בס"ד

Time and Space Complexity Analysis - Part A_1 :

Code Summary: What it does

- Reads a log file in Excel format using `pandas.read_excel`.
- Splits (chunks) the lines into smaller files.
- Counts error codes in each chunk concurrently (using `multiprocessing`).
- Merges the results from all the files.
- Finds the top n most frequent error codes.

Time Complexity Analysis

1. **`read_log_file(filename)`**
   The function loads the entire Excel file into memory. If there are **L** rows in the log:
   Time Complexity: O(L)
   (This includes reading the file and converting it to a list.)

2. **`split_file(lines, chunk_size)`**
   This function splits the lines into lists of size `chunk_size`. For each chunk, a text file is written. If there are **L** rows, this results in **T = ceil(L / chunk_size)** chunks.
   Time Complexity: O(L)
   Space Complexity: O(T) for storing the file names.

3. **`count_errors(chunk_file)`**
   Each chunk file contains at most `chunk_size` rows. The function processes each line and counts the error codes. For **T** files:
   Time Complexity: O(L) (since each line is scanned exactly once)
   Space Complexity: O(U) per chunk, where **U** is the number of unique error codes in the chunk. In the end, the final merged counter will have space complexity O(U).

4. **`heapq.nlargest(n, counter.items(), key=lambda x: x[1])`**
   This function iterates over the **U** unique error codes and returns the top **n**.
   Time Complexity: O(U log n)

Summary of Time Complexity:

- **Reading the file**: O(L)
- **Splitting into chunks**: O(L)
- **Counting errors**: O(L)

- **Merging and final count**: O(U)
- **Finding top-n**: O(U log n)

Thus, the **total time complexity** is **O(L + U log n)**.

Space Complexity:

- **Reading the file into memory (lines)**: O(L)
- **Writing chunk files to disk**: Writing only, so no additional space in memory
- **Error counter**: O(U)
- **List of chunk paths**: O(T)
- **Final result**: O(n)

Thus, the **total space complexity** is **O(L + U)**,

Final Conclusion:

- **Time Complexity**: **O(L + U log n)**
- **Space Complexity**: **O(L + U)**

Since usually **U << L** and **n << U**, we can approximate the complexity as **O(L)** for both time and space — i.e., linear with respect to the file size.