Report

My Github:

https://github.com/NiharikaAdari/datamineresearch/blob/main/gem5_traces/gem5-snoop/functions.py

SEES github:

https://github.com/sees-usf/Model-Synthesis/tree/master

Task:

- Create a general use function that can help remove a pattern of any length from an arbitrary trace, and return the trace with the removed pattern
    - Takes in:
        - Pattern, list of any size
        - Arbitrary trace
    - Gives:
        - Trace with the pattern removed

- Create a function that
    - Takes in:
        - List of patterns (of any length)
        - Arbitrary trace
    - Gives:
        - A txt with the calculated acceptance ratios for each pattern, ranked in descending order

Remove pattern from trace:

- Notes, more fixes made, to increase performance and robustness. Used hash table indexing and pointers.

```
    """
    Remove occurrences of a specified pattern from a trace list.

    Args:
    - trace (list): The trace list from which the pattern occurrences should be removed.
    - pattern (list): The pattern (sequence of numbers) to be removed from the trace.

    Returns:
    - list: A new trace list with the specified pattern occurrences removed.

    Notes:
    - hash table-based indexing and pointer manipulation for efficiency
    """

def remove_pattern_from_trace(trace, pattern):
```

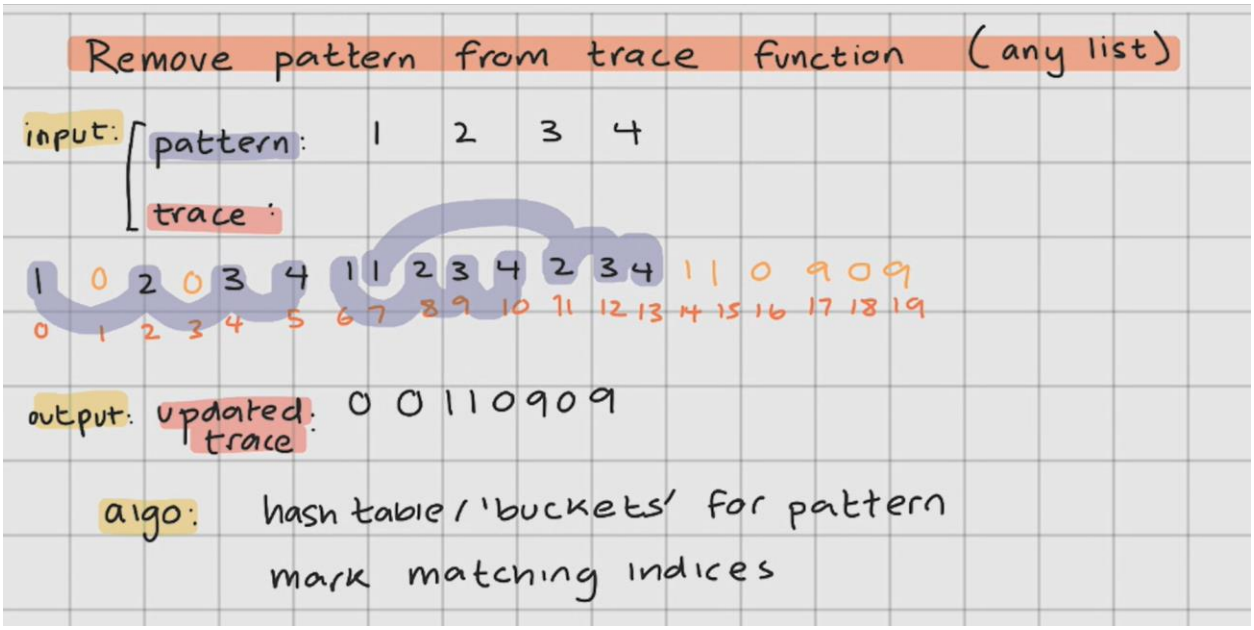
Algorithm Explanation with Examples:

Input:

Trace:

