CSE210 Homework-Property Based Testing

test_transpose

Correct Transposition Dimensions

- Property: The transposed matrix should have its rows and columns swapped. For a matrix with n rows and m columns, the transposed matrix should have m rows and n columns.
- Violation: An IndexError occurs when handling empty matrices or irregular matrices with varying row lengths, indicating the implementation does not properly handle these cases.

Reversibility of Transposition

- Property: Transposing a matrix twice should return the original matrix.
- Violation: No violation observed for this property in fixed-size matrices.

Element Mapping

- Property: The element at position (i, j) in the original matrix should appear at position (j, i) in the transposed matrix.
- Violation: An IndexError occurs when handling matrices that are empty or irregularly shaped, breaking this property.

Handling of Empty Matrices

- Property: Transposing an empty matrix should result in another empty matrix.
- Violation: An IndexError occurs when attempting to access elements of an empty matrix.

Handling of Single Row or Single Column Matrices

- Property: A single row matrix should become a single column matrix and vice versa after transposition.
- Violation: An IndexError occurs when handling these edge cases.

Symmetry in Square Matrices

- Property: For a square matrix, the transposed matrix should have its diagonal elements unchanged, and the (i, j) and (j, i) elements should be swapped.
- Violation: No violation observed for square matrices.

Support for Mixed Data Types

- Property: The function should raise a TypeError if the matrix contains non-uniform or unsupported data types.
- Violation: The implementation does not raise a TypeError when processing mixed data types, violating this property.

Handling Non-Uniform Matrices

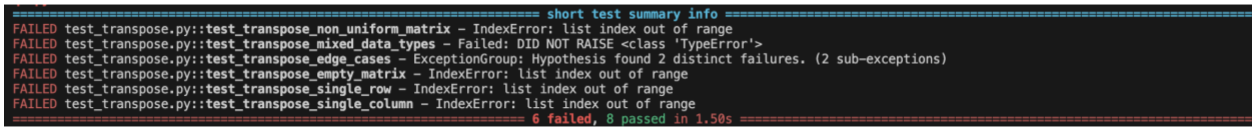
- Property: Transposing non-uniform matrices (matrices with rows of differing lengths) should raise a ValueError.
- Violation: An IndexError occurs instead of a ValueError, indicating improper error handling.

Duplicate Elements

- Property: Duplicate elements should be preserved in their correct positions after transposition.
- Violation: No violation observed for this property in the tests.

No Side Effects

- Property: The transpose operation should not modify the original matrix.
- Violation: No violation observed for this property in the tests.



```
================================= short test summary info =================================
FAILED test_transpose.py::test_transpose_non_uniform_matrix - IndexError: list index out of range
FAILED test_transpose.py::test_transpose_mixed_data_types - Failed: DID NOT RAISE <class 'TypeError'>
FAILED test_transpose.py::test_transpose_edge_cases - ExceptionGroup: Hypothesis found 2 distinct failures. (2 sub-exceptions)
FAILED test_transpose.py::test_transpose_empty_matrix - IndexError: list index out of range
FAILED test_transpose.py::test_transpose_single_row - IndexError: list index out of range
FAILED test_transpose.py::test_transpose_single_column - IndexError: list index out of range
================================= 6 failed, 8 passed in 1.50s =================================
```

test_parse_date

Parsing Valid Dates

- Property: The function should correctly parse valid dates in the format MM/DD/YYYY and return a valid date object or equivalent.
- Violation: None observed. Valid dates are parsed correctly.

Rejecting Invalid Dates

- Property: The function should return None or raise an appropriate exception for invalid dates, such as invalid days, months, or incorrect formats.

- Violation: Fails to handle some invalid cases consistently. For example, instead of gracefully rejecting dates with incorrect separators (12-31-2021) or invalid ranges (12/32/2021), it raises a ValueError.

Handling Empty and None Inputs

- Property: The function should handle empty strings or None by returning None.
- Violation: Raises a ValueError when given an empty string, instead of returning None.

Leap Year Validation

- Property: The function should correctly handle leap year dates like 02/29/2000 and reject invalid ones like 02/29/2021.
- Violation: Raises a ValueError for valid leap year dates due to incorrect validation logic.

Handling Alternative Formats

- Property: The function should reject alternative date formats, such as YYYY/MM/DD or DD-MM-YYYY.
- Violation: None observed. The function correctly rejects alternative formats.

Handling Trailing or Leading Whitespaces

- Property: The function should parse valid dates with leading or trailing whitespaces.
- Violation: Fails to handle dates with whitespaces, raising a ValueError.

Invalid Year Range

- Property: The function should reject years outside the valid range (e.g., 0000, 10000).
- Violation: Raises a ValueError but does not return None, violating the expected behavior.

