hillel IT school

Lesson 15

План заняття

- Stack
- Linked list
- Queue
- Binary tree
- Hash table

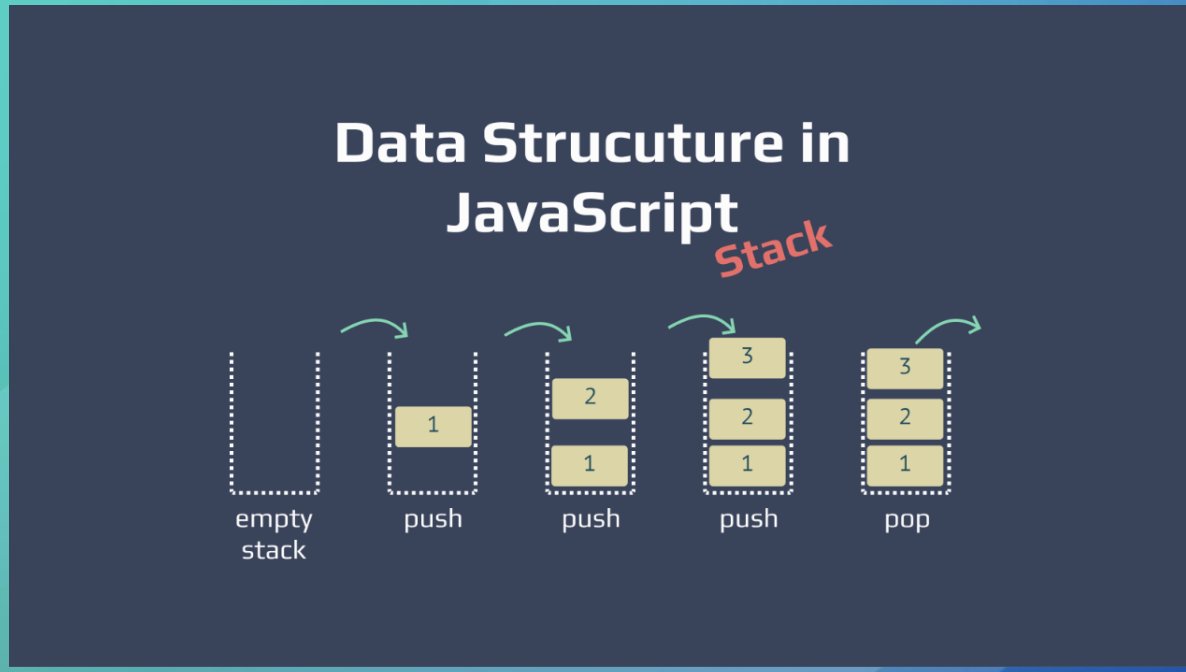
Stack

Це базова, лінійна структура даних, яка дозволяє додавати чи видаляти елементи лише на її початку.

Принцип - першим прийшов – останнім вийшов.

Основні операції:

- Push – додати;
- Pop – видалити;
- toArray – перетворити на масив та показати;
- Peek – забрати.



Складність операцій

Временная сложность стека

Алгоритм	Среднее значение	Худший случай
Space	O(n)	O(n)
Search	O(n)	O(n)
Insert	O(1)	O(1)
Delete	O(1)	O(1)

Stack

Стек корисний коли ми хочемо додати дані у послідовному порядку та видалити ці дані.

