Question 1

For any list of integers, summarise the boundaries of any continuous series (the start and end of the continuous series).

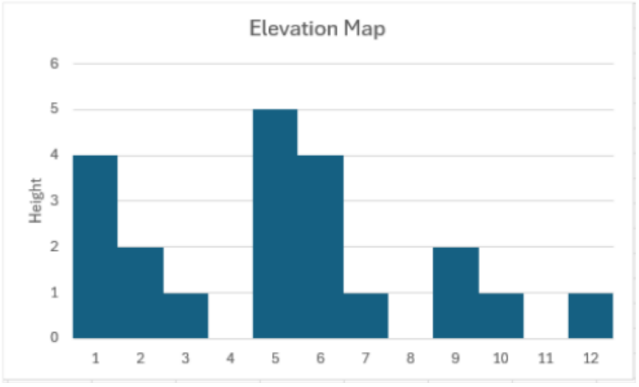
Example: Input: [1,3,4,5,7]
Output: [[1,1],[3,5],[7,7]]

Notes:

1. **First thoughts**: Since the output is a list of lists, it is a good idea to create each sublist and append it to the main one. You can think of this as an $n \times 2$ matrix.
2. **Crux**: I thought, at some point, I have to test if the next index is not strictly one more than the current index.
3. If the next *num* is not strictly one greater than the current *num*, then the current *num* is the ending boundary. The next *num* is also the starting boundary for the next *current_boundary*.
4. **Out of index errors**: Faced many out-of-index errors, but realized you can end the last *current_boundary* with the last element.

Question 2

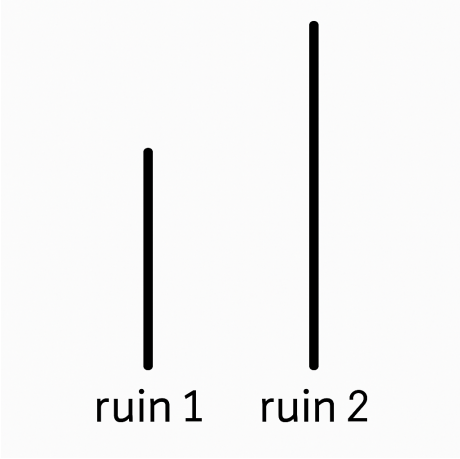
Given an elevation map of ruins represented in the form of an array of positive numbers, where the width of each bar is 1. How much sand would get trapped in the ruins in a sandstorm.



Example: Input: [4,2,1,0,5,4,1,0,2,1,0,1]
Output: 13

Notes:

1. **First thoughts**: I drew many diagrams on paper. It felt like I was spreading out from zero points, but I realized it would be way too inefficient.
2. I realized that the maximum sand blocks between two ruins is limited by the height of the first ruin. Any excess sand would overflow.
   How do I identify these boundary points (poles) in my input?



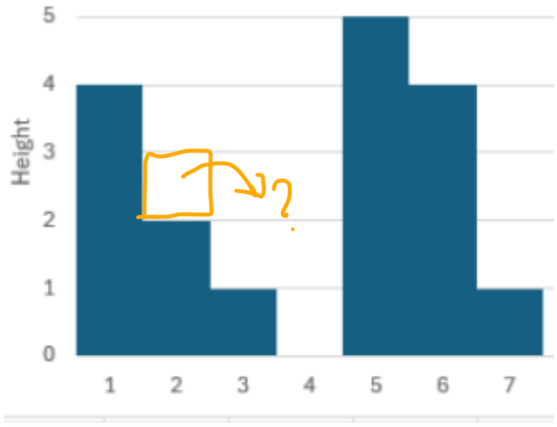
3. **Space Ineficient Algorithm**:

   (a) Find the maximum left height and maximum right height at each index.

   (b) The minimum of these maximums will give you a common depth where sand will be stored.

   (c) Subtract the number of ruin blocks at that index from the minimum of the two maximums.

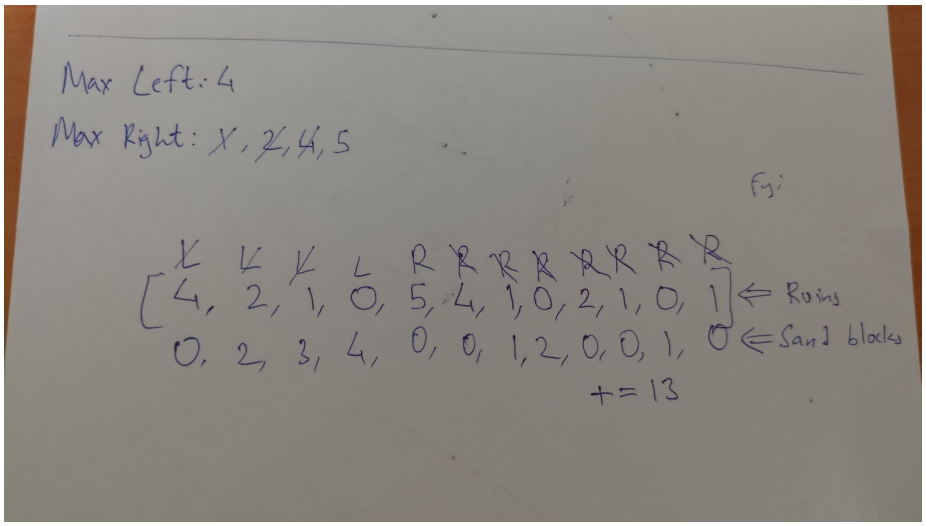
   (d) What if there is still space left? Sand will overflow.



