженнями [1], один користувач мережі залишає цифровий слід розміром близько 10 ГБ за один місяць. Якщо врахувати кількість користувачів мережі, то легко побачити, що дані, які створюються за рік, вже вимірюються зетабайтами.

Значна частина згенерованої інформації зберігається в базах даних, які дозволяють описувати характеристики інформації і взаємозв'язки між її елементами. В загальному випадку базою даних можна вважати будь-який впорядкований набір даних. У сучасних інформаційних системах для забезпечення роботи з базами даних використовують системи керування базами даних (СКБД). СКБД – це система, що заснована на програмних та технічних засобах, яка забезпечує визначення, створення, маніпулювання, контроль, керування та використання баз даних.

Одна з основних вимог, яка ставиться перед СКБД – це можливість швидкого і ефективного отримання даних. Для забезпечення виконання цієї вимоги розробники зазвичай звертаються до індексних структур. Наразі існує широкий вибір таких структур, які задовольняють різні потреби по доступу до даних в залежності від поставленої задачі. Наприклад, якщо потрібен швидкий точковий пошук по ключу, то найкращі результати покаже хеш-таблиця. Для діапазонних запитів загальноприйнятим рішенням є використання Б-дерева. Для виконання перевірки на наявність запису в множині, зазвичай використовують індекс існування, такий як фільтр Блума. Завдяки важливості використання індексів у системах управління базами даних та інших застосунках, за останні кілька десятиліть вони були ретельно налаштовані та покращенні, щоб бути ефективними у плані споживання пам'яті, кеш-пам'яті та ресурсів центрального процесора.

Тим не менш, більшість із цих індексів були винайдені ще у 1960-ті роки і вони залишаються структурами даних загального призначення. У них припускається, що дані розподілені найгіршим чином, і, внаслідок цього, не враховуються закономірності та особливості реальних даних. При цьому, в реальних даних часто прослідковуються деякі загальні правила, використання яких може значно пришвидшити доступ до цих даних і зробити його більш ефективним з точки зору використання ресурсів. Наприклад, якщо постає задача доступу до даних з ключами фіксованої довжини у вигляді неперервних цілих чисел (наприклад, числа від 1 до 100 000 000), то для цього не потрібно будувати хеш-таблицю або бінарне дерево над ключами, оскільки самі ключі можуть бути використані як зміщення, що робить отримання даних значно швидшим.

Звичайно, в більшості реальних випадків розподіл даних не співпадає з певним відомим шаблоном, і не є таким легким для виявлення, як у прикладі вище. При цьому, розробка спеціалізованого індексу для кожного окремого випадку може бути доволі затратною процедурою, адже це потребує прямої участі інженерів, які б досліджували і аналізували ці дані. Проте, якби ми могли навчити модель, яка відображає шаблони у даних, кореляції тощо, то можна було б автоматично синтезувати індексну структуру для кожного окремого випадку. Така структура називається навченим індексом, і внаслідок того, що у ній враховуються особливості розподілу даних, вона потенційно може покращити швидкість і ефективність пошуку.

Нещодавно було проведено одне з перших досліджень [3], яке показало, що в деяких випадках навчені індекси можуть бути значно ефективнішими за класичні індексні структури. Найкраще це проявляється, коли необхідно будувати індекси для дуже великої кількісті записів, адже в навчених індексах використовуються методології машинного навчання, яке у свою чергу показує найкращі результати саме при наявності значного набору тестових даних для тренування моделей.

З іншого боку, при проектуванні індесних структур треба також враховувати те, як вони працюють на сучасному апаратному забезпеченні і які взагалі перспективи розвитку у апаратного забезпечення. Однією з основних тенденцій, що прослідковується в останні роки є те, що закон Мура для центральних процесорів вже

практично не працює, а для графічних процесорів передбачається зростання в продуктивності приблизно в 1000 разів до 2025 року [4]. В свою чергу, варто зауважити, що більшість алгоритмів машинного навчання найкраще пристосовані саме до роботи на графічних процесорах, тому в найближчі роки їх використання буде все більш і більш ефективним.

Метою даної роботи є побудова та дослідження адаптивних індексних структур для точкового пошуку за допомогою технологій машинного навчання, що можуть бути використані у сучасних базах даних. Однією з основних цілей, що ставиться перед цією структурою, є більш ефективне використання пам’яті в порівнянні з класичною хеш-таблицею, що є загальноприйнятим вирішенням задачі точкового пошуку. Результатом роботи є декілька варіантів реалізації адаптивної індексної структури для точкового пошуку з використанням різних моделей машинного навчання, проведення експериментів по роботі цієї структури, а також аналіз ефективності розробленої структури і її порівняння з класичним аналогом.

# 1 ОГЛЯД ІСНУЮЧИХ ІНДЕКСНИХ СТРУКТУР І ПІДХОДІВ ДЛЯ ТОЧКОВОГО ПОШУКУ

Точковий пошук – це пошук необхідних даних по ключу, без врахування порядку, в якому розміщені ці дані. Найпростішим способом пошуку є повний перебір всіх даних, але в такого алгоритму лінійна складність, і він зазвичай може бути прийнятним тільки для маленьких наборів даних. Коли необхідно забезпечити швидкий пошук по ключу для великої кількості інформації, то як правило використовують класичні індексні структури, такі як Б-дерева або хеш-таблиці.

## 1.1 Класичні індексні структури

В системах управління базами даних в більшості випадків використовуються саме Б-дерева (рис. 1.1), які окрім точкових запитів можуть також успішно працювати і з діапазонними запитами. Але в тих ситуаціях, коли достатньо точкового пошуку по ключу (як прямого, так і при виконанні JOIN-операцій) і порядок, в якому розміщені дані є неважливим, хеш-таблиці показують кращу швидкість і ефективність в порівнянні з деревами, адже швидкість пошуку в хеш-таблиці є константною (або O(1) в O-нотації), в той час як бінарні дерева забезпечують доступ до даних за логарифмічний час.

Основна різниця між Б-деревами і хеш-таблицями полягає в тому, які операції проводяться над ключами. Зокрема, в Б-деревах основною операцією є порівняння ключів між собою на предмет того, який з них передує іншому, тобто для ключів має бути визначений порядок. В хеш-таблицях основною операцією є обчислення хеш-значення ключа за допомогою спеціальної хеш-функції.

При виборі оптимальної структури даних для швидкого пошуку треба також враховувати і тип ключів. Зокрема, для числових ключів обчислення хеш-значення є доволі швидким, в той час як для ключів, що є довгими текстовими рядками, обчислення хеш-значення може бути значно повільнішою операцією.

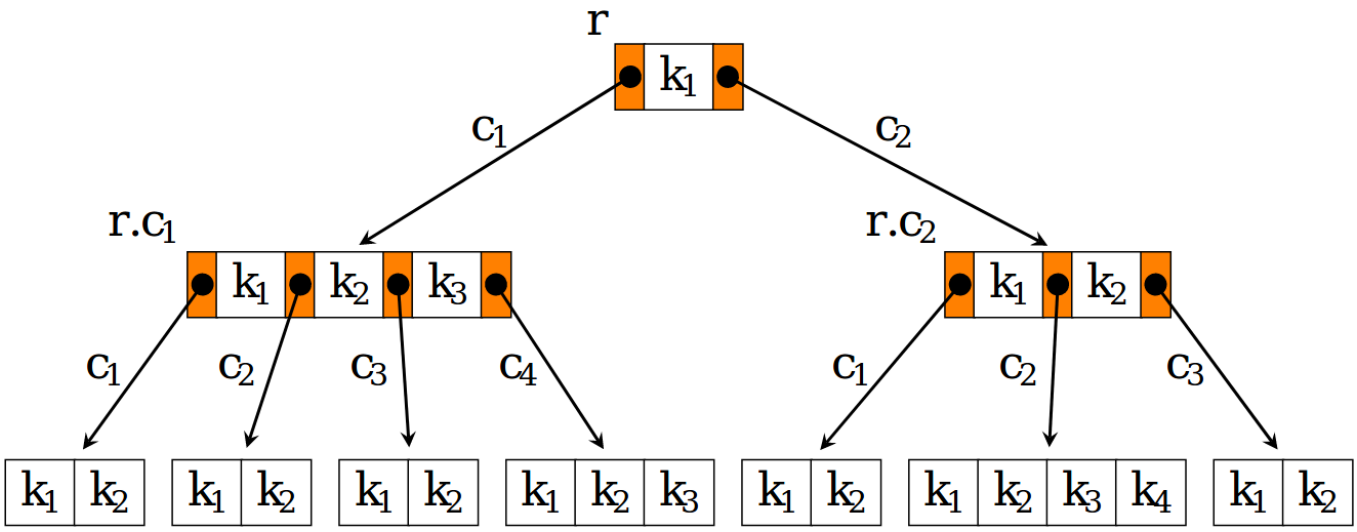


Рисунок 1.1 – Б-дерево [5]

## 1.2 Хеш-таблиці

В обчислювальних технологіях хеш-таблиця – це структура даних, яка реалізує абстрактний тип даних асоціативного масиву, тобто структуру, яка може ставити у відповідність ключам певні значення [6, 7]. В основі хеш-таблиць лежить хеш-функція для обчислення індексу в масиві слотів, з яких можна знайти потрібне значення.

В ідеальному випадку хеш-функція ставить у відповідність кожному ключу унікальний слот, але на практиці таку функцію знайти дуже складно. Внаслідок цього більшість хеш-таблиць використовують недосконалу хеш-функцію, що може призвести до повторень хеш-коду, коли хеш-функція генерує один і той же індекс для декількох ключів. Така ситуаціє називається колізією, і для її обробки необхідно застосовувати додаткові техніки, що в свою чергу уповільнює роботу хеш-таблиці.

У гарно спроектованій хеш-таблиці середня складність (кількість інструкцій) для кожного пошуку не залежить від кількості елементів, що зберігаються в таблиці. Тобто вона є константною, або O(1) в нотації складності роботи алгоритмів. Зазвичай, хеш-таблиці також дозволяють довільні вставки та видалення пар ключ-значення, при амортизованій константній середній складності на одну операцію.

В багатьох випадках хеш-таблиці показали себе в середньому більш ефективними, ніж дерева пошуку або будь-які інші пошукові структури даних. З цієї причини вони широко використовуються в програмному забезпеченні, зокрема для асоціативних масивів, індексування баз даних, кеш-пам'яті та операцій з множинами.

Ефективність роботи хеш-таблиці багато в чому залежить від вибору гарної хеш-функції. Базовою вимогою до хеш-функції є забезпечення рівномірного розподілу вихідних значень в незалежності від вхідних. Нерівномірний розподіл збільшує кількість колізій і вартість операцій для їх вирішення. Рівномірність часто буває складно забезпечити для всіх випадків, але зазвичай підходить і функція, яка проходить відповідні статистичні тести.

При цьому, хеш-функції мають бути також швидкими у роботі, внаслідок чого для хеш-таблиць небажано використовувати стандартні хеш-функції з криптографії, хоча, з іншого боку, вони часто забезпечують доволі гарну рівномірність розподілу вихідних значень.

Іншим важливим аспектом з при проектуванні хеш-таблиці є розв’язання колізій. Для цього існує два основних метода:

- метод ланцюжків
- метод відкритої адресації

### 1.2.1 Розв’язання колізій за допомогою методу ланцюжків

Суть методу ланцюжків полягає в тому, що кожна комірка масиву слотів є вказівником на зв'язаний список (ланцюжок) пар ключ-значення, що відповідають одному і тому ж хеш-значенню ключа. Колізії призводять до того, що з'являються ланцюжки довжиною більше одного елемента. Операції пошуку або видалення елемента вимагають перебору всіх елементів відповідного ланцюжка, щоб знайти в ньому елемент з заданим ключем. Для додавання нового елемента необхідно додати елемент в кінець або початок відповідного списку, і, у випадку якщо коефіцієнт заповнення стане занадто великим, збільшити розмір масиву і перебудувати таблицю.

Схема роботи методу ланцюжків зображена на рис. 1.2

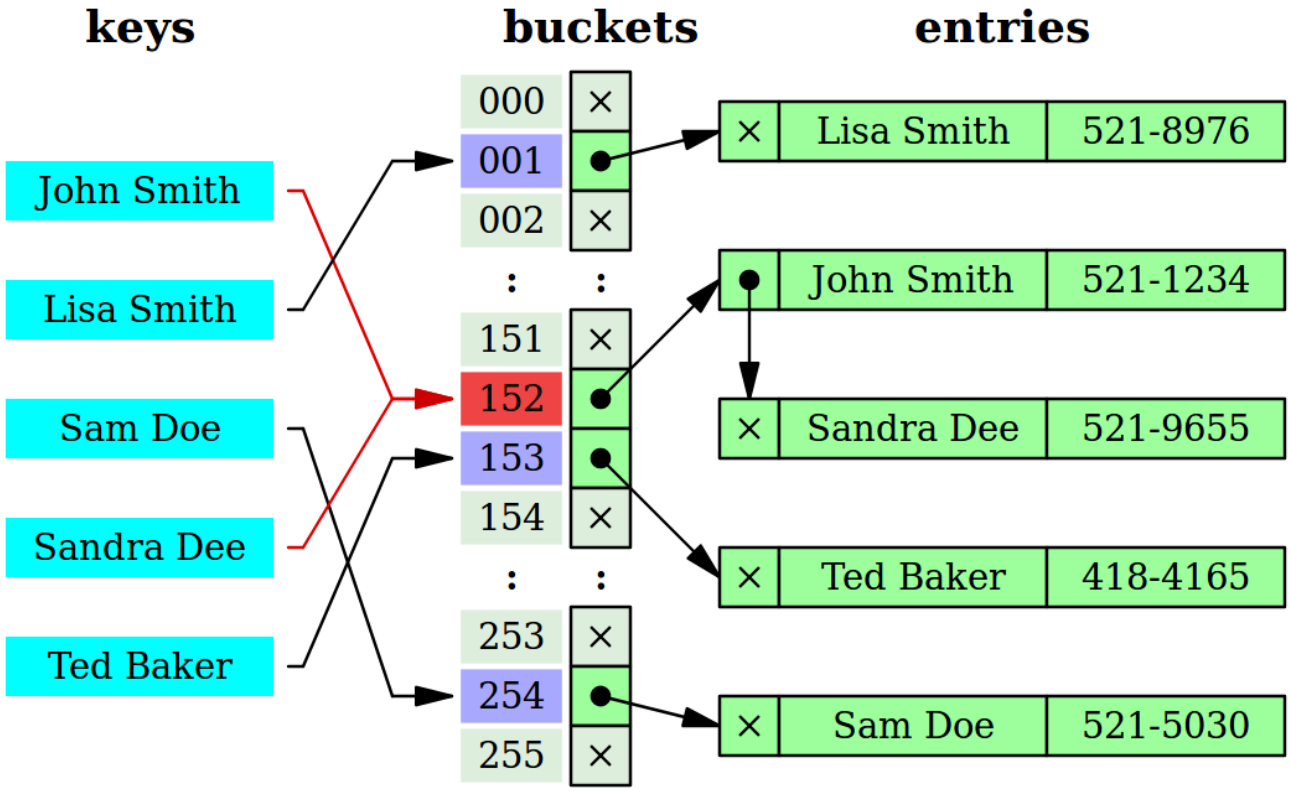


Рисунок 1.2 – Схема роботи хеш-таблиці з використанням методу ланцюжків [8]

### 1.2.2 Розв’язання колізій за допомогою методу відкритої адресації

В стратегії, що називається відкритою адресацією, всі записи зберігаються в самому масиві зі слотами. Коли додається новий запис, масив перевіряється в певному порядку, доки не буде знайдена вільна комірка, куди буде доданий новий елемент. У випадку пошуку елемента, масив сканується в тій самій послідовності, доки потрібний запис або порожня комірка не буде знайдена. При цьому, якщо знайдена порожня комірка – це означає відсутність заданого елемента. Назва «відкрита адресація» вказує на те, що положення (адреса) елемента визначаються не тільки його хеш-значенням. Цей метод також іноді називають закритим хешуванням.

В загальному, послідовність, в якій переглядаються комірки хеш-таблиці, залежить тільки від ключа елемента. Тобто це послідовність h0(x), h1(x), …, hn-1(x), де x – ключ елемента, а hi(x) – довільна функція, яка порівнює кожен ключ комірки з хеш-таблицею. Перший елемент послідовності дорівнює значенню деякої хеш-функції від ключа, а інші обчислюються відносно першого, зазвичай, одним із таких способів:

- лінійне зондування
- квадратичне зондування
- подвійне хешування

Схему роботи хеш-таблиці з використанням методу відкритої адресації показано на рис. 1.3.

Для успішної роботи алгоритму пошуку, послідовність перебору має бути такою, щоб всі комірки хеш-таблиці виявились переглянутими рівно по одному разу.