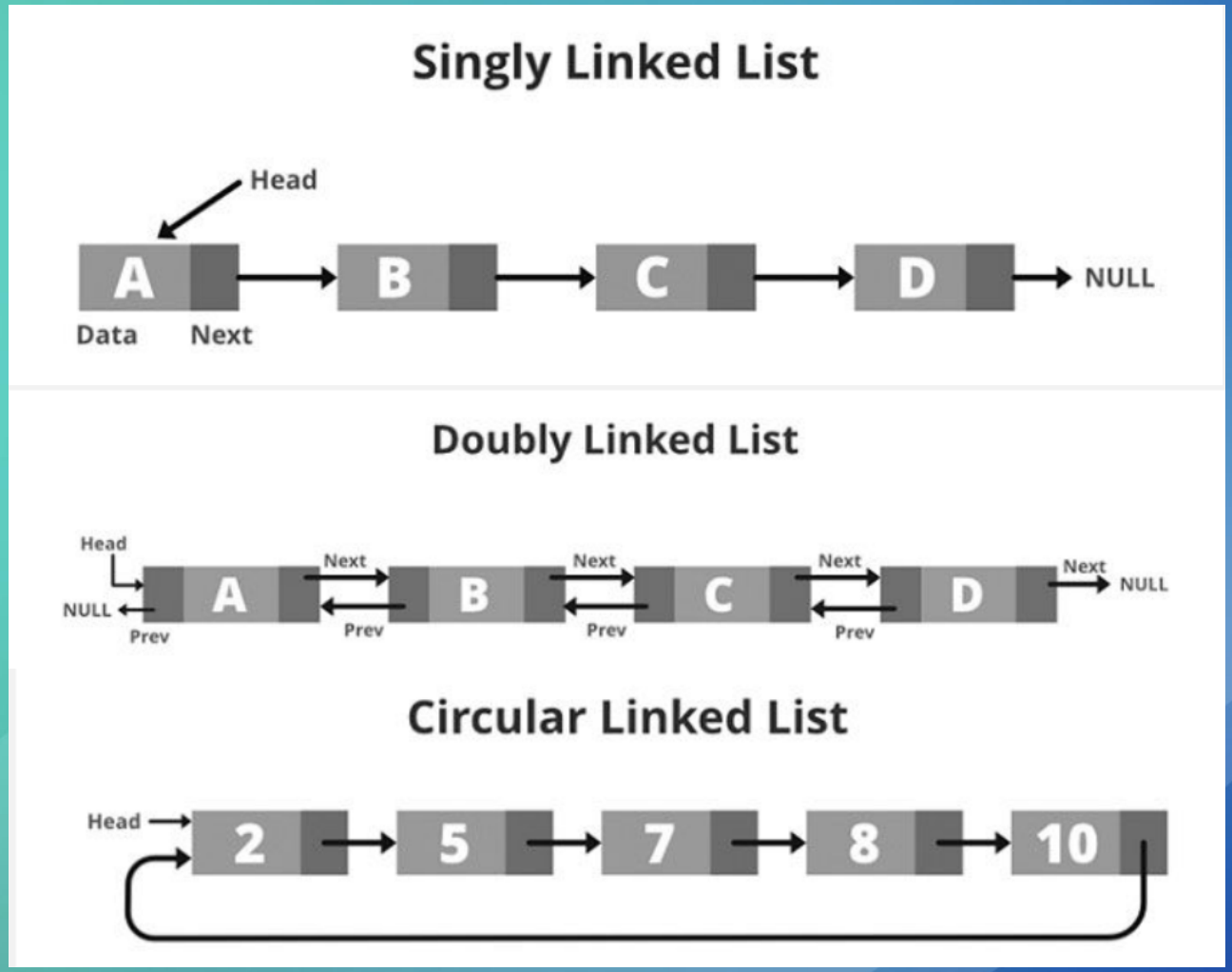
Приклад: call stack

Linked list

Групи вузлів, що разом утворюють послідовність. Кожен вузол містить дві речі: фактичні дані, які в ньому зберігаються і вказівна на наступний вузол у послідовності.

Основні операції:

- Додавання;
- Видалення;
- Пошук елемент;
- Отримання довжини;
- Перевірка на порожнечу;
- Трансформація в масив.



Складність операцій

Временная сложность связного списка

Алгоритм	Среднее значение	Худший случай
Space	O(n)	O(n)
Search	O(n)	O(n)
Insert	O(1)	O(1)
Delete	O(1)	O(1)

Linked list

Концепція використання вузлів, які вказують один на одного, використовується в багатьох інших структурах даних, вбудованих у багато мов програмування вищого рівня. Гарне розуміння того, як працюють зв'язкові списки, важливе для загального розуміння створення та використання інших структур даних.

