Explanation

When a character is not part of the LCS, we add it to the supersequence

The Core Principle

Every character from both strings must appear in the supersequence.

- **Characters that ARE part of the LCS**: Add them once (they're shared)
- **Characters that are NOT part of the LCS**: Add them separately (they're unique to one string)

Why This Works

The **supersequence** must contain both strings as subsequences, which means:

1. **Every character** from `str1` must appear in the supersequence
2. **Every character** from `str2` must appear in the supersequence
3. **Shared characters** (LCS) only need to appear once
4. **Non-shared characters** must be added individually

Simple Example

```
str1 = "abc"
str2 = "ac"
LCS = "ac"  (length 2)
```

Character analysis:

- 'a': Part of LCS ✓ (shared)
- 'b': NOT part of LCS × (unique to str1)
- 'c': Part of LCS ✓ (shared)

Supersequence construction:

- Add 'a' once (shared)
- Add 'b' (unique to str1)
- Add 'c' once (shared)
- Result: "abc"

The Backtracking Logic

```
if str1[i-1] == str2[j-1]:
# Character is part of LCS - add once
    add_to_result(str1[i-1])
elif dp[i-1][j] > dp[i][j-1]:
# str1[i-1] is NOT part of LCS - must add it
    add_to_result(str1[i-1])
else:
# str2[j-1] is NOT part of LCS - must add it
    add_to_result(str2[j-1])
```

So yes, your understanding is perfect: **when a character is not part of the LCS, we add it to the supersequence because it's unique to one string and must be preserved.**