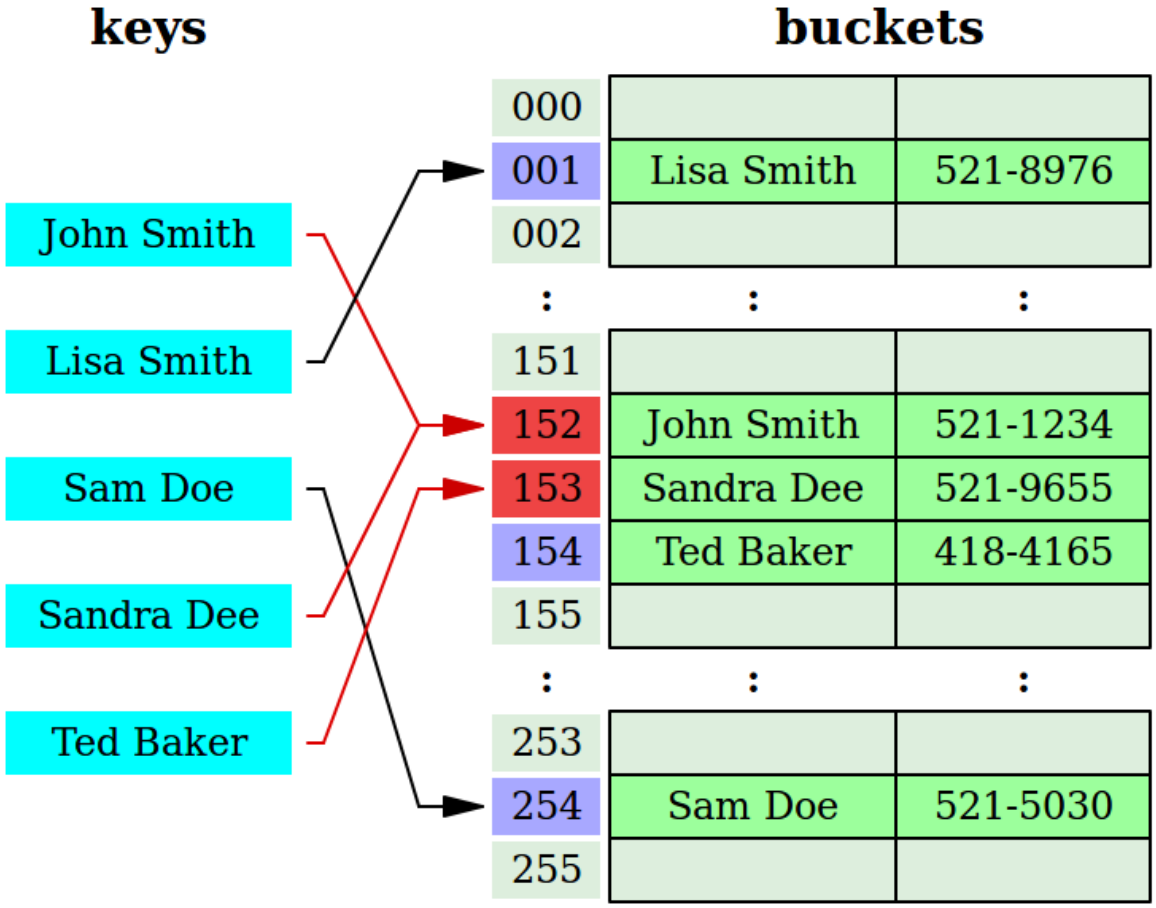


Рисунок 1.3 – Схема роботи хеш-таблиці з використанням методу відкритої адресації [9]

Внаслідок колізій, важливою статистичною метрикою для хеш-таблиць є фактор завантаженості. Він визначається кількістю зайнятих слотів, поділеною на загальну кількість слотів у таблиці. Коли збільшується фактор завантаженості, хеш-таблиця стає працювати повільніше внаслідок збільшення ймовірності того, що обчислене значення хеша для нового ключа буде вказувати на слот, в якому вже є

елементи. Відповідно збільшується як час вставки такого ключа в таблицю, так і час його пошуку

## 1.3 Використання індексних структур в базах даних

Як було зазначено вище, для того щоб пришвидшити пошук і отримання інформації з бази даних, використовуються індексні структури. В загальному випадку, індекс в базі даних – це структура, що дозволяє швидше отримувати дані з заданої таблиці за рахунок виконання додаткових операцій запису в базу і використання більшої кількості пам’яті [10].

Використання індексних структур в базах даних залежить в першу чергу від того, для яких задач була створена база даних, і отримання якого типу даних треба пришвидшити.

Зокрема, класичні реляційні системи керування базами даних зазвичай встановлюють на один окремий сервер, і основними вимогами, які ставляться перед цими системами, є надійність, конзистентність і підтримка ACID-транзакцій. Прикладами таких баз даних є MySQL, PostgreSQL, Oracle. Дані в таких системах зберігаються на жорсткому диску, тому при використанні індексів треба враховувати особливості роботи зчитування/запису з жорсткого диску, а також використання кешування. Основний тип індексу, які прийнято використовувати в таких базах, це B-дерева.

Іншим класом систем для зберігання даних є такі, що зазвичай зберігають інформацію у вигляді ключ-значення в оперативній пам’яті. До них належать Redis, Memcached. В них основною структурою даних для швидкого пошуку є хеш-таблиця

Також, в останні часи значної популярності набули розподілені системи керування базами даних, такі як Cassandra, MongoDB, HBase та інші. В них для швидкого доступу до інформації використовуються спеціалізовані алгоритми.

### 1.3.1 Класифікація індексів в базах даних

Індекси в базах даних можна розрізняти за декількома критеріями. Зокрема, за архітектурою вони поділяються на кластерні і некластерні [11]. За типом індекси розрізняють на щільні, розріджені, а також зворотні.

В кластерному індексі ключі розміщені в такому ж порядку як і записи в таблицях, тому в однієї таблиці може бути тільки один кластерний індекс, який зазвичай будується на основі первинного ключа в таблиці. Кластерні індекси можуть значно пришвидшити отримання записів з БД у випадку, коли запитуваний фізичний порядок цих записів у базі збігається з тим, що зазначено у запиті. Це відбувається внаслідок того, що дані з жорсткого диску зчитуються блоками, в яких зазвичай розміщені багато послідовних записів.

Для некластерних індексів дані в таблиці можуть бути розміщені в будь-якому порядку, адже в індексній структурі зберігаються повні адреси записів. Некластерні індекси зазвичай створюються для колонок, які не виступають в ролі первинних ключів і таких індексів може бути декілька для таблиці.

Різниця між щільними і розрідженими індексами полягає в тому, що у щільних індексах зберігаються ключі і адреси для всіх записів, а в розріджених зберігаються адреси на цілі блоки з даними. Розріджені індекси потребують менше пам’яті, але вони можуть бути побудовані тільки для відсортованих даних.

### 1.3.2 Особливості реалізації індексних структур для точкового пошуку в базах даних

Індекси та дані зазвичай розміщені в окремих файлах. Файл з даними може, наприклад, створюватись для кожної таблиці, і для кожного такого файлу з даними може бути декілька файлів з індексами. В індексних файлах містяться ключі (атрибути записів з даними) і посилання на самі записи. Ці ключі часто є відсортованими і формують певну структуру в залежності від типу індексу.

Як було зазначено вище, основними структурами даних, що використовуються для індексів є Б-дерева і хеш-таблиці. При цьому їхні реалізації дещо відрізняються від класичних варіантів, адже вони повинні забезпечувати швидкий доступ до даних, що зберігаються на диску, а не в оперативній пам’яті. Для того, щоб необхідні записи

можна було отримувати швидше, необхідно враховувати принципи роботи зчитування і запису інформації на диск, переміщення її в оперативну пам’ять і кешування. Зазвичай, для оптимальної швидкодії прийнято оптимізовувати саме кількість зчитувань даних з диску, адже ця операція потребує найбільше часу.

Так як при використанні хеш-таблиць в ролі індексів вони зберігаються на жорсткому диску, в алгоритм їхної роботи треба внести деякі модифікації. Зокрема, масив слотів містить спеціальні блоки, а не прості вказівники на списки з елементами. В одному такому блоці зазвичай зберігаються одразу декілька записів, а також метадані. Якщо в блоці виникає переповнення, то створюється ще один блок, щоб можна було зберегти додаткові записи.

Ефективність таких хеш-таблиць залежить від того, чи достатньо є слотів, а також чи достатньо в відповідних блоках місця для елементів. В ідеальному випадку, коли кількість вільних слотів значно перевищує кількість елементів, то для операції пошуку по ключу необхідно здійснити тільки одне зчитування з диску. Але, якщо розмір файлу з індексом зростає, то врешті-решт виникне ситуація, коли буде багато ланцюжків з блоками, і для однієї операції пошуку вже треба буде виконувати декілька послідовних зчитувань з диску.

Наведені вище хеш-таблиці називаються статичними, адже кількість слотів в них залишається однаковою, незалежно від фактичної кількості елементів в таблиці. Однак, існує також декілька видів динамічних хеш-таблиць, в яких кількість слотів може збільшуватись або зменшуватись в залежності від кількості елементів. Найбільш поширеними стратегіями для побудови динамічних хеш-таблиць є розширене хешування та лінійне хешування.

Основні особливості розширеного хешування полягають в наступному:

- використання додаткового шару (масиву) зі слотами, в якому знаходяться посилання на блоки з даними
- розмір масива з посиланнями може збільшуватись
- у декількох слотів в масиві може бути посилання на спільний блок з даними (але для цього в блоці з даними має бути достатня ємність)

- хеш-функція для кожного ключа може обчислити певну задану кількість біт, наприклад “k”, але для нумерації слотів буде використовуватись менша кількість біт, наприклад “i” (взятих на початку чи в кінці повного хеш-значення); при цьому, розмір масиву зі слотами буде дорівнювати $2^i$

Принцип роботи хеш-таблиці з розширеним хешуванням схематично зображено на рис. 1.4.

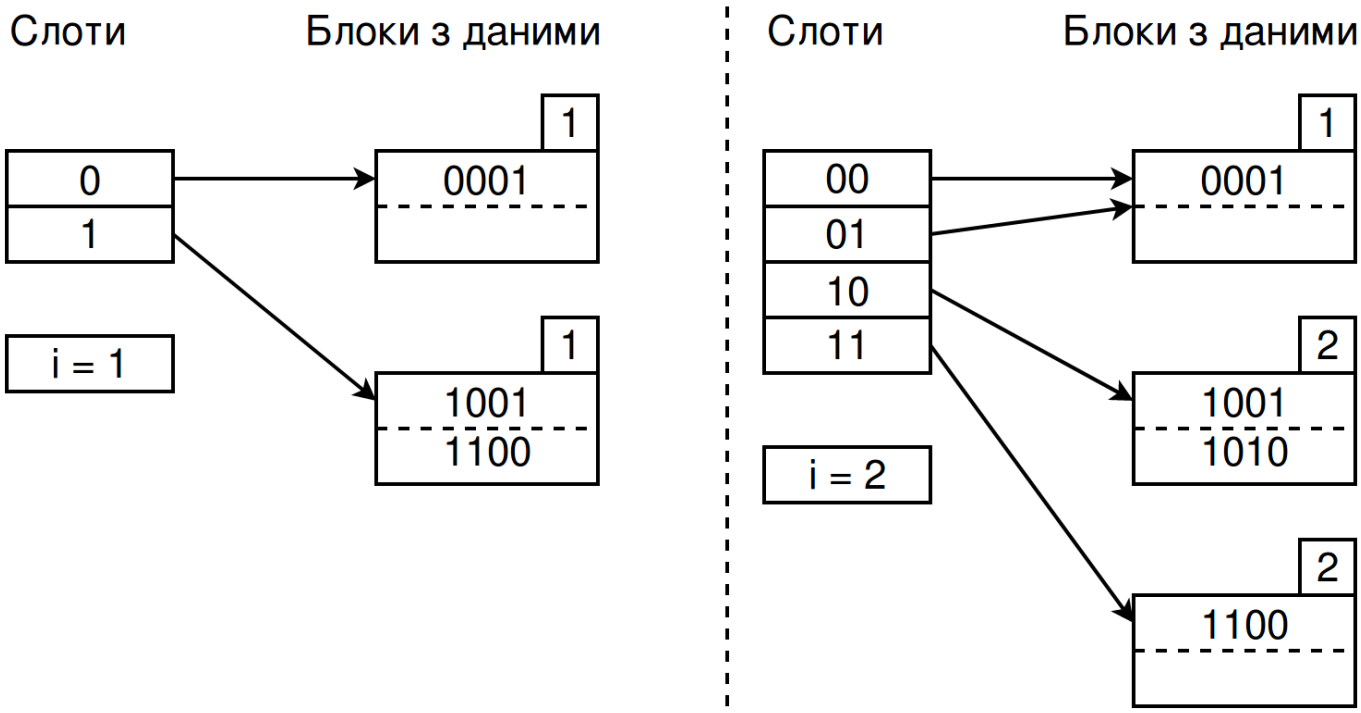


Рисунок 1.4 – Схема роботи хеш-таблиці з розширеним хешуваням. Ліворуч зображений початковий стан хеш-таблиці, праворуч – стан, що утворився після вставки елемента “1010”

У випадку використання стратегії з лінійним хешуванням, розмір таблиці збільшується більш поступово. Основні принципи лінійного хешування наведені нижче:

- розмір масиву зі слотами вибирається таким чином, щоб кількість слотів становила певну частку від фактичної кількості елементів
- дозволяється використовувати ланцюжки блоків
- кількість біт, що використовуються для нумерації масиву зі слотами обчислюється за формулою $\log_2 n$, де n – це поточна кількість слотів

- при вставці нового елемента перевіряється частка вільних слотів; якщо вона менше за задану константу, то кількість слотів збільшується на один

В кожній з цих стратегій є свої переваги та недоліки, але, в цілому, вони працюють більш ефективно за класичний варіант, при умові зберігання хеш-таблиць на диску.

### 1.3.3 Алгоритми розподіленого пошуку

При роботі з великою кількістю даних, одного серверу може виявитись недостатньо. В такому випадку використовують розподілені системи керування базами даних. Ці системи дозволяють розбивати дані на декілька частин (їх зазвичай називають шардами), що будуть зберігатись на окремих серверах. Для того, щоб була можливість виконувати швидкий пошук по цим даним, необхідно використовувати спеціалізовані алгоритми, за допомогою яких можна визначити, на якому сервері знаходяться дані.

Найпростіший варіант вирішення такої проблеми – надіслати запит на всі сервери, і потім зібрати результат в одну колекцію. Але такий варіант не є ефективним, так як вимагає використання ресурсів одразу всіх серверів, незалежно від того, чи містять вони необхідні дані.

Більш розумним підходом до цієї проблеми є розподіл даних по певному ключу (shard key) [12]. В такому випадку можна однозначно визначити, на якому з серверів будуть знаходитись відповідні дані (при умові їх існування). Двома основними стратегіями для такого розподілу є наступні:

- розподіл, базований на інтервалі
- розподіл, базований на хеші

При використанні стратегії розподілу, базованому на інтервалі, необхідно одразу знати приблизний розподіл ключів, щоб була можливість розділити такі ключі рівномірно.

В розподілі, базованому на хеші, в ролі ключа необхідно вибрати такий атрибут даних, в якого є багато різних значень. Зазвичай, в ролі такого ключа використовується первинний ключ і це дозволяє доволі рівномірно розділити дані по різним серверам.

## 1.4 Висновки

Таким чином, основною класичною індексною структурою для точкового пошуку є хеш-таблиця. Для цієї задачі можна використовувати і інші індексні структури, але хеш-таблиця зазвичай є найбільш ефективною.

Однією з основних проблем при використанні хеш-таблиці є колізії. Для їх вирішення існують декілька методів, зокрема метод ланцюжків і метод відкритої адресації. У кожного з цих методів є свої переваги та недоліки, але, в цілому, їх використання збільшує час виконання операцій вставки і витягнення елементів з хеш-таблиці.

При реалізації хеш-таблиці для використання в базі даних треба враховувати додаткові нюанси, що пов’язані з виконанням операцій зчитування і запису на диск, а також особливості функціонування оперативної пам’яті та кеш-пам’яті.

# 2 ПОБУДОВА АДАПТИВНОЇ ІНДЕКСНОЇ СТРУКТУРИ ДЛЯ ТОЧКОВОГО ПОШУКУ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

Як було зазначено в попередньому розділі, основною проблемою, яка виникає при використанні хеш-таблиць, є колізії ключів. Використовуючи схожий принцип, що і в парадоксі днів народжень, можна підрахувати, що при рівній кількості слотів і ключів, математичне сподівання кількості колізій приблизно дорівнює 37% [13]. Це в свою чергу означає, що 37% із виділеної пам’яті для хеш-таблиці буде витрачається марно.

Для кожної з колізій необхідно створити окремий зв’язний список або використовувати метод відкритої адресації. При цьому, при використанні будь-якого з цих варіантів виникне додатковий кеш-промах, що зменшить ефективність точкового пошуку.

Якщо необхідно збільшити швидкість пошуку елементів, то зазвичай кількість слотів роблять значно більшою за фактичну кількість елементів, але в такому випадку зростає кількість пустих слотів, що потребують використання додаткової пам’яті.

При необхідності більш оптимального використання пам’яті, кількість слотів можна зробити меншою за фактичну кількість елементів, але в цьому випадку операції пошуку елементів будуть значно повільнішими.

