C++ STL revision

↳ stl data structures

↳ stl on binary search

↳ some tricks on stl

Some info to cp

constraints →

t

n

[ ]

how many ops in 1sec → $\approx 10^8$

$t = 3$

$n_1 \to 2$

| |

$n_2 \to 3$

| | |

$n_3 \to 4$

| | | |

$n_1 + n_2 + n_3 \le 2 \times 10^5$

$$\underbrace{n + n + n}_{t \text{ times}} \le 2 \times 10^5$$

$\boxed{\underline{n} \times \underline{t}} \le \underline{2 \times 10^5}$

$t \to 10^4$ ✓

$n \to 10^5$ ✓

$n \times t \to \underline{2 \times 10^5}$

$O(n) \to$

$O(n \log n) \to \boxed{t \times n} \times \log n$

$2 \times 10^5 \times \log(10^5)$

$5 \times 2 \times 10^5 \times 3$

$\to \underline{6 \times 10^6} < \underline{10^8}$

TLE

$O(n^2) \to t \times n \times n$

$\to 2 \times 10^5 \times 10^5 \to 2 \times 10^{10}$

200 sec ✗

$n \rightarrow 2 \times 10^5$ $\boxed{nxt} \rightarrow 2 \times 10^5$

int → -10^9 to 10^9

long long → -10^{18} to 10^{18}

u long long → 0 to 2×10^{18}

$^nC_r \rightarrow \dfrac{n!}{((n-r)!(r!))}$

$^{10}C_2 \rightarrow \dfrac{10!}{2!8!} \rightarrow \boxed{45}$

$^{100}C_{10}$ → $100!$ → $100 \times 99 \times 98 \ldots 1$

long long overflow

mod arithmatic

endl → slower

"\n" → faster

containers

→ vectors

→ vector<int> vec; → empty vector

vec.push_back(0);

[ ] → [0] → [0, 1] → - - -

vector<int> vec(n, 0);

size

initial value

vector<int> vec(n);

vector<int> vec(n, 0);

vec.size()

emplace_back()