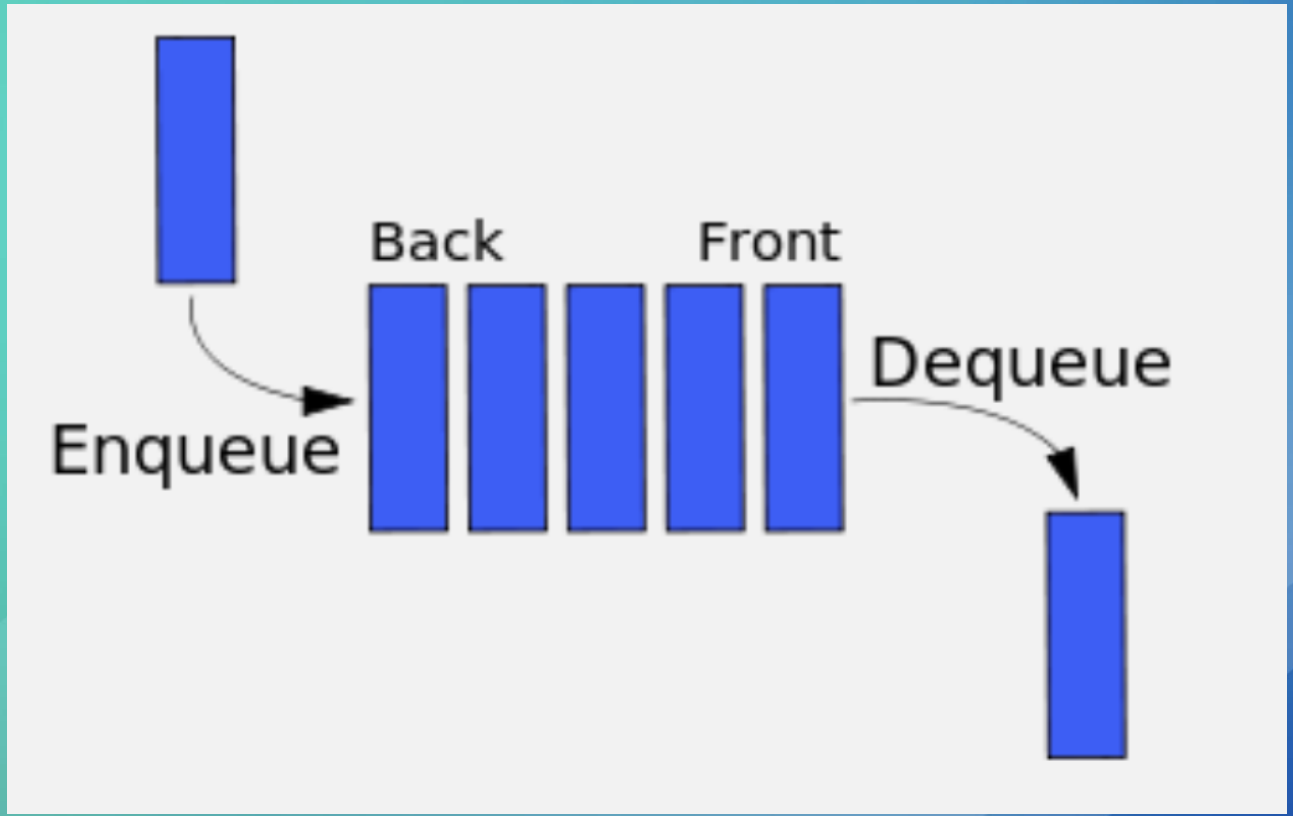
Приклад: слухачі подій C++

Queue

Принцип - першим прийшов – першим вийшов. Це означає, що видалити елемент можна тільки після того, як були прибрані раніше додані елементи.

Основні операції:

- Enqueue – додати до кінця.
- Dequeue – видалити перший елемент.
- Peek – забрати перший.



Складність операцій

Временная сложность очереди

Алгоритм	Среднее значение	Худший случай
Space	O(n)	O(n)
Search	O(n)	O(n)
Insert	O(1)	O(1)
Delete	O(1)	O(1)

Різниця між стеком та чергою

hillel IT school

	Stack	Queue
Принцип	Останнім прийшов, першим вийшов	Першим прийшов, першим вийшов
Склад	Та ж сама ячейка пам’яті використовується для додавання і видалення.	Заданий конець використовується для вставки, перший для видалення.
Напрямок	Один	Два
Оперції	Push & Pop	Enqueue & Dequeue
Порожній стан	Top === -1	Front === -1 Front === Back + 1
Повний стан	Top === Max - 1	Заданий стан === Max - 1
Варіанти	Немає	Кільцева, двонаправлена
Реалізація	Легко	Складно

Queue

Черга зберігає дані у послідовному порядку, але видаляє найстаріші додані дані.

