Data Structures and Algorithms

- Title: Data Structures and Algorithms
- Subtitle: Stacks, FIFO Queues, Sorting Algorithms, and Shortest Path Algorithms
- Presented by: Nguyễn Duy Tùng Lâm
- Date: 16/10/2024

Table of Contents

- Stack Abstract Data Type (ADT)
- FIFO Queue Concrete Data Structure
- Sorting Algorithms
- Shortest Path Algorithms

What is a Stack?

- Definition:

- A stack is a linear data structure that follows the Last In First Out (LIFO) principle.

- The last element added is the first to be removed.

- Common Use Cases:

- Undo functionality

- Call stack in programming

- Expression evaluation.

Specifying Input Parameters

Content:

- How input parameters are used in operations (e.g., values to push onto a stack, elements to search).

Example:

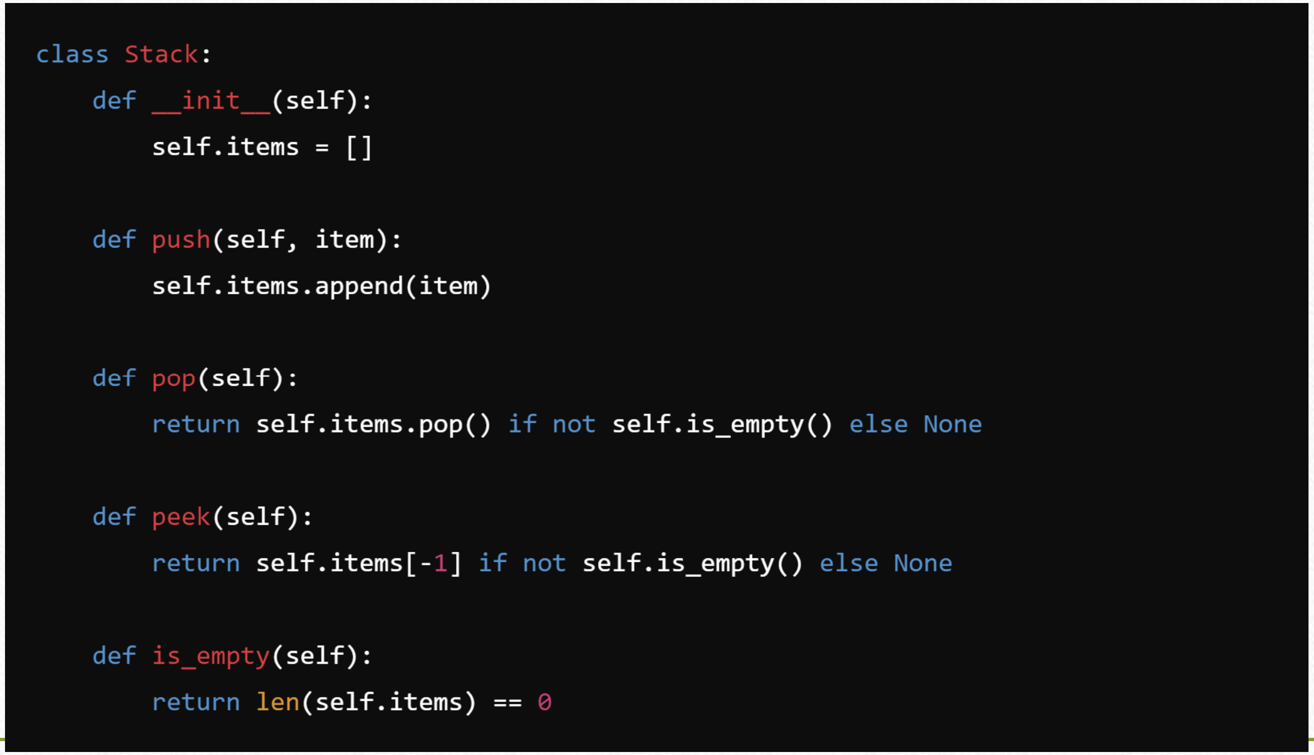
- For stack operations, the input could be an integer or string.
- Visuals:
- Diagram showing function inputs and outputs.

Stack ADT Operations

- **Push:** Adds an element to the top of the stack.
- **Pop:** Removes the element from the top of the stack.
- **Peek/Top:** Views the element at the top without removing it.
- **IsEmpty:** Checks if the stack is empty.

Stack ADT - Example

- Python Code:



```python
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop() if not self.is_empty() else None

    def peek(self):
        return self.items[-1] if not self.is_empty() else None

    def is_empty(self):
        return len(self.items) == 0
```

Stack ADT Implementation

- Linked List vs. Array Implementation:
- Linked List:

- Dynamic size

- No memory wastage

- Array:

- Fixed size

- Faster access time

- Pros & Cons:

- Linked List consumes more memory but is more flexible.

- Array is simpler but has a fixed size.

Use Cases of Stacks

- Programming Call Stack: Handles function calls.
- Undo/Redo Functionality: Tracks user actions.
- Expression Parsing: Evaluates mathematical expressions.

What is a FIFO Queue?

- Definition:
  - A queue is a linear data structure that follows the First In First Out (FIFO) principle.
  - The first element added is the first to be removed.
- Common Use Cases:
  - CPU scheduling
  - Print queue management
  - Network data packet handling

Queue Operations Enqueue: Adds an element to the back of the queue.

- Dequeue: Removes the element from the front of the queue.
- Peek/Front: Views the element at the front without removing it.

IsEmpty: Checks if the queue is empty..

Queue Implementation Using Arrays or Linked Lists: Array: Simple but has a fixed size.

- Linked List: Dynamic size, more flexible.
- Circular Queues: Efficient use of space in arrays.

Queue Implementation Example

- Code example (in Python, C++, or Java)

Applications of Queues

- CPU Scheduling
- Print Queue Management
- BFS in Graphs

Introduction to Sorting Algorithms

- Importance of Sorting
- Categories: Comparison-based & Non-comparison-based.

Selection Sort

- Overview of Selection Sort
- Step-by-step Example

Selection Sort - Time Complexity

- Best, Average, and Worst Case Analysis
- Space Complexity

Merge Sort

- Overview of Merge Sort
- Divide and Conquer Approach

Merge Sort - Time Complexity

- Best, Average, and Worst Case Analysis
- Space Complexity

comparison of Sorting Algorithms

- Comparison Table: Selection Sort vs. Merge Sort

Introduction to Shortest Path Algorithms

- Importance in Graph Theory
- Use Cases: GPS, Network Routing

Dijkstra’s Algorithm

- Overview of Dijkstra’s Algorithm
- Example with Step-by-step Explanation

Dijkstra’s Algorithm - Time Complexity

- Best, Average, and Worst Case Analysis

Bellman-Ford Algorithm

- Overview of Bellman-Ford Algorithm
- Step-by-step Example

Bellman-Ford Algorithm - Time Complexity

- Best, Average, and Worst Case Analysis

Comparison of Shortest Path Algorithms

- Dijkstra vs. Bellman-Ford

Applications of Shortest Path Algorithms

- Network Routing
- Road Navigation Systems
- Packet Switching in Networks

Use Cases in Real-World Systems

- Examples of how these data structures and algorithms are used in software systems

Conclusion

- Summary of Key Points

Quiz/Discussion

- Questions to engage the audience
- Example: "What is the time complexity of Dijkstra's Algorithm?"

References

- Books, Articles, Online Resources