Algorithmic Logic and Data Structures

Problem: Efficiently filter and clean a large dataset of NYC taxi trips, removing invalid records based on missing fields, invalid timestamps, and out-of-bound coordinates, without relying on any libraries.

Approach:

- We implemented a fully manual data cleaning algorithm that includes:
- Custom CSV reading and writing functions to handle file I/O.
- Custom datetime parsing and comparison to validate timestamps.
- Custom coordinate validation to ensure geographic accuracy.
- Custom filtering logic to remove invalid records.

Pseudo-code:

```
function parse_datetime(datetime_str):
split datetime_str into date and time parts
split date into year, month, day
split time into hour, minute, second
validate all components are within valid ranges
return tuple of (year, month, day, hour, minute, second) or None if
invalid

function compare_datetimes(dt1, dt2):
compare year, month, day, hour, minute, second components
return True if dt1 <= dt2, False otherwise

function is_valid_coordinate(lat, lon, nyc_bounds):
convert lat and lon to float
check if they are within NYC geographic bounds
return True if valid, False otherwise

function read_csv(file_path):
open file and read all lines
extract headers from first line
for each subsequent line, split into values and create a dictionary
```

```
return headers and list of rows

function write_csv(file_path, headers, rows):
open file for writing
write headers as first line
for each row, write values as comma-separated line

function clean_data(input_file, output_file, excluded_file):
define NYC geographic bounds
read input CSV file
for each row:
parse pickup and dropoff datetimes
check for missing critical fields
if missing fields, add to excluded_rows and continue
check for invalid timestamps
if invalid timestamps, add to excluded_rows and continue
parse pickup and dropoff coordinates
check for invalid coordinates
if invalid coordinates, add to excluded_rows and continue
round coordinates to 6 decimal places
calculate trip_duration_min
add row to cleaned_rows
add 'trip_duration_min' to headers if not present
write cleaned data to output file
write excluded records to output file
```

Complexity Analysis:

- **Time Complexity**: O(n), where n is the number of rows in the dataset. Each row is processed exactly once.
- **Space Complexity**: O(n), where n is the number of rows in the dataset. We store both cleaned and excluded rows in memory.