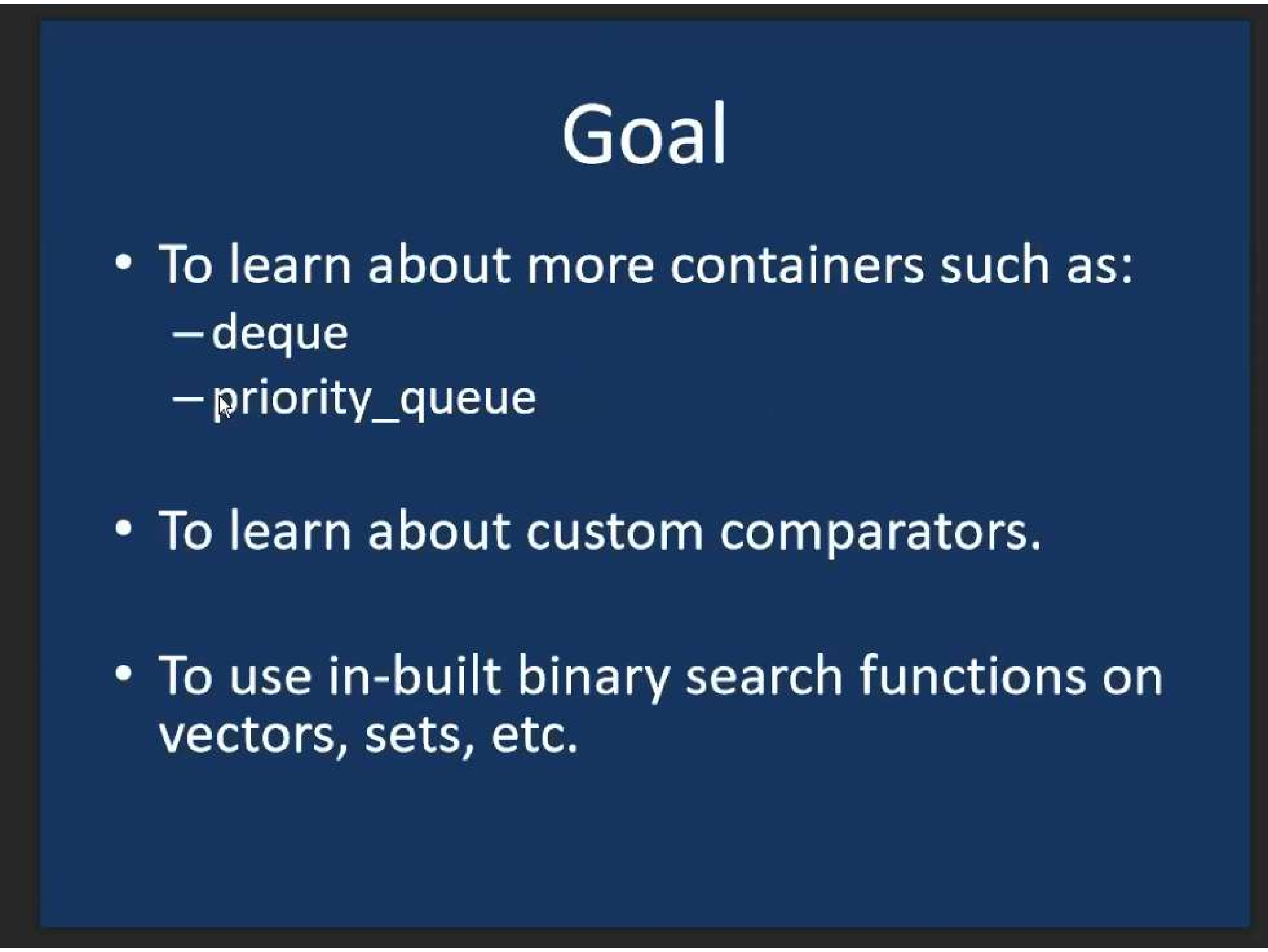
