ISSN: 2582-7219

International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Impact Factor: 8.206

Volume 8, Issue 10, October 2025

© 2025 IJMRSET | Volume 8, Issue 10, October 2025| DOI:10.15680/IJMRSET.2025.0810003

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Comparative Analysis of Data Structures and Algorithms in C, Java, and Python

Mamatha M A[1], Anish Kumar Gaurd[2], Bharath S R[3]

The Oxford College of Science, Bangalore, Karnataka, India[1-3]

ABSTRACT: Data structures and algorithms are the foundation of effective software engineering. While C, Java, and Python are all widely used, they offer distinct approaches due to their language philosophies, syntax, and core features. This article explores fundamental differences, strengths, and weaknesses by comparing implementation, performance, and usability for major structures—arrays, linked lists, stacks, queues, trees, and hash tables. Algorithmic trade-offs are highlighted for sorting and searching, and practical usage insights guide academic and development best practices.

I. INTRODUCTION

Efficient use of data structures and algorithms is crucial for well-performing software. Comparing C, Java, and Python opens insight into system control (C), modularity (Java), and rapid prototyping (Python), impacting memory, speed, and software maintainability. This article provides code samples, complexity analysis, and practical commentary for each language.

Language Overview

Language	Pros	Cons
C	Speed, direct hardware access	Manual memory/error management
Java	OOP, cross-platform, rich libraries	Verbosity, runtime overhead
Python	Readability, flexible syntax	Slower, less control over resources

II. COMPARATIVE DATA STRUCTURE IMPLEMENTATION

Arrays

- **C:** Static or dynamic arrays; direct memory, pointer arithmetic.

```
int arr[10]; // Static array
```

- **Java:** Array objects; dynamic using ArrayList.

```
int[] arr = new int[10];
ArrayList<Integer> list = new ArrayList<>();
```

- **Python:** Lists; dynamic and resizable.

```
arr = [1, 2, 3, 4, 5]
```

© 2025 IJMRSET | Volume 8, Issue 10, October 2025| DOI:10.15680/IJMRSET.2025.0810003

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Linked Lists

- **C:** Manual node and pointer management.

```
struct Node { int data; struct Node* next; };
```

- **Java:** LinkedList class; OOP style.

```
LinkedList<Integer> list = new LinkedList<>();
```

- **Python:** Manual implementation or third-party libraries.

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
```

Stacks and Queues

- **C:** Built with arrays or linked lists; manual push/pop.
- **Java:** Stack and Queue interfaces; easy use.
- **Python:** List as stack, collections.deque for queue.

Trees and Hash Tables

- **C:** Pointers and structs for custom trees; hash tables usually manual or via libraries.
- **Java:** TreeMap, HashMap, and third-party libraries.
- **Python:** Dictionary for hash tables, easy custom trees via classes.

III. ALGORITHMIC EFFICIENCY

Sorting and Searching

- **C:**
  - Sorting: qsort (O(n log n)), manual quicksort/mergesort.
  - Searching: Manual binary search (O(log n)), linear search (O(n)).
- **Java:**
  - Arrays.sort, Collections.sort; optimized mergesort/quicksort.
  - Search: Binary search in Collections, HashMap O(1) lookup.
- **Python:**
  - sorted() uses Timsort (O(n log n)), list.index for search (O(n)), dictionary/set lookup for O(1).

Big-O Complexities (Examples)

Algorithm	C	Java	Python
Linear Search	O(n)	O(n)	O(n)
Binary Search	O(log n)	O(log n)	O(log n)
Hash Lookup	O(1)	O(1) HashMap	O(1) dict
Quick Sort	O(n log n)	O(n log n)	O(n log n)

© 2025 IJMRSET | Volume 8, Issue 10, October 2025| DOI:10.15680/IJMRSET.2025.0810003

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Use Case Scenarios

- **C:** Systems programming, device drivers, embedded systems—when speed and control matter most.
- **Java:** Enterprise applications, Android apps, web backends—strong OOP and scalability.
- **Python:** Data science, scripting, machine learning—ease, rapid iteration, rich library support.

Industry Adoption

- **C** remains dominant in firmware, kernel, and high-performance computing.
- **Java** is standard for cross-platform desktop and backend server work.
- **Python** is exploding in popularity for data-related work, automation, and rapid prototyping.

Sample Table: Structural Comparison

Structure	C	Java	Python
Array	Manual size, static/dynamic	Array/ArrayList	List, dynamic
LinkedList	Manual, pointer-based	LinkedList class	Manual/third-party
Stack	Manual implementation	Stack/Deque classes	List, deque
Queue	Manual implementation	Queue/Deque interfaces	deque, queue module
HashTable	Manual, or via libraries	HashMap, Hashtable	Dictionary

Practical Advantages and Limitations

- **C Strengths:** Unparalleled speed, fine memory control.
- **Java Strengths:** Safety, scalability, easy-to-use libraries.
- **Python Strengths:** Concise, excellent for beginners, huge ecosystem.
- **C Weaknesses:** Hard to debug, longer code, dangerous memory bugs.
- **Java Weaknesses:** Sometimes verbose, slower than C.
- **Python Weaknesses:** Not for low-level performance, less deterministic memory usage.

IV. SAMPLE CODE COMPARISON

- **Stack Push in C:**

