difference between subarry and subsequence

Here is the **clean CP/DSA-style difference** between **subarray** and **subsequence**, exactly how it's explained in interviews:

🔥 SUBARRAY vs SUBSEQUENCE (CP/DSA Style)

✅ SUBARRAY

Definition

A **subarray** is a *contiguous* part of the array.

Key Points

- **Must be continuous** (no gaps allowed)
- Comes from **original array directly**
- Defined by **l to r (range)**
- Count is **$O(n^2)$** possible subarrays

Example

Array:

`[3, 5, 1, 4]`

Valid subarrays:

`[3]`, `[3,5]`, `[5,1]`, `[1,4]`, `[3,5,1]`, `[5,1,4]`, `[3,5,1,4]`

Invalid:

`[3,1]` × (not contiguous)

Where used?

- Sliding window
- Kadane's algorithm
- Prefix/suffix sums
- Range queries

✅ SUBSEQUENCE

Definition

A **subsequence** is formed by deleting ANY elements *without changing order.*

Key Points

- **Not required to be contiguous**
- Order must remain same
- Can skip elements
- Count is **2^n** possible subsequences

Example

Array:

`[3, 5, 1, 4]`

Valid subsequences:

`[3]`, `[5]`, `[3,1]`, `[3,5,4]`, `[5,4]`, `[3,1,4]`, `[3,5,1,4]`

Invalid:

`[5,3]` × (order changes)

Where used?

- DP problems
- LCS (longest common subsequence)
- LIS (longest increasing subsequence)
- Subsequence counting

🚀 ULTRA-SHORT VERSION

Subarray = Contiguous

Subsequence = Ordered, not necessarily contiguous

📌 Quick Code Examples

Subarray Generation (Contiguous)

cpp Copy code

```
for (int l = 0; l < n; l++) {
    for (int r = l; r < n; r++) {
        // subarray is from l to r
    }
}
```

Subsequence Generation (Not Contiguous)

cpp Copy code

```cpp
void gen(int i, vector<int>& a, vector<int>& cur) {
    if (i == a.size()) {
        // cur is a subsequence
        return;
    }
    gen(i+1, a, cur);         // skip
    cur.push_back(a[i]);      // take
    gen(i+1, a, cur);
    cur.pop_back();
}
```

If you want, I can also explain:
✅ substring vs subsequence
✅ subarray vs subset
✅ when interviewer tries to confuse you with examples
or give real LeetCode problems where these matter.

Just tell me!