↙

faster

n

↙

0 to n-1

n=5

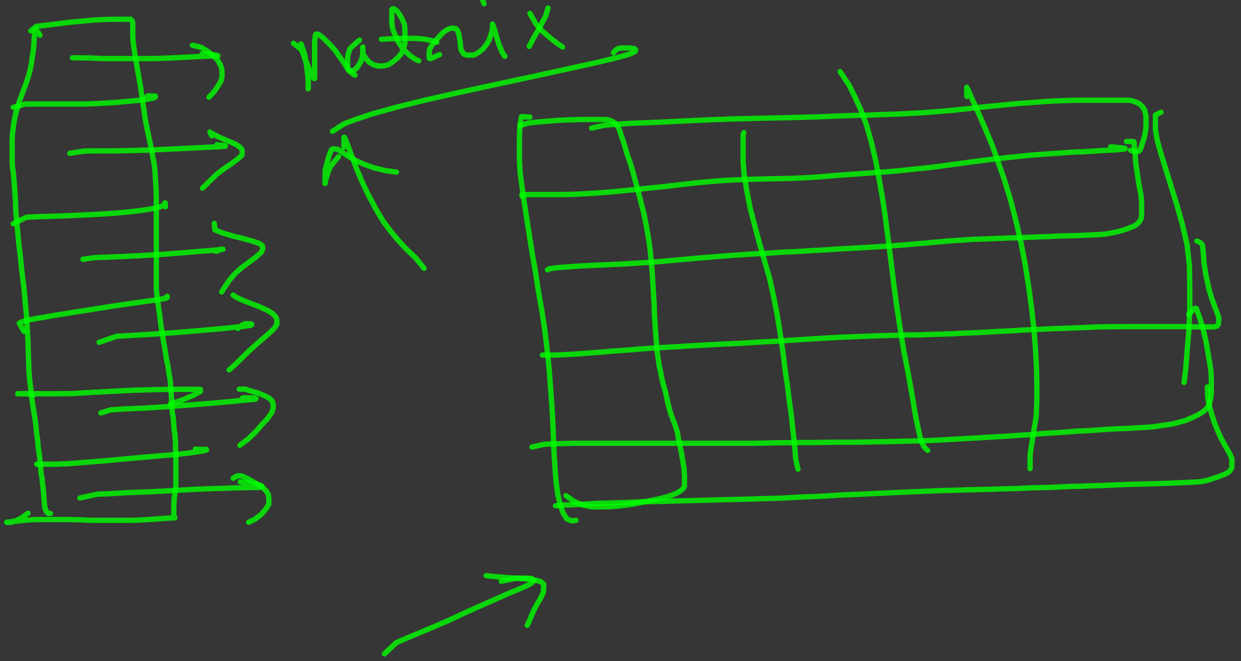
↳

[1, 7, 2, 1, 11]

0 1 2 3 4

2D vectors →

vector<vector<int>>

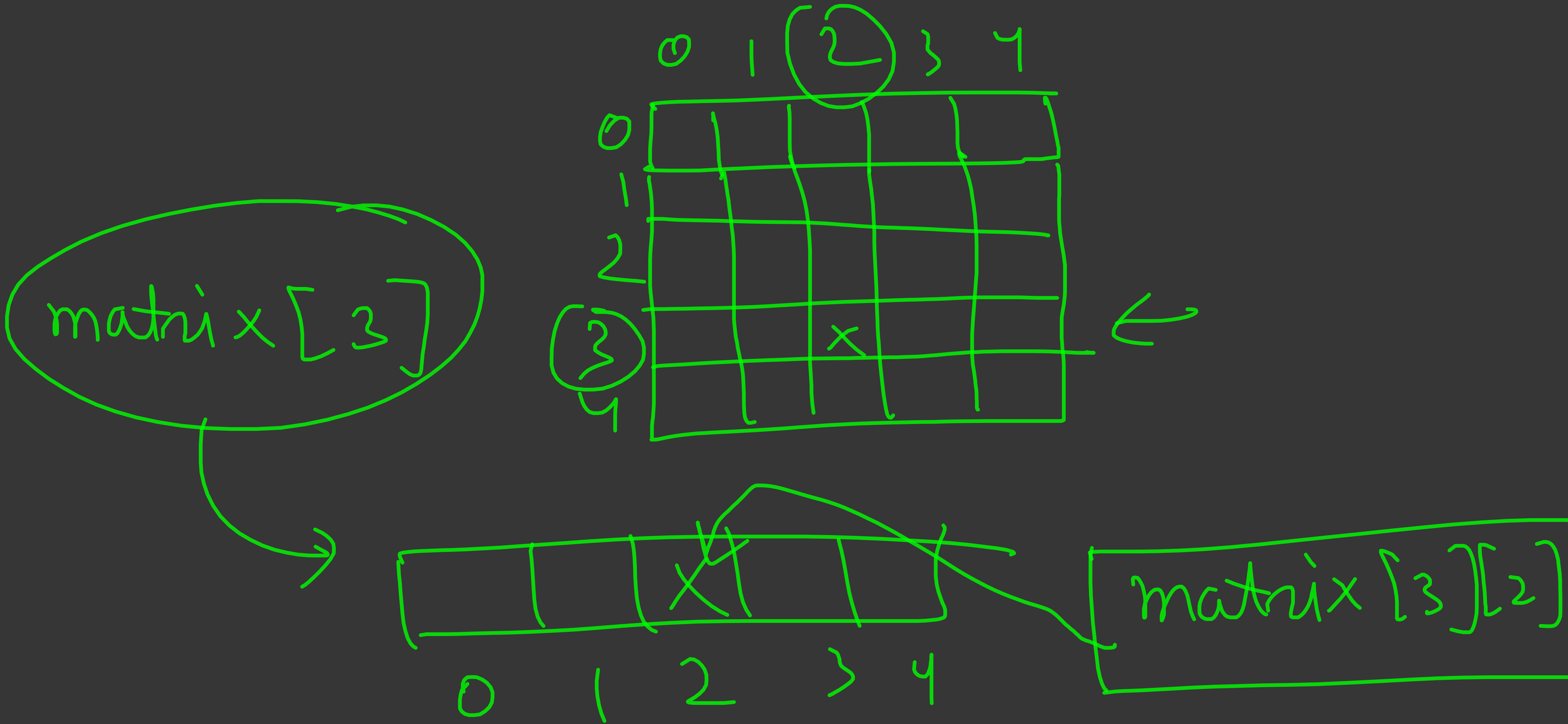


row → 10

col → 15

```
vector <vector<int>> mat(5);
                  (5)

vector<vector<int>(col) > matrix(row);
```

