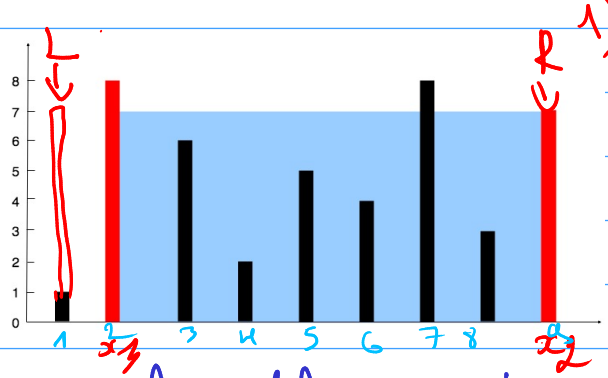


1) problem statement

area = width × min(h1, h2)

area = (x2 − x1) × min(h2, h1)

2) brute force.

3) what makes the area big? wide and tall

we want max area.

what makes the area big: (wide, tall) ×

max wide :: L → 1, R → 9 ⇒ the best possible wide

9 − 1 = 8

4) we have to shrink the width by one. NO choice

L or R must move inward.

shrink width ⇒ ↓ area ↑ height ↑

Currently: L = 1, R = 9

(short) (tall)

height is stuck at 1

bottle neck

Par elimination choice L or R inward.

move R inward: (1, 8) still stuck at (1)

↳ guaranteed move ⇒ no way to get better.

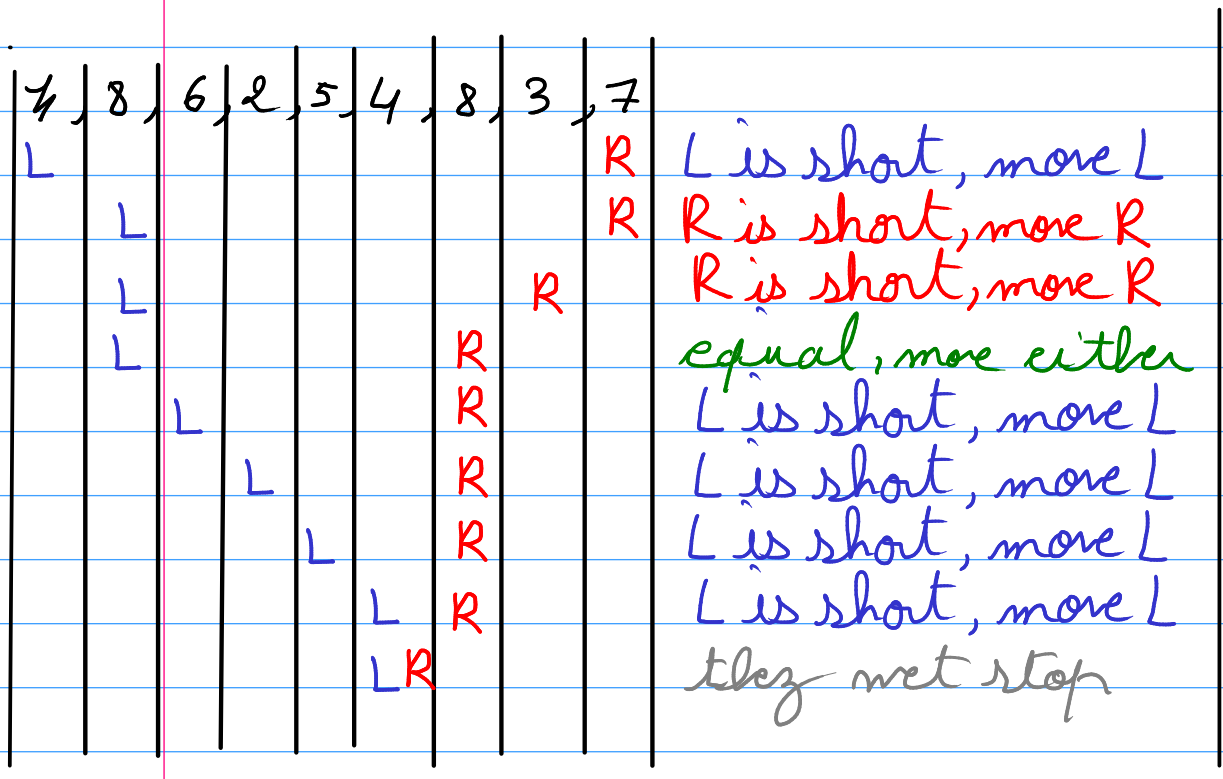
move L inward (8, 7)

↳ width ↓ maybe we find a taller line than 1

↳ maybe better

we move it is the bottleneck, we get rid of it

5) Just keep doing that. Always move the short one



6) Why is this safe? Why we don't miss good pairs.

when we throw the short pointer: we're skipping all the pairs where that short line is one side, with a smaller width.

same short height, less width
↳ less area nothing good