Installation

Prerequisites

- Python 3.8 or higher
- Required libraries:
  - `networkx`
  - `matplotlib`
  - `numpy`
  - `pandas`

Installation Steps

1. Clone the repository:
```
2. git clone <repository-url>
```
3. Navigate to the project directory:
```
4. cd data-analysis-platform
```
5. Install dependencies:
```
6. pip install -r requirements.txt
```
7. Run the program:
```
8. python main.py
```

Module Overview

Graph Module

- **File:** `graph_module.py`
- **Purpose:** Manages tasks and their dependencies using a DAG.
- **Functions:**
  - `add_task(task, dependencies=[])`: Adds a task to the graph with optional dependencies.
  - `detect_cycle()`: Checks for cycles in the graph.
  - `topological_sort()`: Performs topological sorting.
  - `visualize()`: Visualizes the graph.

Data Operations Module

- **File:** `data_operations_module.py`
- **Purpose:** Provides sorting and searching functionalities.
- **Functions:**
  - `merge_sort(arr)`: Sorts an array using Merge Sort.
  - `quick_sort(arr)`: Sorts an array using Quick Sort.
  - `binary_search(arr, target)`: Performs Binary Search on a sorted array.
  - `linear_search(arr, target)`: Performs Linear Search on an array.

Stack and Queue Module

- **File:** `stack_queue_module.py`
- **Purpose:** Implements stack and queue operations.
- **Functions:**
  - `push_stack1(value)`: Pushes a value onto Stack 1.
  - `push_stack2(value)`: Pushes a value onto Stack 2.
  - `pop_stack1()`: Pops a value from Stack 1.
  - `pop_stack2()`: Pops a value from Stack 2.
  - `enqueue(value)`: Adds a value to the queue.
  - `dequeue()`: Removes a value from the queue.

Performance Module

- **File:** `performance_module.py`
- **Purpose:** Measures runtime and memory usage.
- **Functions:**
  - `benchmark(func, *args, **kwargs)`: Benchmarks a function's runtime and memory usage.

Usage

Graph Module

1. Add tasks with dependencies:

```
graph.add_task("Task1")
graph.add_task("Task2", ["Task1"])
graph.add_task("Task3", ["Task2"])
```

5. Detect cycles:

```
print(graph.detect_cycle())
```

7. Perform topological sorting:

```
print(graph.topological_sort())
```

9. Visualize the graph:

```
 graph.visualize()
```

Data Operations Module

1. Sort an array:

```
arr = [3, 2, 1, 5, 4]
sorted_arr = DataOperationsModule.quick_sort(arr)
print(sorted_arr)
```

5. Search in an array:

```
print(DataOperationsModule.binary_search(sorted_arr, 3))
```

Stack and Queue Module

1. Perform stack operations:

```
stack_queue.push_stack1(10)
print(stack_queue.pop_stack1())
```

4. Perform queue operations:

```
queue.enqueue(10)
print(queue.dequeue())
```

Performance Module

1. Benchmark a function:

```
performance.benchmark(DataOperationsModule.quick_sort, [3, 2, 1, 5, 4])
```