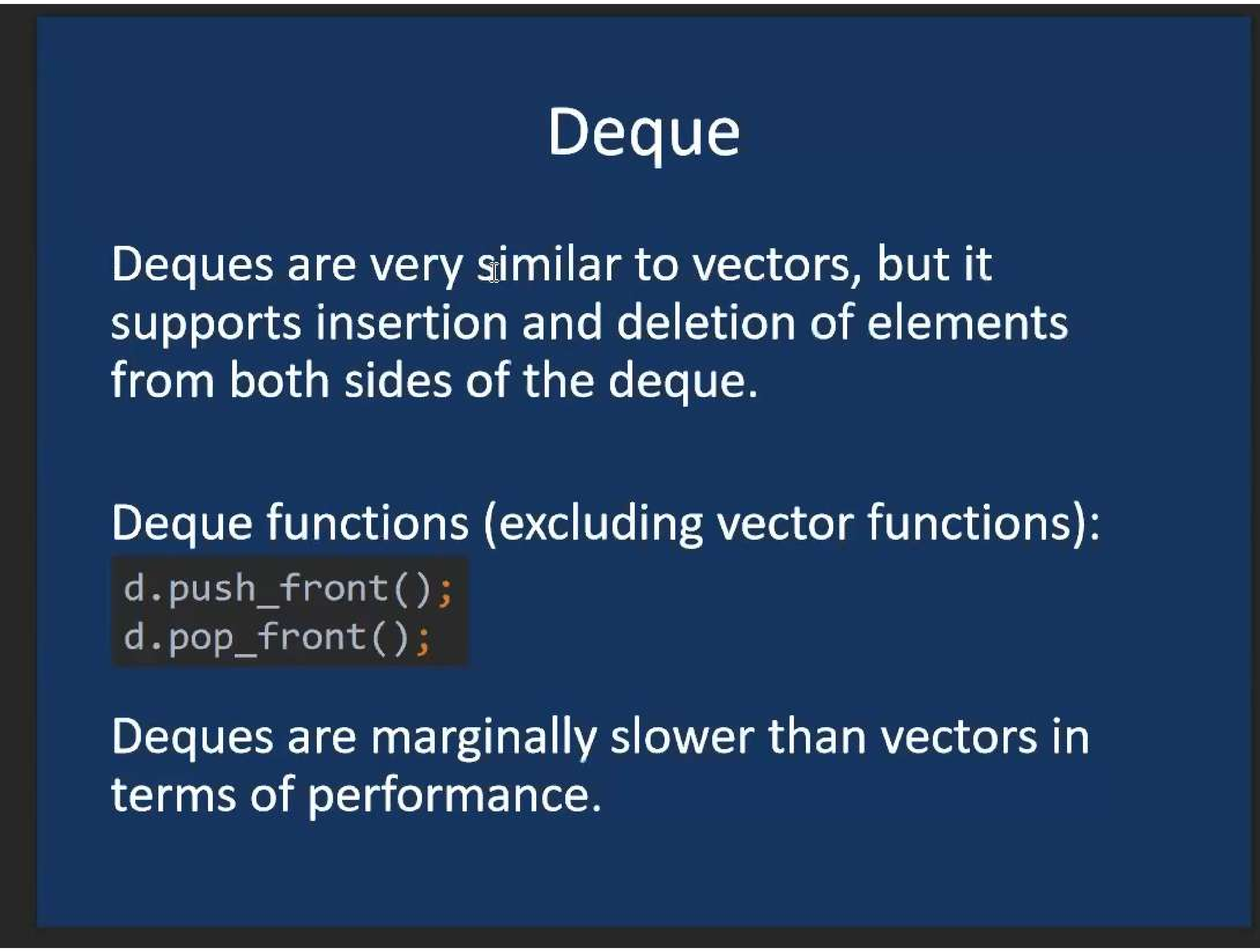
C++ STL (Part 2)

