Makori Chacha 301414318

Library book storage system – Data Structure analysis

To manage book data efficiently, I chose std::vector as the underlying data structure. This is based on the problem's constraints and performance goals:

The library has fixed shelf space(i.e a fixed upper bound on the number of books),

Fast retrieval by ID is a top priority, and

Occasional swapping of books is expected

Why Vectors Are appropriate

Vectors are ideal in this case because:

Each book has a unique ID starting at 0 that is tied to a shelf position. This allows direct access by index in O(1) time.

Swapping or updating a book at a given position is done in constant time using vector[index] = newBook, again O(1).

Unlike static arrays, vectors provide automatic resizing when needed, though resizing won't be frequent in this fixed-size application.

Vectors offer better memory management than static arrays because resizing doesn't require manual copying.

Function analysis (with Big-O)

Function	Description	Big-O	Justification
addBook	Insert book at available ID	O(1)	Direct index assignment using vector
removeBook	Remove book by ID	O(1)	Direct access and set to nullptr or default

swapBook	Replace book at given ID	O(1)	Direct index replacement
checkOutBook	Mark book as checked out	O(1)	Access book by ID and update flag
ReturnBook	Mark book as returned	O(1)	Same as above
isInLibrary	Check availability	O(1)	Flag check via direct index
getBookInfo	Retrieve book data	O(1)	Direct lookup using ID
totalBooks	Count total books	O(n)	Must scan vector for all valid books
booksInLibrary	Count books currently in	O(n)	Traverse and check in/out flag
booksLentOut	Count books currently lent out	O(n)	Same as above

Static Arrays vs. Vectors

If a static array were used:

There would be no dynamic resizing, so any capacity change would require allocating a new array and copying data — an O(n) operation in both time and space.

While access time would remain O(1), modifications like adding or removing books would be more error-prone and less flexible.

Conclusion

Std::vector is well-suited to the library's needs. It supports:

Fast, direct book access by ID (key requirement)

Efficient updates and status flags

Safe dynamic memory management, even if resizing isn’t expected.

Vectors provide O(1) performance for nearly all operations and O(n) only when aggregate data(like total books) must be scanned. This choice ensures both speed and maintainability, which matches the library’s operational model and system goals.