Приклад: події веб-браузера.

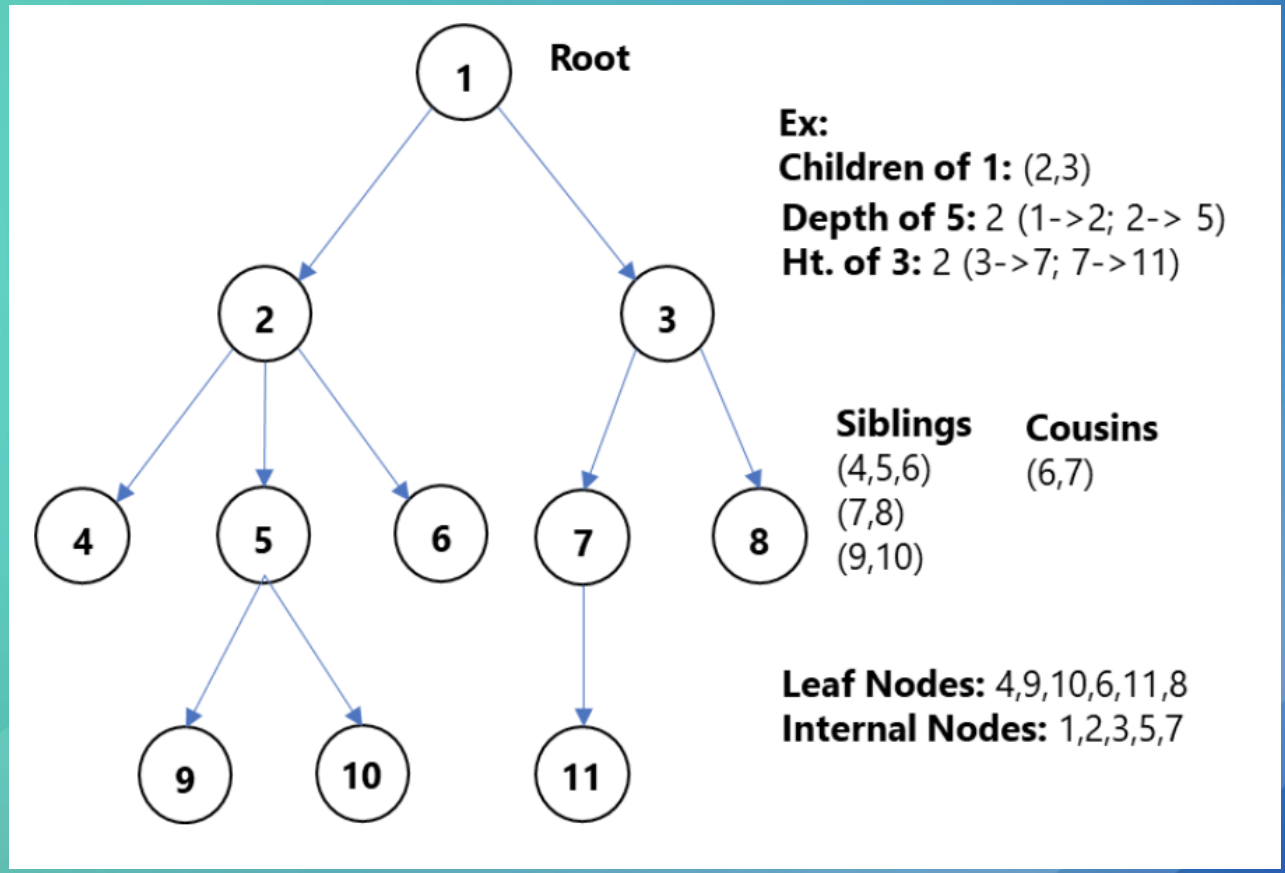
Binary tree

Це структура даних, що складається із вузлів. Їй притаманні такі властивості:

1. Кожне дерево має кореневий вузол (нагорі).
2. Кореневий вузол має нуль чи більше дочірніх вузлів.
3. Кожен дочірній вузол має нуль або більше дочірніх вузлів, і таке інше.

