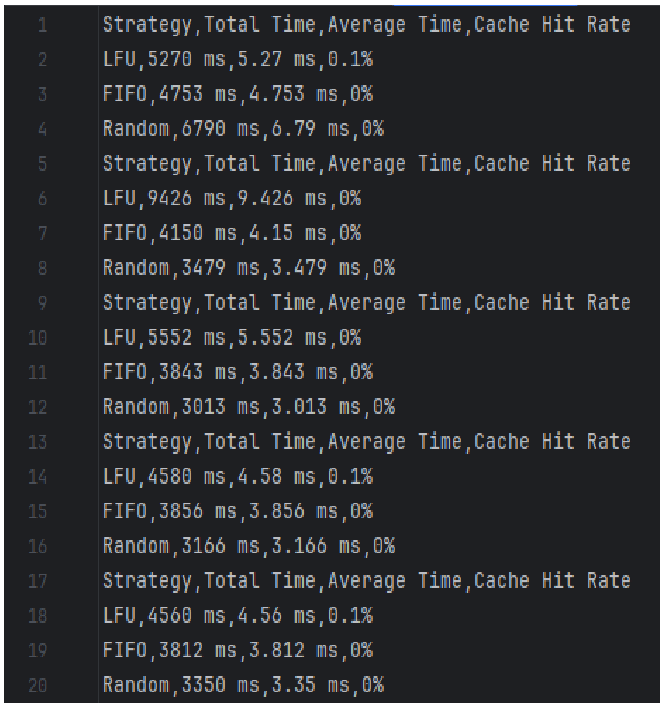
Performance Analysis Report

This performance analysis was created to evaluate the efficiency of three caching strategies: LFU (Least Frequently Used), FIFO (First In, First Out), and Random Replacement. I used the caching strategies for city population lookups using a csv file called world_cities.csv which contained a little under 50,000 cities. This report will explain what I found.

Testing Methodology:

The test was run using a fully automated testing script that tested each caching strategy using the same set of 1000 random cities that was taken from the world_cities.csv file. I kept the same cache size of 10 to keep it consistent with the size we used for the previous milestones. I measured three things: the total time a strategy took to complete 1,000 lookups, the average time it took per look up, and the cache hit rate percentage. The picture to the right is a screenshot of the direct results.



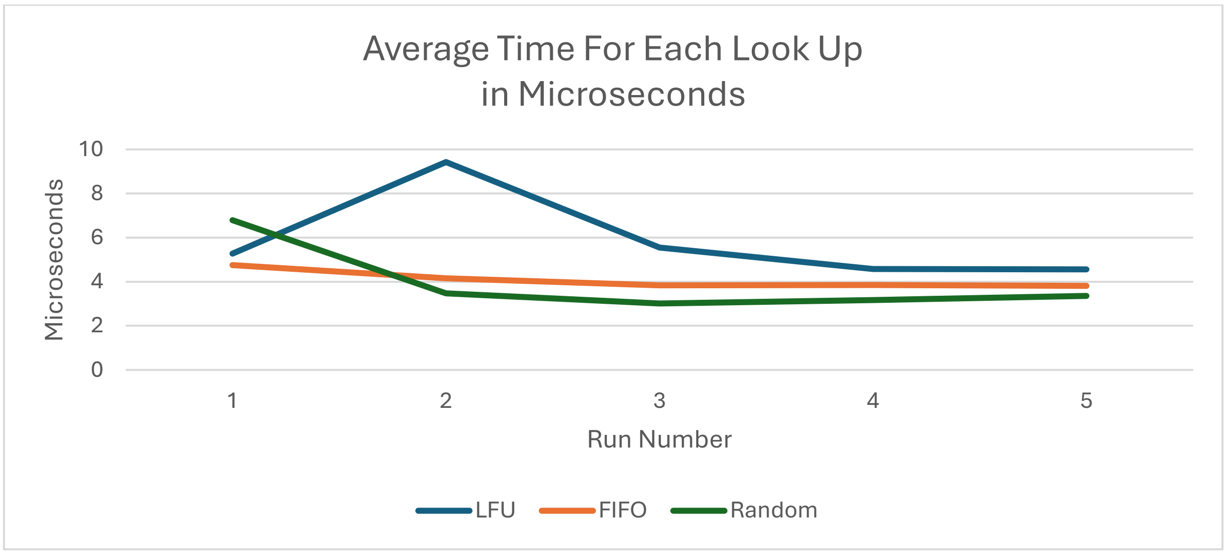
```
Strategy,Total Time,Average Time,Cache Hit Rate
LFU,5270 ms,5.27 ms,0.1%
FIFO,4753 ms,4.753 ms,0%
Random,6790 ms,6.79 ms,0%
Strategy,Total Time,Average Time,Cache Hit Rate
LFU,9426 ms,9.426 ms,0%
FIFO,4150 ms,4.15 ms,0%
Random,3479 ms,3.479 ms,0%
Strategy,Total Time,Average Time,Cache Hit Rate
LFU,5552 ms,5.552 ms,0%
FIFO,3843 ms,3.843 ms,0%
Random,3013 ms,3.013 ms,0%
Strategy,Total Time,Average Time,Cache Hit Rate
LFU,4580 ms,4.58 ms,0.1%
FIFO,3856 ms,3.856 ms,0%
Random,3166 ms,3.166 ms,0%
Strategy,Total Time,Average Time,Cache Hit Rate
LFU,4560 ms,4.56 ms,0.1%
FIFO,3812 ms,3.812 ms,0%
Random,3350 ms,3.35 ms,0%
```

