Comp3121 assignment 2 Question 1
Chen En (z5335039)

Question 1

[30 marks] Antoni is playing a game with some blocks. He starts with a line of n stacks of blocks with heights $h_1, \ldots, h_n$, and in a single move is allowed to split any stack into two (thus increasing the number of stacks by 1).

Your goal is the find the minimum number of moves required so that the resulting stacks are in non-decreasing order.

1.1 **[7 marks]** Show that it's always possible to split some number of split stacks so that they are in non-decreasing order.

1.2 **[23 marks]** Write an algorithm that finds the minimum number of moves required such that the resulting stacks are in non-decreasing order. Your algorithm should run in $O(n)$ time.

1.1
The height of every stack must be a positive integer. That is, we can always split all the stack into the height of 1. If $h_n =$ k, we can split it into k *1 stacks.

1.2
Our goal is to find the minimum number of moves required to form a non-decreasing order of stacks, and splitting the height of stacks as large as possible. As a result, we apply greedy method to this problem.

As non-decreasing order, the last stack should have the maximum height compared to other n-1 stacks. We traverse the line of n stacks to compare the height.
Traverse backward from the last stack till reach the first stack, we check the height of stacks in pairs. We set a variable M = 0 to track the times we split the stacks.

Case 1: If h[i-1] height is less than stack[i] (h[i-1] <= h[i]), which means we have checked the pair are in non-decreasing order, so we check the next pair.
Case 2: If h[i-1] height is greater than h[i], we calculate how many stacks K we are going to split stack[i-1].

Case 2-1: When h[i] is the factor of h[i-1] (h[i-1] mod h[i] == 0), we split h[i-1] into K * h[i] where K = h[i-1]/h[i] and increase M by K-1, which (h[i-1] / h [i]) - 1 is the number of moves for splitting. We let the new value of h[i-1] be h[i].

Case 2-2: Otherwise, h[i] = h[i+1] / (⌊ (h[i-1] / h [i])⌋+1). M increase by ⌊ (h[i-1] / h [i]) ⌋. The following is the explanation:
Suppose we are going to split h[i-1] to K smaller stacks h[i-1][1], h[i-1][2] … h[i-1][K].
After splitting, the tallest stack among the K stacks i.e. h[i-1][K] should be no taller than h[i].
That is h[i] >= h[i-1][K] >= h[i-1] / K (the tallest one must be taller than the average of the K stacks)
We rewrite the inequality above to K >= h[i-1] / h[i].
Our goal is to find the minimum number of moves so we take the minimum value of K which is h[i-1] / h[i]. To be more precisely, K can only be integer, so we will have K = ⌊ (h[i-1] / h [i])⌋ + 1. To make the height of K stacks more evenly, we need to split h[i-1] into some stacks of height ⌊ h[i-1] / K ⌋ and the others of height ⌈h[i-1] / K⌉. As a result, the new value of h[i-1] i.e. the shortest one will be ⌊ h[i-1] / K ⌋.

Repeat the above cases until h[0] was splitted, we return the M which is the minimum move that we made. The greedy strategy provides an optimal solution. Note that when splitting each stack we always take the smallest K as we can. Therefore, after splitting all the stacks we can take the minimum moves at the end.

Time complexity
Since iterating through n-1 stacks and the time complexity for calculation takes O(1), the time complexity for the operation is O(n).