October 29, 2015

CS 577 - Homework 6
Sejal Chauhan, Vinothkumar Siddharth, Mihir Shete

1 Graded written problem

Input: In a library, there are n books that must be stored in alphabetical order in adjustable height shelves. Each book b_i has height h_i and a thickness t_i. Each shelf has width w. The next shelf will be placed on top of $\max(h_i)$ on the given shelf.

Output: Minimize the total height of the shelves to store all the books.

1.1 Algorithm

We have to place books on shelves alphabetically and so the intuition to minimize the height is that we will place a book on a new shelf only if the current shelf does not have any space for the book or if the placing of this book on a new shelf will minimize the overall height in the future. To find if placing a book on new shelf reduces the height of the shelves we need to consider all books which need to be placed after it as well, to achieve this we will start from the last book and calculate the least possible height when that book is kept on a new shelf.

Basically, we are using a top-down Dynamic programming approach to solve this problem. First we consider the last book in the alphabetically ordered list, if the book is placed on a new shelf then the min height of the shelf to place all the books after than will h_n because this is the last book after all. And we store this min height in an Array we can lookup. Next we consider the $(n-1)^{th}$ book, if this book is on a new shelf then to calculate the min height required to place all books we will have to consider all the permutations in which books after $(n-1)^{th}$ book can be arranged on the shelf. Since there is just 1 book after this book we will have only 2 permutations, first is that both $(n-1)^{th}$ and n^{th} books are on the same shelf and second is that they are on different shelf, in first case the minimum height of the shelf is minimum height of both the books and in second case it is the height of $(n-1)^{th}$ book plus the minimum of shelf height required if n^{th} book is placed on a new shelf which we have already calculated and stored in an Array. So the minimum shelf height required to place the $(n-1)^{th}$ book and all the books after than will be the min height obtained after considering every permutation and we will also store this value in the Array.

So, when we want to calculate the minimum height to place all the books after i, including i^{th} book we will consider all the permutations of possible arrangement of the preceding books and choose the permutation which gives minimum height and store its value in an Array to be used in later calculation. The algorithm in pseudocode looks as follows:

Algorithm 1 Algorithm to minimize the height of shelves to store all books

```
1:  procedure ARRANGE-BOOKS(B[n], W[n], h[n])
2:      LeastHeight[n] ← ∞
3:      i ← n
4:      while i ≠ 1 do
5:          if i = n then
6:              LeastHeight[i] = h[i]          ▷ if Last book is on new shelf the min height is the books height
7:              i ← i − 1
8:              continue
9:          AllBooksWidth ← 0
10:         j ← i
11:         h_shelf ← 0
12:         while j ≠ n + 1 do
13:             AllBooksWidth ← AllBooksWidth + W[j]
14:             if AllBooksWidth>ShelfWidth then
15:                 break
16:             h_shelf ← max(h_shelf, h[j])
17:             totalHeight ← h_shelf + LeastHeight[j + 1]
18:             if totalHeight<LeastHeight[j] then
19:                 LeastHeight[j] ← totalHeight
20:             j ← j + 1
21:         i ← i − 1
22:     return LeastHeight[1]
```

The procedure *Arrange-Books* take 3 input arguments $B[n]$, $W[n]$ and $h[n]$, $W[n]$ is the list of width of all the books arranged in alphabetical order and $h[n]$ is a list of their heights. $ShelfWidth$ denotes the width of the shelf which is a constant.

As we can see, we are using 2 loops in our procedure, first one is used to iterate over all the books from the end of the sorted list. The second one will go over all the permutations in which the books coming after the book we are considering can be arranged and will store the least possible height. While considering each permutation we also check if the width of the books in the shelf will exceed the width if we place a new book and if it does we will break from the loop to consider the next book from a new shelf. The work done by the inner loop is bounded by some constant value and it will run $n - 1$ times, the outer loop will run n times and so the performance of our algorithm will be $O(n^2)$.