В описаних вище випадках доводиться знаходити компроміс між швидкістю пошуку та ефективним використанням пам’яті. Проте, більш загальне рішення цієї проблеми може бути знайдено з використанням методів машинного навчання. Наприклад, якщо натренувати модель, яка для кожного значення ключа надавала б унікальну позицію в масиві слотів, то можна було б уникнути колізій. При успішному тренування, ця модель і могла б виступати в ролі адаптивної індексної структури (рис. 2.1)

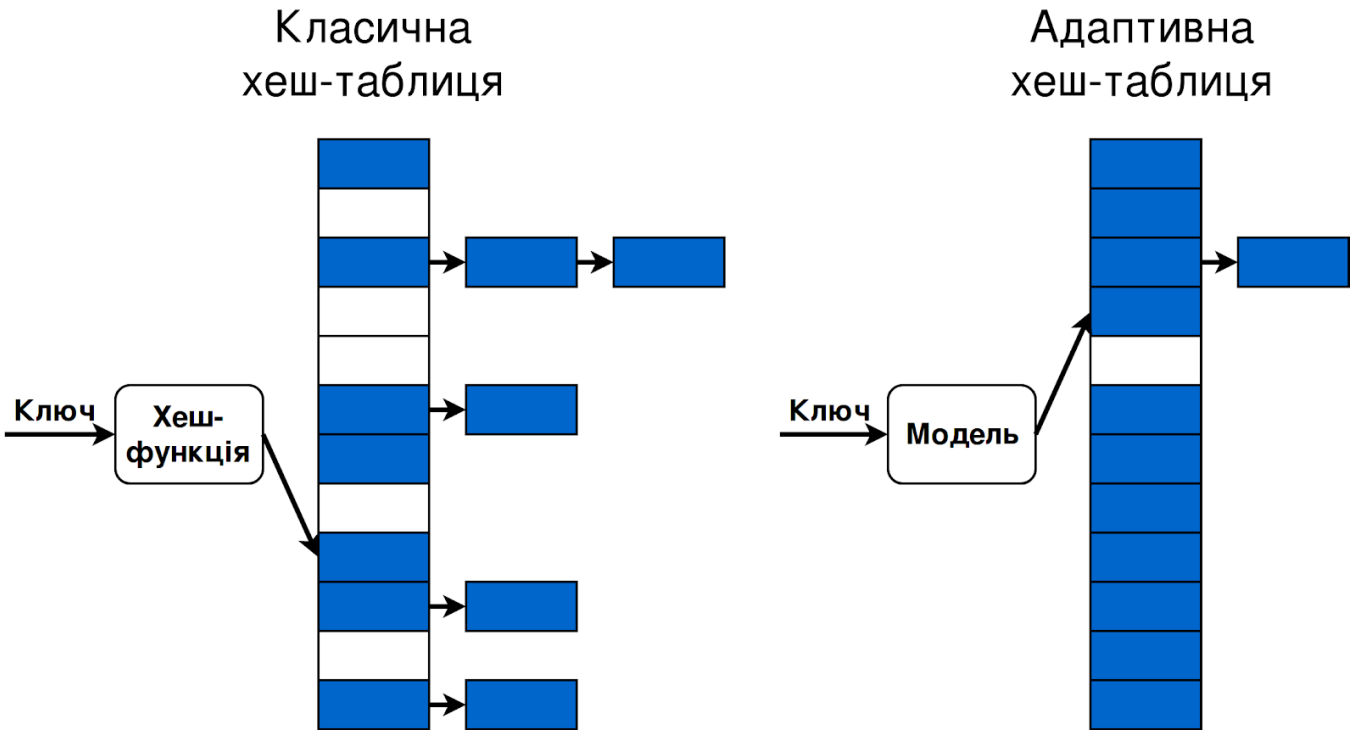


Рисунок 2.1 – Порівняння класичної та адаптивної хеш-таблиць

## 2.1 Використання функції розподілу у ролі хеш-функції

Задача, яка ставиться перед моделлю, полягаю у наступному: для вхідного ключа K необхідно передбачити його позицію у масиві зі слотами. Якщо розмістити ключі у певному порядку (наприклад, від найменшого до найбільшого), і поставити у відповідність кожному ключу позицію в масиві, яка буде дорівнювати порядковому номеру цього ключа, то в результаті це можна описати неспадною функцією. Можливий графік такої функції наведено на рис 2.2.

Легко помітити, що така функція нагадує функцію розподілу (CDF). Внаслідок цього можна спробувати використати функцію розподілу для визначення позиції ключа в масиві:

$$s = F(K) * N \tag{2.1}$$

де s – це передбачене значення позиції ключа в масиві, F(K) – це натренована функція розподілу ключів, яка показує ймовірність того, що позиція випадкового ключа X є меншою за K, а N – це кількість ключів [3].

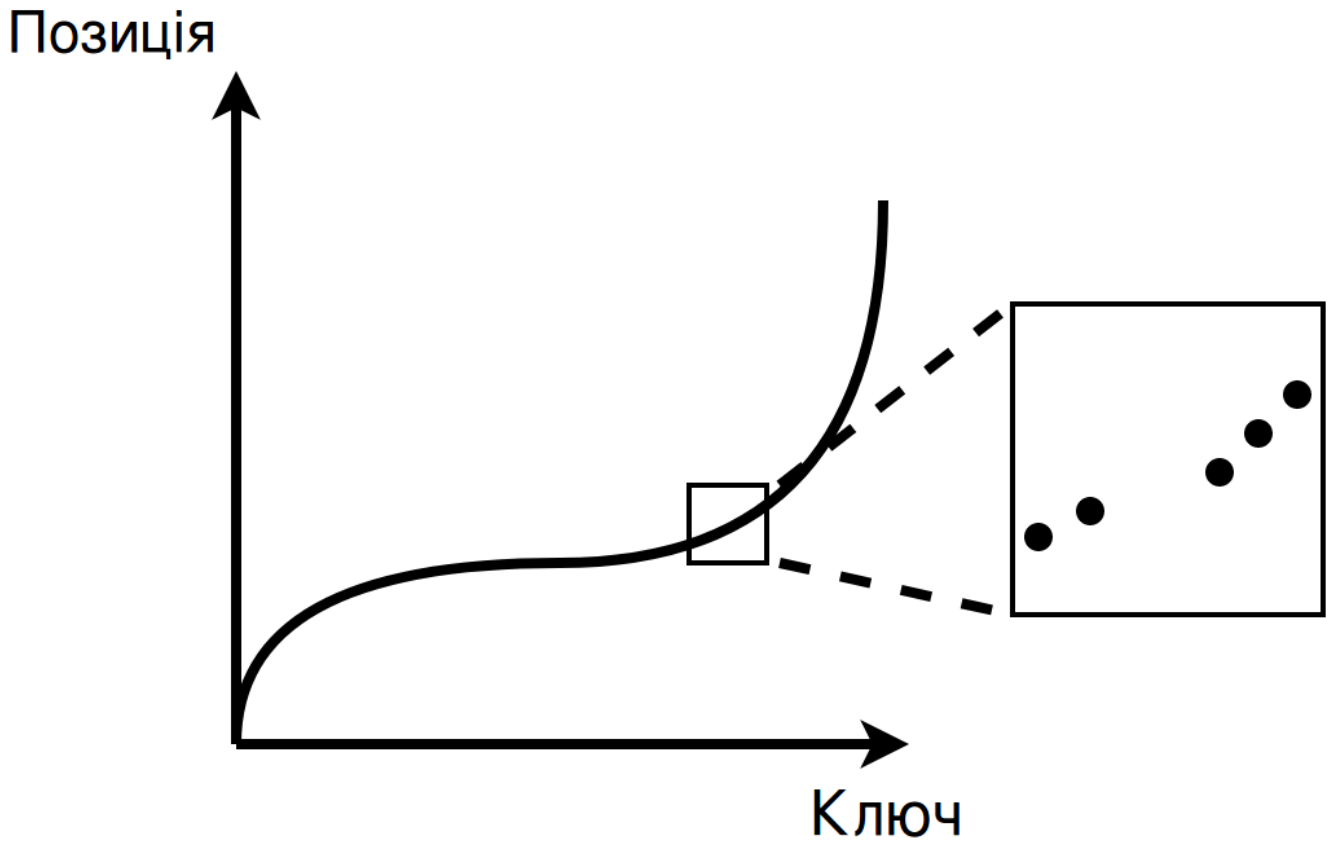


Рисунок 2.2 – Приклад можливого графіку функції розподілу даних

Так як необхідно побудувати адаптивну індексну структуру для точкового пошуку, варто врахохувати, що в масиві слотів ключі не обов’язково мають бути розміщені компактно або у певному відсортованому порядку. Таким чином, можна побудувати таку функцію хешування:

$$h(K) = F(K) * M \tag{2.2}$$

де F(K) – це функція розподілу ключів, а M – це розмір таблиці зі слотами, він може відрізнятись від кількості елементів, що мають бути у ній розміщені.

Якщо запропонована модель повністю вивчить реальний розподіл даних, а кількість слотів в масиві буде не меншою за сумарну кількість ключів, то в результаті не виникне колізій. З іншого боку, якщо навіть і будуть виникати колізії, то для їх вирішення можна використовувати одну зі стратегій, описаних в першому розділі, адже їх реалізація є ортогональною до реалізації самої хеш-таблиці.

При цьому, вставка нових елементів у запропоновану адаптивну структуру будеу працювати схожим чином, як і у звичайних хеш-таблицях: спочатку за

допомогою натренованої моделі h(K) необхідно визначити індекс в масиві зі слотами, а далі перевірити чи є вільним визначений слот, і у разі потреби застосувати стратегію по вирішенню конфліктів.

З точки зору використання пам’яті, дана модель буде працювати краще за класичну хеш-таблицю в тому випадку, якщо вона буде більш рівномірно розміщувати ключі у таблиці. Але варто також враховувати, що навіть при гарному вивченні розподілу даних, при зміні цього розподілу модель необхідно буде перетренувати.

## 2.2 Побудова адаптивної індексної структури на основі лінійної регресії

Одними із базових моделей у машинному навчанні є лінійна регресія, а також похідні від неї. Вони є доволі швидкими у тренуванні та використанні. При цьому, параметри для лінійної регресії можна задати таким чином, що насправді це буде поліномом n-го степеня. Внаслідок того, що ця модель є доволі простою і швидкою, її можна використати для побудови функції розподілу ключів.

### 2.2.1 Лінійна регресія

В лінійні регресії відношення між вхідними і вихідними значеннями даних моделюються за допомогою лінійної функції виду y = ax + b, параметри якої визначаються за допомогою самих даних [14]. В загальному випадку вхідні значення є векторними, тому формула набуває такого вигляду:

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x \tag{2.3}$$

де y – це вихідне значення, яке модель має визначати, вектор x – це вектор вхідних значень, а $\theta$ – вектор параметрів. Під час тренування моделі необхідно визначити такі значення параметрів, щоб мінімізувати помилки. Це зображено на рис. 2.3.

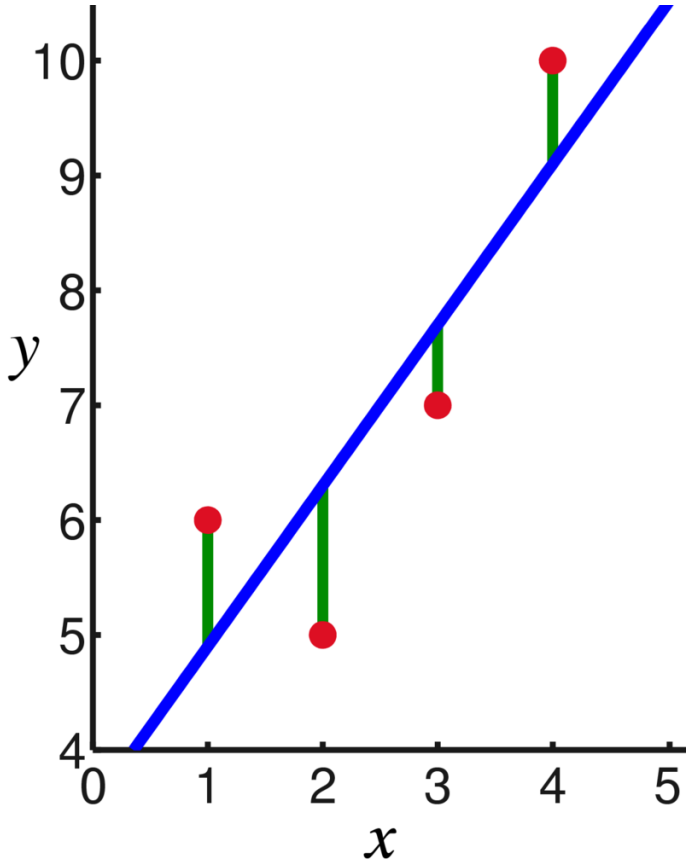


Рисунок 2.3 – Помилки, які враховуються при використанні методу найменших квадратів [15]

Лінійна регресія є однією із найбільш вивчених моделей з багатьма практичними застосуваннями. Це, зокрема, пов'язано і з тим, що в багатьох проблемах моделювання та апроксимації можна припустити, що залежность між вхідними і вихідними значеннями є лінійною. З іншого боку, використовуючи деякі модифікації цього алгоритму, можна також моделювати і нелінійні залежності у даних.

Для того щоб визначити вектор параметрів $\theta$, можна використати метод найменших квадратів. В цьому випадку потрібно мінімузувати суму квадратів помилки між вихідними значеннями, які прогнозує модель, та реальними вихідними значеннями даних. Це можна описати такою функцією:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \quad (2.4)$$

де $J(\theta)$ – функція, що визначає суму квадратів помилок в залежності від вектора параметрів лінійної регресії $\theta$, $x^{(i)}$ та $y^{(i)}$ – це вхідні і вихідні значання для і-го елемента з множини вхідних даних, m – кількість елементів у множині вхідних даних [16].

Для визначення оптимальних значень параметрів $\theta$, функцію $J(\theta)$ необхідно мінімізувати. Це можна зробити, наприклад, одним із градієнтних методів. Але, так як лінійна регресія є простою моделлю, то для вибору оптимальних значень параметрів існує і аналітична формула, яка виглядає таким чином:

$$\theta = (X^T X)^{-1} X^T \vec{y} \tag{2.5}$$

де X – це матриця розміром m*n, що формується з векторів $x^{(i)}$:

$$X = \begin{bmatrix} -(x^{(1)})^T - \\ -(x^{(2)})^T - \\ \vdots \\ -(x^{(m)})^T - \end{bmatrix} \tag{2.6}$$

а y – це вектор довжиною m, що складається зі значень y(i) елементів даних.

Як було зазначено вище, лінійну регресію можна модифікувати таким чином, щоб вона могла моделювати і нелінійні залежності. Роглянемо, наприклад, випадок, коли дані представлені парами значень (x, y), залежність між якими можна описати поліномом третього степеня:

$$y = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0 \tag{2.7}$$

В цьому разі кожне значення x з множини даних можна розширити до вектора $\vec{x} = (x^3, x^2, x, 1)$ і використовувати звичайну лінійну регресію з чотирма параметрами $\theta$.

Описану вище модель можна використовувати і для поліномів вищих порядків, а також для векторних вхідних значень x з множини даних.

### 2.2.2 Побудова індексної структури на основі лінійної регресії

Адаптивна індексна структура для точкового пошуку буде складатись з двох частин:

- хеш-таблиця, в яку при створенні необхідно передати хеш-функцію (патерн стратегія [17])
- хеш-функція, що працює на базі лінійної регресії

Реалізація хеш-таблиця може бути відносно простою, але вона повинна задовольняти наступним вимогам:

- містити базові методи для додавання та отримування елементів з таблиці, такі як put і get.
- мати фіксований розмір, з можливістю вказати його при створенні таблиці.
- визначати хеш-значення ключів за допомогою переданої зовні хеш-функції
- використовувати одну із стратегій для розв'язання колізій (метод ланцюжків або один із методів зондування)
- містити методи для визначення кількості колізій та вільних слотів

Хеш-функція буде працювати на основі натренованої моделі лінійної регресії. З цього випливає, що перед тим, як створювати хеш-таблицю, треба виконати тренування моделі лінійної регресії на множині ключів. Також варто враховувати деякі аспекти, що пов'язані з тренуванням ML-моделей, зокрема те, що процес тренування зазвичай відбувається краще на нормалізованих даних.

Перед початком тренування моделі необхідно поставити у відповідність для кожного ключа позицію в масиві. Ці позиції можуть йти послідовно. Так як значення ключів можуть бути доволі великими, то їх бажано спочатку нормалізувати до проміжку [0; 1]. При цьому, нормалізацію ключів і позицій необхідно також буде враховувати і після тренування, коли дана хеш-функція буде використовуватись в хеш-таблиці.

Іншим важливим питанням є вибір степеню поліному, адже лінійної функції може бути недостатньо для гарної апроксимації більшості розподілів даних. При виборі степеню поліному варто враховувати, що вищий степінь призводить до збільшення часу роботи хеш-функції. Крім цього, підвищення порядку поліному не завжди призводить до більш оптимального використання пам'яті у хеш-таблиці, адже

помилка, яка мінімізується за допомогою методу найменших квадратів, не є еквівалентною до кількості колізій.

Одним із простих алгоритмів, що можна використати для вибору степеню поліному, може бути наступний:

- задати множину можливих значень степенів поліному
- натренувати модель для кожного степеню із множини
- визначити кількість колізій для кожної з натренованих моделей
- у хеш-функції використати модель з найменшою кількістю колізій

## 2.3 Побудова адаптивної індексної структури на основі нейронної мережі

Нейронні мережі є дуже важливим класом моделей у машинному навчанні, що використовуються для вирішення великої кількості задач. Незважаючи на те, що винайдені нейронні мережі були доволі давно, значного поширення вони набули тільки в останнє десятиліття, у зв’язку зі стрімким збільшенням потужностей відеокарт. Однією із найважливіших переваг нейронних мереж є те, що за допомогою них можна апроксимувати практично будь-яку функцію, якою складною вона б не була. У зв’язку з цим нейронні мережі буде доречно використати і для апроксимації функції розподілу даних, щоб на базі цього класу алгоритмів можна було побудувати адаптивну індексну структуру для точкового пошуку.

### 2.3.1 Нейронні мережі

Штучна нейронна мережа (далі просто нейронна мережа) – це обчислювальна система, яка побудована на схожих засадах, що і біологічна нейронна мережа. Вона являє систему собою систему вузлів (нейронів) і зв’язків між ними. Нейрони групуються у декілька шарів, при цьому, в нейронній мережі зазвичай є вхідний шар, вихідний шар, а також можуть бути декілька прихованих шарів. Схематично це зображено на рис 2.4.