Tables and Graphs:

Strategy	Total Time	Average Time	Cache Hit Rate
LFU	5270	5.27	0.10%
LFU	9426	9.426	0%
LFU	5552	5.552	0%
LFU	4580	4.58	0.10%
LFU	4560	4.56	0.10%
LFU Averages	5877.6	5.8776	0.06%

Strategy	Total Time	Average Time	Cache Hit Rate
FIFO	4753	4.753	0%
FIFO	4150	4.15	0%
FIFO	3843	3.843	0%
FIFO	3856	3.856	0%
FIFO	3812	3.812	0%
FIFO Averages	4082.8	4.0828	0.00%

Strategy	Total Time	Average Time	Cache Hit Rate
Random	6790	6.79	0%
Random	3479	3.479	0%
Random	3013	3.013	0%
Random	3166	3.166	0%
Random	3350	3.35	0%
Random Averages	3959.6	3.9596	0.00%



Observations:

I noticed something very odd while looking at the results. Cache hit rates are very low and only seen in the LFU strategy. I thought it was weird at first but when you think about it, only having a cache of 10 for a list of around 50,000 cities is really helping that much. I “unofficially” used a smaller csv file with a list of only 20 cities, kept a cache of 10, and the results made a lot more sense because I was getting a much higher cache rate of around the 50% range for each run. So, with a cache of only 10 with a list of 50,000 cities of course the cache hit rate was going to be very low. Now I will talk about the strengths and weaknesses of each strategy.

For the LFU strategy, I think its strength is that if a user needs to look up the same city over and over it will stay in the cache longer because the cache gets rid of the city that is used the least frequently. I think its weakness then is that if all the cities that need to be looked up are unique, it completely overrules its strength because there isn’t a least frequently used city.

For the FIFO strategy, its strength is that it is simple. The first to go in is the first to go out just like a line or queue. Its weakness is that it doesn’t consider the frequency of use.

And finally, for the random replacement strategy, its strength is that it avoids any overhead from tracking frequency or order meaning it is effective if there is no need to keep track of any frequent cities. Its weakness is that it will have an inconsistent performance in terms of cache hits.

Conclusion:

In conclusion, despite being tied with the lowest cache rate, the random replacement consistently delivered the fastest look up time for the world_cities.csv data set. For this data set, random replacement was the fastest because the cache was almost never in use due to the fact that the dataset was too large for the small cache size making which make the overhead of tracking the frequency or maintaining the order of the cache almost counterproductive. But like I said in the observations section, each strategy has its own strengths and weaknesses. So, for this scenario, random replacement was best but that doesn't mean it would always be best. If there is a scenario where a city was constantly being repeated you would want LFU because it keeps the most frequently used cities in the cache. Or if you need a predictable cache layout the FIFO would work best because you know exactly when order cities are going in and out. So, the choice of what caching strategy to use should be determined on the scenario in which it is being put into use.