**For finding 2-5 , which nodes will be considered and under which condition for range query : 1, 2, 3, 4, 5, 6**

For the segment tree built from \*\*\*\*, the range sum query for indices **2–5** (inclusive of 2, exclusive of 5) will consider specific nodes depending on whether the left or right pointer is odd or even at each step. Here is the step-by-step breakdown:

**Nodes Considered for Range [2, 5)**

**Initialization**

* Start pointers: $ l = 2, r = 5 $
* Adjusted for tree size (size = 8):
  + $ l = 10, r = 13 $

**Query Steps and Conditions**

**Iteration 1**

* **l = 10 (even), r = 13 (odd)**
  + **Left pointer (l):** Even, so no action.
  + **Right pointer (r):** Odd, decrement right and add node:
    - $ r -= 1 \rightarrow r = 12 $
    - Add **tree** (value = 5) to the result.

**Pointer Update**

* $ l = 10 \rightarrow 5 $ ($ l // 2 $)
* $ r = 12 \rightarrow 6 $ ($ r // 2 $)

**Iteration 2**

* **l = 5 (odd), r = 6 (even)**
  + **Left pointer (l):** Odd, add node and increment left:
    - Add **tree** (value = 7) to the result.
    - $ l += 1 \rightarrow l = 6 $
  + **Right pointer (r):** Even, so no action.

**Pointer Update**

* $ l = 6 \rightarrow 3 $
* $ r = 6 \rightarrow 3 $

**Iteration End**

* Loop stops ($ l \geq r $).
* Result is sum of **tree** and **tree** (i.e., 7 + 5 = 12).

**Table: Nodes Used in Range [2, 5)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node | Tree Index | Covers | Value | Condition Used |
| tree | 5 | (3, 4) | 7 | l odd (add, increment l) |
| tree | 12 | (5) | 5 | r odd (decrement r, add) |

**How and Why These Nodes Are Selected**

* **Odd index (l)**: Add node, move left pointer forward. Used for covering the leftmost element or block in range.
* **Odd index (r)**: Decrement, then add node for the rightmost element/block just inside the range.
* **Result**: Only the minimal set of nodes (**tree** and **tree**) that fully cover indices 2, 3, and 4 (**data=3, data=4, data=5**) is chosen efficiently.