# CS 341 A3 Q2

Tareef Dedhar - 20621325

June 15, 2018

## 1 Bookshelf Problem

### 1.1 Description/Correctness

First, we assume W >= the max thickness, otherwise this is impossible to solve. So, given that, for each book, if it fits in the space left on the current shelf, shelve it. If not, start a new shelf and add it there. Repeat this until all books are shelved. This is an optimal solution because since we have a defined order of books, for any given book being shelved, there are only 2 options: shelf on the current shelf, or a new one. This algorithm chooses the former, which does not increase our target variable (number of shelves used), whenever possible. So it must be the minimal solution.

### 1.2 Pseudocode

```
Librarian(books, numShelves, spaceLeft)
{
  if (books.empty) return numShelves

  width = pop(book)

  if width <= spaceLeft
  {
    spaceLeft -= width
    return Librarian(books, numShelves, spaceLeft)
  }
  else
  {
    ++numShelves
    spaceLeft = W - width
    return Librarian(books, numShelves, spaceLeft)
  }
}

call used: Libarian(books, 0, W)
```

### 1.3 Time Complexity

This is a recursive algoritm that processes 1/n books each call, so the runtime will be Theta(n * loop), where loop is the runtime of the inner loop code. The first check made (if books is empty) should be a constant time task. The if/else check is a numeric comparison, so this is also constant time. Inside each if/else case, we have either 1 or 2 constant time assignment/incrementing operations, and recursive call. All of these should each (not considering the runtime of the recursive call's code, but the call itself) take constant time.

So, since we have a function comprised of constant time functions being called n times, our runtime is Theta(n)