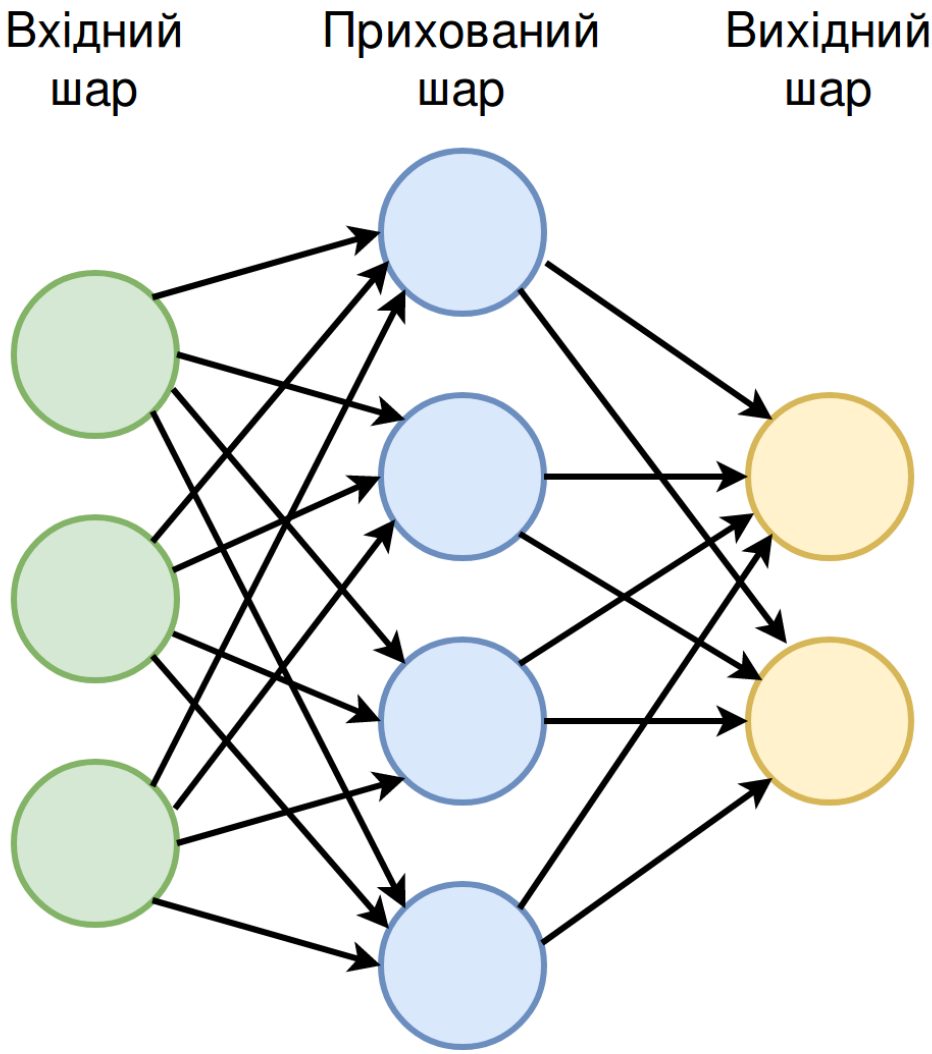


Рисунок 2.4 – Схема нейронної мережі з одним прихованим шаром

Основними компонентами, з яких складається нейронна мережа, є такі [18]:

- нейрони
- з’єднання та ваги
- функція поширення
- правило навчання

Нейрон є базовим вузлом у нейронній мережі. Він складається з множини входів, вектора з ваговими коефіцієнтами для кожного з входів, а також функції активації, яка визначає значення на виході з нейрона, використовуючи зважену суму вхідних значень. Вихідне значення нейрона може бути пізніше використане іншими нейронами. Схему нейрона зображено на рис. 2.5.

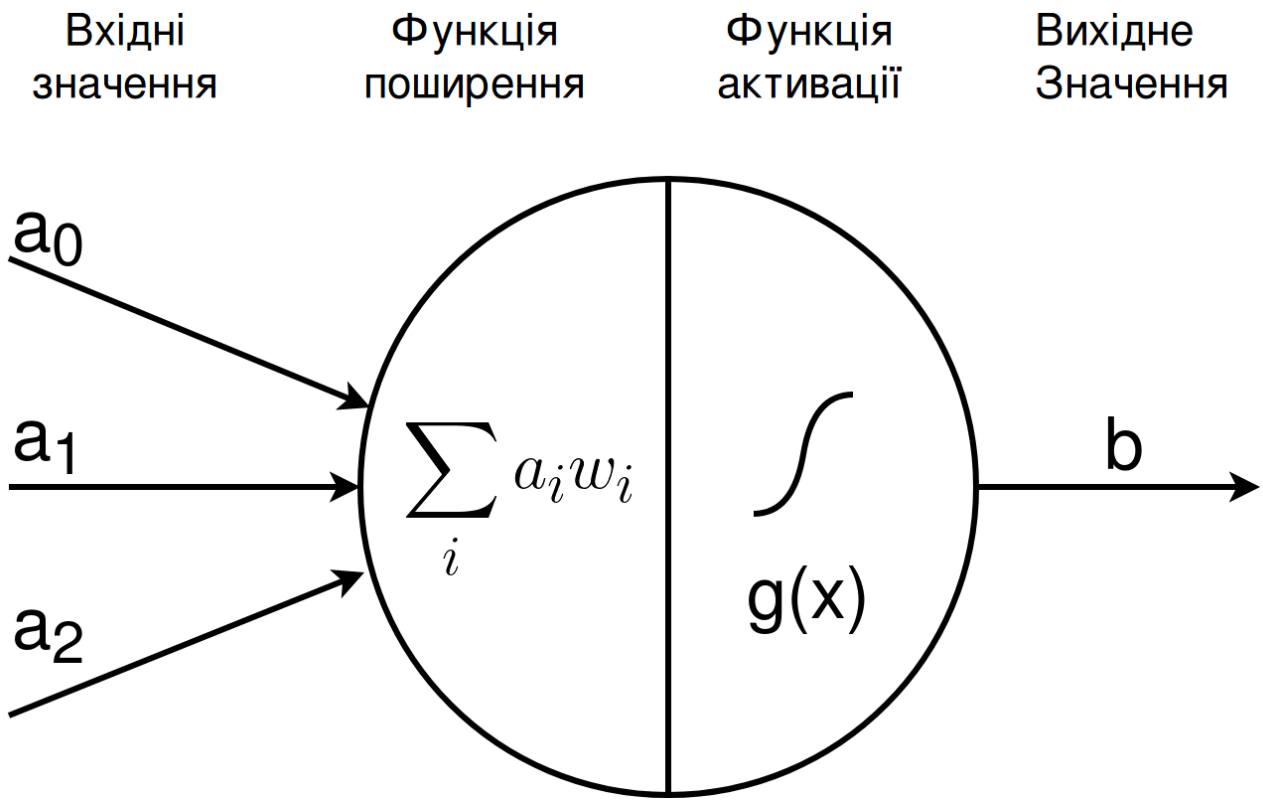


Рисунок 2.5 – Схематичне зображення будови штучного нейрону

Робота нейронної мережі багато в чому залежить від архітектури мережі та вибору функції активації. Зазвичай, в якості функції активації вибирають одну із наступних:

- сигмоїдна функція (логістична)
- гіперболічний тангенс
- ReLU (Rectified Linear Unit)

Сигмоїдна функція має формулу σ (x) = 1 / (1 + $e^{-tx}$), де параметр t визначає її крутизну. Область значень даної функції знаходиться в проміжку (0,1). Однією з основних переваг цієї функції є те, що вона доволі легко диференціюється: при параметрі t = 1, $\frac{d\sigma}{dx}$ = σ * (1 - σ), тобто значення похідної цієї функції можна обчислити через значення самої функції. Така властивість похідної значно полегшує обчислення в алгоритмі зворотного поширення помилки [19].

Функція гіперполічного тангенса також має s-подібний графік, але область її значень лежить в інтервалі (-1; 1). Однією із основних її переваг перед логістичною

сигмоїдною функцію є те, що негативні вхідні значення результують у негативні вихідні значення, а позитивні вхідні – у позитивні вихідні.

ReLU є найбільш поширеною функцією активації на даний момент, адже вона використовується практично у всіх згорткових нейронних мережах, а також при глибинному навчанні [20]. Ця функція має формулу y = max(0, x), а її область значень лежить у проміжку (0; ∞). Одними з основних переваг ReLU-активації перед іншими є те, що нейронні мережі працюють значно швидше, адже значення функції та значення похідної від ReLU обчислюється доволі простими операціями. Крім цього, ReLU-активація більш схожа на біологічні аналоги, а також допомагає вирішити проблему “зникнення” градієнта [21]. Порівняння сигмоїдної функції активації та ReLU показано на рис. 2.6

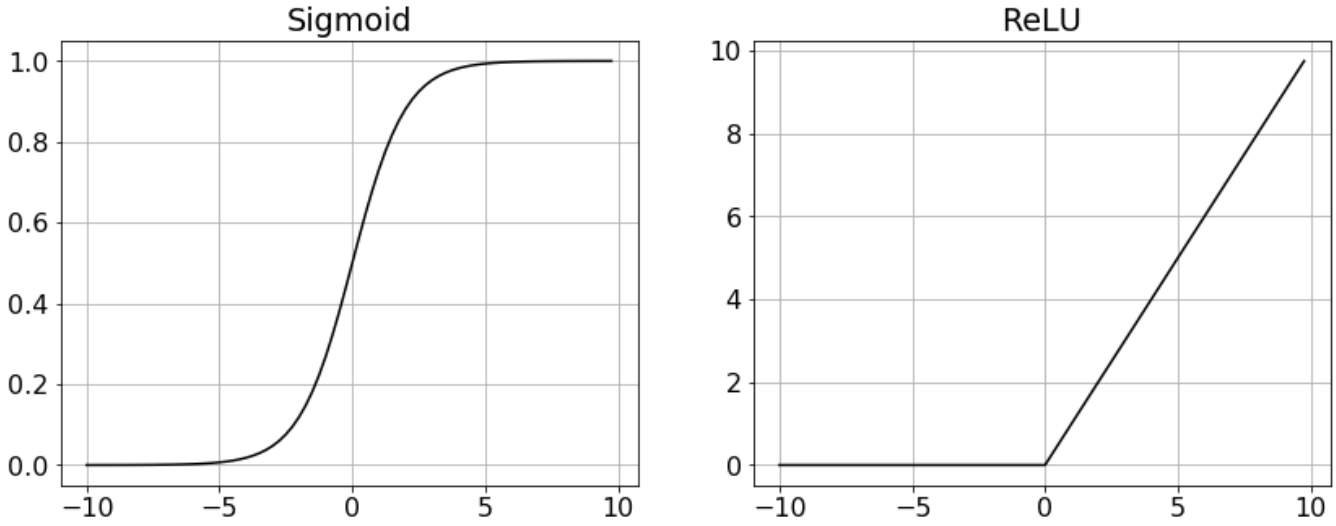


Рисунок 2.6 – Порівняння сигмноїдної функції активації та ReLU

Архітектура мережі визначає яким чином нейрони зв’язані між собою. Двома найбільш поширеними типами архітектури нейронних мереж є наступні:

- нейронна мережа прямого поширення (feedforward neural network)
- рекурентна нейронна мережа (reccurent neural network)

В кожній з зазначених вище архітектур є свої переваги та недоліки, які визначають задачі, для яких їх можна застосувати.

Важливим класом нейронних мереж є також глибокі нейронні мережі, в яких наявні як мінімум два приховані шари. Глибокі нейронні мережі значно складніше

тренувати, але, з іншого боку, вони можуть моделювати та розрізняти доволі складні закономірності у даних.

### 2.3.2 Метод зворотного поширення помилки

Метод зворотного поширення помилки є ефективним градієнтним методом для тренування нейронної мережі. Основна його ідея полягає в тому, що інформація про помилки поширюється від вихідних шарів мережі назад до вхідних, через всі проміжні шари.

Для роботи методу зворотного поширення помилки важливо спочатку означити функцію для визначення помилки, яку зазвичай називають loss-функцією. В нейронних мережах у її ролі зазвичай виступає різниця між передбаченими вихідними значеннями мережі для вхідних даних та очікуваними вихідними значеннями. При цьому, для цієї функції часто використовують суму квадратів помилок для якогось набору елементів із множини вхідних даних.

Даний алгоритм можна поділити на дві основні фази:

- пряме та зворотнє поширення
- оновлення вагових коефіцієнтів мережі

Під час першої фази спочатку послідовно обчислюються вихідні значенняя кожного нейрону. Потім обчислюється помилка між отриманими вихідними значеннями у останньому шарі та очікуваними значенням. Після цього у зворотному шляху обчислюється вплив вагових коефіцієнтів на результат і формується градієнт, елементами якого є часткові похідні вагових коефіцієнтів мережі. Завдяки методу зворотного поширення помилки вказаний градієнт можна обчислити доволі ефективно, адже багато результатів обчислень використовуються повторно.

У другій фазі відбувається оновлення вагових коефіцієнтів мережі з використанням обчисленого градієнту. При цьому, важливим параметром є коефіцієнт швидкості навчання (learning rate), який впливає на те, наскільки сильно змінюються вагові коефіцієнти.

Зазначені вище фази необхідно повторювати до того моменту, поки нейронна мережа не почне передбачувати правильні вихідні значення.

### 2.3.3 Оптимізація за допомогою градієнтних методів

Градієнтний спуск є одним із найбільш популярних методів для виконання оптимізації. Цей метод та його похідні в тому числі активно використовуються і для оптимізації нейронних мереж [22].

Основна ідея градієнтного методу полягає в тому, що для оптимізації цільової функції $J(\theta)$, де $\theta \in R^d$, параметр $\theta$ необхідно оновлювати у напрямку, протилежному до градієнту функції J відносно $\theta$. При оновленні параметрів також використовується коефіцієнт швидкості навчання $\mu$. Метод градієнтного спуску можна виразити такою формулою:

$$\theta = \theta - \mu \nabla_\theta J(\theta) \tag{2.8}$$

Використання методу градієнтного спуску для тренування ML-моделей поділяють на три основні варіанти:

- Пакетний градієнтний спуск (Batch gradient descent), в якому на кожній ітерації використовується весь набір даних.
- Стохастичний градієнтний спуск (Stochastic gradient descent, SGD), в якому на ітерації використовується тільки один елемент з множини даних.
- Міні-пакетний градієнтний спуск (Mini-batch gradient descent), що поєднує переваги першого і другого варіантів, використовуючи невелику частину випадково обраних елементів з датасету.

При тренуванні моделей на великих обсягах даних, пакетний варіант може спричиняти довгу тривалість однієї ітерації, а також значне використання пам’яті, що зазвичай є непрактично. Внаслідок цього для машинного навчання зазвичай використовують стохастичний або міні-пакетний варіанти.

Використання mini-batch варіанту градієнтного спуску не гарантує гарну збіжність, при цьому виникають деякі додаткові проблеми:

- складність вибору коефіцієнту швидкості навчання
- використання одного і того ж коефіцієнту швидкості навчання для всіх вагових параметрів

- велика кількість неоптимальних локальних мінімумів цільової функції, які бажано уникати

Для часткового вирішення зазначених вище проблем існує доволі багато модифікацій методу градієнтного спуску, основними з яких є наступні:

- Momentum
- NAG
- Adagrad
- Adadelta
- RMSProp
- Adam
- AdaMax
- Nadam

Однією з основних переваг вказаних вище модифікацій є те, що в них коефіцієнт швидкості навчання вибирається динамічно, тому на практиці використання цих методів часто покращує ефективність тренування моделей.

### 2.3.4 Побудова індексної структури на основі нейронної мережі

Як і у випадку адаптивної індексної структури на базі лінійної регресії, система буде складатись із двох основних частин: хеш-таблиці та хеш-функції на базі нейронної мережі. При цьому, можна використати ту ж саму реалізацію хеш-таблиці, що і для попереднього варіанту.

Хеш-функція буде формуватись на базі натренованої нейронної мережі. При виборі архітектури нейронної мережі та функції активації необхідно враховувати, що модель не може бути занадтно складною, адже це призведе до суттєвого сповільнення роботи індексної структури. Внаслідок цього необхідно використовувати нейронну мережу прямого поширення з невеликою кількістю прихованих шарів. В якості функції активації можна вибрати ReLU, адже для її обчислень треба значно менша кількість операцій ніж для сигмоїдної

При тренуванні моделі на конкретному наборі даних можна провести декілька сесій тренувань з нейроними мережами різної складності. Після тренувань необхідно вибрати ту з них, що є найшвидшою при передбаченні позицій і надає прийнятні результати з точки зору кількості колізій.

## 2.4 Висновки

Отже, для мінімізації кількості колізій було запропоновано схему побудови адаптивної індексної структури, робота якої ґрунтується на застосуванні методів машинного навчання.

Основною ідеєю є використання функції розподілу даних у ролі хеш-функції. Для вивчення функції розподілу даних можна використовувати одну із моделей машинного навчання. При умові, що така модель достатньо точно вивчить функцію розподілу, кількість колізій буде значно зменшена.

Зважаючи на те, що хеш-функція має працювати швидко, необхідно використовувати прості моделі машинного навчання. Враховуючи цю вимогу, було запропоновано використання двох видів моделей: лінійну регресію та нейронну мережу з невеликою кількістю прихованих шарів.