```
stack[++top] = value;
```

- **Java Stack Push:**

```
stack.push(value);
```

- **Python Stack Push:**

```
stack.append(value)
```

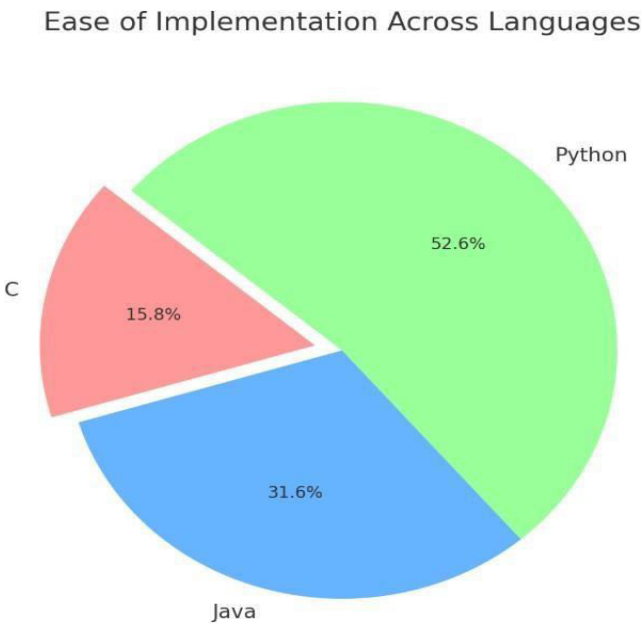
© 2025 IJMRSET | Volume 8, Issue 10, October 2025| DOI:10.15680/IJMRSET.2025.0810003

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Result



Here's the **pie chart** showing **Ease of Implementation** across C, Java, and Python.

- **C**: 3/10 → smallest slice (most complex to implement)
- **Java**: 6/10 → medium slice
- **Python**: 10/10 → largest slice (easiest to implement)

V. CONCLUSION

This comparative analysis of data structures and algorithms in C, Java, and Python highlights the unique strengths and trade-offs of each language. C offers unparalleled control and efficiency, making it ideal for performance-critical and system-level programming, although it demands careful manual memory management and low-level coding expertise. Java balances performance with usability by providing a robust object-oriented framework, automatic memory management, and extensive built-in data structures, which suits enterprise-scale and cross-platform development. Python excels in simplicity and rapid development, offering versatile high-level data structures and vast library support, mainly favored in data science, scripting, and prototyping, despite its slower execution speed.

The choice among these languages depends on project requirements balancing speed, memory control, ease of use, and development speed. While algorithms maintain their theoretical complexity across languages, implementation convenience and ecosystem support significantly influence productivity and code maintainability. Understanding these subtle differences empowers developers and students to select the appropriate language and data structure paradigm tailored to their specific tasks.

This exploration underscores the importance of foundational data structures and algorithms knowledge combined with awareness of programming language characteristics to build efficient, scalable, and maintainable software solutions.

REFERENCES

1. Kernighan, B. W., Ritchie, D. M. (1988). The C Programming Language, Prentice Hall.
2. Oracle. Java Platform Documentation. https://docs.oracle.com/javase
3. Python Software Foundation. https://docs.python.org/3
4. Goodrich, M. T., Tamassia, R., Goldwasser, M. H. (2014). Data Structures and Algorithms in Java, Wiley
5. Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). Introduction to Algorithms, MIT Press

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com