

# *Cache Performance*

Cache Performance Effects on Search Algorithms

CSE 340 Computer Architecture

Chris Wilkins & Matt Sgambato

---

---

# *Table of Contents*

- Introduction / Background
  - Array – Sequential Search
  - Binary Search Trees
  - Sequential Binary Search Tree
  - Example
  - The Program
  - The Results
  - Conclusions / Moving Forward
- 
-

# *Introduction / Background*

- Search Data Structures
    - Array
    - Binary Search Tree
    - Sequential Binary Search Tree
  - Cache Implementations
    - Direct Mapped
    - Fully Associative
    - 2-way Set Associative
  - Search Times
    - Array -Sequential Search :  $O(n)$
    - Binary Search Tree :  $O(\log n)$
    - Sequential Binary Search Tree :  $O(\log n)$
- 
-

# *Array – Sequential Search*

- Search through the data in order
  - The Good
    - Values are stored sequentially in memory
  - The Bad
    - $O(n)$  search time

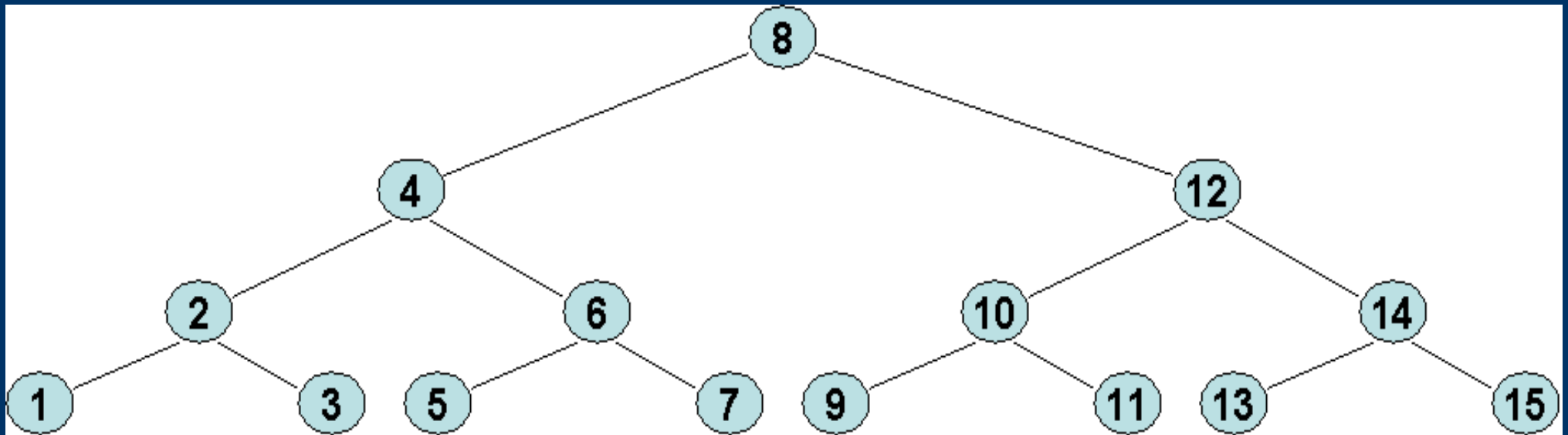
# *Binary Search Trees*

- The Good:
    - Search time of  $O(\log n)$
  - The Problem:
    - A Cache miss does not guarantee other tree elements will be placed in the cache.
    - Nodes are non-sequentially stored in memory
    - Many more Cache misses.
  - The Goal:
    - Search time of  $O(\log n)$
    - Store values sequentially in memory
- 
-

# *Sequential Binary Search Tree*

- The Good:
  - Sequential Memory Storage
  - Binary Search Tree speed:  $O(\log n)$
- The Bad:
  - Preprocessing overhead

# Example:



value	8	4	2	1	3	6	5	7	12	10	9	11	14	13	15
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

