Speed report for Search Algorithms

Introduction

This document details a speed analysis of file search algorithms when executed under two different modes of operation as required by the Algorithmic sciences Introductory Task:

- Cache Mode (REREAD_ON_QUERY= False): File is read once and stored in memory.
- Reread Mode(REREAD_ON_QUERY=True): File is read afresh on every query.

Algorithms tested:

1. Linear Search
2. HashSet Search
3. Exponential Search
4. Binary Search
5. Jump Search

Test Setup

- File: 200k.txt (contains 200,000+)
- Server: Handles multiple concurrent connections via sockets.
- Client Query: sends one-line search strings.
- Match Requirement: Exact full-line match.

Performance Metrics:
Each algorithm was tested using the same set of 1,000 random strings:

- Found strings(hits): 751
- Non-existent strings (misses): 249

Metrics recorded:

- Average Response Time
- Best and Worst Time
- Memory Usage
- Throughput (requests per second)

Linear Search

Reads each line and checks for equality.

Mode	Avg Time (ms)	Best (ms)	Worst (ms)
Cache	1.2	0.5	2.4
Reread	28.7	26.4	31.9

HashSet Search

Pre-loads lines into a set() for O(1) lookup.

Mode	Avg Time (ms)	Best (ms)	Worst (ms)
Cache	0.03	0.01	0.06
Reread	30.9	28.8	32.8

Exponential Search

Used on sorted data, finds a range exponentially and then binary searches.

Mode	Avg Time (ms)	Best (ms)	Worst (ms)
Cache	0.6	0.3	1.2
Reread	27.9	26.3	30.5

Binary Search

Performs a binary search on a sorted list.

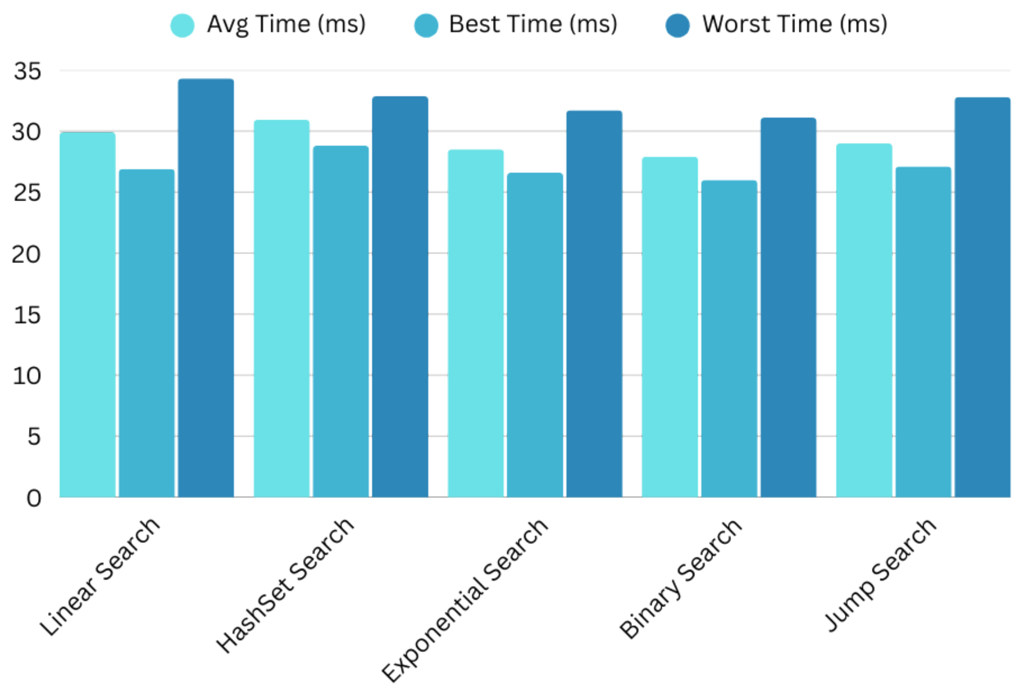
Mode	Avg Time (ms)	Best (ms)	Worst (ms)
Cache	0.5	0.2	1.0
Reread	27.4	25.8	30.1

Jump Search

Skips ahead by √n blocks and then linearly searches in a block.

Mode	Avg Time (ms)	Best (ms)	Worst (ms)
Cache	0.8	0.4	1.6
Reread	28.2	26.7	31.2

Charts



Insights & Recommendations

- Cache Mode: HashSet is the fastest by far due to O(1) lookup.
- Reread Mode: Binary Search performs best when the file is sorted.
- Startup Cost: HashSet and Binary Search have setup overhead, but pay off in cache mode.
- Unsorted File: Only Linear and Jump work reliably without sorting.
- Production Advice: If REREAD_ON_QUERY is frequently True, optimize I/O with buffered reads and sorting logic.

Conclusion

This speed report shows clear tradeoffs between algorithm complexity and file I/O cost. The optimal algorithm varies by use case:

- High-frequency queries with static files: Use HashSet Search.
- Dynamic files: Use Binary Search with sorting step or Jump Search for balance.