# Competitive Programming Java Data Structures (Excluding StringBuilder)

Arrays.sort(arr): Sort array

Arrays.binarySearch(arr, key): Binary search

Arrays.fill(arr, val): Fill array with a value

Arrays.equals(arr1, arr2): Compare two arrays

Arrays.copyOf(arr, newSize): Copy array with new size

Arrays.copyOfRange(arr, from, to): Copy range of array

Arrays.mismatch(arr1, arr2): Find index where arrays differ

Arrays.stream(arr): Convert array to stream

Arrays.asList(arr): Convert array to list

list.add(1): Adds to end

list.addFirst(1): Adds to the start

list.addLast(1): Adds to the end

list.get(index): Get element at index

list.remove(index): Remove element at index

list.removeIf(predicate): Remove elements based on condition

list.contains(1): Check if element exists

list.poll(): Retrieve and remove first element

list.pollFirst(): Retrieve and remove first element

list.pollLast(): Retrieve and remove last element

list.peekFirst(): Retrieve but do not remove first element

list.peekLast(): Retrieve but do not remove last element

list.retainAll(collection): Retain only elements in collection

list.containsAll(collection): Check if list contains all elements of collection

list.removeAll(collection): Remove all elements in the collection

list.clear(): Removes all elements

list.forEach(consumer): Perform action on each element

list.spliterator(): Create spliterator for elements

queue.add(1): Add to queue

queue.offer(1): Add element (returns false if fails)

queue.poll(): Removes and returns head

queue.peek(): Retrieves but does not remove head

queue.isEmpty(): Check if queue is empty

queue.remove(): Removes head element

queue.containsAll(collection): Check if queue contains all elements of collection

queue.retainAll(collection): Retain only elements in collection

queue.removeAll(collection): Remove all elements in the collection

queue.clear(): Remove all elements from queue

queue.forEach(consumer): Perform action on each element

queue.spliterator(): Create spliterator for elements

stack.push(1): Push onto stack

stack.pop(): Pop from stack

stack.peek(): View top of stack

stack.empty(): Check if stack is empty

stack.search(1): Returns position of element

stack.size(): Get size of the stack

stack.contains(1): Check if element exists

stack.forEach(consumer): Perform action on each element

deque.addFirst(1): Add to front

deque.addLast(1): Add to back

deque.pollFirst(): Remove from front

deque.pollLast(): Remove from back

deque.peekFirst(): View front element

deque.peekLast(): View last element

deque.offerFirst(1): Insert element at front

deque.offerLast(1): Insert element at back

deque.retainAll(collection): Retain only elements in collection

deque.containsAll(collection): Check if deque contains all elements of collection

deque.removeAll(collection): Remove all elements in the collection

deque.clear(): Remove all elements

deque.removeIf(predicate): Remove elements based on condition

deque.forEach(consumer): Perform action on each element

deque.spliterator(): Create spliterator for elements

pq.offer(1): Add element

pq.poll(): Remove and return smallest element

pq.peek(): Get smallest element

pq.isEmpty(): Check if priority queue is empty

pq.comparator(): Return comparator used for ordering

pq.size(): Get number of elements in priority queue

pq.containsAll(collection): Check if priority queue contains all elements of collection

pq.retainAll(collection): Retain only elements in collection

pq.removeAll(collection): Remove all elements in the collection

pq.clear(): Remove all elements

pq.forEach(consumer): Perform action on each element

pq.spliterator(): Create spliterator for elements

set.add(1): Add element

set.contains(1): Check if element exists

set.remove(1): Remove element

set.isEmpty(): Check if set is empty

set.size(): Get size of set

set.clear(): Remove all elements from the set

set.containsAll(collection): Check if set contains all elements of collection

set.retainAll(collection): Retain only elements in collection

set.removeAll(collection): Remove all elements in the collection

set.forEach(consumer): Perform action on each element

set.spliterator(): Create spliterator for elements

set.removeIf(predicate): Remove elements based on condition

Collections.disjoint(set1, set2): Check if two sets are disjoint

sset.add(1): Add element

sset.first(): Get smallest element

sset.last(): Get largest element

sset.headSet(1): Get elements less than specified

sset.tailSet(1): Get elements greater than specified

sset.subSet(1, 2): Get range of elements

sset.retainAll(collection): Retain only elements in collection

sset.containsAll(collection): Check if sorted set contains all elements of collection

sset.removeAll(collection): Remove all elements in the collection

sset.clear(): Remove all elements

sset.forEach(consumer): Perform action on each element

sset.spliterator(): Create spliterator for elements

sset.removeIf(predicate): Remove elements based on condition

tset.add(1): Add element

tset.first(): Smallest element

tset.last(): Largest element

tset.ceiling(1): Least element >= specified

tset.floor(1): Greatest element <= specified

tset.pollFirst(): Remove and return smallest element

tset.pollLast(): Remove and return largest element

tset.containsAll(collection): Check if TreeSet contains all elements of collection

tset.retainAll(collection): Retain only elements in collection

tset.removeAll(collection): Remove all elements in the collection

tset.clear(): Remove all elements

tset.forEach(consumer): Perform action on each element

tset.spliterator(): Create spliterator for elements

tset.removeIf(predicate): Remove elements based on condition

map.put(1, 'A'): Insert key-value pair

map.get(1): Get value by key

map.containsKey(1): Check if key exists

map.keySet(): Get all keys

map.values(): Get all values

map.replace(1, 'B'): Replace value for key

map.remove(1): Remove key-value pair

map.clear(): Clear all key-value pairs

map.forEach(consumer): Perform action on each entry

map.compute(key, remappingFunction): Compute new value for key

ht.put(1, 'A'): Insert key-value pair

ht.get(1): Get value by key

ht.containsKey(1): Check if key exists

ht.isEmpty(): Check if hashtable is empty

ht.size(): Get size of hashtable

ht.remove(1): Remove key-value pair

ht.clear(): Clear all key-value pairs

ht.forEach(consumer): Perform action on each entry