# *The Program*

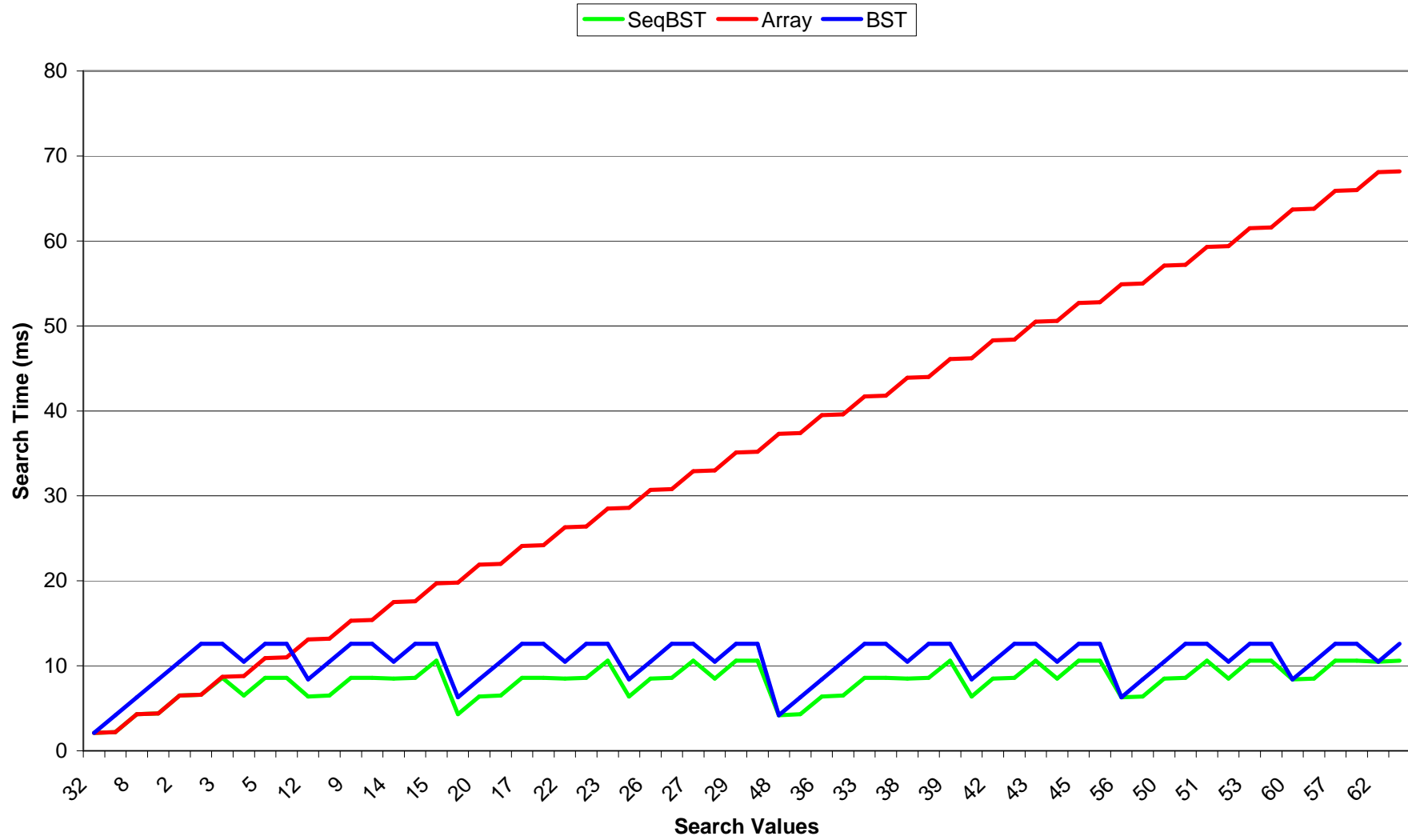
- Main Memory Size:  $16777216 = 2^{24}$
  - Cache Size:  $= 65536 = 2^{16}$
  - Block Sizes: 8 & 64
  
  - The Program allows us to modify the following values:
    - Cache Size, Memory Size, Block Size, Cache Access Time, Memory Access Time, and Cache Type
- 
-

# *Results*

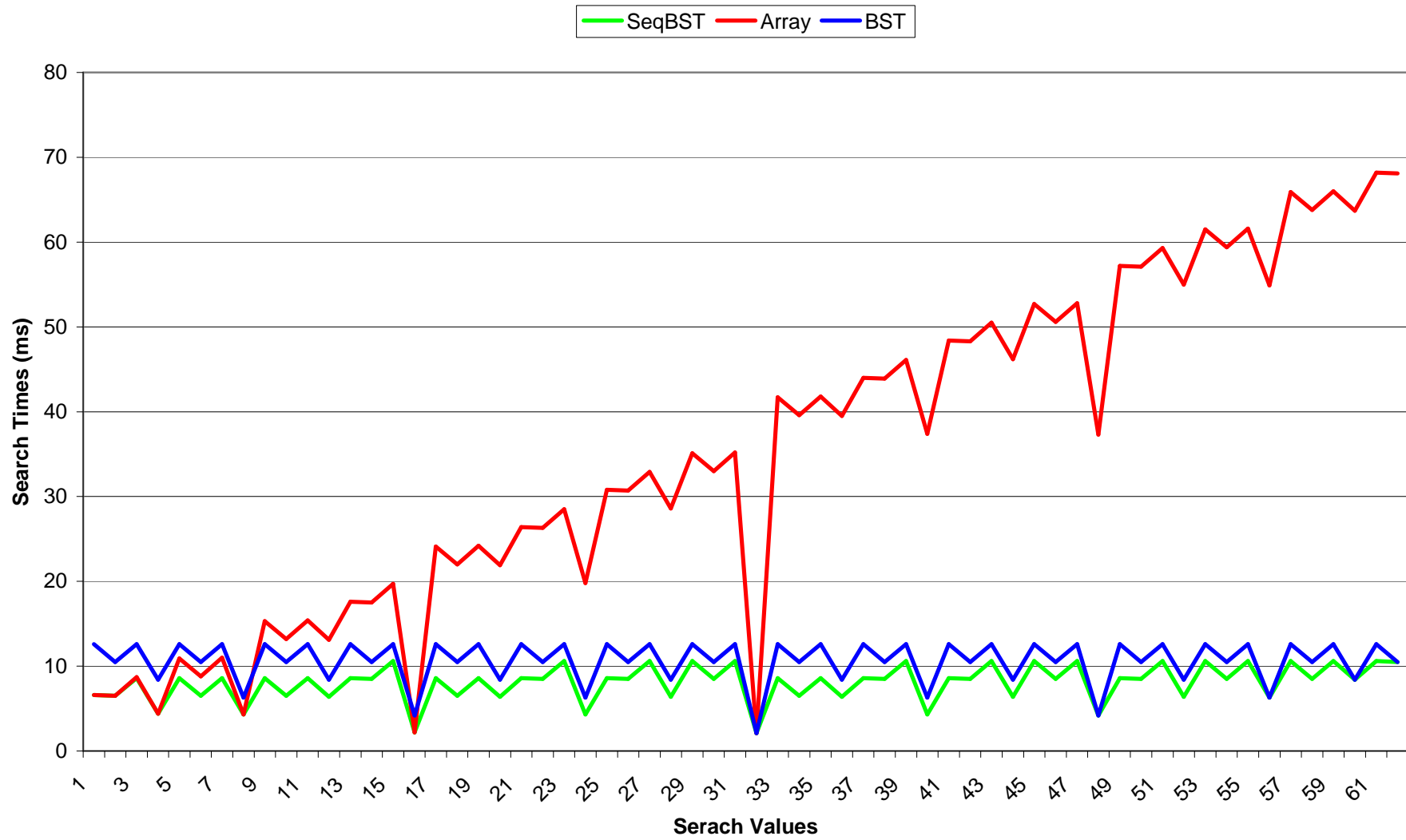
- The Following Pages Illustrate our results.