Основні операції:

- Додавання.
- Пошук



Терміни

- Root: Вузол без батьківського елемента. Самий верхній вузол дерева.
- Children: Вузли безпосереднього наступника.
- Siblings: Діти одного батька.
- Leaf node: Будь-який вузол, який не має дітей
- Internal Nodes: Усі вузли з принаймні одним дочірнім елементом
- Depth of Node x: Довжина шляху від кореня до вузла x
- Height of Node x: Кількість ребер на найдовшому шляху від вузла x до кінцевого вузла
- Height of tree: Висота дерева - це висота кореня
- Level of node: Сукупність усіх вузлів на заданій глибині називається рівнем дерева

Обхід графів

Обхід графа - це перехід від однієї його вершини до іншої з цілю пошуків властивостей та зв’язків.

Варіанти пошуку:

1. Breath – First – Search - Пошук у ширину;
2. Deph – First – Search - Пошук у глибину;

Breath – First – Search

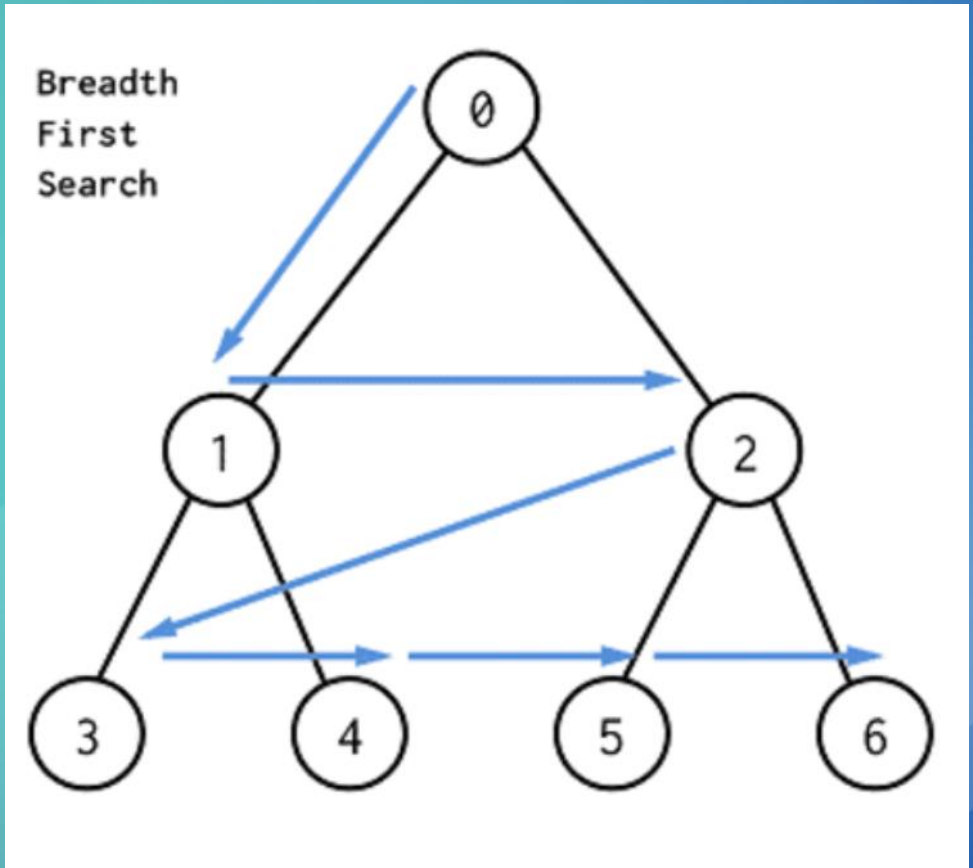
Концепція - Розширюйся на висоту пташиного польоту (go wide, birds eye view).

Ідея рухатись вперед по одному сусіду до іншого. Цей підхід вивчає усіх сусідів у межах одного кроку. Гарно підходить для перебору варіанти черги. До черги додається вершина, потім її сусіди, і так доки ми не знайдемо данні.

