Quiz 18-25: Time Complexity, Set, and Dictionary

1) Choose the best container (synergy)

1. **set — check membership (`"Ben" in guests`) — O(1)**
2. **dict — look up value (`phones["Amy"]`) — O(1)**
3. **list — keep order + print all (`for score in scores:`) — O(n)**

2) Fill in the blanks (set vs list membership)

- `x in my_list` is usually **O(n)**.
- `x in my_set` is usually **O(1)** (average).
- `x in my_dict` checks only the dictionary's **keys**.

3) What can be a set element / dict key?

Item	Can be in a `set`?	Can be a `dict` key?
`7`	YES	YES
`"cat"`	YES	YES
`(1, 2)`	YES	YES
`[1, 2]`	NO	NO

Rule: A set element / dict key must be **hashable** (usually **immutable**).

4) Common keys between dictionaries

```
a = {"Amy": 3, "Ben": 5, "Chloe": 1}
b = {"Ben": 2, "Drew": 9}

s1 = set(a.keys())
s2 = set(b.keys())
s = s1 & s2
print(s)
```

(That makes `common == {"Ben"}`.)

---

5) Fast “seen before?” logic

Option B (a `set`) is better because `word in words_set` is **O(1)** on average, while `word in words_list` is **O(n)**.

---

6) Time complexity of a “build then look up” plan

- Step 1 (build dict from `n` pairs): **O(n)**
- Step 2 (look up one key): **O(1)** (average)

---

7) Unique vs counts (set + dict together)

One correct output is:

```
{'A', 'B', 'C'}
2
3
```

(The set order may vary.)

---

8) Common players (dict keys → set intersection)

```
{'Ben'}
1
```

(Set order may vary, but there is only one item.)

---

9) Updating scores (dict overwrite)

Final dictionary (order may vary):

```
{'Amy': 2, 'Ben': 7, 'Chloe': 1}
7
```

10) “Any overlap?” two ways

Answer: **O(n²)**

Explanation:

- The loop runs up to `n` times.
- `x in b` is a list membership check. Its time complexity is O(n).
- Thus the overall time complexity is O(n²).

11) “Any overlap?” using a set (faster idea)

Answer: **O(n)**

Reason:

- Building `b_set` has **O(n)** time complexity.
- Then we do `n` membership checks in a set. Each costs **O(1)** time complexity.
- Totally **O(n)** time complexity.

12) One bug: set method name

Corrected lines:

```
colors.add("red")
colors.add("blue")
```

13) Build a frequency dictionary (fill in blanks)

```
counts[x] = counts.get(x, 0) + 1
```

14) Find duplicates (dict + set together)

One correct solution:

```
for f in fruits:
    counts[f] = counts.get(f, 0) + 1
    if counts[f] >= 2:
        dupes.add(f)
```

Expected results:

- `counts` is `{'apple': 3, 'banana': 2, 'orange': 1}` (order may vary)
- `dupes` is `{'apple', 'banana'}` (order may vary)

15) Set operations from lists (union + intersection)

One correct solution:

```
math_set = set(math)
music_set = set(music)

print(math_set | music_set)   # union
print(math_set & music_set)   # intersection
```

16) First repeated word (fast check with a set)

One correct solution:

```
seen = set()

for w in words:
    if w in seen:
        print(w)
        break
    seen.add(w)
```

It prints `cat`.

17) Mini inverted index (dict of sets) — fill in blanks

Blanks:

- `set()`
- `add`

So the completed part is:

```
if w not in index:
    index[w] = set()
index[w].add(i)
```

18) Pick the faster plan (big-O + explanation)

1) Plan A: **O(n²)**
2) Plan B: **O(n)**
3) Because Plan A does up to `n` list membership checks and each check can take up to `n` steps, while Plan B scans the list once to build a set.