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:

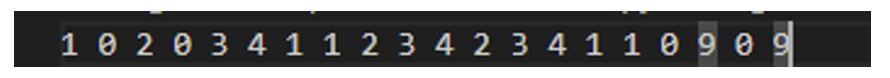
* Notes, more fixes made, to increase performance and robustness. Used hash table indexing and pointers.

A screenshot of a computer program

Description automatically generated

## Algorithm Explanation with Examples:

### Input:

**Trace: **

**Pattern to remove:**

[1,2,3,4]

A close-up of a graph

Description automatically generated

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:**

A graph with numbers and numbers

Description automatically generated

**Code:**

A screenshot of a computer program

Description automatically generated

* 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**

A close-up of a piece of paper

Description automatically generated

**Code**

A screenshot of a computer program

Description automatically generatedBucket 1: Pick index 0

Bucket 2: pick index 2

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

A close up of numbers

Description automatically generated

Bucket 1: pick index 6

Bucket 2: pick index 8

Etc, find next valid set

A screenshot of a computer program

Description automatically generatedThen, start from bucket 1 again, pick index 7

In bucket 2, it picks index 11, etc

A close up of numbers

Description automatically generated

Returns the trace with the patterns removed.

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

A graph with numbers and tick marks

Description automatically generated

A black and white background with text

Description automatically generated

A black screen with white text

Description automatically generated

# Compute Pattern Ratios (Write results to folder)

A computer screen shot of a black and white screen

Description automatically generated

* 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

A screenshot of a computer

Description automatically generated

A graph paper with black writing

Description automatically generated

## 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:** A computer screen shot of a number

Description automatically generated

* 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:

A black screen with white text

Description automatically generatedThis 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