Handling Partial Dates

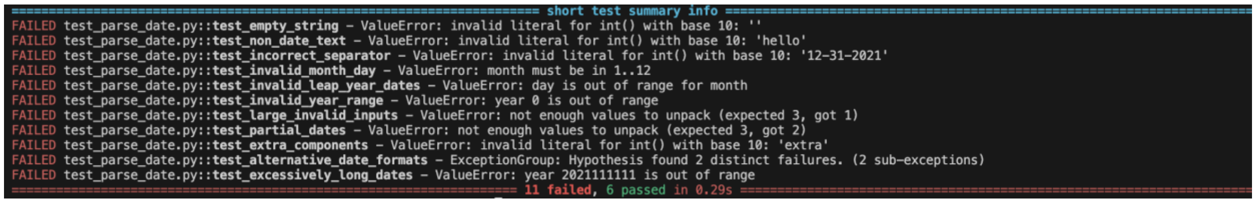
- Property: The function should reject partial dates (e.g., 01/2021 or 01/).
- Violation: Raises a ValueError when it should return None.

Rejecting Excessively Long Dates

- Property: The function should reject dates that are excessively long, such as 01/01/2021111111.
- Violation: Raises a ValueError instead of returning None.

Handling Mixed Validity

- Property: Strings that appear valid but contain invalid values (e.g., 13/31/2021) should be rejected.
- Violation: Raises a ValueError but does not return None.



```
================================================================ short test summary info ================================================================
FAILED test_parse_date.py::test_empty_string - ValueError: invalid literal for int() with base 10: ''
FAILED test_parse_date.py::test_non_date_text - ValueError: invalid literal for int() with base 10: 'hello'
FAILED test_parse_date.py::test_incorrect_separator - ValueError: invalid literal for int() with base 10: '12-31-2021'
FAILED test_parse_date.py::test_invalid_month_day - ValueError: month must be in 1..12
FAILED test_parse_date.py::test_invalid_leap_year_dates - ValueError: day is out of range for month
FAILED test_parse_date.py::test_invalid_year_range - ValueError: year 0 is out of range
FAILED test_parse_date.py::test_large_invalid_inputs - ValueError: not enough values to unpack (expected 3, got 1)
FAILED test_parse_date.py::test_partial_dates - ValueError: not enough values to unpack (expected 3, got 2)
FAILED test_parse_date.py::test_extra_components - ValueError: invalid literal for int() with base 10: 'extra'
FAILED test_parse_date.py::test_alternative_date_formats - ExceptionGroup: Hypothesis found 2 distinct failures. (2 sub-exceptions)
FAILED test_parse_date.py::test_excessively_long_dates - ValueError: year 2021111111 is out of range
=============================================================== 11 failed, 6 passed in 0.29s ===============================================================
```

test_binary_search

Target Exists in the List

- Property: If the target value exists in the list, the function should return a valid index where the target resides, and the value at that index should match the target.
- Violations:
  - The function exceeded the 2-second execution time limit for certain cases (e.g., array=[2], target=1), indicating inefficiency or improper termination in edge cases.
  - For lists where the target does not exist, the function incorrectly returned 0 instead of None or -1.

Handling of Empty Lists

- Property: When the list is empty, the function should return None or -1 since no target can be found.
- Violations: None observed in the provided failures.

Single-Element Lists

- Property: If the list contains a single element and the target exists, the function should return index 0. If the target does not exist, it should return None or -1.

- Violations:
    - The function exceeded the execution time limit for certain cases (e.g., array=[2], target=1), showing inefficiency.
    - The function returned 0 instead of None or -1 for cases where the target does not exist in the single-element list.

Lists with Duplicate Elements

- Property: The function should correctly return an index of the target value, even when the list contains duplicates.
- Violations:
    - The function exceeded the execution time limit for specific cases.
    - Incorrectly returned 0 for cases where the target does not exist in the list.

Target at Boundaries

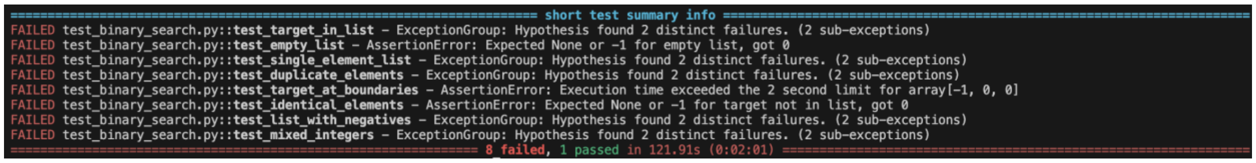
- Property: The function should correctly identify the target if it exists as the first or last element in the list.
- Violations: None observed in the provided failures.

