Topic: Advanced Directory Structure with Path Finding, Search, and Optimization
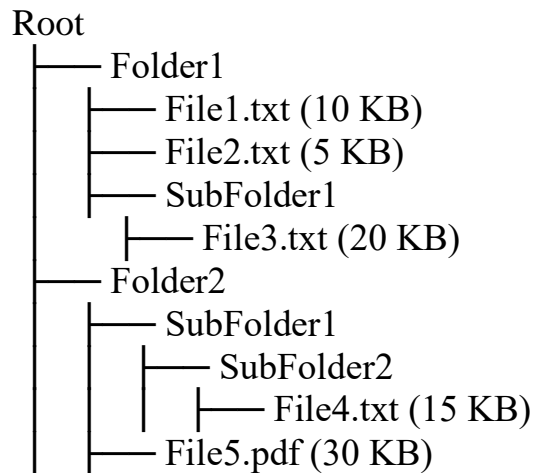
Overview: In modern computing environments, **file systems** need to handle **large-scale directory structures** with millions of files efficiently. The goal of this project is to create a **directory management system** that can:

- **Locate files quickly** using optimized search algorithms.
- **Compute storage usage of directories recursively**.
- **Identify duplicate files and recommend storage optimizations**.
- **Support wildcard and metadata-based search queries**.
- **Optimize pathfinding for accessing deeply nested files efficiently**.

This project will **simulate a file system** where directories contain subdirectories and files. Using **graph and tree-based algorithms**, the system will efficiently **search, traverse, and manage** file structures while optimizing storage.

Example directory structure:

```
Root
├── Folder1
│   ├── File1.txt (10 KB)
│   ├── File2.txt (5 KB)
│   ├── SubFolder1
│   │   ├── File3.txt (20 KB)
├── Folder2
│   ├── SubFolder1
│   │   ├── SubFolder2
│   │   │   ├── File4.txt (15 KB)
│   ├── File5.pdf (30 KB)
```

2. Functionalities

2.1 Find a File and Show Its Path

- Given a **file name**, find its location in the directory tree.
- **Output:** The **absolute path** from the root.

Example Query:
Find File: File4.txt
Path Found: Root/Folder2/SubFolder1/SubFolder2/File4.txt
Search Time: 0.0021 seconds

2.2 Compute Total Size of a Folder

- Given a **folder name**, calculate the **total storage used** by its contents (including nested subfolders).

Example Query:
Folder: Folder2
Total Size: 85 KB
Computation Time: 0.0012 seconds

2.3 Optimized Pathfinding in Deeply Nested Directories

- Find the **fastest access path** to a file in a deeply nested structure.

Example Query:
Find Shortest Access Path to: File4.txt
Shortest Path: Root → Folder2 → SubFolder1 → SubFolder2
Path Cost: 4 hops
Algorithm Used: A* Search
Execution Time: 0.0008 seconds

2.4 Advanced Search Queries (Wildcard, Metadata-Based)
Search for files using wildcards, file size, type, or last access date. Example Queries:
Find all .txt files in Folder1 → Outputs list of all .txt files.
Find files larger than 20 KB in Folder2 → Lists large files.

Example Output:
Search Query: Find all .txt files in Folder1
Results:
- File1.txt (10 KB)
- File2.txt (5 KB)