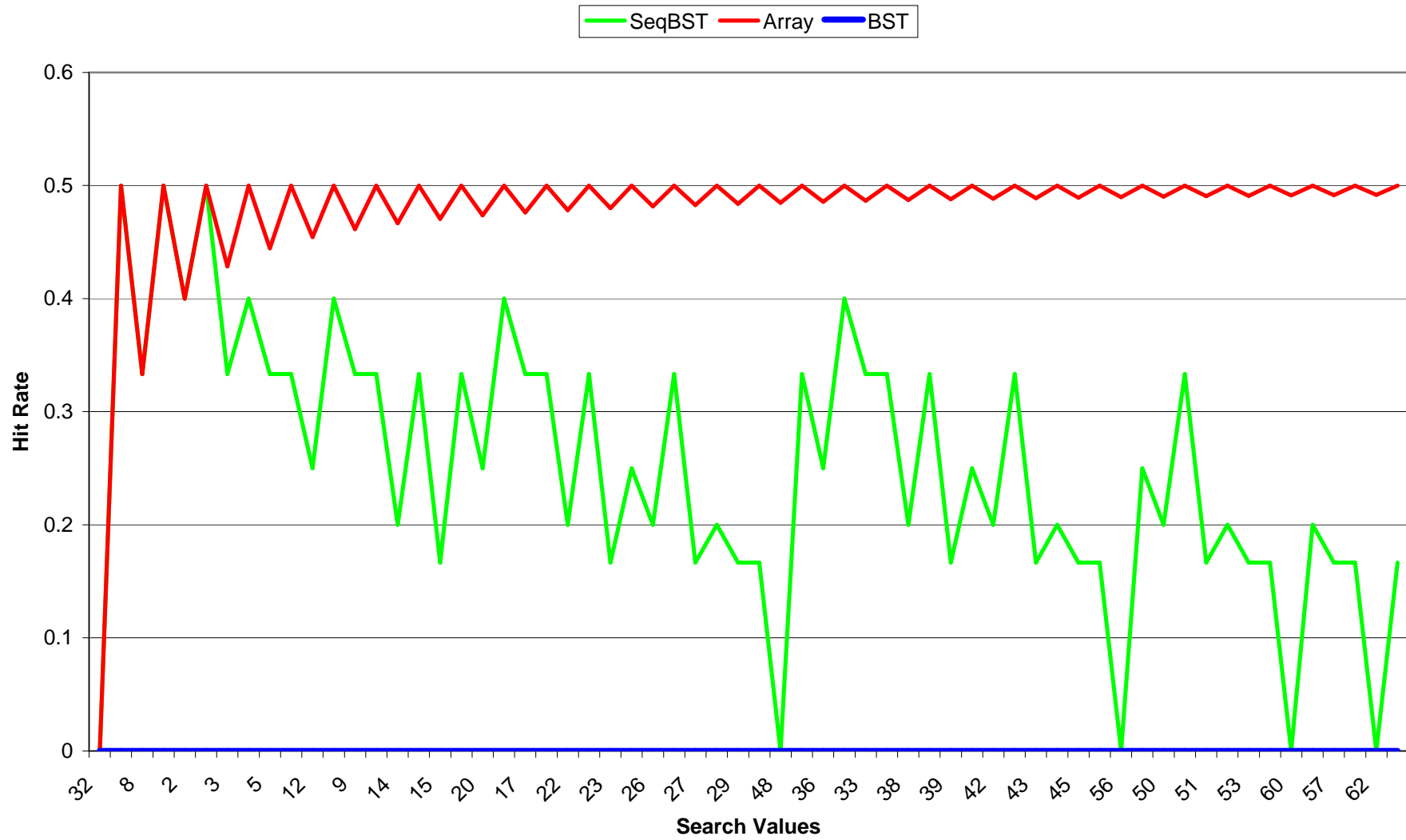
Search Time (ms) (x-values sorted based on memory location)  
Nodes: 63 Cache Size: 4 Block Size: 2