# 3 РЕАЛІЗАЦІЯ АЛГОРИТМУ РОБОТИ АДАПТИВНОЇ ІНДЕКСНОЇ СТРУКТУРИ ДЛЯ ТОЧКОВОГО ПОШУКУ ТА ОГЛЯД РЕЗУЛЬТАТІВ

## 3.1 Вибір технологій

Вибір технологій для реалізації адаптивної індексної структури поділяється на вибір середовища та мови програмування, а також фреймворка для роботи з нейронними мережами. При цьому, важливо звертати увагу на такі основні аспекти:

- поширеність технології
- легкість у використанні
- ефективність

Внаслідок того, що в даній роботі пропонується прототипний варіант реалізації індексної структури, то більш важливими аспектами у технологій є саме поширеність та легкість використанні, а ефективність матиме суттєвий вплив вже в наступних версіях програми.

### 3.1.1 Вибір середовища програмування

Так як робота індексної структури базується на методах машинного навчання, то необхідно враховувати наявність його підтримки в стандартних засобах мови та середовища програмування. Важливим фактором також є наявність відповідних бібліотек.

Для реалізації прототипного програмного забезпечення пов’язаного з використанням машинного навчання найчастіше використовують одну із наступних мов програмування:

- Python [23]
- R [24]
- Matlab [25]

Python є мовою програмування загального призначення, але в ній наявні багато бібліотек для роботи з даними і машинним навчанням. Крім цього, в останні часи ця мова доволі активно застосовується при проведенні досліджень в галузі машинного навчання і в неї є велика спільнота.

R та Matlab є більш вузькоспеціалізованими мовами, тому для виконання деяких задач, зокрема реалізації хеш-таблиці, вони можуть бути менш зручними у використанні.

Враховуючи зазначену вище інформацію, можна зробити висновок, що для реалізації прототипного варіанту адаптивних індексних структур на базі машинного навчання найкраще підійде мова програмування Python.

### 3.1.2 Вибір фрейморка для роботи з нейронними мережами

Для мови Python існує велика кількість фреймворків, що дозволяють працювати з нейронними мережами, проте найбільш поширеними серед них є наступні:

- TensorFlow [26]
- Keras [27]
- PyTorch [28]

TensorFlow був розроблений компанією Google та має на даний момент найбільшу спільноту серед трьох зазначених фрейморків. TensorFlow є бібліотекою для розподілених обчислень загального призначення, що в свою чергу означає можливість виконувати на ньому обчислення для нейронних мереж, але, окрім цього, ще багато інших видів обчислень. Важливою його перевагою є також те, що він вже доволі довго використовується для комерційних цілей, і може працювати як на серверному обладнанні так і на звичайних смартфонах. Проте, внаслідок своєї загальності, реалізація на ньому навіть доволі простих нейронних мереж може виявитись занадто громіздкою.

Keras є високорівневою бібліотекою для роботи з нейронними мережами, що працює поверх інших обчислювальних фрейморків. Keras є доволі простим у використанні та має зрозумілий API, тому є доволі популярним вибором для

прототипування. Але, з іншого боку, внаслідок своєї простоти йому іноді не вистачає гнучкості.

PyTorch був спочатку розроблений у компанії Facebook, але на даний момент підтримується широкою спільнотою програмістів. Цей фреймворк надає високорівневий функціонал для тензорних обчислень з підтримкою GPU, а також зручне API для побудови і використання глибоких нейронних мереж. Однією з основних переваг PyTorch перед іншими фреймворками є те, що він дозволяє будувати нейронні мережі динамічно, тому це значно спрощує побудову і відлагодження нейронних мереж. Іншою важливою перевагою є тісна інтеграція з мовою Python, внаслідок чого код, написаний за допомогою даного фреймворку, є лаконічним та простим для розуміння.

Зважаючи на наведені вище переваги та недоліки різних фреймворків, для реалізації адаптивної індексної структури на базі нейронної мережі було обрано PyTorch.

## 3.2 Набори даних для експериментів

Для проведення експериментів було обрано два набори даних:

1) Грошові транзакції з кредитних карток [29]
2) Населення США за поштовими кодами [30]

Перший датасет являє собою список грошових транзакцій в Європі за декілька днів вересня 2013 року. Оригінальні дані цих транзакцій були перетворені за допомогою методу PCA, щоб не розкривати конфіденційну інформацію, але часові мітки транзакцій були збережені. Ці часові мітки і будуть використані в якості ключів хеш-таблиці.

Перед використанням цього набору даних була проведена його попередня обробка:

- всі поля, крім часових міток, були виключені
- часові мітки, що дублюють попередні, були також виключені

- для використання в ролі ключів хеш-таблиці були взяті перші 100 тис. записів з набору

Графік розподілу даних у першому датасеті наведено на рис. 3.1.

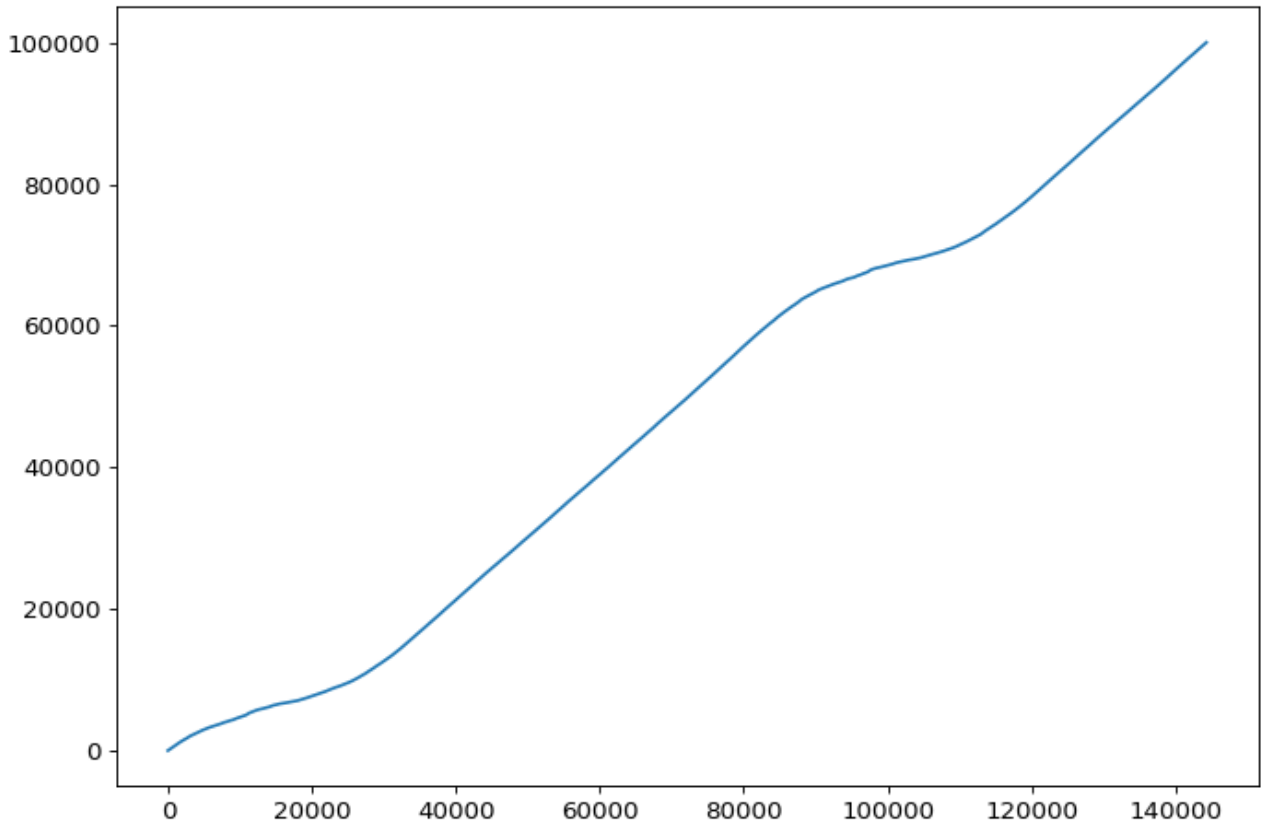


Рисунок 3.1 – Графік функції розподілу даних у першому наборі даних

У другому наборі даних міститься інформація про кількість населення у різних регіонах США в залежності від поштового коду. При цьому, дані розбиті і по декільком іншим чинникам, таким як вік та стать.

Перед використанням цього датасету, була також проведена його попередня обробка, зокрема, всі записи були згруповані по поштовому коду. У якості ключів для хеш-таблиці будуть використані поштові коди, які представляють собою цілі числа, що складаються з п’яти або менше цифр. Всього в наборі даних наявно приблизно 33 тис. поштових кодів.

Графік розподілу поштових кодів у другому наборі даних зображено на рисунку 3.2.

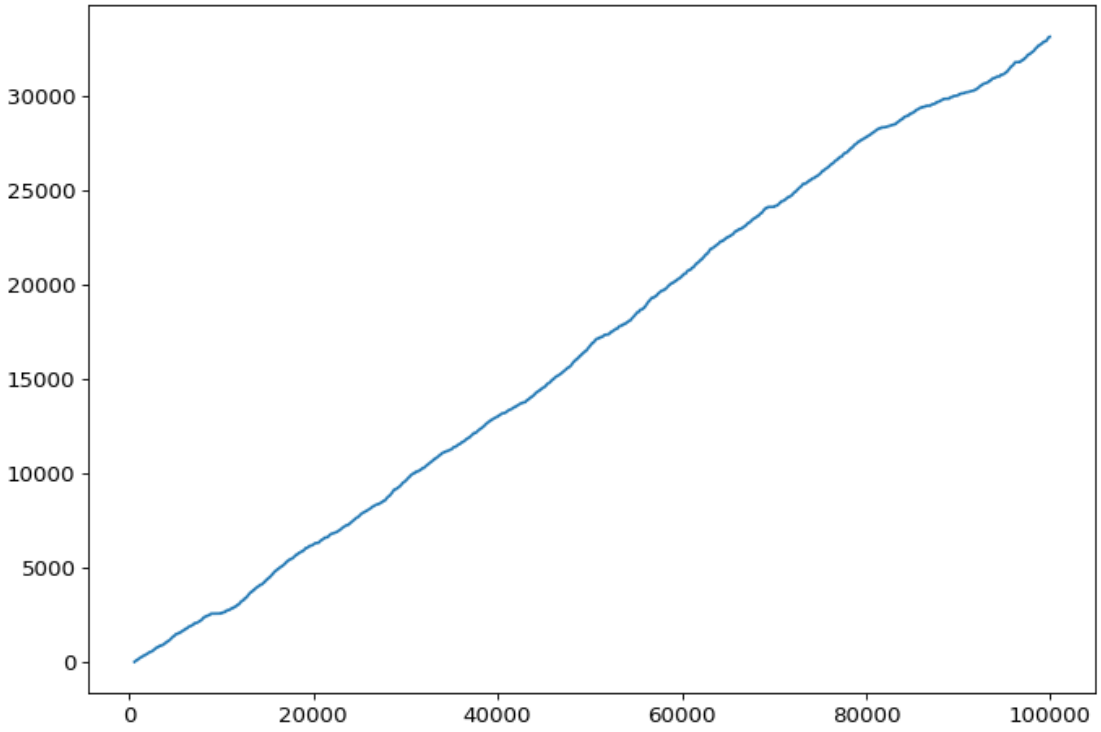


Рисунок 3.2 – Графік функції розподілу даних у другому наборі даних

Незважаючи на те, що графік на рис. 3.2 виглядає доволі лінійно, варто зазначити, що при збільшенні масштабу помітні значні розриви у ключах. Це показано на рис. 3.3

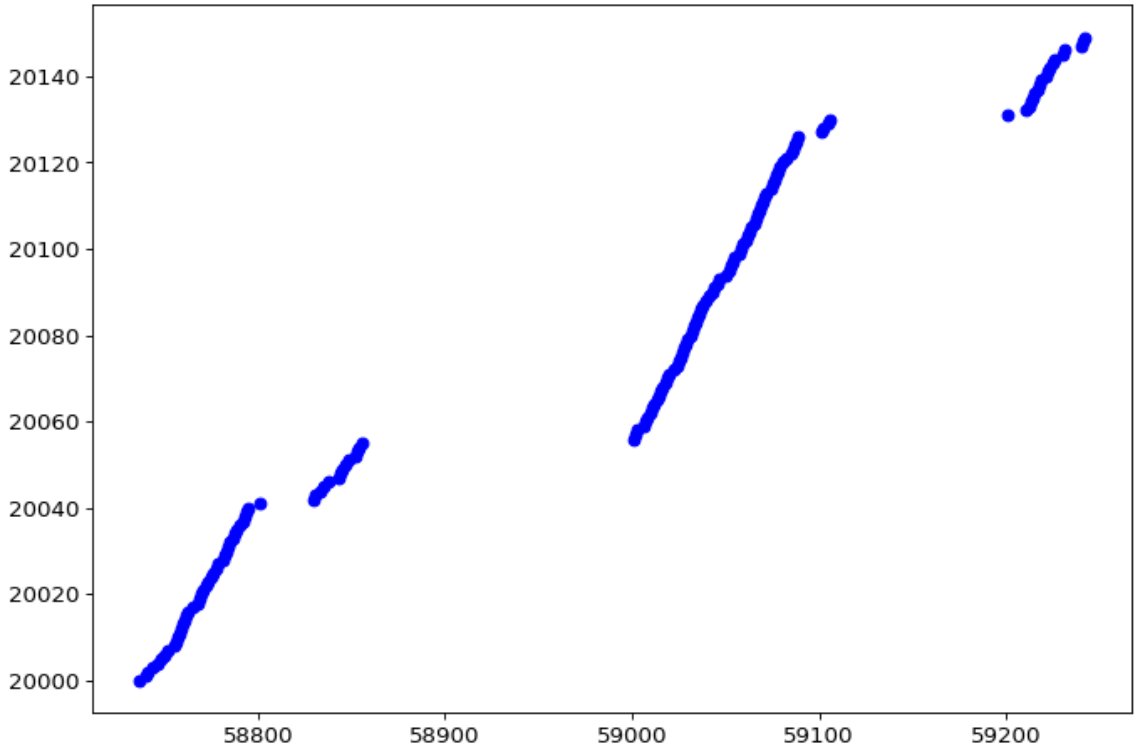


Рисунок 3.3 – Приклади розривів у функції розподілу даних другого набору

## 3.3 Реалізація загальної хеш-таблиці для експериментів

Для полегшення визначення кількості колізій та невикористаних слотів було реалізовано загальну хеш-таблицю, в яку для роботи необхідно передати хеш-функцію. Ця хеш-функція може бути як класичною, так і базованою на натренованій моделі машинного навчання. В зазначеній реалізації було використано метод ланцюжків для розв’язання колізій. Приклад методу для зберігання ключа і значення в хеш-таблиці наведено у коді нижче:

```
def put(self, key, value):
  # визначення хеш-коду ключа і відповідного слота
  hash = self.hash_func(key)
  node = self.slots[hash]
  # збереження ключа і значення
  # при відсутності колізій
  if node is None:
    self.slots[hash] = list_node(key, value, None)
    self.size += 1
  # розв'язання колізій методом ланцюжків
  else:
    while True:
      if (node.key == key):
      node.value = value
      return
    elif (node.next_node is None):
      node.next_node = list_node(key, value, None)
      self.size += 1
      return
    else:
      node = node.next_node
```

## 3.4 Реалізація адаптивної індексної структури на основі лінійної регресії

Перед тренуванням моделі лінійної регресії, спочатку необхідно відсортувати ключі та поставити у відповідність кожному ключу одну позицію. Після цього, значення ключів та позицій бажано нормалізувати до проміжку [0; 1] для кращого проходження тренування. Окрім тренування, нормалізацію даних треба буде враховувати і при використанні моделі для роботи хеш-функції.

Для того, щоб лінійна регресія могла враховувати нелінійності в розподілі ключів, вхідні дані для тренування необхідно розширити до поліноміального виду. Для полегшення цього процесу, можна використати утиліти з бібліотеки scikit-learn [31]. Нижче наведено приклад коду, що розширяє вхідні дані до поліноміального виду, тренує модель та повертає її у вигляді лямбда-функції для подальшого використання:

```
def train_linreg_cdf(xs, ys, degree):
  # трансформація вхідних даних
  # для підтримки нелінійностей у розподілі
  poly = PolynomialFeatures(degree=degree)
  t = lambda xs: poly.fit_transform(xs.reshape(-1, 1))
  # тренування моделі
  reg = linear_model.LinearRegression()
  reg.fit(t(xs), ys)
  # повернення моделі у вигляді лямбда-функції
  # для передбачення позицій
  return lambda x: reg.predict(t(np.array([x])))
```

В наведеній вище функції один із аргументів відповідає за степінь поліному. Важливою задачею є правильний вибір цього степеня. Як було зазначено в попередньому розділі, для цього можна вибрати деяку множину степенів та

натренувати модель для кожного з них. Після цього необхідно вибрати ту з моделей, в якій наявна найменша кількість колізій.

## 3.5 Огляд результатів роботи адаптивної індексної структури на основі лінійної регресії

Розглянемо спочатку результати роботи адаптивної індексної на першому наборі даних (числові мітки грошових транзакцій). У табл. 3.1 наведено порівняння кількостей колізій у хеш-таблиці при різних степенях поліному.