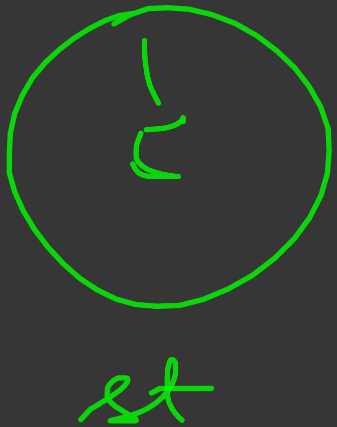


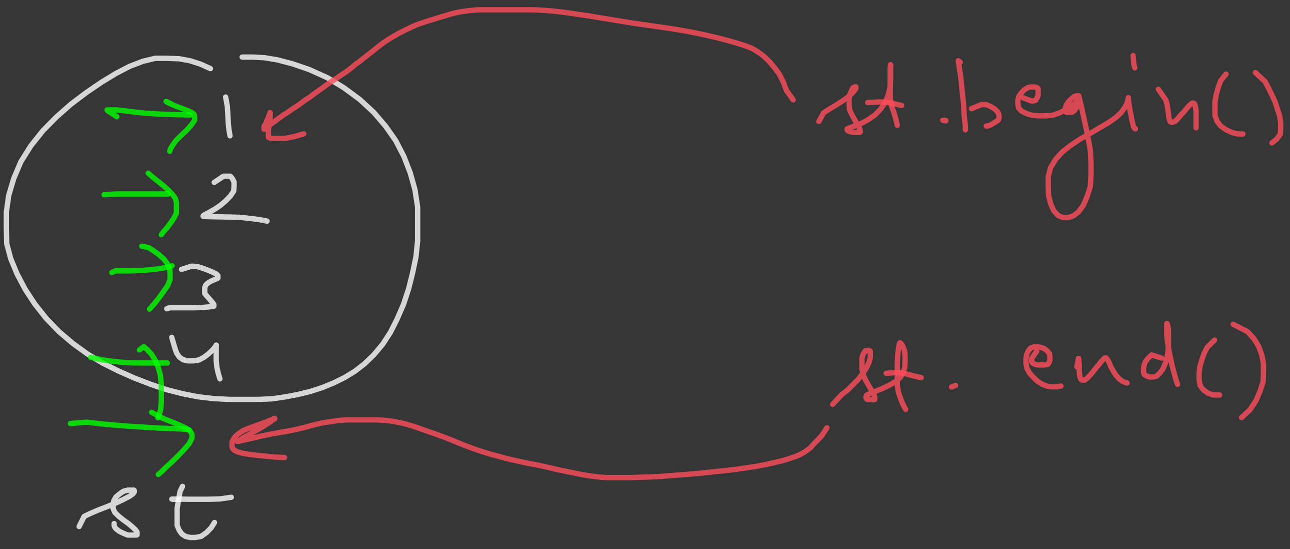
vector<int> dev;

```
dev pb (5),          [5]
dev pb (10);         [5, 10]
dev. pob_back(); →   [5]
```

set → unique elements in sorted order

```
st. insert(5);
st. insert(5);
st. insert(1);
```





```
for (auto it = st.begin(); it != st.
                                    end();
                                 it++){
  cout << *it << endl;
}
```

1 2 3 4

for each

~~for loop~~

```
for (auto it : st){
        cout << it << " ";
}
```

O(log n)

