Explanation

Why it works:

1. **Initial check**: `n != m+1` ensures `word_2` has exactly one more character than `word_1`
2. **Your example**: `abc` vs `abcd`
   - `i` will exhaust (reach `m=3`)
   - `j` will be at position 3 in `abcd`
   - Since we know `word_2` has exactly one extra character, and we've matched all of `word_1`, that remaining character must be the valid addition
3. **Edge case handling**: When the loop ends:
   - If `i < m`, it means we couldn't match all characters of `word_1` → impossible
   - But since `n = m+1` and we process both pointers, if we exit normally, `i` will always equal `m`

Your Complexity Analysis is Also Correct

- **Time**: $O(n^2)$ - the string comparison is O(L) but since L is typically small and constant relative to n, it's reasonable to consider it $O(n^2)$
- **Space**: O(n) for the dp array