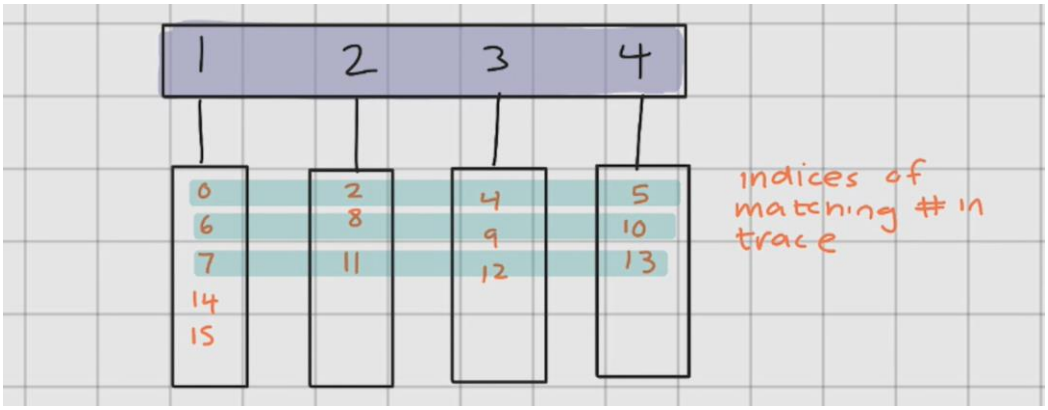
Pattern to remove:

[1,2,3,4]



So first, it makes a hash table for the pattern itself. And each number is a key to a bucket, which stores the indices in the trace that matches.

Diagram:

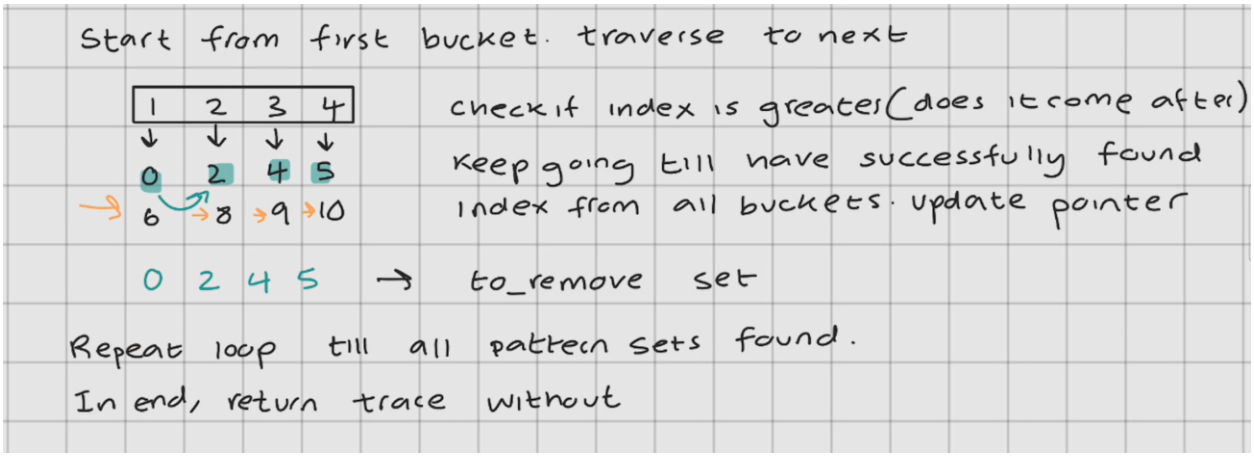


Code:

```
buckets {1: [0, 6, 7, 14, 15], 2: [2, 8, 11], 3: [4, 9, 12], 4: [5, 10, 13]}
```

- Start from first bucket, pick index. Pick: 0.
- Move to next bucket. Traverse until you find an index that is sequentially greater. In this case, 2. Update pointer.
- Get a index from all buckets, till you can form the pattern.
- Repeat in a loop until all pattern sets are found. If a set is found, it is saved to set_to_remove, which updates each time
- In the end, return trace without all marked indices

Diagram



Code

```
buckets {1: [0, 6, 7, 14, 15], 2: [2, 8, 11], 3: [4, 9, 12], 4: [5, 10, 13]}
current indices:  [0]
checking bucket of 2  in  [1, 2, 3, 4]
index found from bucket. 2
checking bucket of 3  in  [1, 2, 3, 4]
index found from bucket. 4
checking bucket of 4  in  [1, 2, 3, 4]
index found from bucket. 5
valid pattern found. indices: [0, 2, 4, 5]
current indices:  [6]
checking bucket of 2  in  [1, 2, 3, 4]
index found from bucket. 8
checking bucket of 3  in  [1, 2, 3, 4]
index found from bucket. 9
checking bucket of 4  in  [1, 2, 3, 4]
index found from bucket. 10
valid pattern found. indices: [6, 8, 9, 10]
current indices:  [7]
```

