Min size subarray

Let's go very slowly and clarify what "top" means first.

We are using a stack.

- The stack stores indexes of bars (not heights).
- The top means the last pushed index in the stack (the most recent one).

For example:

```ini
heights = [10,20,30]
stack = [0,1]
```

- `stack[-1] = 1` is the top index
- `heights[stack[-1]] = heights[1] = 20` is the top bar height

So when I say top(50) it means:

- Look at `stack[-1]`, get that index → say it's 4
- Then `heights[4] = 50` is the height of the top bar.

Now let's repeat the same example step by step with this clarified.

---

heights = [10,20,30,40,50,0]

We'll process i = 0 → 5.

---

i = 0 → height = 10

- Stack is empty → push this index
- Stack = `[0]`

Now the top = 0 → `heights[top] = 10`

---

i = 1 → height = 20

- Compare `heights[i]=20` with `heights[top]=10`

- 20 >= 10 → taller, so just push

Stack = `[0,1]`
Now top = 1 → `heights[1] = 20`

---

i = 2 → height = 30

- Compare `heights[i]=30` with `heights[top]=20`
- 30 >= 20 → taller → push

Stack = `[0,1,2]`
Now top = 2 → `heights[2] = 30`

---

i = 3 → height = 40

- Compare with `heights[top]=30`
- 40 >= 30 → push

Stack = `[0,1,2,3]`
Now top = 3 → `heights[3] = 40`

---

i = 4 → height = 50

- Compare with `heights[top]=40`
- 50 >= 40 → push

Stack = `[0,1,2,3,4]`
Now top = 4 → `heights[4] = 50`

---

i = 5 → height = 0

Now 0 < heights[top]=50 → so we start popping.

First pop

- Pop index `4`
- height = heights[4] = 50
- Now top becomes `3`

- Width = `i - top - 1 = 5 - 3 - 1 = 1`
- Area = 50 × 1 = 50

Second pop

- Pop index `3`
- height = heights[3] = 40
- Now top becomes `2`
- Width = `i - top - 1 = 5 - 2 - 1 = 2`
- Area = 40 × 2 = 80

Third pop

- Pop index `2`
- height = heights[2] = 30
- Now top becomes `1`
- Width = `i - top - 1 = 5 - 1 - 1 = 3`
- Area = 30 × 3 = 90

Fourth pop

- Pop index `1`
- height = heights[1] = 20
- Now top becomes `0`
- Width = `5 - 0 - 1 = 4`
- Area = 20 × 4 = 80

Fifth pop

- Pop index `0`
- height = heights[0] = 10
- Stack is now empty → width = `i = 5`
- Area = 10 × 5 = 50

Finally push `i=5`
Stack = `[5]`

MAX AREA FOUND = 90

Key idea about top:

- top is always `stack[-1]`, the most recent bar that hasn't been closed yet.
- If the current bar is shorter than top, it closes the rectangle for that top bar.
- While popping:
  - height = popped bar height
  - width = current index `i` - new top after popping - 1

Would you like me to:

1. Draw a simple diagram showing bars and stack visually after each step?
2. Or use an even simpler example `[2,1,2]` so the width idea is clearer?