Час виконання: O (V + E)

- V - сума вершин.
- E - сума граней.

Складність: O (V)



Deph – First – Search

Концепція - Розширюйся на висоту пташиного польоту (go wide, birds eye view).

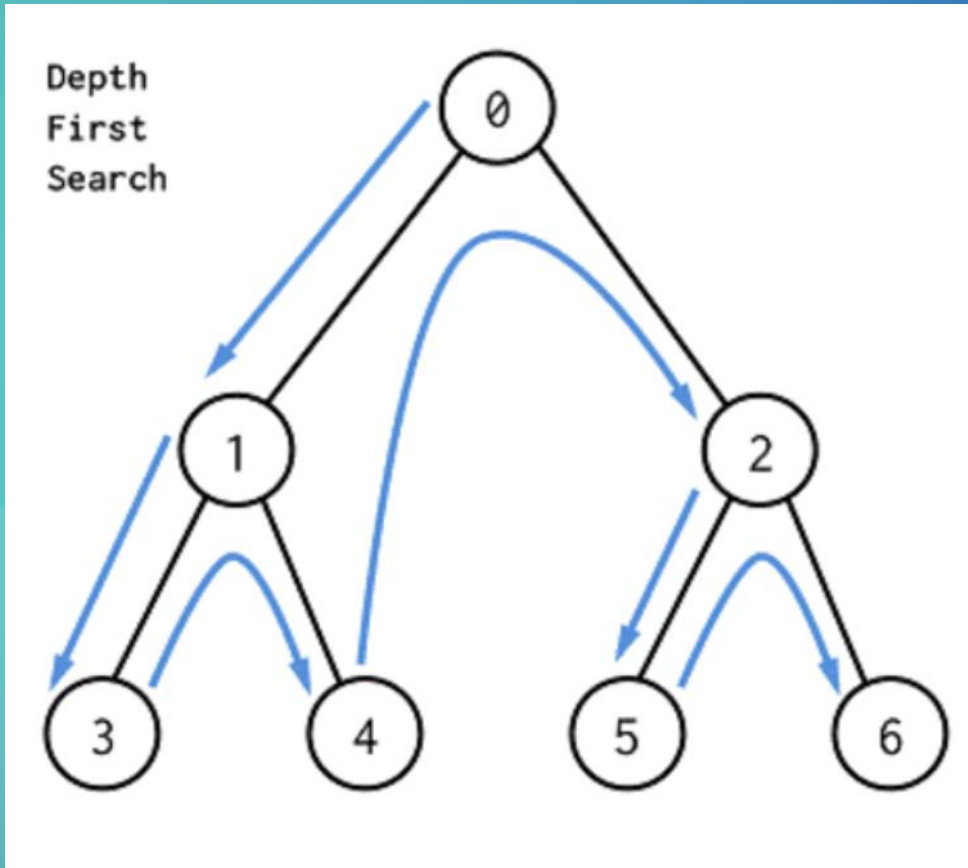
Ідея рухатись від начальної вершини, в одному напрямку, доки не дійдемо до кінця шляху або пункту призначення. Якщо ми дійшли до кінця, а то не є нашою метою, то повернись до першої розвилки, та їди іншим шляхом.

Реалізована через рекурсію. Може використовуватись для пошуку маршруту додавання.

Час виконання: O (V + E)

- V - сума вершин.
- E - сума граней.

Складність: O (V)



Складність операцій

Временная сложность двоичного дерева поиска

Алгоритм	Среднее значение	Худший случай
Space	O(n)	O(n)
Search	O(log n)	O(n)
Insert	O(log n)	O(n)
Delete	O(log n)	O(n)

Binary tree

Двійкові дерева пошуку дозволяють швидко знаходити, додавати та видаляти елементи. Вони влаштовані так, що час кожної операції пропорційний до логарифму загального числа елементів у дереві.

