Solution 18: Time Complexity

1) Fill in the blanks (meaning)

- Time complexity describes how the amount of work grows when the input size **grows**.
- **O(1)** means the program does about the **same** number of steps no matter what `n` is.
- We usually report the **worst** case.

2) What is the input size `n`?

1. `numbers = [2, 5, 1, 4]` → `n = len(numbers)` (here, `n = 4`)
2. `s = "ABCDE"` → `n = len(s)` (here, `n = 5`)

3) First item (no loop)

Time complexity: **O(1)**
Reason: One list access and one print, no loops.

4) Swapping numbers

Time complexity: **O(1)**
Reason: A fixed number of assignments (no loop).

5) Print every item (one loop)

Time complexity: **O(n)**
Reason: One loop runs once per item.

6) Two loops in a row (still linear)

Time complexity: **O(n)**
Reason: `n` prints in the first loop + `n` prints in the second loop → `2n` → still **O(n)**.

---

7) Pairwise print (nested loops)

Time complexity: **O(n^2)**
Reason: Outer loop runs `n` times, inner loop runs `n` times each → `n × n = n²`.

---

8) Inner loop runs a constant number of times

Time complexity: **O(n)**
Reason: Outer loop runs `n` times; inner loop runs 5 times (constant) → `5n` → **O(n)**.

---

9) Triangle nested loops

Time complexity: **O(n^2)**
Reason: Total prints are `0 + 1 + 2 + ... + (n-1)` which grows like `n²`.

---

10) Coefficients don’t matter in Big-O

Time complexity: **O(n)**
Reason: `3n + 100` is a linear function of `n`.

---

11) Search with `break`

- Best case time complexity: **O(1)** (target is the first item; break immediately)
- Worst case time complexity: **O(n)** (target is last item or not found)

---

12) Two-pointer palindrome check (concept review)

Worst case time complexity: **O(n)**
Reason: Each loop step compares two characters and moves pointers inward; at most about `n/2` iterations.

---

13) How many prints? (n = 4)

Number of `print` calls: **4**

14) How many prints? (n = 3, nested loops)

Number of `print` calls: **9** (`3 × 3`)

15) Unique pairs only (i < j)

1. Time complexity: **O(n^2)**
   Reason: Still two nested loops; the inner loop is smaller but total work grows like `n²` .
2. If `n = 4` , how many pairs are printed? **6**
   Pairs count is `4 choose 2 = 6` : `(0,1), (0,2), (0,3), (1,2), (1,3), (2,3)` .