HW 5

Yunhao Lin

October 16, 2018

1. **Algorithm:**
Since we are arranging books on m shelves. Know for j $\in \{1, ..., m\}$, and $x_j = W - \sum_{i:book\ i\ is\ on\ shelf\ j} wi$, where W is the width of each shelf. In order to minimize $\sum_{j\in[m]} x_j$, the total waste space across m shelves, we are arranging books in order. Assuming the next book to be put into j_{th} shelf is z_i. While index of i is less than the total number of books n, we need to check if $x_j - w_i$ is greater than 0. If it is, then book z_i is able to put in j_{th} shelf and we just put it there. If not, z_i would be the first book in the next shelf.
Proof of Correctness:
Claim: The algorithm is going to help us minimize the total waste space , which is $\sum_{j\in[m]} x_j$.
Base Case: Since the first book is always going to add at the first book shelf, else we are wasting all the space in first shelf. It does not make sence to put it in the second shelf.
So in this specific case , base case is when have 2 books to put in shelf. In this scenario, the $1st$ book is already in the $1st$ shelf. And the algorithm determine if the $2nd$ can be put in the current shelf. If it can, then both book are in the same shelf, the total waste space would be $(W - w_1 - w_2)$. If we did not put the book in $1st$ shelf, even though it can be put there, then $2nd$ book is put on the $2nd$ shelf, and the total waste is $(2W - w_1 - w_2)$ which is much larger then the algorithm proposed. If $2nd$ book is too wide to put in $1st$ shelf, then it can only be put in the $2nd$ shelf as suggested by the algorithm.
Inductive Hypothesis: P(n):Suppose when we have k number of books, and the algorithm correctly provide with the minimized wasted space.
Inductive Steps: Since we already all for k books, algorithm would arrange it to minimize the waste space. We need to prove that for $(k + 1)_{th}$ book, the algorithm would still minimize the waste space. After k_{th} book is added to the shelf, we are at l_{th} shelf. And there is no need to worry about any shelf before l_{th} shelf, since every shelf at this moment is the minimized in waste space. So right now, in order to add in the $(k + 1)_{th}$ book, there is two possibilities.

- First, $(k + 1)_{th}$ book can fit on l_{th} shelf, then the algorithm add the $(k + 1)_{th}$ book on l_{th} shelf. And the totoal waste would be all the waste for k books $-w_i$. If we put this on the shelf below l_{th} even though we can fit it in l_{th} shelf. The total waste would be all the waste for k books $W - w_i$, which means we are wasting a whole length of W. And the algorithm provided with the best arrangement of the book.
- Second, $(k + 1)_{th}$ book cannot fit on l_{th} shelf, $x_l - w_{k+1}$ is smaller than 0. Then $k + 1$ book can only be put on the $(l + 1)_{th}$ shelf. The algorithm also provide the correct output.

Therefore, P(k) holds then P(k+1) holds. By induction, the algorithm would correctly arrange books to minimize the total waste space, which is $\sum_{j\in[m]} x_j$.

Running time: Since there is a total number of n books, we are doing comparsion and adding for each one of them. The total running time could be analysed to be $O(n)$.

2. Here is the psudo-code for No.2

Algorithm 1 Optimized book arrangement

```
1:  Initialize Arraylist q                                  ▷ Store the index of begin book of each shelf
2:  procedure MINPOWER(i, j)
3:      if i = n then
4:          return W − (∑_{k=j}^{i} w_k)^3                  ▷ The result by definition is just x_j
5:      else
6:          if x_j < w_{i+1} then                           ▷ When the book cannot fit in current shelf
7:              Record index q.add(0, i + 1)
8:              return minpower(i + 1, i + 1) + x_j^3
9:          else if minpower(i + 1, j) > minpower(i + 1, i + 1) + x_j^3 then
10:             Record index q.add(0, i + 1)                ▷ Add to the first position in q
11:             return minpower(i + 1, i + 1) + x_j^3
12:         else
13:             return minpower(i + 1, j)
```