4. **Sample Algorithm Run**: I ran a sample algorithm as shown in the comments:

```
# Q: how do you find max left height and max right height for each index?
# Input/ruin bloc                        [4, 2, 1, 0, 5, 4, 1, 0, 2, 1, 0, 1]
# Max left                               [0, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5]
# Max right                              [5, 5, 5, 5 ,4 ,2 ,2 ,2 ,1 ,1 ,1 ,0]
# Min(Max left, Max right)               [0, 4, 4, 4, 4, 2, 2, 2, 1, 1, 1, 0] => max sand height
# Min(Max left, Max right) - ruin blocks  [-4, 2, 3, 4,-1,-2, 1, 2,-1, 0, 1,-1]
# sum the positive answers                = 13 water blocks
```

5. **New algorithm:**


Max Left: 4
Max Right: X, X, 4, 5
Fy:
L L L L R R R R R R R
[4, 2, 1, 0, 5, 4, 1, 0, 2, 1, 0, 1] ⇐ Ruins
0, 2, 3, 4, 0, 0, 1, 2, 0, 0, 1, 0 ⇐ Sand blocks
+= 13

Question 3

By using the dataset given below, create a dataset containing all the common substrings between String_1 and String_2 for each ID and use it to identify the longest common substring. (Please show the intermediate table containing all the common substrings).

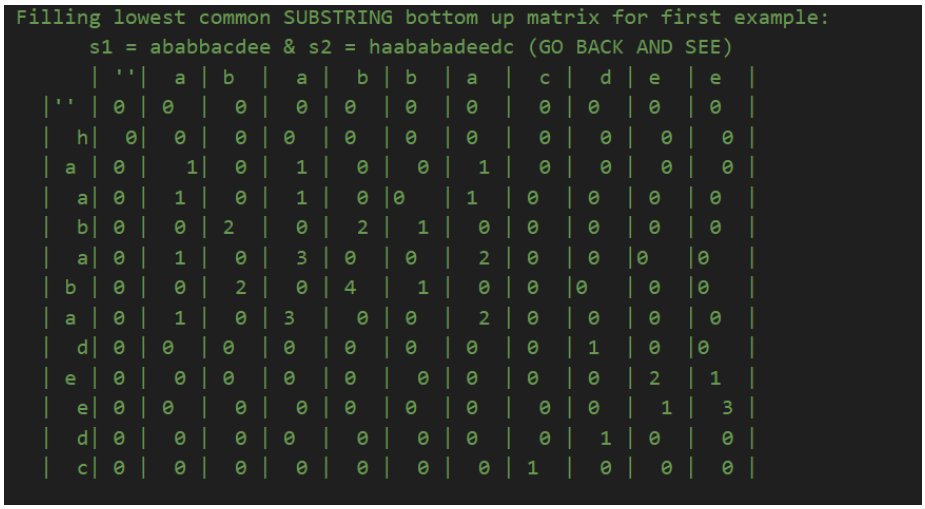
ID	String 1	String 2
1	ababbacdee	haababadeedc
2	Thisisadocumentcontainingpatienthistory	Theletteringinthisstoryisquiteunique
3	abcdefgxyz123	xyz789abcdef
4	The adventurous cat explored the mysterious cave.	A curious cat ventured into the dark cave for exploration.
5	Sunflowers bloomed in the radiant sunlight.	Radiant sunlight illuminated the field of blooming sunflowers.
6	Gentle waves lapped against the sandy shore.	The shore echoed with the soothing sounds of lapping waves.

Present the final output in the following form:

ID	Longest_common_substring
1	abab
2	...
3	...
4	...
5	...
6	...

Notes:

- **First thoughts**: Brute force approach. $\mathcal{O}(n^3, m)$
- **Similar problem from university**: Longest common subsequnce. Explain its cases, with an example, and bottom-up approach.
- **Modify LCS bottom-up to get Longest Common Substring**: Show that the second case is not the same, with an example.
- **Bottom-up on first example**: Explanation or steps for the botto-up approach on the first example
- **Discuss assumptions.**


Filling lowest common SUBSTRING bottom up matrix for first example:
s1 = ababbacdee & s2 = haababadeedc (GO BACK AND SEE)
'' h a a b a b a c d e e
'' 0 0 0 0 0 0 0 0 0 0 0
h 0 0 0 0 0 0 0 0 0 0 0
a 0 1 0 1 0 0 1 0 0 0 0
a 0 1 0 1 0 0 1 0 0 0 0
b 0 0 2 0 2 1 0 0 0 0 0
a 0 1 0 3 0 0 2 0 0 0 0
b 0 0 2 0 4 1 0 0 0 0 0
a 0 1 0 3 0 0 2 0 0 0 0
d 0 0 0 0 0 0 0 0 1 0 0
e 0 0 0 0 0 0 0 0 0 2 1
e 0 0 0 0 0 0 0 0 0 1 3
d 0 0 0 0 0 0 0 0 1 0 0
c 0 0 0 0 0 0 0 1 0 0 0