Таблиця 3.1 – Порівняння кількості колізій у хеш-таблиці з використанням лінійної регресії при різних степенях поліному

| Степінь поліному | Пусті слоти |
|---|---|
| 1 | 25,29% |
| 2 | 26,00% |
| 3 | 27,19% |
| 4 | 20,40% |
| 5 | 20,79% |
| 6 | 16,63% |
| 7 | 17,82% |
| 8 | 17,76% |
| 9 | 16,50% |
| 10 | 15,41% |
| 11 | 15,51% |
| 12 | 14,63% |
| 13 | 14,99% |
| 14 | 14,83% |
| 15 | 14,70% |

Аналізуючи результати у табл. 3.1, можна зробити висновок, що для даного розподілу даних найкращий результат показала модель лінійної регресії з дванадцятим степенем поліному. Варто також зауважити, що збільшення степеню поліному не обов’язково надає кращий результат, адже при тренуванні лінійної регресії мінімузується середньоквадратична помилка, що не є еквівалентом кількості колізій.

На рис. 3.4 показано порівняння реального розподілу даних у датасеті з тим, що передбачає модель.

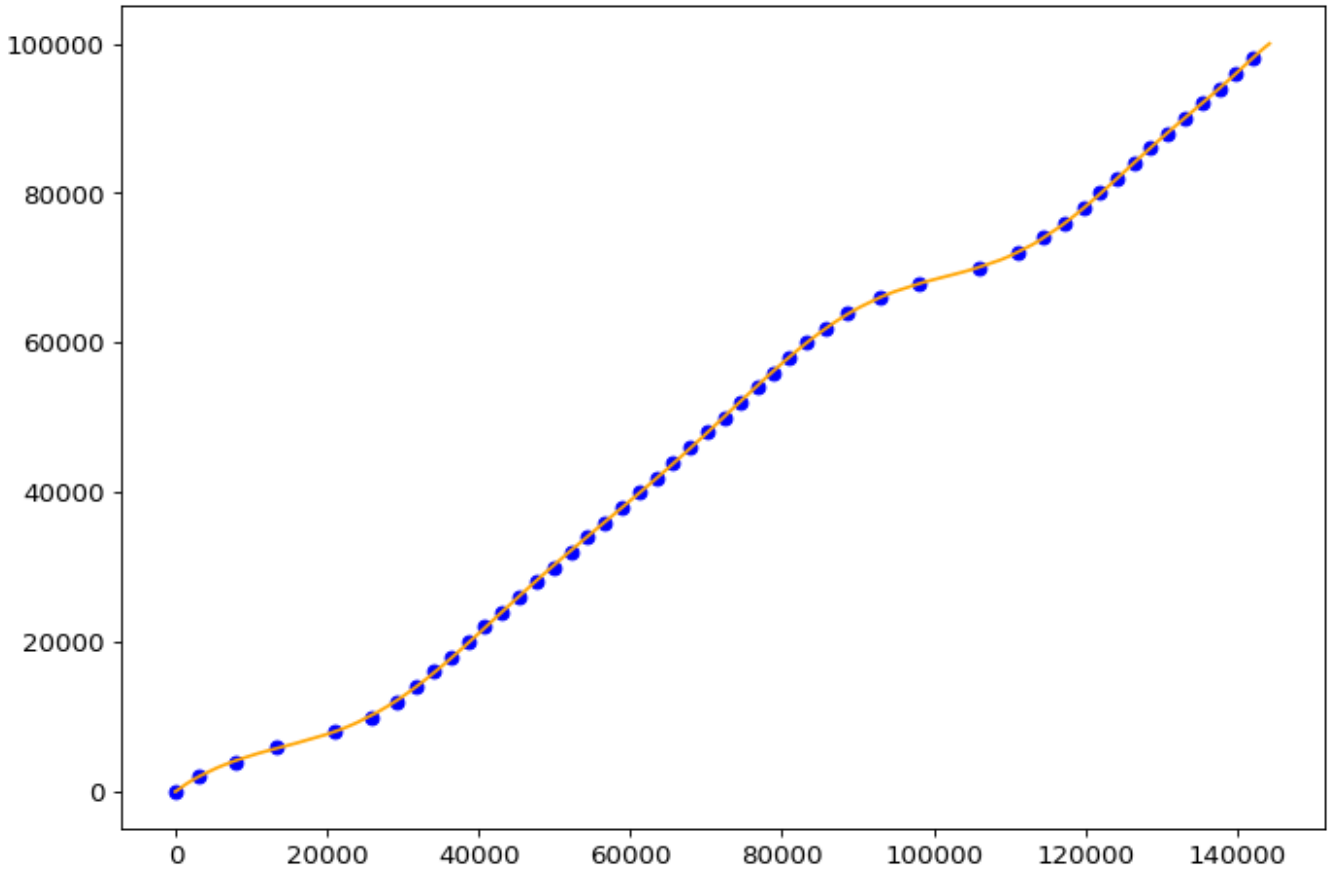


Рисунок 3.4 – Порівняння функції розподілу для першого набору даних: пунктирною лінією зображено реальний розподіл даних, суцільною – розподіл, що передбачає модель лінійної регресії

У таблиці 3.2 показано порівняння роботи адаптивної та класичної індексних структур з точки зору ефективності використання пам’яті

Таблиця 3.2 – Порівняння роботи адаптивної та класичної індексних структур для першого набору даних

| Коефіцієнт завантаженості хеш-таблиці | Пусті слоти в адаптивній індексній структурі | Пусті слоти в класичній індексній структурі |
|---|---|---|
| 75% | 28.21% | 47.30% |
| 100% | 14.63% | 36.68% |
| 125% | 10.58% | 28.53% |

Аналізуючи результати з таблиці 3.2, можна зробити висновок, що для розподілу даних з транзакціями використання адаптивної індексної структури на базі лінійної регресії показало значно кращий результат з точки зору використання пам’яті.

Тепер розглянемо результати роботи на другому наборі даних (поштові індеси у США).

У таблиці 3.3 показано порівняння роботи адаптивної та класичної індексних структур з точки зору ефективності використання пам’яті.

Таблиця 3.3 – Порівняння роботи адаптивної та класичної індексних структур для другого набору даних

| Коефіцієнт завантаженості хеш-таблиці | Пусті слоти в адаптивній індексній структурі | Пусті слоти в класичній індексній структурі |
|---|---|---|
| 75% | 53.63% | 47.18% |
| 100% | 49.88% | 37.01% |
| 125% | 47.20% | 28.53% |

Аналізуючи результати з таблиці 3.3, можна зробити висновок, що для розподілу даних з поштовими кодами використання адаптивної індексної структури на базі лінійної регресії показало гірший результат з точки зору використання пам’яті. Це відбулось внаслідок того, що в цьому наборі даних існують великі розриви між

групами ключів, і незважаючи на те, що модель вивчила загальний розподіл, на “останній милі” вона давала значні помилки. Це можна побачити на рис. 3.5

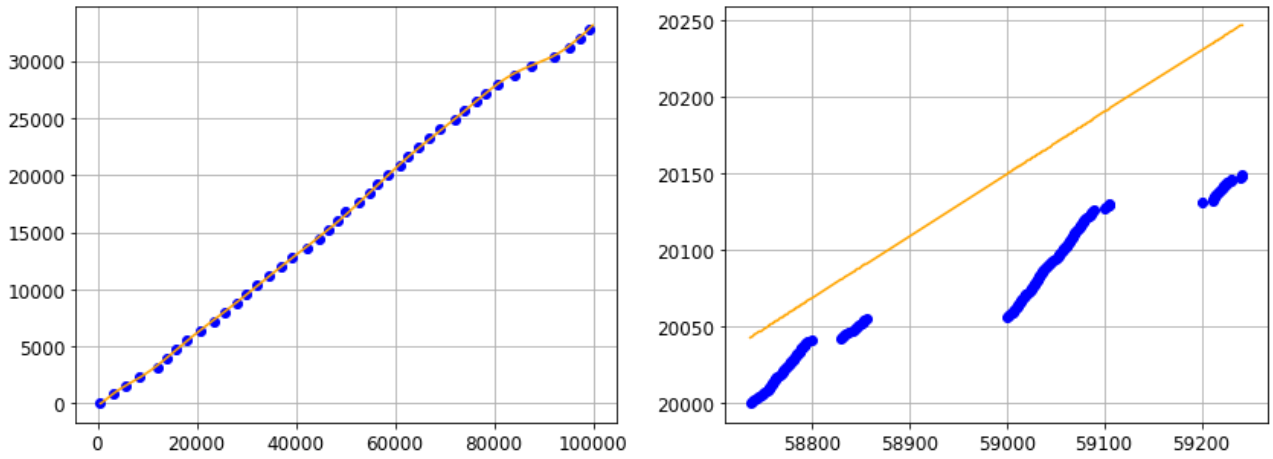


Рисунок 3.5 – Ліворуч зображено повний графік функції розподілу для другого набору даних, праворуч – частину графіку зі збільшеним масштабом

## 3.6 Реалізація адаптивної індексної структури на основі нейронної мережі

Як і в попередньому випадку, перед тренуванням моделі дані необхідно доповнити та нормалізувати, щоб вони лежали в проміжку [0; 1]. Після цього можна починати тренування нейронної мережі.

Зважаючи на те, що обчислення хеш-функції має працювати швидко, в якості моделі було обрано нейронну мережу з одним прихованим шаром. Кількість нейронів для цього шару можна вказати в залежності від кількості даних і складності їх розподілу, але зазвичай вона знаходиться в межах від 10 до 50. В якості функції активації було обрано ReLU, так як вона є більш ефективною в порівнянні з альтернативами. Частина коду, що використовується для побудови нейронної мережі та налаштування параметрів оптимізації, наведено нижче:

```
# вказання архітектури мережі
model = torch.nn.Sequential(
    torch.nn.Linear(1, hidden_layer_size),
    torch.nn.ReLU(),
```

```
    torch.nn.Linear(hidden_layer_size, 1)
    )
# налаштування параметрів оптимізації
criterion = nn.MSELoss()
optimiser = torch.optim.SGD(model.parameters(), lr = 0.1)
```

## 3.7 Огляд результатів роботи адаптивної індексної структури на основі нейронної мережі

Спочатку розглянемо результати роботи адаптивної індесної структури на базі нейронної мережі для першого набору даних. На рис. 3.6 показано порівняння реального розподілу даних у датасеті з тим, що передбачає модель:

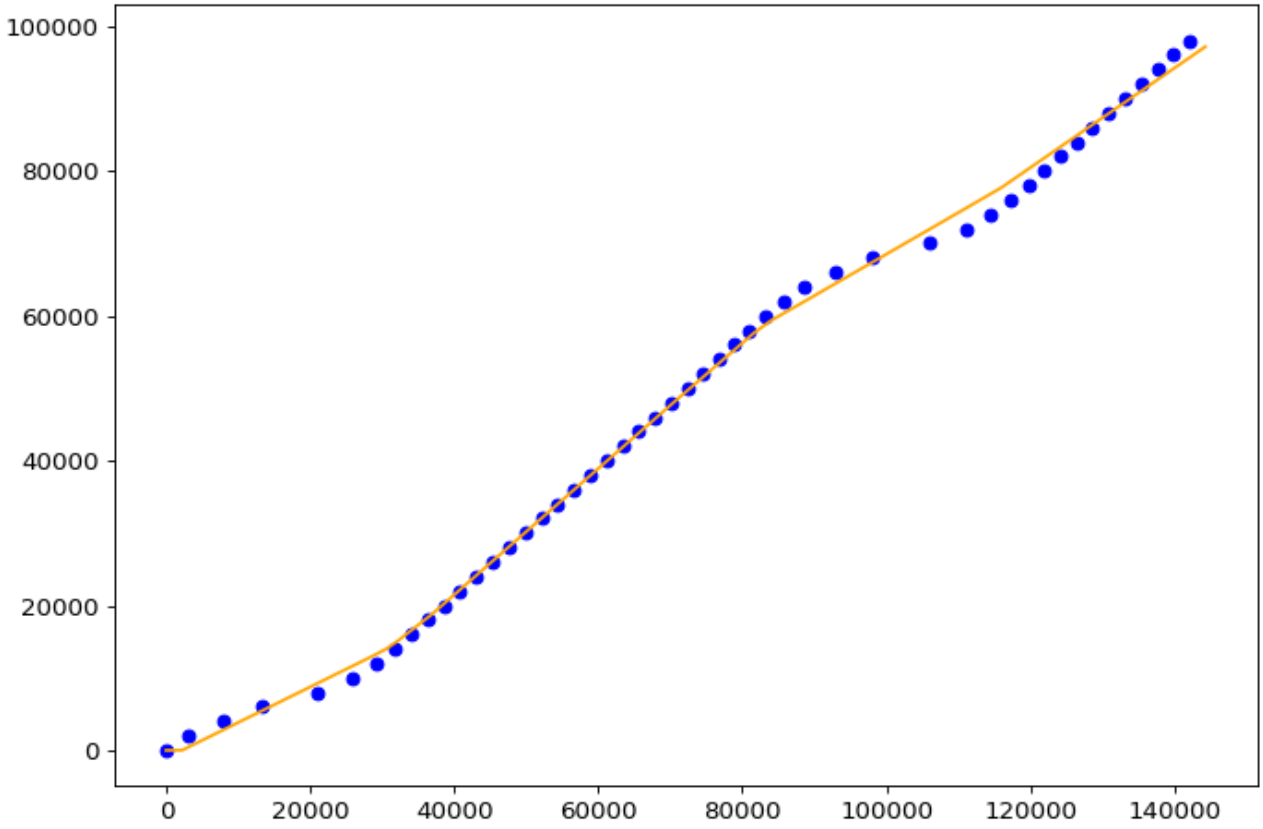


Рисунок 3.6 – Порівняння функції розподілу для першого набору даних: пунктирною лінією зображено реальний розподіл даних, суцільною – розподіл, що передбачає нейронна мережа

У таблиці 3.4 показано порівняння роботи адаптивної та класичної індексних структур з точки зору ефективності використання пам’яті:

Таблиця 3.4 – Порівняння роботи адаптивної та класичної індексних структур для першого набору даних

| Коефіцієнт завантаженості хеш-таблиці | Пусті слоти в адаптивній індексній структурі | Пусті слоти в класичній індексній структурі |
|---|---|---|
| 75% | 30.33% | 47.30% |
| 100% | 20.83% | 36.68% |
| 125% | 16.62% | 28.53% |

Аналізуючи результати з таблиці 3.4, можна зробити висновок, що для розподілу даних з транзакціями використання адаптивної індексної структури на базі нейронної мережі показало кращий результат в порівнянні з класичною індексною структурою. Але, якщо порівнювани з адаптивною структурою на базі лінійної регресії, то результати є дещо гіршими.

Тепер розглянемо результати роботи на другому наборі даних.

У таблиці 3.5 показано порівняння роботи адаптивної та класичної індексних структур з точки зору ефективності використання пам’яті.

Таблиця 3.5 – Порівняння роботи адаптивної та класичної індексних структур для другого набору даних

| Коефіцієнт завантаженості хеш-таблиці | Пусті слоти в адаптивній індексній структурі | Пусті слоти в класичній індексній структурі |
|---|---|---|
| 75% | 55.03% | 47.18% |
| 100% | 51.52% | 37.01% |
| 125% | 48.90% | 28.53% |

Результати в таблиці 3.5 вказують на те, що на другому наборі даних структура на базі нейронної мережі також не змогла перевершити класичну хеш-таблицю. Як і у випадку з лінійною регресією основні проблеми виникли на “останній милі”.

## 3.8 Висновки

Отже, в даному розділі було описано деталі реалізації адаптивних індексних структур для точкового пошуку, а також проведено експерименти, які показали ефективність використання таких структур.

Спочатку було проведено аналіз існуючих середовищ програмування та фреймворків для роботи з нейронними мережами. В результі цього було обрано середовище програмування Python та фреймворк PyTorch.

Потім було показано деталі реалізації двох видів адаптивних індексних структур для точкового пошуку: на базі використання лінійної регресії та нейронної мережі. Окрім цього, було також описано реалізацію загальної хеш-таблиці, за допомогою якої можна перевірити кількість колізій та пустих слотів при використанні різних моделей машинного навчання в ролі хеш-функцій.

Також було наведено результати експериментів по використанню запропонованих адапнивних індесних структур і їхнє порівняння з класичними. Ці результати показали, що на одних наборах даних адаптивні індексні структури працюють краще за класичні з точки зору ефективності використання пам’яті, а на інших наборах гірше. Крім цього, використання лінійної регресії у ролі хеш-функції показало дещо кращі результати у порівнянні з використанням нейронної мережі.

# 4 РЕАЛІЗАЦІЯ СТАРТАП ПРОЕКТУ

## 4.1 Опис ідеї та технологічний аудит стартап-проекту

У даному розділі описано економічне обґрунтування реалізації стартап-проекту на тему «Створення системи керування базами даних з адаптивними індексними структурами».

Таблиця 4.1 – Опис ідеї стартап-проекту

| *Зміст ідеї* | *Напрямки застосування* | *Вигоди для користувача* |
|---|---|---|
| Ідея полягає у тому, щоб створити СКБД, в якій будуть використовуватись адаптивні індексні структури даних для швидкого пошуку на основі підходів машинного навчання | 1. Вирішення задачі швидкого доступу до даних | Користувачу необхідно буде встановити СКБД, створити базу даних і додати в неї свої дані. Після цього в нього буде можливість швидкого доступу до даних |
| | 2. Вирішення задачі адаптування індексів до конкретних розподілів даних | В користувача буде можливість ефективніше використовувати обчислювальні ресурси внаслідок використання у базі даних адаптивних індексних структур |

Аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів передбачає:

1. визначення переліку техніко-економічних властивостей та характеристик ідеї
2. визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;

3. проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні).