Bucket 1: Pick index 0

Bucket 2: pick index 2

Etc, till you get valid pattern with indices [0,2,4,5]

Bucket 1: pick index 6

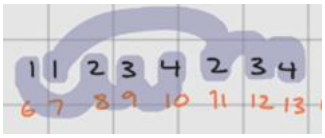
Bucket 2: pick index 8

Etc, find next valid set

```
current indices:  [7]
checking bucket of 2  in  [1, 2, 3, 4]
index found from bucket. 11
checking bucket of 3  in  [1, 2, 3, 4]
index found from bucket. 12
checking bucket of 4  in  [1, 2, 3, 4]
index found from bucket. 13
valid pattern found. indices: [7, 11, 12, 13]
current indices:  [14]
checking bucket of 2  in  [1, 2, 3, 4]
current indices:  [15]
checking bucket of 2  in  [1, 2, 3, 4]
[0, 0, 1, 1, 0, 9, 0, 9]
```

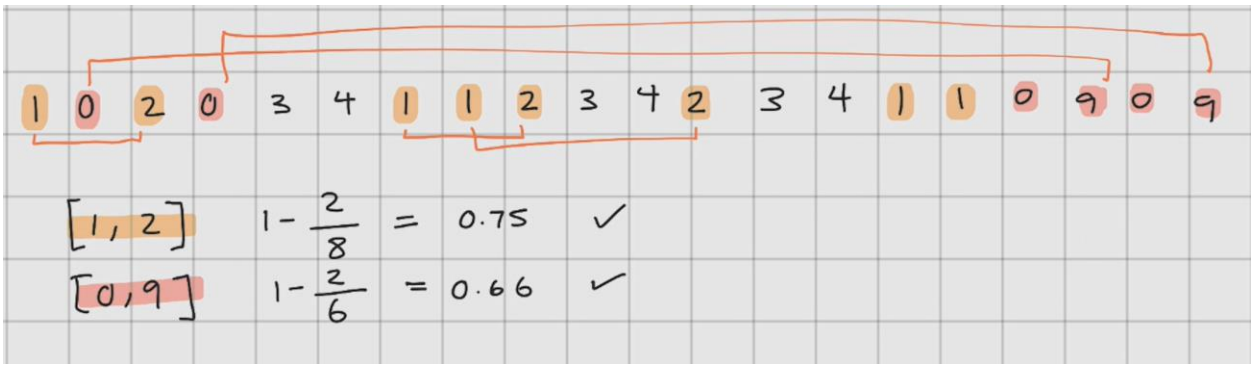
Then, start from bucket 1 again, pick index 7

In bucket 2, it picks index 11, etc



Returns the trace with the patterns removed.

When calculating acceptance ratio, count how many in the original, then how many left after removing.



Pattern_AcceptanceRatios.txt in the unslicedtrace-1-patterns folder gives:

```
patterns = [[0,9], [1,2]]
compute_pattern_ratios(trace, output_folder, patterns)
```

```
------------------------
1. [1, 2], Acceptance Ratio: 0.75
2. [0, 9], Acceptance Ratio: 0.6666666666666667
------------------------
```

This is correct.

Compute Pattern Ratios (Write results to folder)

```
"""
Computes acceptance ratios of each pattern from a list of patterns on a trace and
writes results to an output folder.
    Parameters:
    - trace: pass in a trace file
    - output_folder: Path to the output folder to write results.
    - patterns: pass in the list of patterns to get ratios from
"""


def compute_pattern_ratios(trace, output_folder, patterns):
```

- Input: Pass in a trace, an output folder to write results, and a list of patterns
- Uses the function below, which in turn uses the remove pattern from trace function
- Output: txt file with patterns ranked by acceptance ratio

Find acceptance ratios

Pass in a list of patterns. It will find the acceptance ratio for each one on the trace passed in.