Srivaths P



Goal

- To learn about more containers such as:
  - deque
  - priority_queue
- To learn about custom comparators.
- To use in-built binary search functions on vectors, sets, etc.



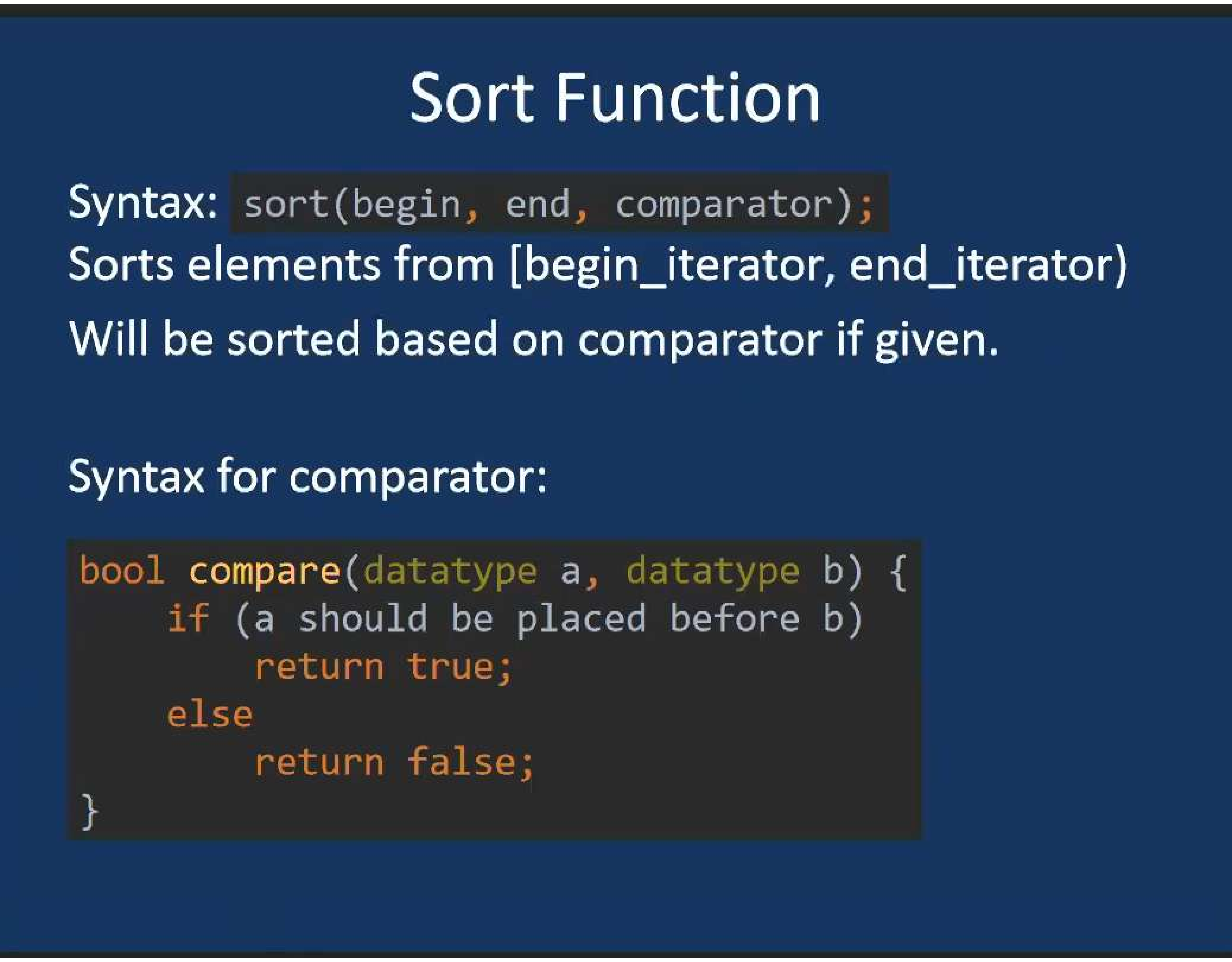
Deque

Deques are very similar to vectors, but it supports insertion and deletion of elements from both sides of the deque.

Deque functions (excluding vector functions):

```
d.push_front();
d.pop_front();
```

Deques are marginally slower than vectors in terms of performance.



