Rui Yin

HW5

2018/10/16

1. Without loss of generality, suppose that the input array of widths is in the Dewey Decimal System order. Use a greedy paradigm to arrange the books as follow:

   Denote that the former shelf ends at book $k-1$ (and the first shelf, $k=1$). Then, for the present shelf, place books as more as possible. That is to say, find an index k' such that:

$$\sum_{i=k}^{k'} x_i \leq W \quad \text{and} \quad \sum_{i=k}^{k'} x_i > W$$

Time complexity: with the paradigm stated above, the algorithm traverse each item of the input array one time. So the running time is linear to the number of books and shelves, $O(n+m)$.

Correctness: denote the algorithm outputs the partitions as A. If it is not the optimal solution, then there exists an optimal solution O, which is closest to A. Denote the first shelve that the two solutions differ as k. Exchange the shelf to form a new solution O'. Note that each shelf still has extra space, so O' is feasible.

Case 1: if the books in $A[k]$ is more than that of $O[k]$, then there exists some positive integer p, such that books in $A[k+p]$ is less than that of $O[k+p]$. In this case, the remaining extra space from k to $k+p$ remains the same. i.e., O' is optimal.

Case 2: if the books in $A[k]$ is less than that of $O[k]$, with the same consideration, O' is optimal too.

So, O' is optimal and more like A, which is contradictory to that O is the optimal solution closest to A. So A must be an optimal solution.

The whole algorithm can be expressed as below:

```
Init an array bookToShelf of size n
tmpSum = 0, j = 1
For i from 1 to n, do
    tmpSum += w[i]
    if tmpSum > W, do
        j++
        tmpSum = 0
        i--
    else
```

```
            bookToShelf[i] = j
        EndIf
    EndFor
```

2. This problem should be solved by dynamic programming:

$$\min_{j\in[1,2,\ldots,m]} x_j^3 = \min_{j\in[2,\ldots,m]} \left\{ x_j^3 + \left[W - \sum_{i:\text{book } i \text{ in shelf } 1} w_i\right]^3 \right\}$$

Use a matrix P of size $m \times n$ to restore the calculation. Consider the first shelf, it can be filled with book of width $w_1, w_2, \ldots$, until no more books can be placed in. And each case corresponds to a different value of sub-problem. So the recursive formula is as below:

$$P[j,k] = \begin{cases} \infty & \text{if} \sum_{i\in[k,\ldots,k']} w_i > W \\ \min\left[W - \sum_{i\in[k,\ldots,k']} w_i\right]^3 + P[j-1,k-1] & \text{otherwise} \end{cases}$$

<u>Time complexity</u>: for each shelf, the algorithm traverse AT MOST each item of the input array per possible case. So the running time is polynomial to shelf and books, $O(n^2 m)$.

<u>Correctness</u>: here is proof by contradiction. For the case $\sum_{i\in[k,\ldots,k']} w_i \le W$, suppose that there exists some solution such that $P[j,k] < \min\left[W - \sum_{i\in[k,\ldots,k']} w_i\right]^3 + P[j-1,k-1]$. That is to say, $P[j,k] - P[j-1,k-1] < \min\left[W - \sum_{i\in[k,\ldots,k']} w_i\right]^3$, then books k, $k+1$, …, $k' + p(p>0)$ can be placed into the shelf j, which is contradictory to the condition.

<u>The whole algorithm can be expressed as below</u>:

```
Init an array P[j,k] of size m*(n+1)
Set P[:,0] = W^3
For j from 1 to m, do
    k = 1
    tmpSum = w[k]
    While tmpSum<W, do
        residual = W
        i = k
        While residual - w[i] < W, do
            residual = residual - w[i]
        EndWhile
```

$P[j,k] = P[j-1,k-1] +$ *residual* residual* residual;*

EndWhile

EndFor

Return min $P[m,:]$