1105. Filling Bookcase Shelves

ou are given an array `books` where `books[i] = [thicknessi, heighti]` indicates the thickness and height of the `i`th book. You are also given an integer `shelfWidth`.

We want to place these books in order onto bookcase shelves that have a total width `shelfWidth`.

We choose some of the books to place on this shelf such that the sum of their thickness is less than or equal to `shelfWidth`, then build another level of the shelf of the bookcase so that the total height of the bookcase has increased by the maximum height of the books we just put down. We repeat this process until there are no more books to place.

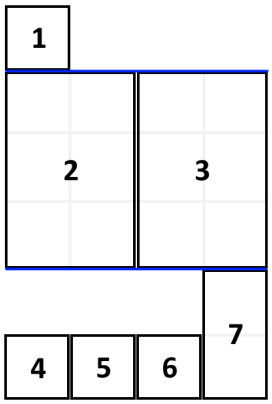
Note that at each step of the above process, the order of the books we place is the same order as the given sequence of books.

- For example, if we have an ordered list of `5` books, we might place the first and second book onto the first shelf, the third book on the second shelf, and the fourth and fifth book on the last shelf.

Return *the minimum possible height that the total bookshelf can be after placing shelves in this manner*.

Example 1:

```
Input: books = [[1,1],[2,3],[2,3],[1,1],
[1,1],[1,1],[1,2]], shelfWidth = 4
Output: 6
Explanation:
The sum of the heights of the 3 shelves
is 1 + 3 + 2 = 6.
Notice that book number 2 does not have
to be on the first shelf.
```



Example 2:

```
Input: books = [[1,3],[2,4],[3,2]], shelfWidth = 6
Output: 4
```

Constraints:

- `1 <= books.length <= 1000`
- `1 <= thicknessi <= shelfWidth <= 1000`
- `1 <= heighti <= 1000`

1105. Filling Bookcase Shelves

```
/*
    DFS+DP memoization
    Time complexity: O(n^2)
    Extra space complexity: O(2n)
*/
class Solution {
    public:
        int minHeightShelves(std::vector<std::vector<int>>& books,
int shelfWidth) {
            int n=books.size();
            std::vector<int> memo(n,-1);
            auto solve=[&](int i,auto& self)->int{
                if(i==n) return 0;
                if(memo[i]!=-1) return memo[i];
                int shelf_w=shelfWidth;
                int max_height=INT_MIN;
                int ans=INT_MAX;
                for(int j=i;j<n;++j){
                    int w=books[j][0];
                    int h=books[j][1];
                    if (shelf_w<w) break;
                    shelf_w-=w;
                    max_height=std::max(max_height,h);
                    ans=std::min(ans,self(j+1,self)+max_height);
                }
                return memo[i]=ans;
            };
            return solve(0,solve);
        }
};
```

1105. Filling Bookcase Shelves

```
/*
    DP
    Time complexity: O(n^2)
    Extra space complexity: O(n)
*/
class Solution {
    public:
        int minHeightShelves(std::vector<std::vector<int>>& books,
int shelfWidth) {
            int n=books.size();
            std::vector<long long> dp(n+1,INT_MAX);
            dp[0]=0;
            for(int i=1;i<=n;++i){
                int shelf_w=shelfWidth;
                int max_height=INT_MIN;
                for(int j=i;j>0;--j){
                    int w=books[j-1][0];
                    int h=books[j-1][1];
                    shelf_w-=w;
                    if (shelf_w<0) break;
                    max_height=std::max(max_height,h);
                    dp[i]=std::min(dp[i],dp[j-1]+max_height);
                }
            }
            return dp[n];
        }
};
```