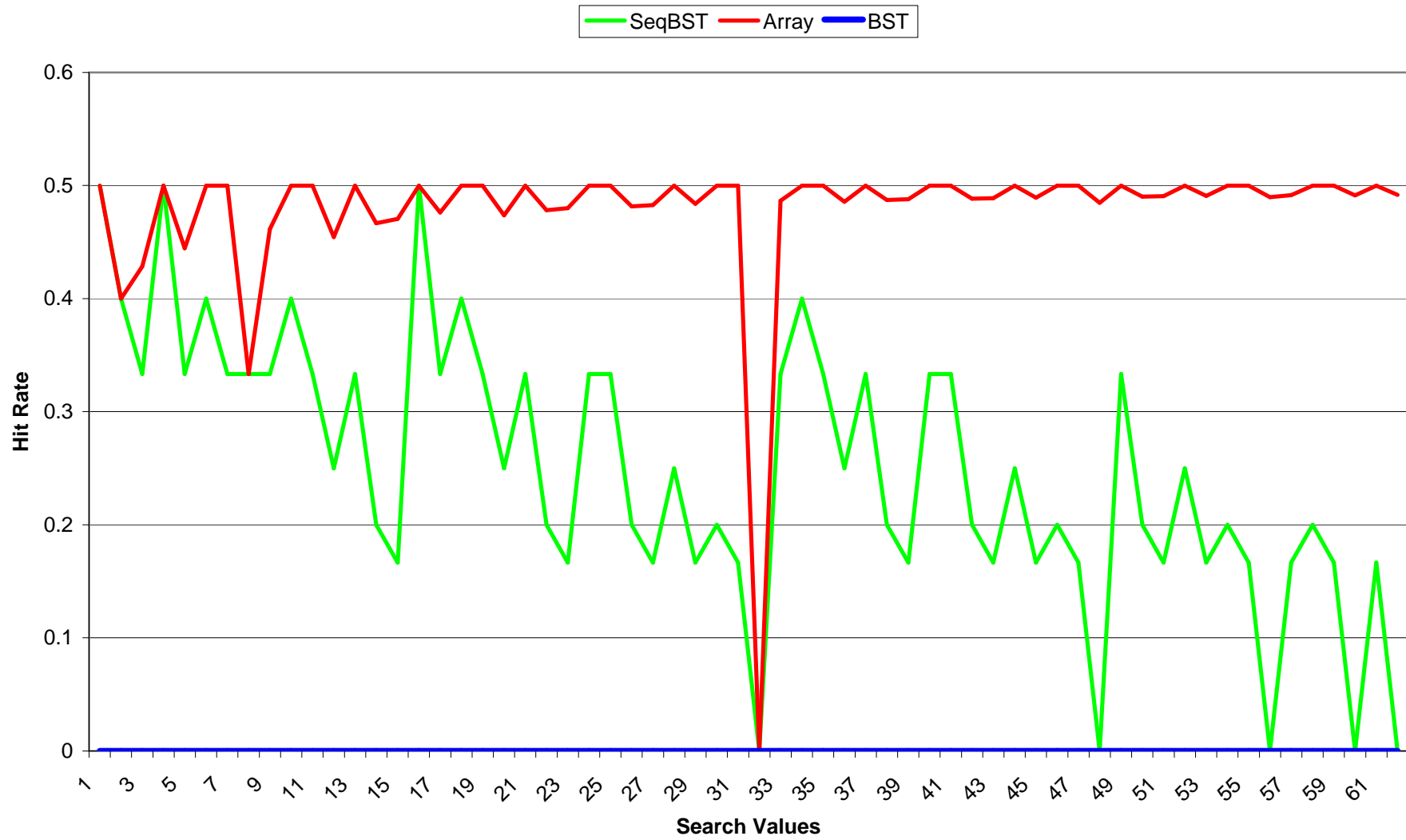
Search Time (ms) (x-values sorted numerically)  
Nodes: 63 Cache Size: 4 Block Size: 2