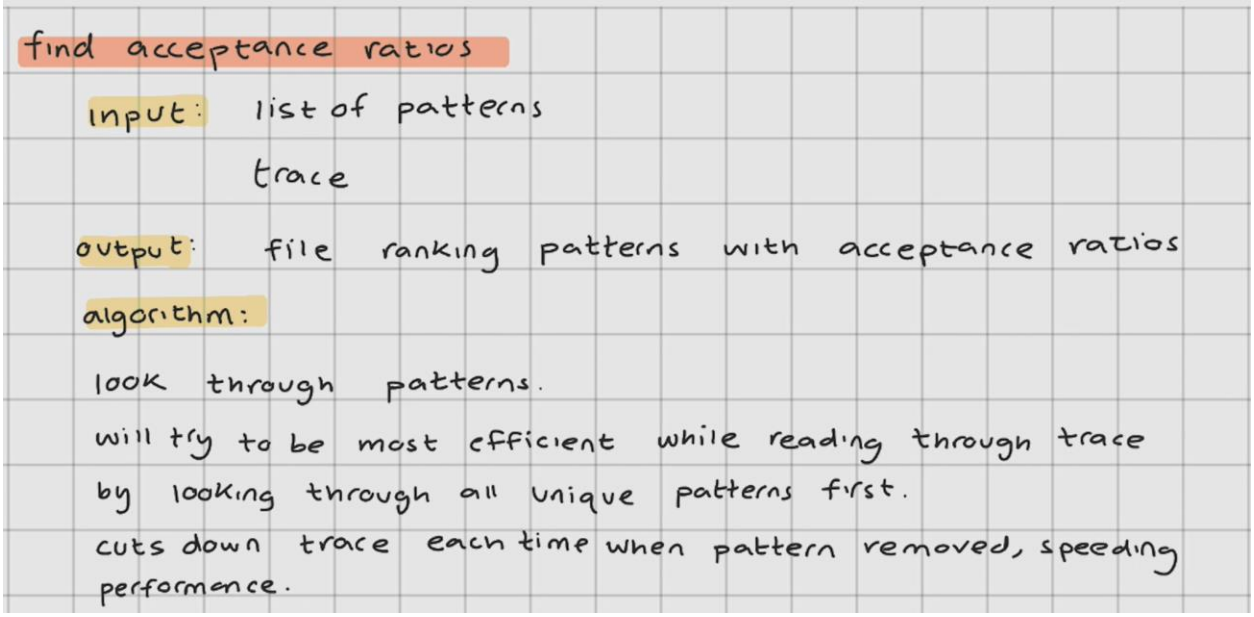
```
"""
    Find patterns in the trace and calculate acceptance ratios for each pattern.

    Args:
    - trace (list): List of integers representing the trace.
    - patterns (list): A list of patterns (each a list of numbers) to find in the
trace.

    Returns:
    - pair_acceptance_ratios: List of tuples representing pairs and their
acceptance ratios.

"""

def find_acceptance_ratios(trace, patterns):
```



Algorithm Explanation with Examples

Input

Trace: [0 9 0 9 1 9 2 3 4]

List of Patterns: [[0,9], [1,9], [2,3,4]]

Diagram:

```
Trying pattern 1/3: [0, 9]
[0, 9, 0, 9, 1, 9, 2, 3, 4]
removing: [0, 9]
Acceptance ratio: 0.8
used numbers:  {0, 9}

Trying pattern 2/3: [1, 9]
[1, 9, 2, 3, 4]
skip pattern

Trying pattern 3/3: [2, 3, 4]
[1, 9, 2, 3, 4]
removing: [2, 3, 4]
Acceptance ratio: 1.0
used numbers:  {0, 2, 3, 4, 9}

Trying pattern 1/1: [1, 9]
[0, 9, 0, 9, 1, 9, 2, 3, 4]
removing: [1, 9]
Acceptance ratio: 0.5
used numbers:  {1, 9}
[([0, 9], 0.8), ([2, 3, 4], 1.0), ([1, 9], 0.5)] 3
```

- First, try removing [0,9] on the original trace.
- Original: [0,9,0,9,1,9,2,3,4]
- Result: [1,9,2,3,4]
- 1 – 1/5 = 0.8
- Get acceptance ratio 80%
- Skip next pattern for now, since we already removed 9
- Remove pattern 2,3,4 from shortened trace
- Reset trace to original, and try pattern [1,9]

Output:

```
------------------------
1. [2, 3, 4], Acceptance Ratio: 1.0
2. [0, 9], Acceptance Ratio: 0.8
3. [1, 9], Acceptance Ratio: 0.5
------------------------
```

This txt ranks the patterns in the original list by the acceptance ratios

Task:

Try on list of allPatterns with 54 rows and 6024481 patterns, on unsliced gem5 snoop trace

Problem: Takes too long

Fixes: WIP

- Algorithm works best when sequential patterns differ in numbers/don't reuse
- So I transposed the allPatterns list
- Working on how to speed it up further