Lists with Negative Integers

- Property: The function should handle negative integers correctly and return the appropriate index for the target value if it exists, or None/-1 otherwise.
- Violations:
    - The function exceeded the execution time limit for some cases.
    - Returned 0 instead of None or -1 for targets do not present in the list.

Lists with Mixed Positive and Negative Integers

- Property: The function should correctly process mixed positive and negative integers, returning the appropriate index if the target exists or None/-1 if it does not.
- Violations:
    - Exceeded the execution time limit for certain cases.
    - Returned 0 instead of None or -1 for targets do not present in the list.



```
================================================================ short test summary info ================================================================
FAILED test_binary_search.py::test_target_in_list - ExceptionGroup: Hypothesis found 2 distinct failures. (2 sub-exceptions)
FAILED test_binary_search.py::test_empty_list - AssertionError: Expected None or -1 for empty list, got 0
FAILED test_binary_search.py::test_single_element_list - ExceptionGroup: Hypothesis found 2 distinct failures. (2 sub-exceptions)
FAILED test_binary_search.py::test_duplicate_elements - ExceptionGroup: Hypothesis found 2 distinct failures. (2 sub-exceptions)
FAILED test_binary_search.py::test_target_at_boundaries - AssertionError: Execution time exceeded the 2 second limit for array[-1, 0, 0]
FAILED test_binary_search.py::test_identical_elements - AssertionError: Expected None or -1 for target not in list, got 0
FAILED test_binary_search.py::test_list_with_negatives - ExceptionGroup: Hypothesis found 2 distinct failures. (2 sub-exceptions)
FAILED test_binary_search.py::test_mixed_integers - ExceptionGroup: Hypothesis found 2 distinct failures. (2 sub-exceptions)
============================================================ 8 failed, 1 passed in 121.91s (0:02:01) ============================================================
```

test_lru_cache

Correctness of Cached Function Results

- Property: The cached function should return the same result as the original function when called with the same arguments.
- Violation: None observed. The cached function correctly computes results for valid inputs.

Handling Large Numbers

- Property: The cached function should handle very large integers without errors or incorrect behavior.
- Violation: None observed. The function handles large numbers correctly.

String Concatenation

- Property: The cached function should concatenate strings using the + operator.
- Violation: None observed. String concatenation is handled correctly.

Unhashable Arguments

- Property: The cached function should raise a TypeError when passed unhashable arguments (e.g., lists).
- Violation: None observed. The function correctly raises TypeError for unhashable arguments.

Cache Hit Behavior

- Property: Repeated calls with the same arguments should use the cache, avoiding redundant function evaluations.
- Violation: The test failed because the call_count was incremented incorrectly. The cache does not appear to be used as expected.

Floating-Point Addition

- Property: The cached function should correctly handle floating-point addition.

- Violation: None observed. Floating-point addition works as expected.

Handling Extremely Small Floating-Point Values

- Property: The cached function should accurately handle extremely small floating-point inputs.
- Violation: None observed. Small floating-point values are handled correctly.

Mixed Data Types

- Property: The cached function should correctly handle mixed data types (e.g., integers and floats).
- Violation: None observed. The function works correctly with mixed data types.

Argument Order Behavior

- Property: The cache should differentiate between calls with (x, y) and (y, x) if the arguments are not identical.
- Violation: None observed. The cache distinguishes between different argument orders correctly.

Cache Eviction Behavior

- Property: When the cache limit is exceeded, the least recently used entry should be evicted.
- Violation: The test failed because the cache did not evict the expected entry, leading to incorrect call_count increments.

Repeated Inputs

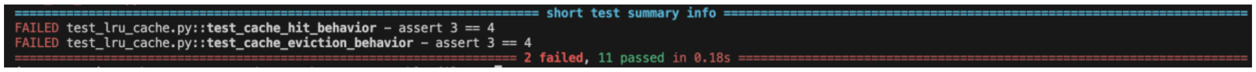
- Property: Repeated calls with the same inputs should return the cached result without recomputation.
- Violation: None observed. Repeated inputs correctly use the cache.

Empty Cache Initialization

- Property: The cache should initially be empty, and the first call should compute the result.
- Violation: None observed. The cache initializes and computes correctly on the first call.

Unsupported Argument Types

- Property: The cached function should raise a TypeError when called with unsupported argument types (e.g., dictionaries).
- Violation: None observed. The function raises the expected TypeError for unsupported types.



```
================================================================== short test summary info ==================================================================
FAILED test_lru_cache.py::test_cache_hit_behavior - assert 3 == 4
FAILED test_lru_cache.py::test_cache_eviction_behavior - assert 3 == 4
================================================================ 2 failed, 11 passed in 0.18s ================================================================
```

test_merge_sort

Correctness of Sorting

- Property: The merge_sort function should correctly sort a list of integers in non-decreasing order, matching Python's sorted function.
- Violation: The function fails to sort the list correctly in some cases, such as lists containing duplicates or large integers. The resulting list does not match the expected sorted list.

