Functions

→ Non-Mutating Algo

IV)

(i) find function →

iterator of datatype | $\boxed{\text{auto}}$ it = find(v.begin(), v.end, value).
↳ $\Theta(n)$

→ recommended to use unorder set & unordered map find (— takes O(1))

→ class specific find function.

V)

VI) T|F

(ii) 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)

(iii) Upperbound → Returns Iterator to first greater element in sorted range

(iv) is_permutation → used to check whether all elements of one containers are permutation of other

is_permutation (v1.begin, v1.end(), v2.begin)

Lo 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);

$\underbrace{}$ (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(int))
minus)

rand() → random value
srand(time(NULL)) → sets different time
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 lexographically next permutation
→ next_permutation(v.begin(), v.end())

vi) prev_permutation →
gives prev permutation of
given no.

A rand() → random value
(srand(time(NULL))) → sets different time
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) ~~t~~next_permutation →
prints lexographically next permutation
→ next_permutation(v.begin(), v.end())

vi) prev_permutation →
gives prev permutation of
given no.