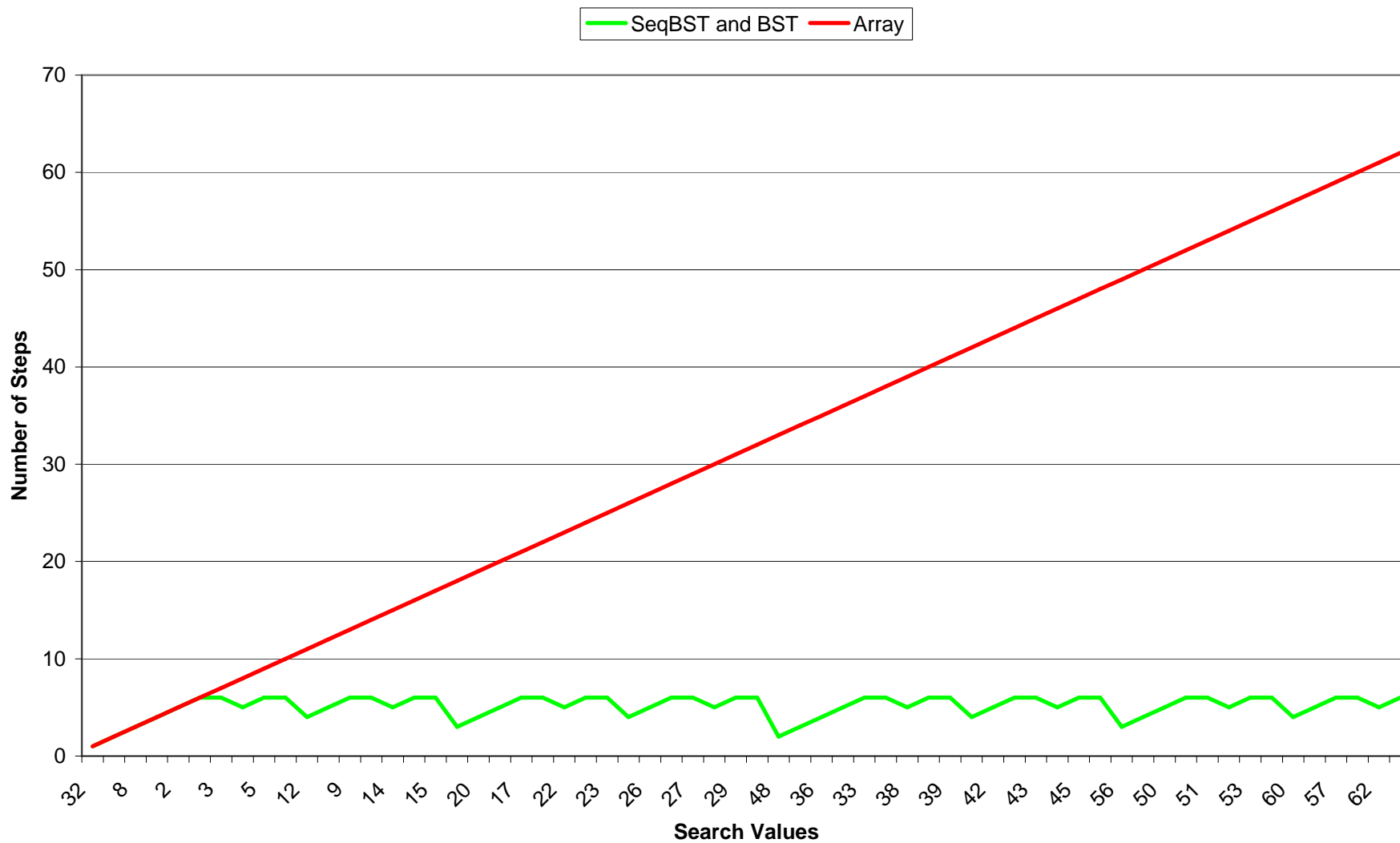
Hit Rate (x-values sorted based on memory location)  
Nodes: 63 Cache Size: 4 Block Size: 2