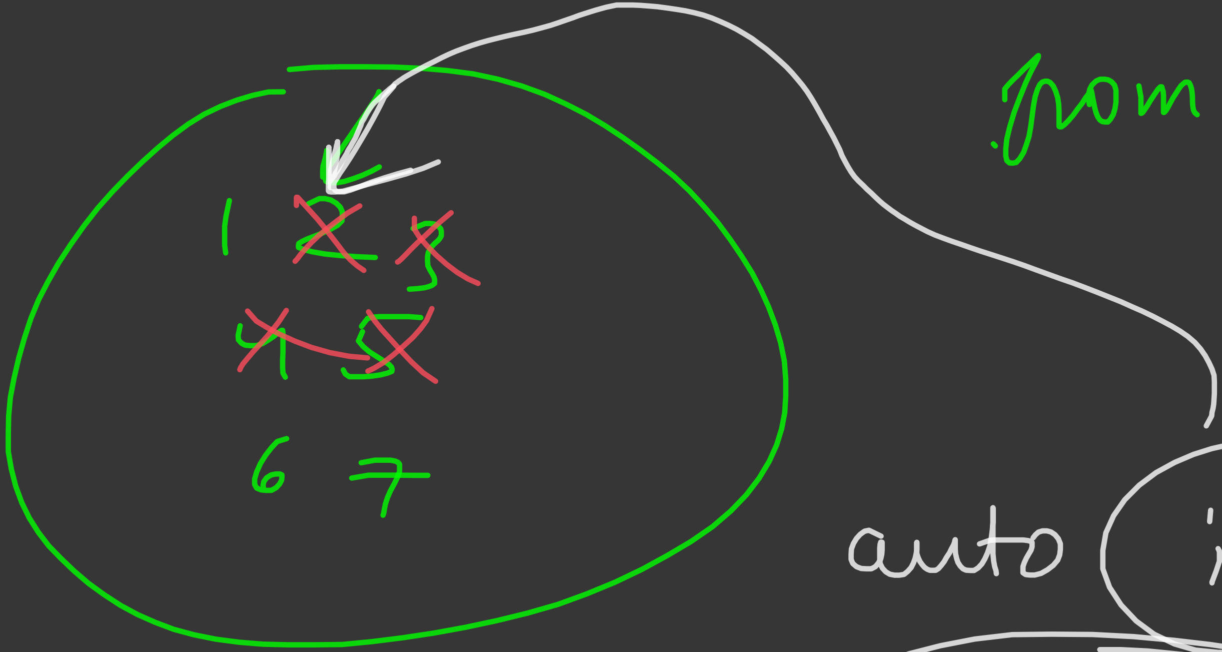
```
st.erase(3);
```

set<int>::iterator

```
auto it = st.begin()
```

insert → log(n)

st.size() → size



from 2 delete 3 more elements.

auto it = st.find(2)

st.end()

nxlogn ← st.erase(it, it+4);

→ if element does not exists, it points to st.end()

unordered set → constant time unique elements any order

unordered_set<int> st;

time → O(1)

→ 10^5 $\log(10^5) \rightarrow 5 \times \log(10)$

→ set

20

multiset

→ sorted order

→ NOT unique

→ [1, 1, 1, 2, 2, 3]

```
multiset<int> st;
```

st.count(1); → 3

1 1 1
2 2 3

→ nlogn

st.clear() → O(n)

erase all 1 → st.erase(1);

erase only one 1 →

st erase (st.find(1))

map → key value pair

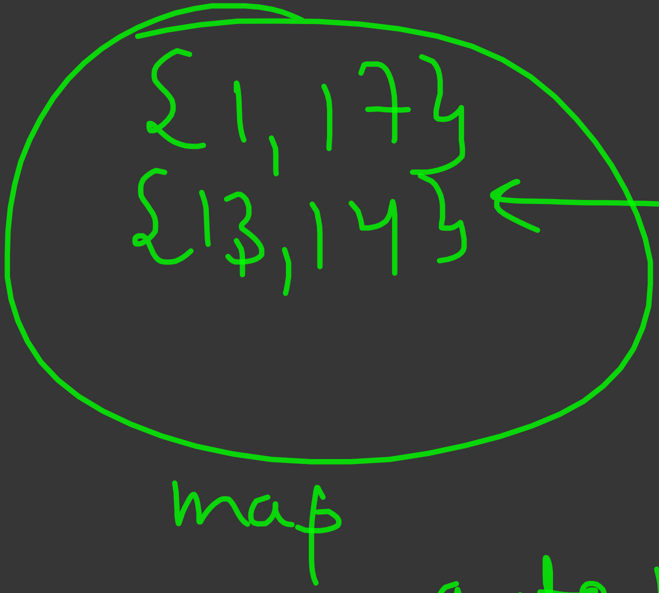
→ unique keys

→ sorted order in keys.