Приклад: Часткова реалізація у DOM

Hash table

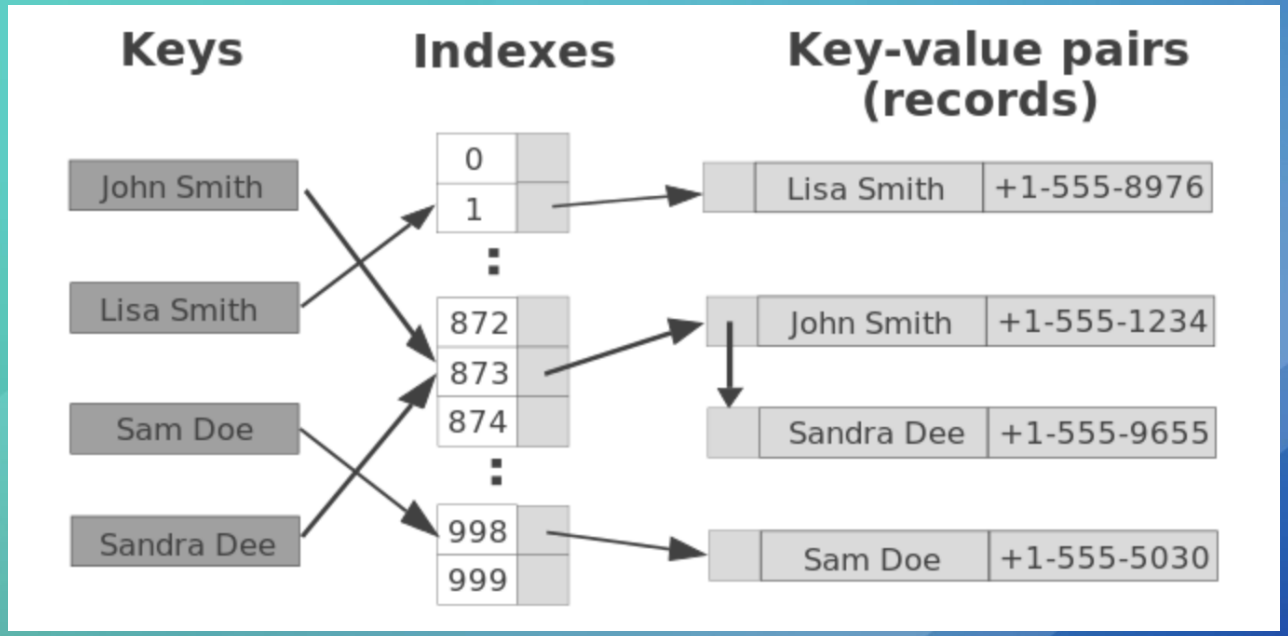
Є структурою даних, яка відображає ключі та значення.

Вона використовує хеш-функцію для обчислення індексу в масиві блоків даних, щоб знайти бажане значення.

Зазвичай хеш-функція приймає рядок символів як вступні дані і виводить числове значення. Для того самого введення хеш-функція повинна повертати однакове число.

Основні операції:

- Insert – додати;
- Search – знайти;
- Remove – видалити;
- Print - показати.



Хеш функція

Хеш-функція повинна перетворювати ключ із пари ключ-значення на індекс масиву, в якому зберігаються значення. Проблема функції у тому, що з двох різних ключів може повернути той самий індекс. Такі ситуації називають колізіями.

Два шляхи вирішення коллізій:

1. Відкрита адресація (Open addressing) - Припустимо що ключ вже присутній у таблиці. У такій ситуації ми починаємо ітерування за індексами у пошуках вільного місця – такий спосіб називається лінійне пробування.
2. Метод ланцюжків (Separate chaining) - При такому підході в масиві зберігаються ключі, а пов'язані списки ключів. У разі колізії хеш-функції новий ключ додається до того списку, який лежить за тим же індексом.

Складність операцій

Временная сложность хэш-таблицы

Алгоритм	Среднее значение	Худший случай
Space	O(n)	O(n)
Search	O(1)	O(n)
Insert	O(1)	O(n)
Delete	O(1)	O(n)

Hash table

Хеш-таблиці використовуються для реалізації структур даних карт і наборів у більшості поширених мов програмування. У C++ і Java вони є частиною стандартних бібліотек, тоді як Python і Go мають вбудовані словники та карти.

Приклад: Map()

Дякую за увагу