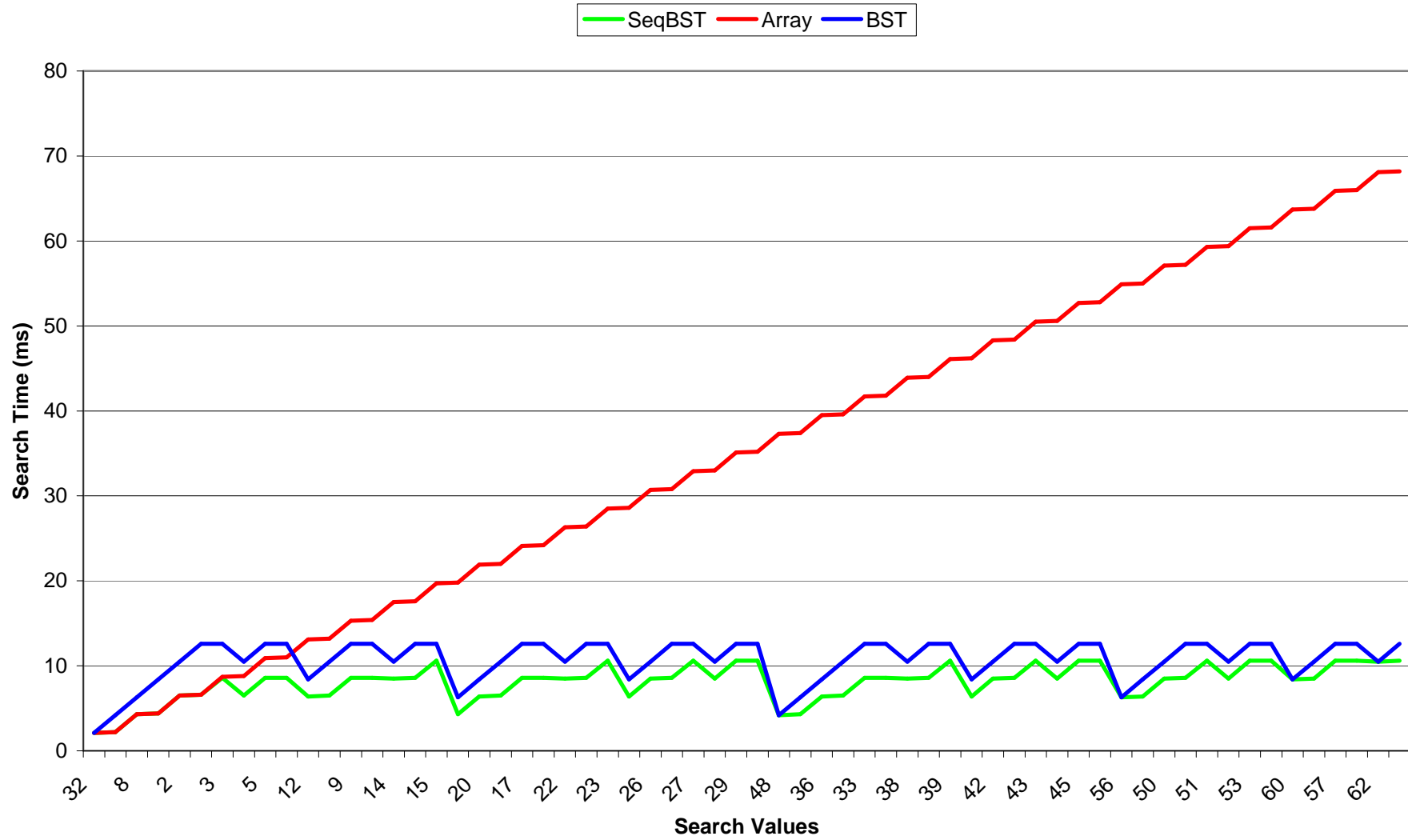
Hit Rate (x-values sorted numerically)  
Nodes: 63 Cache Size: 4 Block Size: 2



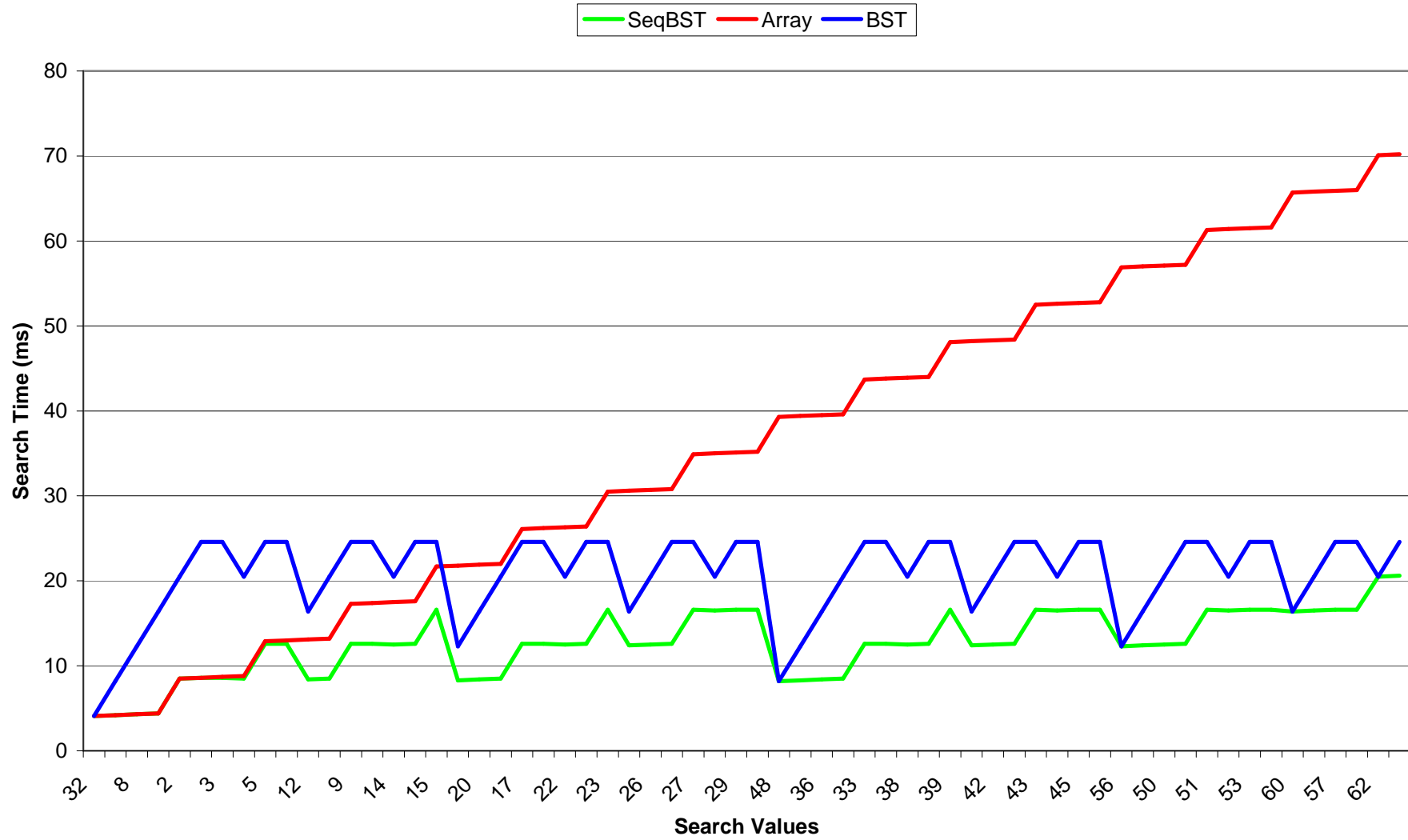
**Search Steps Hit Rate (x-values sorted based on memory location)**  
**Nodes: 63 Cache Size: 4 Block Size: 2**