mp<int, int> mp,

mp[13] = 14;

mp[1] = 17



auto it = mp.find(13);

→ {13, 14}

it first ← → it.second.

mp erase (mp find (13));

(*it) first

it → first

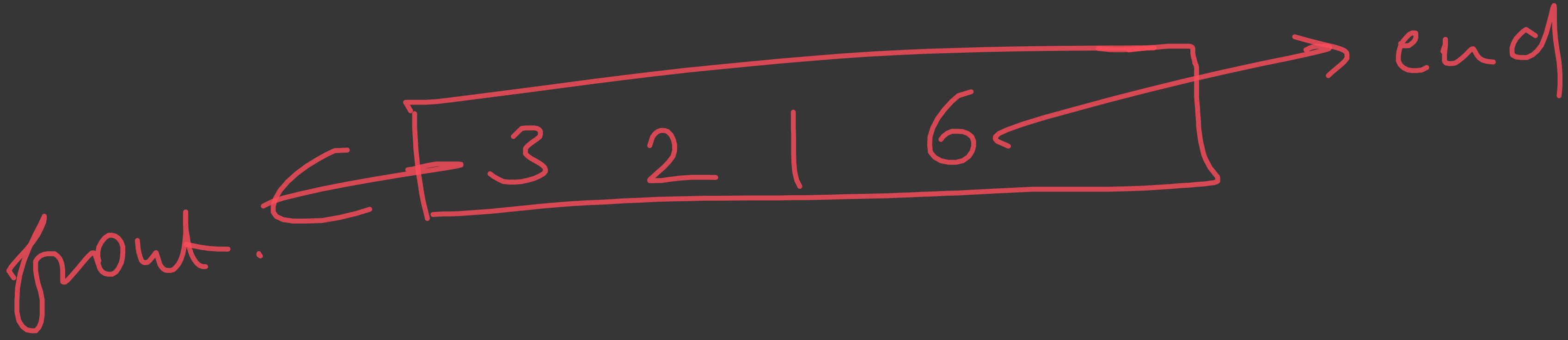
unordered_map → unsorted (any order)

→ O(1)

→ unique keys

multimap → sorted
→ log(n)
→ we can have duplicate

queue → fifo



q.pop()

```
queue <int) q;
q.push(3);
q push(2)
q push(1)
q.push(6)
```

q.front() → 3

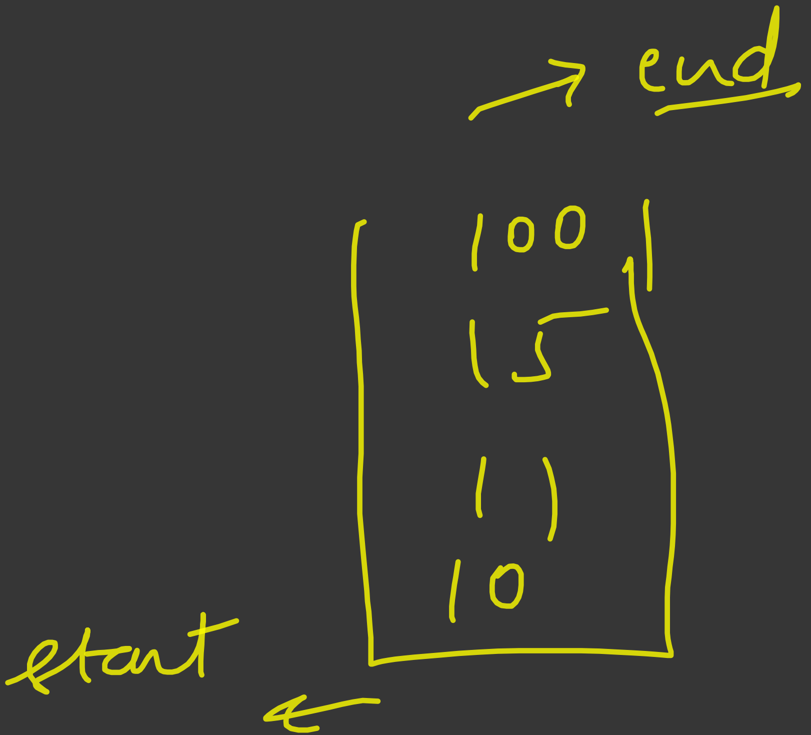
→ → q.empty() → true

→ false

q size()

stack → lifo

stack <int) st;