For the above algorithm, there is few things need to clarify. We define i as the index of the current book(the last book we already put in shelf), and j as the index of the first book in the current shelf. The algorithm has a recursive call $minpower(i, j)$ that would first determine if the next book $(i + 1)_{th}$ book can be put in the current shelf j. If not, we call recursive call $minpower(i + 1, i + 1)$ putting $(i + 1)_{th}$ book in the next shelf and assign the first book in the next shelf to be the $(i + 1)_{th}$ and the waste is going to be add the space $W - (\sum_{k=j}^{i} w_k)^3$. Else, the $(i + 1)_{th}$ book can be fit in j_{th} shelf and here we consider two cases if we should include $((i + 1)_{th})$ book in this shelf or put it in the next shelf. If don't include is going to minimize x_j^3, add the total cost of the current shelf with the power need if we put $(i + 1)_{th}$book in the next shelf. Else just return $minpower(i + 1, j)$, the total power used if we include the book in this current level. Every time changing shelf is the better or only solution , we just record it in the first place in q. In the end it would result in a beautiful list with all the indexs of books at the begining of each shelf.

Proof of Correctness:
The algorithm would correctly return an list of all the first indexes of each shelf at completion.
Claim: The algorithm would correctly give a list of all the indexes of the first book in each shelf.
Base Case: Our base case is the last call to the recursive call which is $minpower(i, j)$, where i is equal to n. By the definition of i, the last book is already in the shelf and in this case , the algorithm returns the waste of this level which is $W - (\sum_{k=j}^{i} w_k)^3$.
Inductive Hypothesis: P(x) Everytime we call $minpower(a, b)$ we are going to find the minimum power used in shelves from $(a + 1)_t h$ book to $n_t h$ book.
Inductive Steps: We need to prove that $minpower(a - 1, b)$ also find the minimum power used in all the shelves from $(a)_t h$ book to n book. Since at this case, $a - 1$ cannot equal to n, therefore no need to worry about line 3 to 4. Then we call $minpower(a - 1, b)$, there is three possible situations.

- If a_{th} book cannot fit in this current shelf, then the algorithm is going to call $minpower(a+1, b)$, where $b = a + 1$, and by our Inductive Hypothesis that is going to get the minimum power used form $(a + 1)_t h$ book to n and we add the waste on this level is still the smallest waste, this case holds. We add the index of the $(a + 1)_{th}$ book to the q, because this is when we start a new line.
- If a_{th} book can fit in this current shelf, and the power of including this a_{th} book in the current shelf is greater than the power making $a_t h$ book the first book in the next shelf. Since both call to $minpower()$ on line 9 is going to return the correct output by inductive hypothesis, we chose the solution that put a_{th} book in the next shelf. And every time we choose this possibility, we are adding the index of $(a+1)_{th}$ book to the q, which makes sure the index is recorded. And we return the totoal waste used by now to the upper level by adding waste on this level $W - (\sum_{k=j}^{i} w_k)^3$ to the $minpower(a + 1, b)$. If power of including this a_{th} book in the current shelf is smaller than the power making $a_t h$ book the first book in the next shelf. We choose to put the book in the current shelf and the algorithm calls $minpower(a + 1, b)$ which by hypothesis is going to return

the minimum power used from $(a+1)_{th}$ to n_{th} book. So here in whichever case, we are choosing the one that uses smaller power, which means we are giving the minimized power from a_{th}

In any case, the algorithm is going to help us minimize the power from a_{th} book to n_{th} book and since we record the first index of each shelf whenever we choose to put book on the new shelf. By induction, the algorithm is going to find the minimized power in all the shlef and since we record all the index of the first book in each shelf. It will give us an list of all the index of the first book in each shelf.

Time complexity:

Since there is n_th book to consider and the worst case for y is n shelves. And every time in the algorithm we are just doing comparsion and addition, since the value from next level is already known. Therefore, the time complexity is going to be $O(n)$.