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

| *No п/п* | *Техніко-економічні характеристики ідеї* | *(потенційні) товари/концепції конкурентів* | | | | *W (слабка сторона )* | *N (нейтральна сторона )* | *S (сильна сторона )* |
|---|---|---|---|---|---|---|---|---|
| | | *Мій проект* | *Конкурент1* | *Конкурент2* | *Конкурент3* | | | |
| 1. | Форма виконання | Програма | Програма | Веб-додаток | Програма | | | + |
| 2. | Собівартість | Низька | Висока | Низька | Висока | | | + |
| 3. | Наявність адміністратора для налаштування | Треба | Не треба, дистанційно | Треба | Треба | | + | |
| 4. | Наявність інтернету | Не треба | Не треба | Треба | Не треба | | | + |
| 5. | Кросплатформенність | Так | Так | Так | Ні | + | | |

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових:

1. за якою технологією буде виготовлено товар згідно ідеї проекту?
2. Чи існують такі технології,чи їх потрібно розробити/доробити?

3. чи доступні такі технології авторам проекту?

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

| *No n/n* | *Ідея проекту* | *Технології її реалізації* | *Наявність технологій* | *Доступність технологій* |
|---|---|---|---|---|
| 1. | Створення нової СКБД | Amazon Web Services | Наявна | Платна, недоступна |
| 2. | | C/C++, TensorFlow | Наявна | Безкоштовна, доступна |
| Обрана технологія реалізації ідеї проекту: для створення системи керування базою даних даних обрано технології C/C++ TensorFlow, яка є безкоштовною та доступною. | | | | |

## 4.2 Аналіз ринкових можливостей

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проводимо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

| *No n/n* | *Показники стану ринку (найменування)* | *Характеристика* |
|---|---|---|
| 1 | Кількість головних гравців, од | 3 |
| 2 | Загальний обсяг продаж, грн/ум.од | 20000 грн./ум.од |
| 3 | Динаміка ринку (якісна оцінка) | Зростає/спадає/стагнує |
| 4 | Наявність обмежень для входу (вказати характер обмежень) | Немає |
| 5 | Специфічні вимоги до стандартизації та сертифікації | Немає |

| | | |
|---|---|---|
| 6 | Середня норма рентабельності в галузі (або по ринку), % | R = (3000000 * 100) / (1000000 * 12) = 25% |

Рентабельність— поняття, що характеризує економічну ефективність виробництва, за якої за рахунок грошової виручки від реалізації продукції (робіт, послуг) повністю відшкодовує витрати на її виробництво й одержується прибуток як головне джерело розширеного відтворення

[http://buklib.net/books/29473/]. Суть одного із найважливіших методів оцінки економічної ефективності інвестицій полягає у розрахунку їх середньої рентабельності за формулою [http://pidruchniki.com/1566072162240/turizm/prognozuvannya_efektivnosti_investi tsiynogo_proektu]

$$R = \frac{P}{1 \times n} \times 100$$

де Р- прибуток за час експлуатації проекту; / - повна сума інвестиційних витрат; п - час експлуатації проекту. Інвестувати грошові засоби доцільно тоді, коли від цього можна отримати більший прибуток, ніж від їх зберігання у банку. Порівнюючи середньорічну рентабельність інвестицій зі ставкою банківського відсотка, можна дійти висновку, що вигідніше [http://pidruchniki.com/1566072162240/turizm/prognozuvannya_efektivnosti_investi tsiynogo_proektu].

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

| *No п/п* | *Потреба, що формує ринок* | *Цільова аудиторія (цільові сегменти ринку)* | *Відмінності у поведінці різних потенційних цільових груп клієнтів* | *Вимоги споживачів до товару* |
|---|---|---|---|---|
| 1. | Необхідно програмне забезпечення для обробки великих наборів даних і | Потенційними цільовими групами є дослідницькі центри, університети та | Цільова група займається дослідженнями або має обробляти даних великих розмірів | Рішення має бути швидким, що дозволить концентруватись на інших задачах, і |

| | | | | |
|---|---|---|---|---|
| | швидкого доступу до них | компанії, специфіка роботи яких пов‘язана із великими даними | | використовувати даний програмний продукт як інструментарій |

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. NoNo 6-7). Фактори в таблиці подавати в порядку зменшення значущості.

Ринкові можливості - це сприятливі обставини, які підприємство може використовувати для отримання переваг. Слід зазначити, що можливостями з погляду SWOT-аналізу є не всі можливості, які існують на ринку, а тільки ті, які можна використовувати
[http://pidruchniki.com/1974070454048/menedzhment/strategichniy_analiz_pidpriyemstva].

Ринкові загрози - події, настання яких може несприятливо вплинути на підприємство.

[http://pidruchniki.com/1974070454048/menedzhment/strategichniy_analiz_pidpriye mstva].

Таблиця 4.6 – Фактори загроз

| *No n/n* | *Фактор* | *Зміст загрози* | *Можлива реакція компанії* |
|---|---|---|---|
| 1. | Конкуренція | Вихід на ринок великої компанії | 1) Вихід з ринку<br>2) Запропонувати великій компанії поглинути себе<br>3) Передбачити додаткові переваги власного ПЗ для того, щоб повідомити про них саме після виходу міжнародної компанії на ринок |

| | | | |
|---|---|---|---|
| 2. | Зміна потреб користувачів | Користувачам необхідне програмне забезпечення з іншим функціоналом | 1) Передбачити можливість додавання нового функціоналу до створюваного ПЗ |

Таблиця 4.7 – Фактори можливостей

| *No n/n* | *Фактор* | *Зміст можливості* | *Можлива реакція компанії* |
|---|---|---|---|
| 1 | Зростання можливостей потенційних покупців | Зростанням держфінансувауня досліджень у галузі обробки великих наборів даних | Запропонувати свої послуги державним підприємствам та дослідницьким центрам |
| 2 | Зниження довіри до конкурента 1 | У ПЗ конкурента No1 нещодавно була знайдена помилка, завдяки якій дані досліджень усіх клієнтів стали доступні в інтернеті для всіх користувачів | При виході на ринок звертати увагу покупців на безпеку нашого ПЗ |

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку (табл. 8).

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

| *Особливості конкурентного середовища* | *В чому проявляється дана характеристика* | *Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)* |
|---|---|---|
| 1. Вказати тип конкуренції - досконала | Існує 3 фірми-конкурентки на ринку | Врахувати ціни конкурентних компаній на початкових етапах створення бізнесу, реклама (вказати на конкретні переваги перед конкурентами) |
| 2. За рівнем конкурентної боротьби - міжнародний | Одна з компаній – з ішної країни, дві – з України | Додати можливість вибору мови ПЗ, щоб легше було у майбутньому |

| | | |
|---|---|---|
| | | вийти на міжнародний ринок |
| 3. За галузевою ознакою - внутрішньогалузева | Конкуренти мають ПЗ, яке використовується лише всередині даної галузі | Створити основу ПЗ таким чином, щоб можна було легко переробити дане ПЗ для використання у інших галузях |
| 4. Конкуренція за видами товарів: - товарно-видова | Види товарів є однаковими, а саме – програмне забезпечення | Створити ПЗ, враховуючи недоліки конкурентів |
| 5. За характером конкурентних переваг неціновa | Вдосконалення технології створення ПЗ, щоб собівартість була нижчою | Використання менш дорогих технологій для розробки, ніж використовують конкуренти |
| 6. За інтенсивністю - не марочна | Бренди відсутні | - |

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі. *М. Портер* вирізняє п‘ять основних факторів, що впливають на привабливість вибору ринку з огляду на характер конкуренції.

Сильні позиції компанії за кожним з факторів означають її можливості забезпечити необхідні темпи обороту капіталу та її здатність впливати на інших агентів ринку, диктуючі їм власні умови співпраці.

Характеристики факторів моделі відрізняються для різних галузей та змінюються із часом. Сила кожного фактору є функцією від структури галузі та її техніко-економічних характеристик.

На основі аналізу складових моделі 5 сил М. Портера розробляється перелік факторів конкурентоспроможності для певного ринку.

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

| *Складові аналізу* | *Прямі конкуренти в галузі* | *Потенційні конкуренти* | *Постачальники* | *Клієнти* | *Товари-замінники* |
|---|---|---|---|---|---|
| | *Навести перелік* | *Визначити бар’єри* | *Визначити фактори* | *Визначити фактори* | *Фактори загроз з боку замінників* |

| | *прямих конкурентів* | *входження в ринок* | *сили постачальників* | *сили споживачів* | |
|---|---|---|---|---|---|
| Висновки: | Існує 3 конкуренти на ринку. Найбільш схожим за виконанням є конкурент 1, так як його рішення також представлене у вигляді програми. | Так, можливості для входу на ринок є, бо наше рішення спрощує та пришвидшує роботу спеціаліста. | Постачальники відсутні. | Важливим для користувача є швидкість роботи ПЗ. | Товари-замінники можуть використати більш дешеву техно- логію створення ПЗ та зменшити собівартість товару. |

За результатами аналізу таблиці робиться висновок щодо принципової можливості роботи на ринку з огляду на конкурентну ситуацію. Також робиться висновок щодо характеристик (сильних сторін), які повинен мати проект, щоб бути конкурентоспроможним на ринку. Другий висновок враховується при формулюванні переліку факторів конкурентоспроможності у п. 3.6.

На основі аналізу конкуренції, проведеного в п. 3.5 (табл. 9), а також із урахуванням характеристик ідеї проекту (табл. 2), вимог споживачів до товару (табл. 5) та факторів маркетингового середовища (табл. NoNo 6-7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за табл. 10.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

| *No n/n* | *Фактор конкурентоспроможності* | *Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)* |
|---|---|---|
| 1. | Виконання ПЗ у вигляді програми у розподіленій системі | Це рішення дозволяє швидко обробляти дані розміром значно більше за 4 ГБ. |
| 2. | Простота інтерфейсу користувача | Користувач має лише завантажити дані і запустити програму на виконання. |

За визначеними факторами конкурентоспроможності (табл. 10) проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 11). Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін проекту

| *No п/ п* | *Фактор конкурентоспроможності* | *Бали 1-20* | *Рейтинг товарів-конкурентів у порівнянні з нашим підприємством* | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | *–3* | *–2* | *–1* | *0* | *+1* | *+2* | *+3* |
| 1 | Виконання ПЗ у вигляді програми у розподіленій системі | 15 | | | + | | | | |
| 2 | Простота інтерфейсу користувача | 20 | + | | | | | | |

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. Наприклад: зниження доходів потенційних споживачів – фактор загрози, на основі якого можна зробити прогноз щодо посилення значущості цінового фактору при виборі товару та відповідно, – цінової конкуренції (а це вже – ринкова загроза).

Таблиця 4.12 – SWOT-аналіз стартап-проекту

| | |
|---|---|
| Сильні сторони: простий інтерфейс користувача, виконання ПЗ у вигляді програми | Слабкі сторони: необхідно мати власний сервер (або орендувати його) |

| | |
|---|---|
| Можливості: у конкурента 1 виявлена проблема із безпекою ПЗ, додаткове держфінансування для досліджень у підприємствах, які є потенційними покупцями | Загрози: конкуренція, зміна потреб користувачів |

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів.

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

| *No п/п* | *Альтернатива (орієнтовний комплекс заходів) ринкової поведінки* | *Ймовірність отримання ресурсів* | *Строки реалізації* |
|---|---|---|---|
| 1. | Створення доповнення до існуючої СКБД, наприклад PostgresSQL | 80% | 6 місяців |
| 2. | Створення програми без використання будь- яких фреймворків для обробки даних | 30% | 12 місяців |

З означених альтернатив обирається та, для якої: а) отримання ресурсів є більш простим та ймовірним; б) строки реалізації – більш стислими. Тому обираємо альтернативу 1.

## 4.3 Розробка ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

| *No п/п* | *Опис профілю цільової групи потенційних клієнтів* | *Готовність споживачів сприйняти продукт* | *Орієнтовний попит в межах цільової групи (сегменту)* | *Інтенсивність конкуренції в сегменті* | *Простота входу у сегмент* |
|---|---|---|---|---|---|
| 1. | Університетські дослідження | Час виконання не є критичним у даному випадку, а однією із найголовніших переваг є саме це | Дослідження в області великих даних проводяться постійно | Існує 3 конкуренти, які надають схожі, але менш швидкі рішення. | У сегмент увійти непросто, бо бюрократія всередині університеті в занадто складна, також відсутня дуже впливова перевага ПЗ |
| 2. | Дослідницькі центри | Пришвидшення часу виконання додає їм можливості виконати більшу кількість замовлень | Дослідження в області великих даних проводяться постійно, до того ж – за замовлення м приватних компаній | | Маючи перевагу у зручності інтерфесу користувача, що пришвидшує роботу та отримання результату, вийти на ринок нескладно |
| 3. | Підприємства | Пришвидшення часу виконання додає їм можливості виконати більшу кількість замовлень, до того ж в цьому році з держбюджету виділені додаткові кошти на цей напрям досліджень | | | Маючи перевагу у тому, що рішення є зручним для користування і швидким вийти на ринок не є складно |

| |
|---|
| Які цільові групи обрано: обираємо дослідницькі центри та підприємства, які підтримуються держзамовленням |

За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку. За М. Портером, існують три базові стратегії розвитку, що відрізняються за ступенем охоплення цільового ринку та типом конкурентної переваги, що має бути реалізована на ринку (за витратами або визначними якостями товару).

Стратегія лідерства по витратах передбачає, що компанія за рахунок чинників внутрішнього і/або зовнішнього середовища може забезпечити більшу, ніж у конкурентів маржу між собівартістю товару і середньоринковою ціною (або ж ціною головного конкурента).

Стратегія диференціації передбачає надання товару важливих з точки зору споживача відмітних властивостей, які роблять товар відмінним від товарів конкурентів. Така відмінність може базуватися на об‘єктивних або суб‘єктивних, відчутних і невідчутних властивостях товару(у ширшому розумінні – комплексі маркетингу), бути реальною або уявною. Інструментом реалізації стратегії диференціації є ринкове позиціонування.

Стратегія спеціалізації передбачає концентрацію на потребах одного цільового сегменту, без прагнення охопити увесь ринок. Мета тут полягає в задоволенні потреб вибраного цільового сегменту краще, ніж конкуренти. Така стратегія може спиратися як на диференціацію, так і на лідерство по витратах, або і на те, і на інше, але тільки у рамках цільового сегменту. Проте низька ринкова доля у разі невдалої реалізації стратегії може істотно підірвати конкурентоспроможність компанії.

Наступним кроком є вибір стратегії конкурентної поведінки.

Стратегія лідера. Залежно від міри сформованості товарного(галузевого) ринку, характеру конкурентної боротьби компанії-лідери обирають одну з трьох стратегій:

розширення первинного попиту, оборонну або наступальну стратегію або ж застосувати демаркетинг або диверсифікацію.

Таблиця 4.15 – Визначення базової стратегії розвитку

| *No п/п* | *Обрана альтернатива розвитку проекту* | *Стратегія охоплення ринку* | *Ключові конкурентоспроможні позиції відповідно до обраної альтернативи* | *Базова стратегія розвитку** |
|---|---|---|---|---|
| 1. | Створення програми з використанням C/C++, TensorFlow | Ринкове позіціювання | Простота інтерфейсу, пришвидшення роботи | Диференціації |

Стратегія розширення первинного попиту доцільна у разі, якщо фірмі- лідерові недоцільно розмінюватися на боротьбу з невеликими конкурентами, вона може отримати велику економічну віддачу від розширення первинного рівня попиту. В цьому випадку компанія займається реалізацією заходів по формуванню попиту(навчанню споживачів користуванню товаром, формування регулярного попиту, збільшення разового споживання), також пропаганду нових напрямів застосувань існуючих товарів, виявлень нових груш споживачів.

У міру зростання ринку, його становлення позиції компанії-новатора починають атакувати конкуренти-імітатори. В цьому випадку, компанія може вибрати оборонну стратегію, метою якої є захист власної ринкової долі.

Наступальна стратегія припускає збільшення своєї частки ринку. При цьому переслідувана мета полягає в подальшому підвищенні прибутковості роботи компанії на ринку за рахунок максимального використання ефекту масштабу.

Якщо фірма потрапляє під дію антимонопольного законодавства, вона може удатися до стратегії демаркетинга, що припускає скорочення своєї частки ринку, зниження рівня попиту на деяких сегментах за рахунок підвищення ціни. При цьому ставиться завдання недопущення на ці сегменти конкурентів, а компенсація втрат

прибутку через зменшення обсягів виробництва компенсується встановленням надвисоких цін.

Стратегія виклику лідера. Стратегію виклику лідерові найчастіше вибирають компанії, які є другими, третіми на ринку, але бажають стати лідером ринку. Теоретично, ці компанії можуть прийняти два стратегічні рішення: атакувати лідера у боротьбі за частку ринку або ж йти за лідером.

Рішення атакувати лідера є досить ризикованим. Для його реалізації потрібні значні фінансові витрати, know – how, краще співвідношення «ціна- якість», переваги в системі розподілу і просування і т. д. У разі не реалізації цієї стратегії, компанія може бути відкинута на аутсайдерські позиції на досить довгий час. Залежно від цього компанія може вибрати одну з альтернативних стратегій: фронтальної або флангової атаки.

Стратегія наслідування лідеру. Компанії, що приймають слідування за лідером – це підприємства з невеликою часткою ринку, які вибирають адаптивну лінію поведінки на ринку, усвідомлюють своє місце на нім і йдуть у фарватері фірм-лідерів. Головна перевага такої стратегії – економія фінансових ресурсів, пов‘язаних з необхідністю розширення товарного(галузевого) ринку, постійними інноваціями, витратами на утримання домінуючого положення.