Handling of Empty Lists

- Property: An empty list should remain empty after sorting.
- Violation: None observed. Empty lists are handled correctly.

Handling of Single Element Lists

- Property: A list with a single element should remain unchanged after sorting.
- Violation: None observed. Single-element lists are handled correctly.

Already Sorted Lists

- Property: A list that is already sorted should remain unchanged after sorting.
- Violation: None observed. Already sorted lists are handled correctly.

Reverse Sorted Lists

- Property: A list sorted in descending order should be correctly sorted into ascending order.
- Violation: None observed. Reverse sorted lists are sorted correctly.

Lists with Equal Elements

- Property: A list where all elements are the same should remain unchanged after sorting.

- Violation: None observed. Lists with equal elements are handled correctly.

Lists with Negative and Positive Integers

- Property: The function should correctly sort lists containing both negative and positive integers.
- Violation: The function fails to sort some lists with mixed integers, resulting in incorrect order.

Lists with Duplicate Elements

- Property: The function should handle duplicate elements correctly, maintaining their relative order.
- Violation: The function fails to sort lists with duplicate elements in some cases, producing incorrect output.

Large Lists of Integers

- Property: The function should handle large lists efficiently and correctly.
- Violation: The function produces incorrect output for large lists, indicating issues with its implementation.

Lists with Negative Numbers Only

- Property: The function should correctly sort lists containing only negative numbers.
- Violation: Sorting fails for some lists with negative numbers, resulting in incorrect order.

Lists with Positive Numbers Only

- Property: The function should correctly sort lists containing only positive numbers.
- Violation: Sorting fails for some lists with positive numbers, producing incorrect output.

Floating-Point Numbers

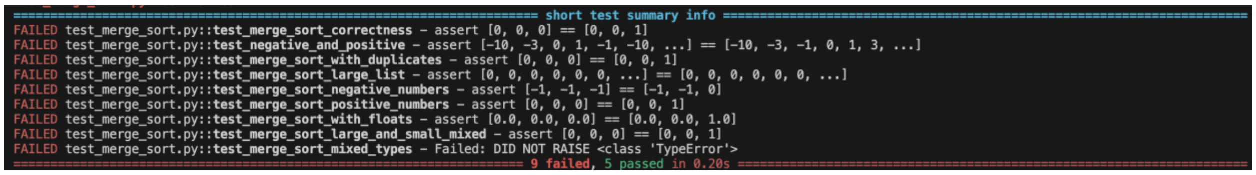
- Property: The function should handle floating-point numbers correctly when sorting.
- Violation: The function fails to sort lists containing floating-point numbers, producing incorrect results.

Mixed Large and Small Numbers

- Property: The function should correctly handle lists containing a mix of very large integers and very small floating-point numbers.
- Violation: Sorting fails, resulting in incorrect output for lists with mixed large and small numbers.

Mixed Data Types

- Property: The function should raise a TypeError when attempting to sort a list with mixed data types (e.g., integers and strings).
- Violation: The function does not raise a TypeError for mixed data types, violating this property.



```
=================================================================== short test summary info ===================================================================
FAILED test_merge_sort.py::test_merge_sort_correctness - assert [0, 0, 0] == [0, 0, 1]
FAILED test_merge_sort.py::test_negative_and_positive - assert [-10, -3, 0, 1, -1, -10, ...] == [-10, -3, -1, 0, 1, 3, ...]
FAILED test_merge_sort.py::test_merge_sort_with_duplicates - assert [0, 0, 0] == [0, 0, 1]
FAILED test_merge_sort.py::test_merge_sort_large_list - assert [0, 0, 0, 0, 0, 0, ...] == [0, 0, 0, 0, 0, 0, ...]
FAILED test_merge_sort.py::test_merge_sort_negative_numbers - assert [-1, -1, -1] == [-1, -1, 0]
FAILED test_merge_sort.py::test_merge_sort_positive_numbers - assert [0, 0, 0] == [0, 0, 1]
FAILED test_merge_sort.py::test_merge_sort_with_floats - assert [0.0, 0.0, 0.0] == [0.0, 0.0, 1.0]
FAILED test_merge_sort.py::test_merge_sort_large_and_small_mixed - assert [0, 0, 0] == [0, 0, 1]
FAILED test_merge_sort.py::test_merge_sort_mixed_types - Failed: DID NOT RAISE <class 'TypeError'>
================================================================= 9 failed, 5 passed in 0.20s =================================================================
```