**Weather Data Storage System Report**

**Course:** Data Structures (ENCS205 / ENCA 201)

**Assignment No:** 01

**Session:** 2025-26

**Semester:** 3rd

**Submitted By:** Ishan Jha

**Class:** BTech CSE Core, Section-B

**Roll No:** 2401010022

**Submitted To:** Dr. Swati Gupta

## 1. Introduction

This project develops a **Weather Data Storage System** to systematically store and retrieve temperature records based on **city** and **year**. The system uses **Abstract Data Types (ADT)** and **2D arrays** for structured storage and supports **sparse datasets**, enabling memory-efficient storage when some records are missing.

## 2. Weather Record ADT

**Attributes:** - **Date:** String in the format dd/mm/yyyy, e.g., “01/06/2025” - **City:** String, e.g., “Gurgaon” - **Temperature:** Float/Double, e.g., 34°C

**Methods:** - insert(record) → Add a weather record - delete(city, date) → Remove a record for a city on a given date - retrieve(city, year) → Retrieve temperature for the city in that year - traverse() → Iterate through all records

## 3. Memory Representation Strategy

### Row-Major Order

* Rows represent years (2025, 2026)
* Columns represent cities (Gurgaon)
* Traversal accesses all cities for a year before moving to the next

### Column-Major Order

* Columns represent cities
* Rows represent years
* Traversal accesses all years for a city before moving to the next

## 4. Handling Sparse Data

* **Dense Matrix with Sentinel Values:** Missing entries represented as None
* **Sparse Dictionary Representation:** Stores only existing records as (year, city): temperature

**Example Sparse Dictionary:**

{ (2025, 'Gurgaon'): 34,  
 (2026, 'Gurgaon'): 31 }

This allows memory-efficient storage for incomplete datasets.

## 5. Time and Space Complexity

| Operation | Dense (2D Array) | Sparse (Dictionary) |
| --- | --- | --- |
| Insert | O(1) | O(1) |
| Delete | O(1) | O(1) |
| Retrieve | O(1) | O(1) |
| Row Traversal | O(n × m) | O(k) |
| Column Traversal | O(n × m) | O(k) |

* n = number of years (2)
* m = number of cities (1)
* k = number of filled records

**Space Complexity:** - Dense → O(n × m) - Sparse → O(k)

## 6. Sample Execution

from weather\_ds import WeatherRecord, DataStorage  
  
years = [2025, 2026]  
cities = ["Gurgaon"]  
ds = DataStorage(years, cities, sentinel=None)  
  
r1 = WeatherRecord("01/06/2025", "Gurgaon", 34)  
r2 = WeatherRecord("15/07/2026", "Gurgaon", 31)  
  
ds.insert(r1)  
ds.insert(r2)  
  
print(ds.retrieve("Gurgaon", 2025))  
print(ds.row\_major\_access())  
print(ds.handle\_sparse\_data())

**Output:**

34  
[(2025, 'Gurgaon', 34), (2026, 'Gurgaon', 31)]  
{'total': 2, 'filled': 2, 'missing': 0}

## 7. Conclusion

The Weather Data Storage System demonstrates: - Efficient storage and retrieval of weather records using **ADT and 2D arrays** - **Row-major and column-major traversal** strategies - **Sparse data handling** for memory-efficient storage - Predictable **time and space complexity** for operations

This system meets all **assignment objectives** and provides a practical example of structured data management.

## 8. Project Structure

├── weather\_ds.py # ADT & DataStorage implementation  
├── main.py # Demo / Example usage  
├── README.md # Project description and report  
└── report.docx # Assignment report

**End of Report**