Стратегія заняття конкурентної ніші. При прийнятті стратегії зайняття конкурентної ніші (інші назви – стратегія фахівця або нішера) компанія в якості цільового ринку вибирає один або декілька ринкових сегментів. Головна особливість – малий розмір сегментів/сегменту. Ця конкурентна стратегія являється похідною від такої базової стратегії компанії, як концентрація. Головне завдання для компаній, що вибирають стратегію нішера або фахівця, – це постійна турбота про підтримку і розвиток своєї конкурентної переваги, формування лояльності і прихильності споживачів, підтримка вхідних бар‘єрів.

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

| *No n/n* | *Чи є проект* | *Чи буде компанія* | *Чи буде компанія копіювати основні* | |
|---|---|---|---|---|

| | *«першопрохідцем» на ринку?* | *шукати нових споживачів, або забирати існуючих у конкурентів?* | *характеристики товару конкурента, і які?* | *Стратегія конкурентної поведінки** |
|---|---|---|---|---|
| 1. | Ні | Так | Буде, а саме: основною задачею ПЗ є зберігання даних і швидкий доступ до них (конкуренти 1, 2, 3), простий інтерфейс користувача (конкурент 2) | Зайняття конкурентної ніші |

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17 – Визначення стратегії позиціонування

| *No п/ п* | *Вимоги до товару цільової аудиторії* | *Базова стратегі я розвитку* | *Ключові конкурентоспроможні позиції власного стартап- проекту* | *Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)* |
|---|---|---|---|---|
| 1. | Простота інтерфейсу, швидкість | Дифе-ренціації | Простота користувацького інтерфейсу, що дозволяє пришвидшити та спростити роботу працівників, швидкість роботи, що дозволяє підвищити швидкість експериментів | |

## 4.4 Розробка маркетингової програми

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

| *No п/п* | *Потреба* | *Вигода, яку пропонує товар* | *Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити* |
|---|---|---|---|
| 1. | Швидкість отримання даних | ПЗ враховує реальний розподіл даних, внаслідок цього краще враховує їхні особливості і забезпечує швидший доступ до даних | Переваги у швидкості |
| 2. | Спрощення інтерфейсу користувача | Простота роботи з ПЗ | Користувачам не потрібно замислюватись над тим, як саме обробити велики дані. Лише необхідно надати дані, та запустити програму. |

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 19).

1-й рівень. При формуванні задуму товару вирішується питання щодо того, засобом вирішення якої потреби і / або проблеми буде даний товар, яка його основна вигода. Дане питання безпосередньо пов'язаний з формуванням технічного завдання в процесі розробки конструкторської документації на виріб.

2-й рівень. Цей рівень являє рішення того, як буде реалізований товар в реальному/ включає в себе якість, властивості, дизайн, упаковку, ціну.

3-й рівень. Товар з підкріпленням (супроводом) - додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості , доставка, умови оплати та ін). [http://pidruchniki.com/19670511/marketing/razrabotka_produktovoy_strategii_kompanii#891] [http://marketing-helping.com/konspekti-lekcz/21-konspekt-lekczj- qosnovi-marketinguq/412-marketingove-ponyattya-ta-klasifkaczya-vidv-tovaru.html]

Таблиця 4.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Об‘єкт допомогає фахівцям у області великих даних вирішувати задачу швидкого доступу до даних. Користувач має лише завантажити необхідні дані у систему та запустити обробку.</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Простота інтерфейсу користувача 2. Швидкість роботи 3. Безпека згідно до світових стаднартів</td><td>-</td><td>-</td></tr>
<tr><td colspan="3">Якість: згідно до стандарту ISO 4444 буде проведено тестування</td></tr>
<tr><td colspan="3">Маркування відсутнє.</td></tr>
<tr><td colspan="3">Моя компанія. «Біг Дата – В»</td></tr>
<tr><td rowspan="2">III. Товар із підкріплю-нням</td><td colspan="3">1-місячна пробна безкоштовна версія та безкоштовне встановлення</td></tr>
<tr><td colspan="3">Постійна підтримка для користувачів</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: ноу-хау.</td></tr>
</table>

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. Захист може бути організовано за

рахунок захисту ідеї товару (захист інтелектуальної власності), або ноу-хау, чи комплексне поєднання властивостей і характеристик, закладене на другому та третьому рівнях товару.

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 20). Аналіз проводиться експертним методом.

Таблиця 4.20 – Визначення меж встановлення ціни

| *No n/n* | *Рівень цін на товари-замінники* | *Рівень цін на товари-аналоги* | *Рівень доходів цільової групи споживачів* | *Верхня та нижня межі встановлення ціни на товар/послугу* |
|---|---|---|---|---|
| 1. | 25000 | 30000 | 200000 | 20000 |

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 21).

Таблиця 4.21 – Формування системи збуту

| *No n/n* | *Специфіка закупівельної поведінки цільових клієнтів* | *Функції збуту, які має виконувати постачальник товару* | *Глибина каналу збуту* | *Оптимальна система збуту* |
|---|---|---|---|---|
| 1. | Купують ПЗ та роблять щорічні внески для подовження ліценції | Продаж | 0(напряму), 1(через одного посередника) | Власна та через посередників |

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 22).

Таблиця 4.22 – Концепція маркетингових комунікацій

| *No п/ п* | *Специфіка поведінки цільових клієнтів* | *Канали комунікацій, якими користуються цільові клієнти* | *Ключові позиції, обрані для позиціонування* | *Завдання рекламного повідомлення* | *Концепція рекламного звернення* |
|---|---|---|---|---|---|
| 1. | Купівля ПЗ через інтернет, робота з ПЗ на комп‘ютерах з різними ОС | Інтернет | Швидкодія, простота використання, безпека | Показати переваги ПЗ, у тому числі і перед конкурентами | Демо-ролик із використання |

## 4.5 Елементи фінансової підтримки стартапу та аналіз ризиків

Таблиця 4.23 – Сукупні інвестиційні витрати на реалізацію стартап-проекту

| No п/п | Стаття витрат | Сукупні витрати, тис. грн. |
|---|---|---|
| *1.* | *Загальні податкові витрати* | *405* |
| 1.1. | Проведення пошукових та прикладних досліджень | 5 |
| 1.2. | Розробка проектних матеріалів і ТЕО | 10 |
| 1.3. | Робоче проектування і прив'язка проекту | 10 |
| 1.4. | Витрати на придбання обладнання та устаткування та пристроїв | 100 |
| 1.5. | Витрати на придбання нематеріальних активів | 40 |
| 1.6. | Витрати на утримання обладнання та приміщень | 40 |
| 1.7. | Витрати на передвиробничі маркетингові дослідження | 50 |
| 1.8. | Витрати на створення збутової мережі | 50 |
| 1.9. | Витрати на просування та рекламу | 50 |
| 1.10. | Оплата юридичних послуг | 50 |
| *2.* | *Витрати на матеріальні ресурси* | 0 |
| 2.1. | матеріали | 0 |

| | | |
|---|---|---|
| 2.2. | комплектуючі | 0 |
| 2.3. | сировина | 0 |
| *3.* | *Витрати на оплату праці команди стартапу* | 300 |
| Разом: | | 705 |

Таблиця 4.24 – Визначення основних фінансово-економічних показників проекту

| *No з/п* | *Стаття витрат* | *Сукупні витрат и, тис. грн.* |
|---|---|---|
| 1. | Обсяг виробництва продукції в натуральних показниках | 1 |
| 2. | Собівартість одиниці продукції, тис. грн. | 705000 |
| 3. | Собівартість виробництва продукції, тис. грн. (3 = 1 · 2) | 705000 |
| 4. | Обсяг реалізації продукції в натуральних показниках | 100 |
| 5. | Ціна реалізації продукції без ПДВ, тис. грн. | 20000 |
| 6. | Виручка від реалізації продукції без ПДВ, тис. грн. (6 = 4 · 5) | 2000000 |
| 7. | Податок на додану вартість (ПДВ), тис. грн. | 200000 |
| 8. | Валовий прибуток (8 = 6 – 3) | 1295000 |
| 9. | Податок на прибуток | 259000 |
| 10. | Чистий прибуток (10 = 8 – 9) | 1036000 |

Рентабельність продажів (або маржа прибутку) показує, скільки прибутку приносить кожна гривня з обсягу реалізації. Маржу прибутку, як правило, визначають окремо за кожним видом діяльності або за кожною групою реалізованої продукції за формулою:

$$Rs = \frac{\text{П}}{\text{В}} \times 100\%,$$

де П – прибуток;

В – виручка від реалізації продукції.

$$Rs = 51.8\%$$

Період окупності проекту відображає час, який потрібен для того, щоб

сума надходжень від реалізації проекту відшкодувала суму витрат на його впровадження. Період окупності звичайно вимірюється в роках або місяцях та може бути розрахований за формулою:

$$PBP = \frac{II}{ACI}$$

де PBP – період окупності інвестицій, роки;

II – сума інвестиційних витрат, тис. грн.;

ACI – щорічні надходження (річний чистий прибуток), тис. грн.

$$PBP = 0.35(\text{роки})$$

Рентабельність довгострокових інвестицій – коефіцієнт повернення інвестицій, показник рентабельності вкладень, що у відсотковому співвідношенні демонструє прибутковість (у разі, коли його значення більше 100%) або збитковість (у разі, коли його значення менше 100%) інвестицій в проект. Якщо розрахований показник менший 100%, то інвестиційні вкладення не окупуються. Показник може бути розрахований за формулою:

$$ROI = \frac{\text{Д} - \text{С}}{II} \cdot 100\%$$

де Д – дохід (виручка від реалізації продукції), тис. грн.;

С – повна собівартість, тис. грн.;

II – сума інвестиційних витрат, тис. грн.

$$ROI = 46.9\%$$

Таблиця 4.25 – Програма запобігання та реагування на ризики проекту

| *Ризики проекту* | *Заходи запобігання ризиків* | *Заходи реагування при виникненні ризиків* |
|---|---|---|
| Вихід з ладу системи контролю версій | Збереження копій вихідних кодів проекту, проектної документації та інших даних на інших серверах | Отримання копії даних з інших серверів |
| Звільнення члена команди | Детальна декомпозиція завдань, щоб зробити кожне з них максимально простим та незалежним. | Продовження роботи без цієї людини |

| | | |
|---|---|---|
| | Використання систем контролю версій. | |
| Збільшення собівартості через проблеми роботи з бібліотеками | Немає | Необхідні додаткові час і фінансування для вирішення проблем |
| Зміна системного коду TensorFlow | Приймати участь в обговореннях переваг та недоліків цієї технології на офіційному сайті TensorFlow, абстрагувати деякі рівні ПЗ для незалежності від системного API | Змінити код у тих місцях, де використовується системне API |

## 4.6 Висновок

Згідно до проведених досліджень існує можливість ринкової комерціалізації проекту. Також, варто відмітити, що існують перспективи впровадження з огляду на потенційні групи клієнтів, бар‘єри входження не є високими, а проект має дві значні переваги перед конкурентами.

Для успішного виконання проекту необхідно реалізувати програму із використанням засобів C/C++ та TensorFlow. В рамках даного дослідження були розраховані основні фінансово-економічні показники проекту, а також проведений менеджмент потенційних ризиків. Проаналізувавши отримані результати, можна зробити висновок, що подальша імплементація є доцільною.

# ВИСНОВКИ

У даній роботі було досліджено переваги та недоліки класичних індексних структур для точкового пошуку, а також, в результаті цього аналізу, запропоновано і реалізовано дві адаптивні індексні структури, що працюють на базі лінійної регресії на нейронної мережі.

По-перше, було проведено загальний огляд і аналіз класичних індексних структур для точкового пошуку. Основною структурою для вирішення цієї задачі є хеш-таблиця. Було показано способи побудови і функціонування хеш-таблиць, описано їхні основні проблеми, зокрема колізії ключів, а також традиційні способи вирішення цих проблем. Крім цього, було зазначено особливості побудови хеш-таблиць для використання у базах даних.

Аналіз класичних рішень проблеми колізій показав, що всі такі рішення погіршують ефективність роботи хеш-таблиць. Тому було запропоновано альтернативні рішення, що базуються на використанні підходів машинного навчання.

Було показано, що для зменшення кількості колізій, в якості хеш-функції доцільно використовувати функцію розподілу даних. При цьому, внаслідок особливостей точкового пошуку, порядок цих даних є неважливим. Крім цього, було також показано, що для апроксимації цієї функцію варто використовувати моделі машинного навчання.

Було запропоновано реалізацію адаптивної індексної структури, що складається з двох частин: загальної хеш-таблиці та хеш-функції, що може базуватись на моделях машинного навчання.

При виборі моделей машинного навчання варто було враховувати, що вони мають працювати доволі швидко. Внаслідок цього, було запропоновано використання лінійної регресії на нейронної мережі з одним прихованим шаром, адже такі моделі є простими та ефективними.

Було реалізовано адаптивні індексні структури для точкового пошуку, що базуються на використанні зазначених моделей машинного навчання. Для реалізації

цих індексних структур були використані сучасні засоби програмування на бібліотеки для машинного навчання, що значно полегшують процес тренування та використання моделей.

Після цього були проведені експерименти з реалізованими індексними структурами на двох наборах даних. Результати експериментів показали, що на деяких наборах даних запропоновані адаптивні індексні структури показують значно меншу кількість колізій за класичні аналоги, та, внаслідок цього, більш ефективне використання пам’яті. З іншого боку, на тих наборах даних, в яких розподіл даних містить велику кількість груп, що йдуть послідовно, та значні розриви між цими групами, запропоновані індесні структури показали дещо гірші результати використання пам’яті. Крім цього, результати експериментів також показали, що використання лінійної регресії (зокрема, її узагальненої поліноміальної версії) у ролі хеш-функції, надає дещо кращі результати з точки зору ефективності використання пам’яті в порівнянні з нейронною мережею з одним прихованим шаром.

Внаслідок того, що розроблені адаптивні індексні структури для точкового пошуку показують більш ефективне використання пам’яті на деяких наборах даних, їх можна інтегрувати у сучасні системи керування базами даних, за умови подальшого вдосконалення.

Важливо також зауважити, що результати, отримані в цій роботі, підтверджують загальну ідею доцільності використання машинного навчання для заміни класичних індексних структур.

# ПЕРЕЛІК ПОСИЛАНЬ


1. The Zettabyte Era: Trends and Analysis [Електронний ресурс] – Режим доступу до ресурсу: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html.
2. База даних [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/База_даних.
3. T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. The Case for Learned Index Structures [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1712.01208.
4. Moore Law is Dead but GPU will get 1000X faster by 2025 [Електронний ресурс] – Режим доступу до ресурсу: https://www.nextbigfuture.com/2017/06/moore-law-is-dead-but-gpu-will-get-1000x-faster-by-2025.html.
5. Зображення Б-дерева [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%91-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE#/media/File:B-Baum.svg.
6. Геш-таблиця [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Геш-таблиця.
7. Robert Sedgewick. Algorithms in C++, Parts 1-4: Fundamentals, Data Structure, Sorting, Searching, Third Edition / Robert Sedgewick. – Addison-Wesley, 1998. – С. 587.
8. Розв'язання колізій за допомогою ланцюжків [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Геш-таблиця#/media/File:Hash_table_5_0_1_1_1_1_1_LL.svg.
9. Розв'язання колізій за допомогою методу відкритої адресації [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Геш-таблиця#/media/File:Hash_table_5_0_1_1_1_1_0_SP.svg.

10. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom. Database Systems, The Complete Book, Second Edition / Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom. – Pearson Prentice Hall, 2009. – С. 619.
11. Database index [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Database_index.
12. Hashed Sharding [Електронний ресурс] – Режим доступу до ресурсу: https://docs.mongodb.com/manual/core/hashed-sharding/.
13. Probability in Hashing [Електронний ресурс] – Режим доступу до ресурсу: https://www2.cs.duke.edu/courses/cps102/spring09/Lectures/L-18.pdf.
14. Linear Regression [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Linear_regression.
15. Least squares scheme [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_least_squares_example2.png.
16. CS229 Lecture notes [Електронний ресурс] – Режим доступу до ресурсу: http://cs229.stanford.edu/notes/cs229-notes1.pdf.
17. Design Patterns: Elements of Reusable Object-Oriented Software / Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. – Addison-Wesley, 1994. – С. 349.
18. Artificial Neural Network [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Artificial_neural_network.
19. Штучний нейрон [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Штучний_нейрон.
20. Activation Functions: Neural Networks [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.
21. Glorot X. Deep sparse rectifier neural networks / X. Glorot, A. Bordes, Y. Bengio. // 14th International Conference on Artificial Intelligence and Statistics. – 2011. – С. 315–323.

22. Sebastian Ruder. An overview of gradient descent optimization algorithms [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1609.04747.
23. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/.
24. The R Project for Statistical Computing [Електронний ресурс] – Режим доступу до ресурсу: https://www.r-project.org/.
25. MATLAB [Електронний ресурс] – Режим доступу до ресурсу: https://www.mathworks.com/products/matlab.html.
26. TensorFlow [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/.
27. Keras [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/.
28. PyTorch [Електронний ресурс] – Режим доступу до ресурсу: https://pytorch.org/.
29. Credit Card Fraud Detection Dataset [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/mlg-ulb/creditcardfraud.
30. US Population By Zip Code Dataset [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/census/us-population-by-zip-code.
31. Scikit-Learn: machine learning in Python [Електронний ресурс] – Режим доступу до ресурсу: http://scikit-learn.org/.