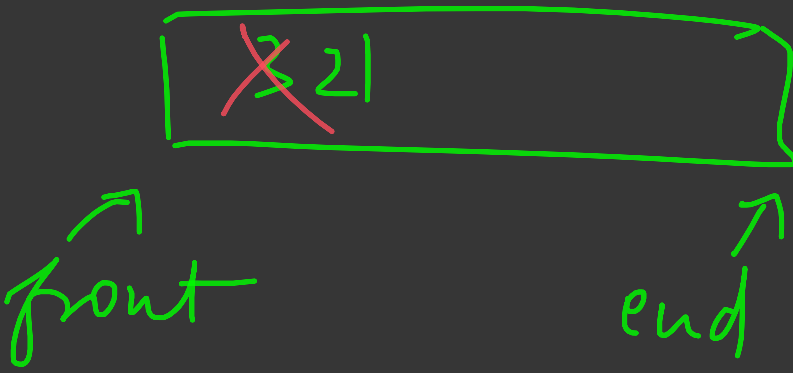


```
st. push(10)
    push(11)
    push(15)
```

```
st.top() → 100
st. pop() →
st. size()
st. empty()
```

priority queue → → max heap

```
priority_queue<int> pq;
  pq push(1),
  pq push(3),
  pq push(2)
```



pq.top() → 3

pq pop() → 3 gone

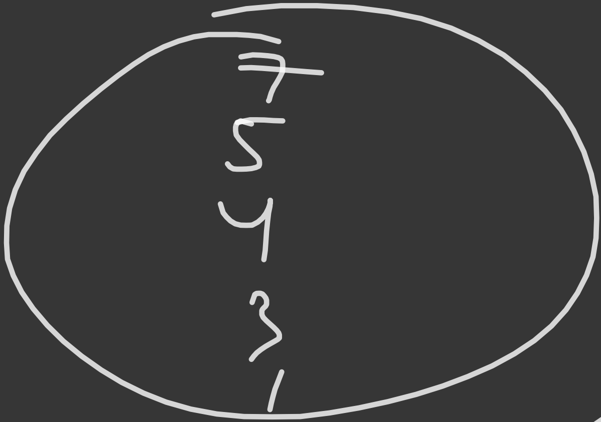
pq size()

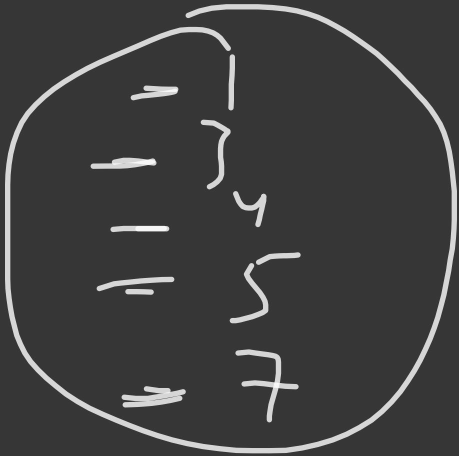
pq.empty()

method1

```
priority_queue<int, vector<int>,
                    greater<int>>
                    pq,
```