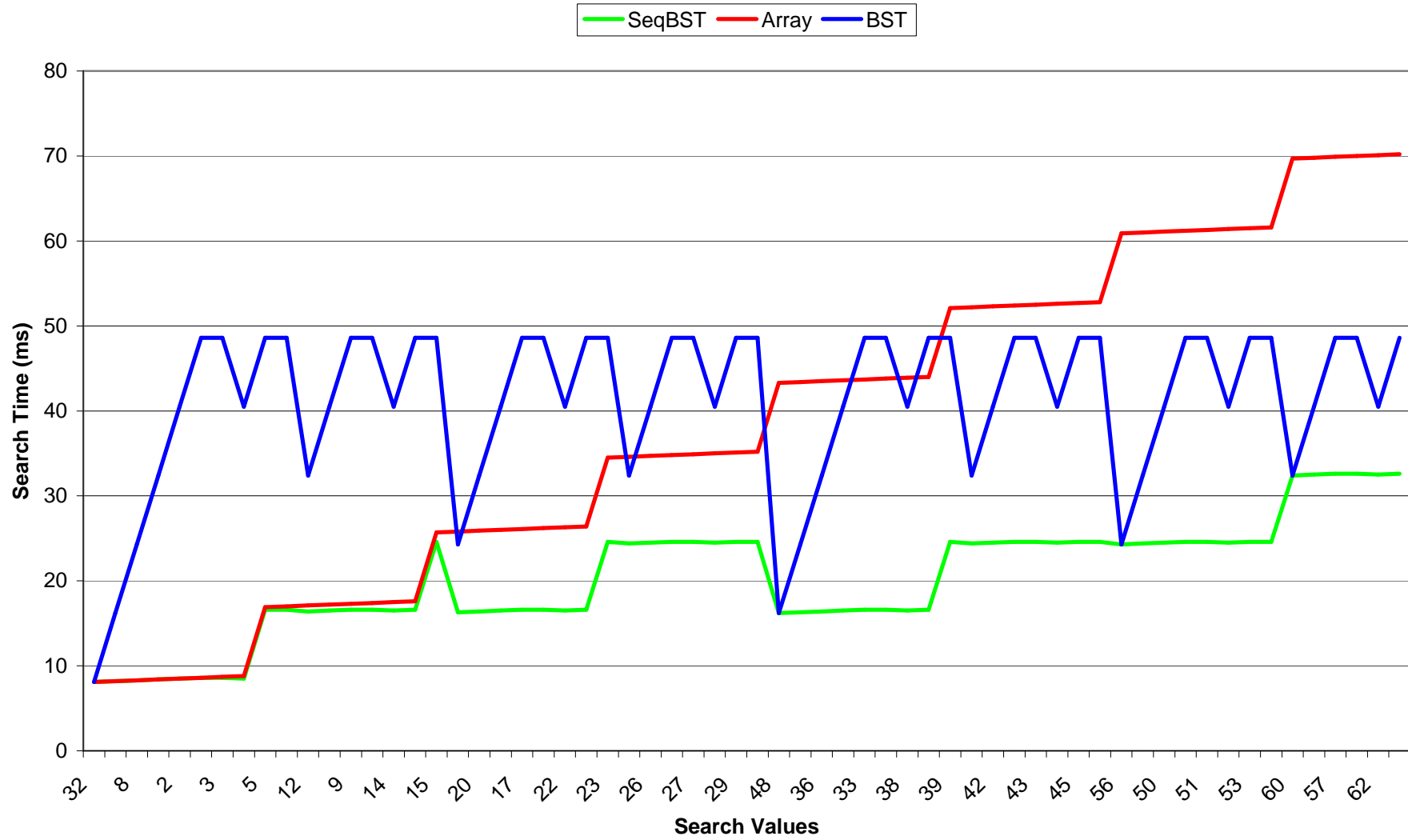
Search Time (ms) (x-values sorted based on memory location)  
Nodes: 63 Cache Size: 32 Block Size: 2



