Programming Practicum Report:
Meeting 1
Implementing a Book Inventory System

R. Ethan Halim

August 26th, 2024

1 Determining the Data Structure

The necessary data structure that is required has to be able to store a series of ISBN codes that represent the books stored in the inventory. The data structure needs to endure these three operations efficiently for the purpose of the system, with the stored elements being the ISBN codes:

1. checking for element membership;
2. element appendage;
3. element removal;

It is assumed that the most common function that would be used is to check whether an ISBN code is present inside of the structure. Therefore, the said data structure must be optimized for fast look-up operations.

The first option that one may consider is dynamic arrays (equivalent to the STL type `std::vector` in C++ or the builtin type `list` in Python). However, in order to check element membership among elements stored in a dynamic array, one must iterate over the entire array, therefore resulting in the time complexity of $O(n)$, where n is the amount of elements that the array has.

A better option is sets (equivalent to the STL type `std::set` in C++ or the builtin type `set` in Python). They provide a faster time complexity of $O(\log n)$ for checking element membership, as they utilize red-black trees for logarithmically faster look-up operations. Additionally, hash map sets (equivalent to the STL type `std::unordered_set` in C++) provide the fastest time complexity for look-up—$O(1)$, with the trade-off being more memory usage.

Moreover, as for element insertions and removals, sets can perform with the similar time complexity of $O(\log n)$. The average time complexity of hash map sets for the two operations is similarly $O(1)$, and the worst time complexity is $O(n)$ upon the instance of hash map rehashing, as documented in **cppreference.com**.

Thus, preceding all functions, there is to be a global variable that is a set, which is accessible from anywhere in the code, whose name shall be set as `Inventory`. It will store the ISBN codes of all books in the inventory.

Pseudocode

```
let Inventory be an empty set of integers
```

ISBN codes are sequences of 13 (formerly, 10) decimal digits separated by dashes. Because the positions of the separating dashes are fixed, and newer standards are backwards-compatible by adding trailing zero (s) preceding the most significant digit, they can therefore be safely stored inside of a scalar integer. While the pseudocode snippets in this document do not include the integer type, in C/C++, the integer type should ideally be an `uint64_t` (64-bit unsigned integer), which can safely store any 19-digit combination of decimal numbers.

2 Inventory Checking

The function detailed in the pseudocode below implements a `check` function which checks whether the provided `ISBN` argument is present in the `Inventory` set. It utilizes the set type's membership-look-up method, represented in the pseudocode as '`is in`' (equivalent to the `.find` method in C++ `std::set`s). The function returns `true` if the given `ISBN` is in the `Inventory`, and `false` otherwise.

Pseudocode

```
function check(ISBN) -> boolean:
    let ISBN be an integer argument

    if ISBN is in Inventory:
        print "Book with ISBN ", ISBN, " is in the inventory."
        return true
    else:
        print "Book with ISBN ", ISBN, " is not present."
        return false
```

3 Adding to the Inventory

The function detailed in the pseudocode below implements an `add` function which appends the provided `ISBN` argument to the `Inventory`, given that there aren't any duplicates, in which case the function would return `false` upon unsuccessful appendage, and `true` otherwise. This function utilizes the `check` function from before, in order to anticipate for the aforementioned duplicate case condition. The appendage of the provided `ISBN` into the `Inventory` is alike to the `.insert` method in C++ `std::set`s.

Pseudocode

```
function add(ISBN) -> boolean:
    let ISBN be an integer argument

    if check(ISBN):
        # This prevents duplicates.
        print "The book will not be added,",
              "as it is already present in the inventory."
        return false
    else:
        append ISBN to Inventory
        print "Book has been added to the inventory."
        return true
```

4 Removing from the Inventory

The function detailed in the pseudocode below implements a `remove` function which removes the provided `ISBN` argument from the `Inventory`, given that the ISBN has already been logged in the `Inventory` and from thereon, returns `true` for successful removal. Otherwise, if no such ISBN exists, the function returns `false` to indicate failure. The removal of the provided `ISBN` from the `Inventory` is alike to the `.erase` method in C++ `std::set`s.

Pseudocode

```
function remove(ISBN) -> boolean:
    let ISBN be an integer argument

    if check(ISBN):
        remove ISBN from Inventory
        print "Book has been removed from the inventory."
        return true
    else:
        # This prevents triggering an exception.
        print "Nothing is removed."
        return false
```

5 Command-Line Interface

In order to ease the usage of the three functions implemented above, below is the pseudocode of a command-line interface (CLI) for utilizing the functions through the command-line. The output comes from the `print` statements included in the implementation of the functions.

Pseudocode

```
function main():
    let Input be an integer variable
    let ISBN be an integer variable

    loop indefinitely:
        print "Which operation would you like to do?"
        print "(1) Checking a book's availability"
        print "(2) Add a new book to the inventory"
        print "(3) Remove a book from the inventory"
        request Input from user

        match Input:  # the same as ‘switch‘ in C++
            case 1:
                print "Enter the ISBN code of the book whose ",
                      "availability you'd like to check:"
                request ISBN from user
                check(ISBN)
                break

            case 2:
                print "Enter the ISBN code of the book you'd like "
                      "to add in the inventory:"
                request ISBN from user
                add(ISBN)
                break

            case 3:
                print "Enter the ISBN code of the book you'd like "
                      "to remove from the inventory:"
```

```
        request ISBN from user
        remove(ISBN)
        break

    default:
        print "Invalid option!"
```