method2





- pq top() → 1 → **min heap:**

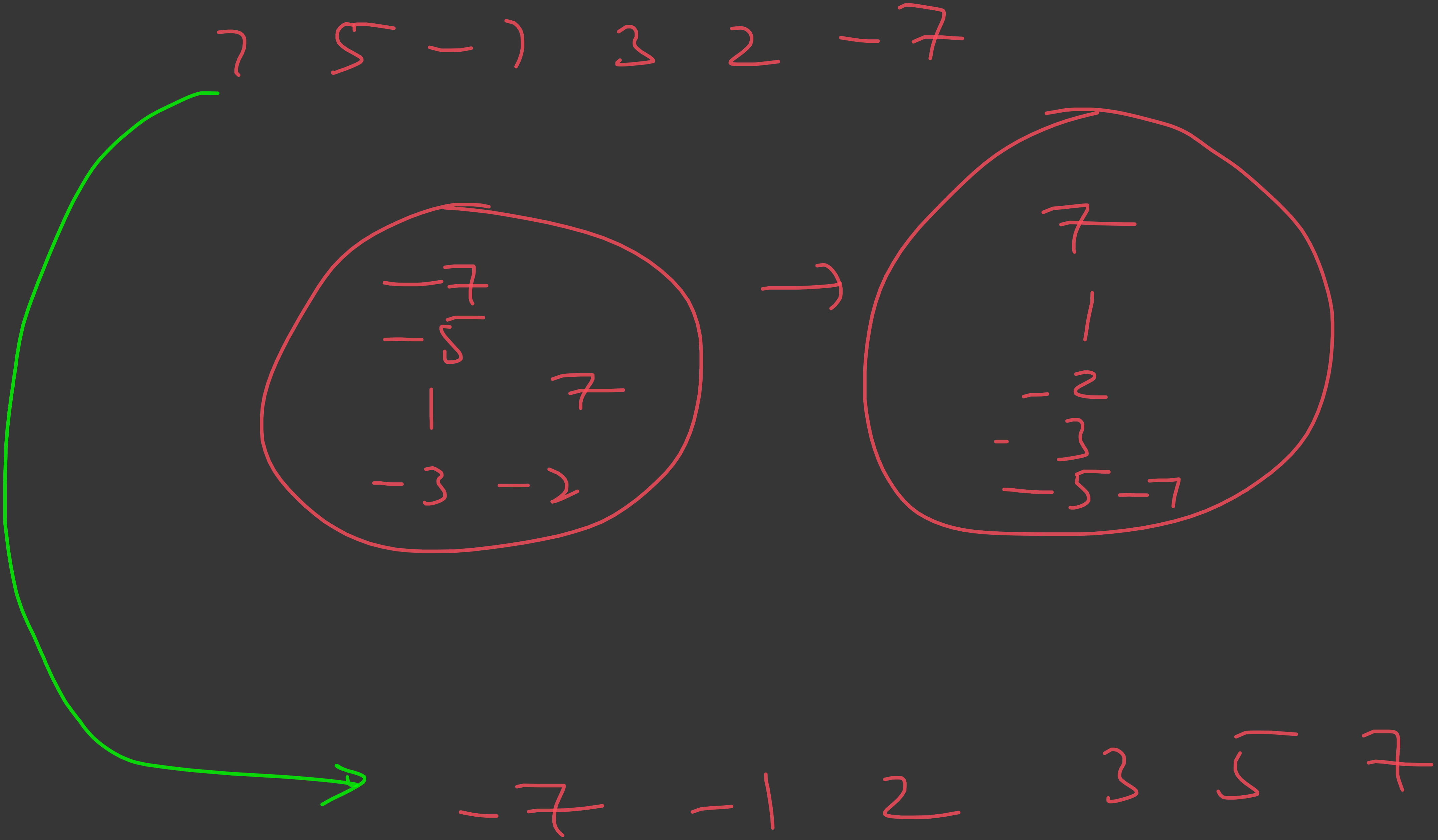
steps ② (max heap → min heap)

