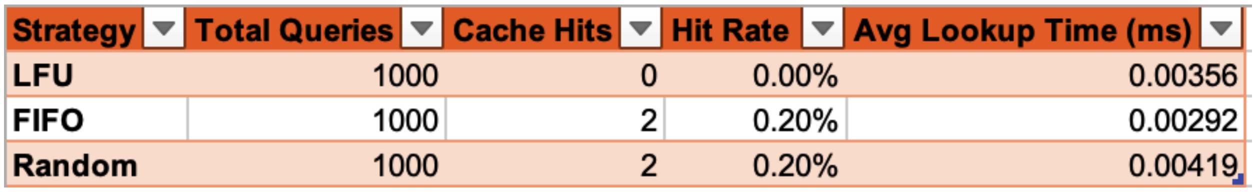
In this milestone, I evaluated the performance of three caching strategies: LFU (Least Frequently Used), FIFO (First-In First-Out), and Random Replacement, used in a city population lookup program that loads data into a Trie from a CSV file. The goal was to measure and compare each strategy's performance under a simulated high-load scenario.

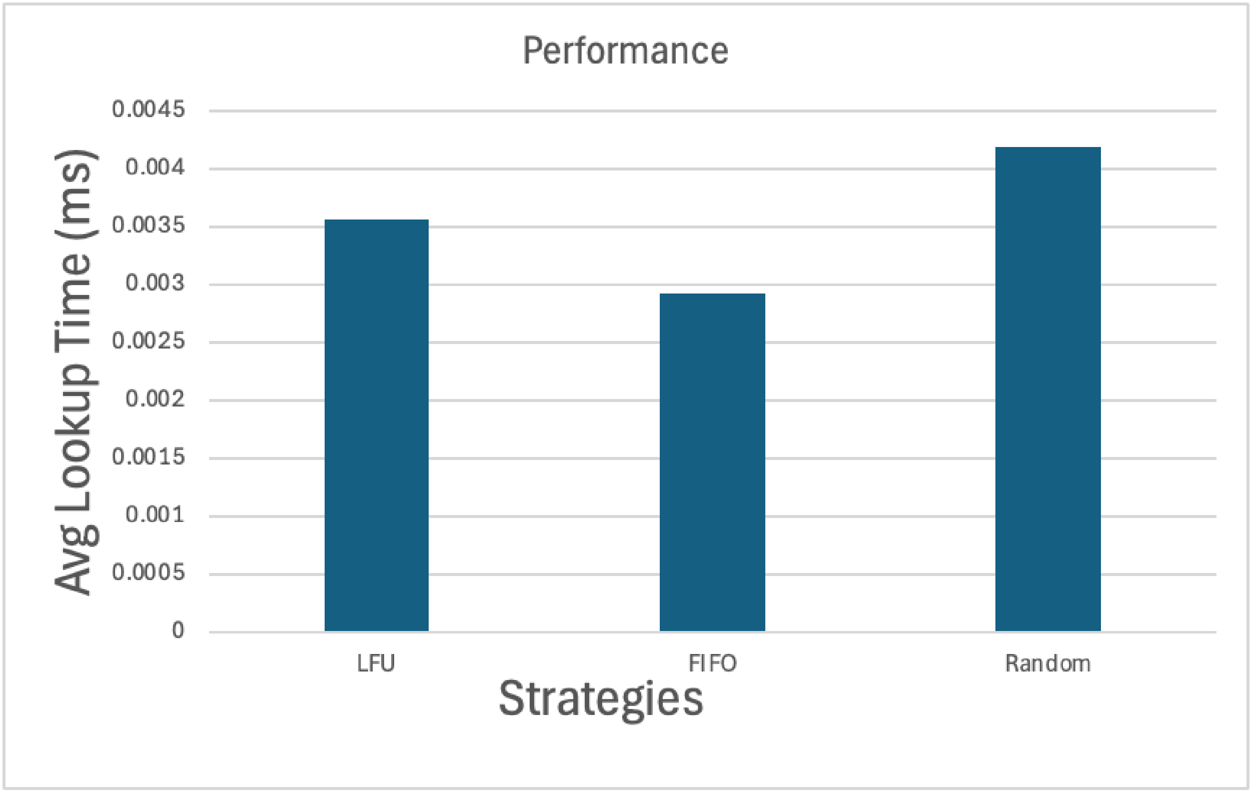
2. Testing Methodology

- A load testing script was developed to simulate 1000 random city + country code queries.
- Each query attempted to retrieve the city's population, either from the cache (if present) or from the Trie.
- The same set of queries was used across all three cache strategies.
- Each cache had a fixed size of 10 entries.
- For each run, the following metrics were collected:
    - Total number of queries
    - Cache hits and hit rate
    - Average lookup time in milliseconds

3. Performance Metrics



Strategy	Total Queries	Cache Hits	Hit Rate	Avg Lookup Time (ms)
LFU	1000	0	0.00%	0.00356
FIFO	1000	2	0.20%	0.00292
Random	1000	2	0.20%	0.00419



4. Analysis

- LFU Cache
    - Performed the worst in terms of cache hits.
    - With random queries, frequency-based replacement offers no advantage, as entries are likely to not be accessed more than once.
    - Best suited for workloads with repeated accesses to the same items.
- FIFO Cache
    - Slightly better than LFU in this test.

    - The simple queue-based eviction allowed a few repeated entries to remain long enough to be hit again.
    - Performs well when data access has some recency pattern.
- Random Replacement Cache
    - Also had very low cache hit rate, as expected.
    - Random eviction offers no learning or pattern-following, leading to hits mostly by chance.
    - More suitable for small datasets with unpredictable patterns of access.
- Performance Time
    - All strategies had very fast lookup times, with averages ranging between 0.0029–0.0042 ms.
    - Trie lookups combined with lightweight cache access contributed to relatively low response times.

5. Conclusion

In a random access scenario with many unique queries and a small cache size, all three strategies performed poorly in terms of cache hits. However:

- FIFO and Random slightly outperformed LFU, mainly due to chance-based retention of repeated entries
- LFU struggled because no meaningful frequency pattern existed to exploit
- All strategies maintained low average lookup times, highlighting the efficiency of the underlying Trie and cache design