

Functions → Non-Mutating Algo

① find function →

iterators of datatype `auto it = find(v.begin(), v.end(), value);`
 $\hookrightarrow O(n)$

- recommended to use `unordered_set` & `unordered_map` find takes $O(1)$
- class specific find function;

② Lower Bound → Returns an iterator having address of element greater than equal to given value in a given sorted range.

(Syntax) `auto it = lower_bound(v.begin(), v.end(), value);`

③ Upper bound → Returns iterator to first greater element in sorted range

④ is_permutation → used to check whether all elements of one containers are permutation of another
is_permutation(v1.begin(), v1.end(), v2.begin(), v2.end())
 $\hookrightarrow n2$

iv) Max & Min element →

auto it = max_element(v.begin(), v.end())
= min_element()

in case of array →

auto it = *max_element(arr, arr+n)

v) Count →

→ count(v.begin(), v.end(), value)

vi) Binary Search →

T/F → binary_search(v.begin(), v.end(), value)

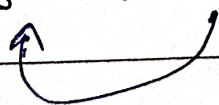
vii) fill → utility function, fills the value in iterator (used for vector, list, deque, etc.)

→ fill(v.begin(), v.end(), value)

(range) (can be altered)

viii) rotate → (forward iterators)

rotate(v.begin(), v.begin() + 2, v.end())



ix) accumulate → res = 0 → add all the elements

accumulate(v.begin(), v.end(), res)
(, minus(0))
minus

rand() → random value
(srand(time(NULL))) → sets different times
Based on → linear congruential generator.

Mutating STL Alg.

i) Sort → `sort(arr.begin(), arr.end())`

ii) make_heap → `make_heap(v.begin(), v.end())`
(by default max heap) (makes heap)

iii) Merge → merge in sorted order
`merge(v1.begin(), v1.end(), v2.begin(), v2.end(), v3.begin())`
(sorted)

iv) reverse → reverse the container
`reverse(arr.begin(), arr.end())`

v) next_permutation →
prints lexicographically next permutation
→ `next_permutation(v.begin(), v.end())`

vi) prev_permutation →
gives prev permutation of
given no.

* $\text{rand}() \rightarrow$ random value
($\text{srand}(\text{time}(\text{NULL}))$) \rightarrow sets different time
Random \rightarrow linear congruential Generator.

Mutating STL Alg.

i) $\text{Sort} \rightarrow \text{sort}(\text{arr.begin}(), \text{arr.end}())$
ii) $\text{make_heap} \rightarrow \text{make_heap}(v.begin(), v.end())$
(by default max heap) (makes heap)

iii) $\text{Merge}() \rightarrow$ merge in sorted order
 $\text{merge}(v1.begin(), v1.end(), v2.begin(), v2.end(), v3.begin())$

(sorted)

iv) $\text{reverse}() \rightarrow$ reverse the container
 $\text{reverse}(\text{arr.begin}(), \text{arr.end}())$

v) $\text{next_permutation} \rightarrow$
prints lexicographically next permutation
 $\rightarrow \text{next_permutation}(v.begin(), v.end())$

vi) $\text{prev_permutation} \rightarrow$
gives prev permutation of
given no.