→ ① multiply by -1 while inserting in pq

1 ←

② multiply by -1 while taking it out

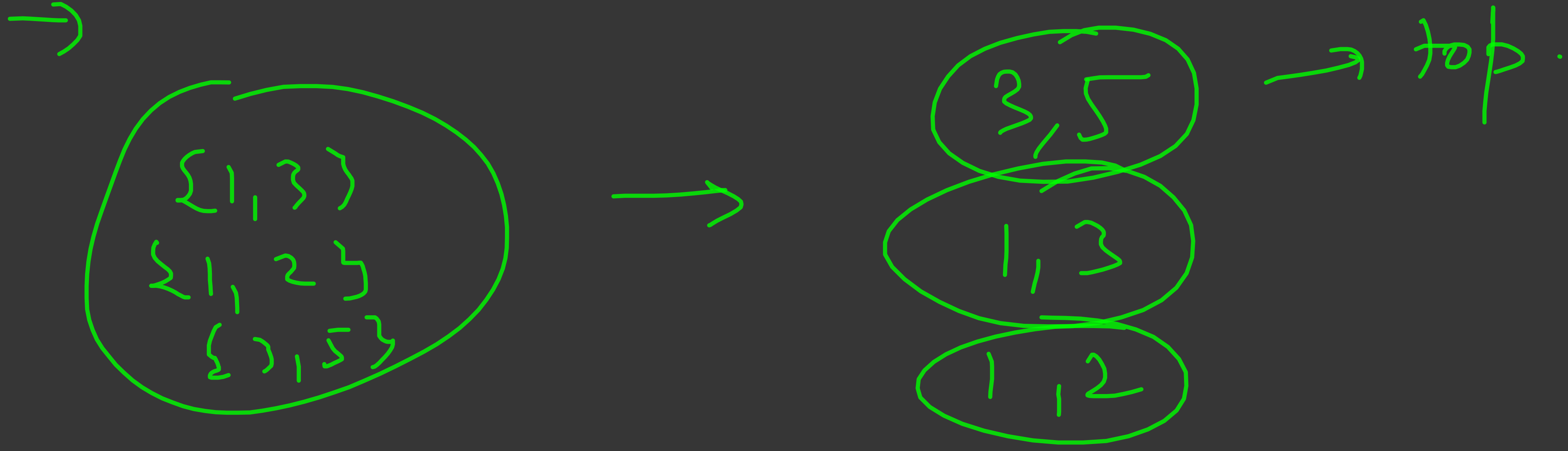
1 ←


7 5 -1 3 2 -7
-7
-5
1
7
-3 -2
7
1
-2
-3
-5 -7
-7 -1 2 3 5 7

$\boxed{\text{pair}} \rightarrow \{\underline{a}, \underline{b}\}$

$\{1, 3\} \quad \{1, 2\} \quad (3, 5\}$

$\rightarrow \{1, 2\} \quad \{1, 3\} \quad \{3, 5\}$



-1, -2

-1, -3

-3, -5

first ↓ asc

sec ↓ desc

-1 3

-1 2

-3 5

-1	2
-1	3
-3	5

Binary search stl → vector → [1, 3, 5, 7, 11

(1) Binary search → → true ✓ / false ✓

✓ log(n)

binary_search(vec.begin(), vec.end(), 5]

(2) lower_bound → you will get first $x \geq 5$

✓ log(n)

(3) upper_bound → you will get first $x > 5$

✓ log(n)

vec.begin()

$$\text{vector} \rightarrow \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ [1, & 3, & 5, & 7, & 11] \end{array}$$

15

auto it = lower_bound(vec.begi
vec.end(), 5);

index = it - vec.begin();

int

vec.begin()

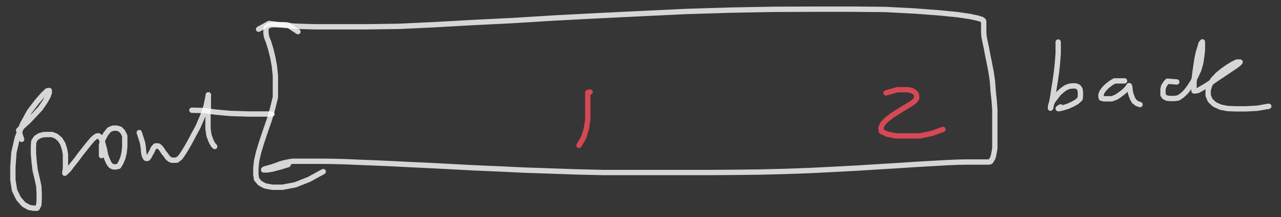
vectors → vec.begin()
→ vec.end()

```
int arr[n];
```

arrays → arr

arrays → arr+n

deque → list



deque

```
deque<int> dq;

dq.push_front(1);
dq.push_back(2);
```

```
dq pop-front()
dq pop-back()
```