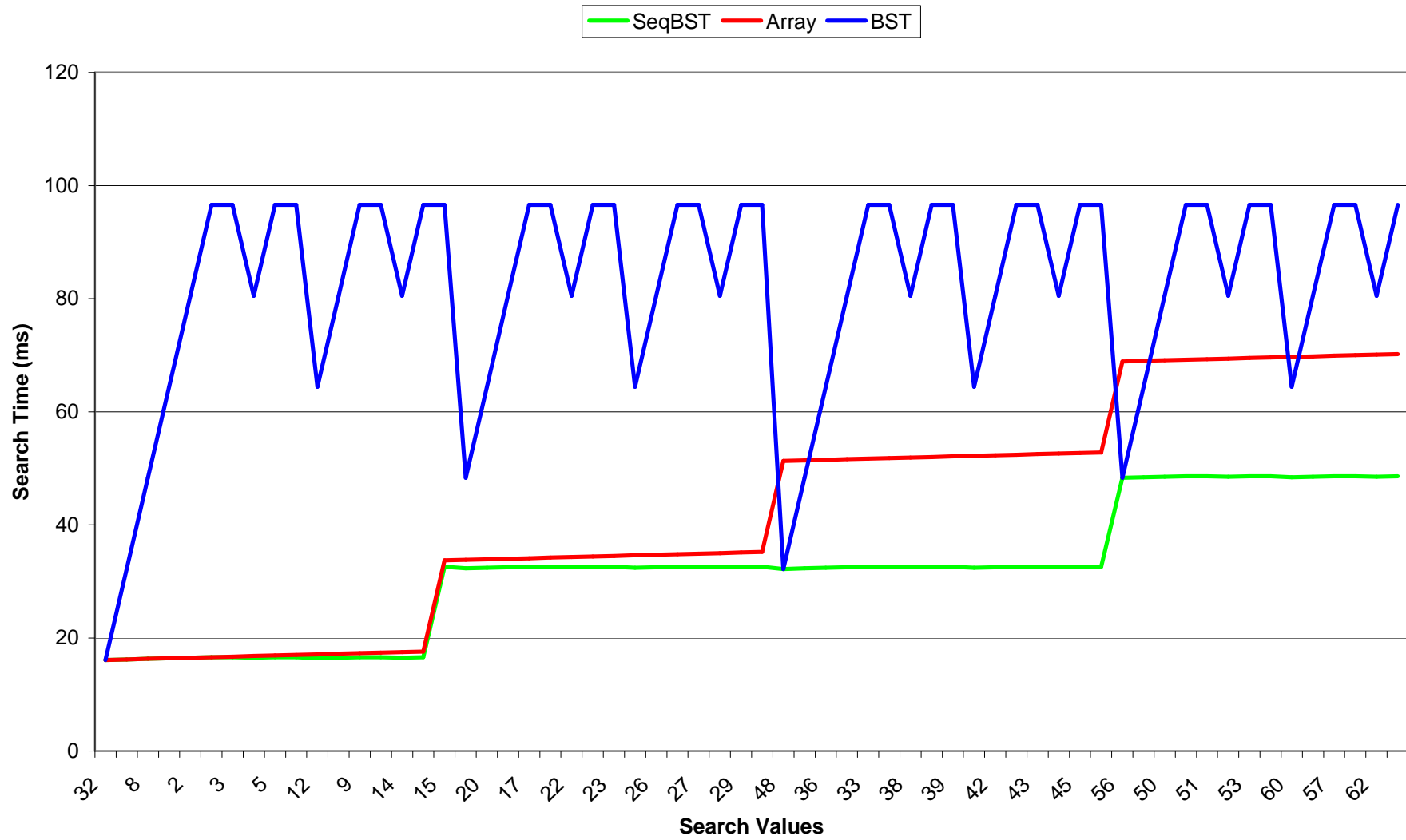
Search Time (ms) (x-values sorted based on memory location)  
Nodes: 63 Cache Size: 32 Block Size: 4



Search Time (ms) (x-values sorted based on memory location)  
Nodes: 63 Cache Size: 32 Block Size: 8



Search Time (ms) (x-values sorted based on memory location)  
Nodes: 63 Cache Size: 32 Block Size: 16



# *Conclusions / Moving Forward*

- Our data structure achieved the goal of  $O(\log n)$  search time and its realtime performance is greater than that of arrays and binary trees.
- Improvements / Future Work:
  - AVL Trees
  - Broaden Test Scope
  - Eliminate Reference Binary Search Tree