**To find minimum of sub-array query (a, b)**

**1) Look-Up Table Technique**

* Array = arr [n]
* To find-minimum value in the range (a, b) where a = query start, b = query end and 0<=a<=b<n
* Construct a 2D lookup table (n \* n)
* Storing index of the least value in the range i to j where, I = [0, n) and j = [i, n)
* Minimum element in given range (a, b) = element corresponding to index in row-a and column-b
* Memory= O(n\*n)
* Processing time= O(n\*n)
* Fetching time= 1 unit (faster and efficient queries)

**2) Square root decomposition**

* Divide array into groups of size, x = sqrt(n)
* Let minimum of each grp= min(grp)
* Minimum element in range (a, b) = minimum of all min(grp) which are part of range (a, b)
* Advantage - Minimizes look-up table to 1D and Reduces time complexity by a factor of sqrt(n)
* Disadvantage- Minimum of groups can be directly compared only if entire range of that group is a part of range (a, b). If not, then start comparison from lower limit i.e., a
* Memory=O(n)
* Processing time=approx. sqrt(n)

**3) Segment tree**

* It’s a binary tree build in bottom-up approach
* Each node stores least value of its successive node
* Consider an array A[N] corresponding Segment Tree ST:

-Root of ST will represent the entire array A [0: N−1]

-Each leaf in the Segment Tree T will represent a single element A[i] where, 0≤i<N

-The internal nodes represent the intervals A[i:j] where 0≤i<j<N.

* Advantage-

-It works even when the range (a, b) doesn't exactly match with range of node unlike square root decomposition

-Segment tree is a better option if there are lot of queries and the values are static

* Disadvantage-

-Becomes complex if values are dynamic or elements need to be added /deleted

* Memory= [st3](https://www.geeksforgeeks.org/wp-content/uploads/st3.png) where, n is number of leaves
* Processing time
* tree construction=O(n)
* for query or update=O(Logn)