Sort Function

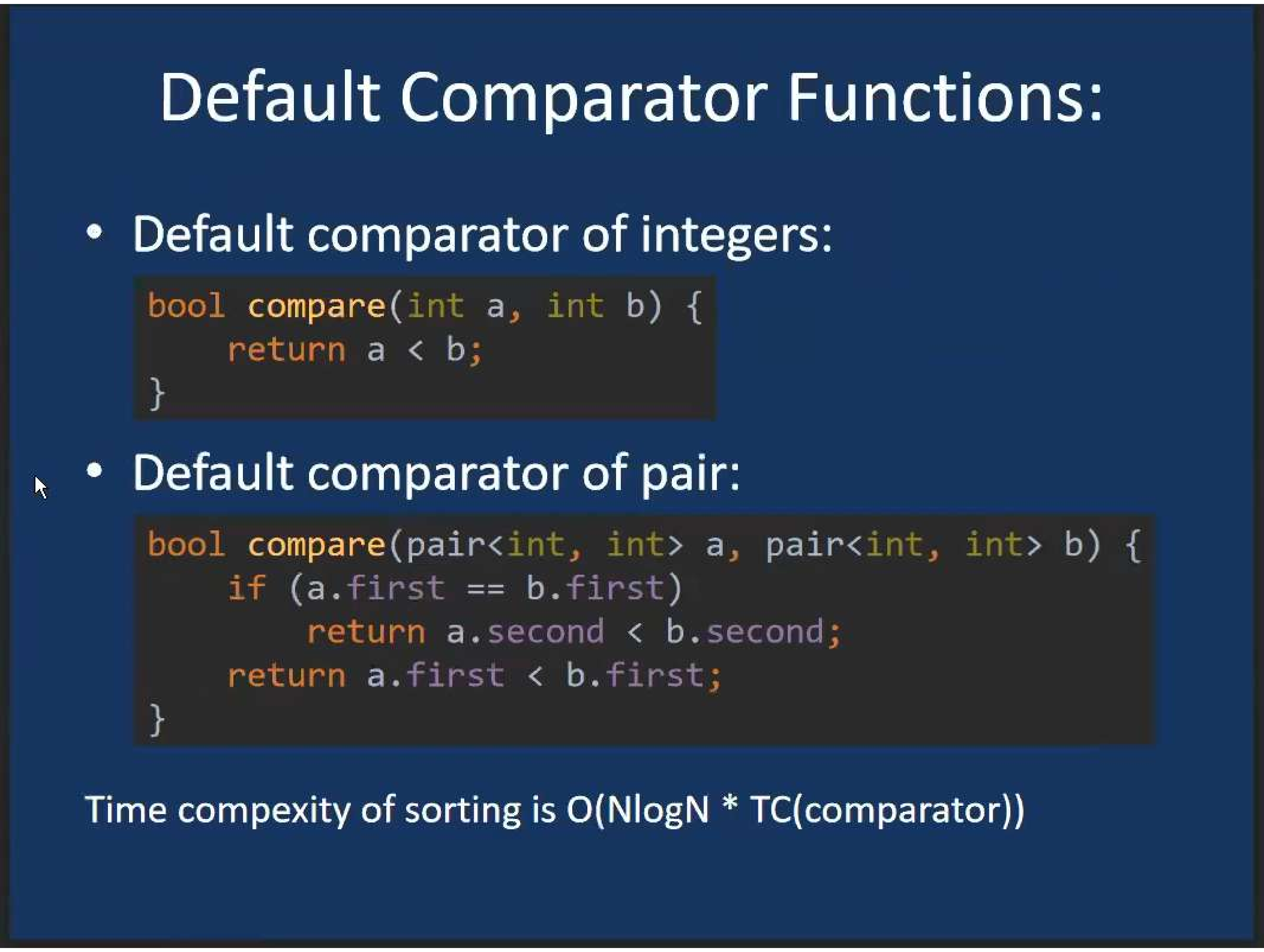
Syntax: `sort(begin, end, comparator);`

Sorts elements from [begin_iterator, end_iterator)

Will be sorted based on comparator if given.

Syntax for comparator:

```
bool compare(datatype a, datatype b) {
    if (a should be placed before b)
        return true;
    else
        return false;
}
```



Default Comparator Functions:

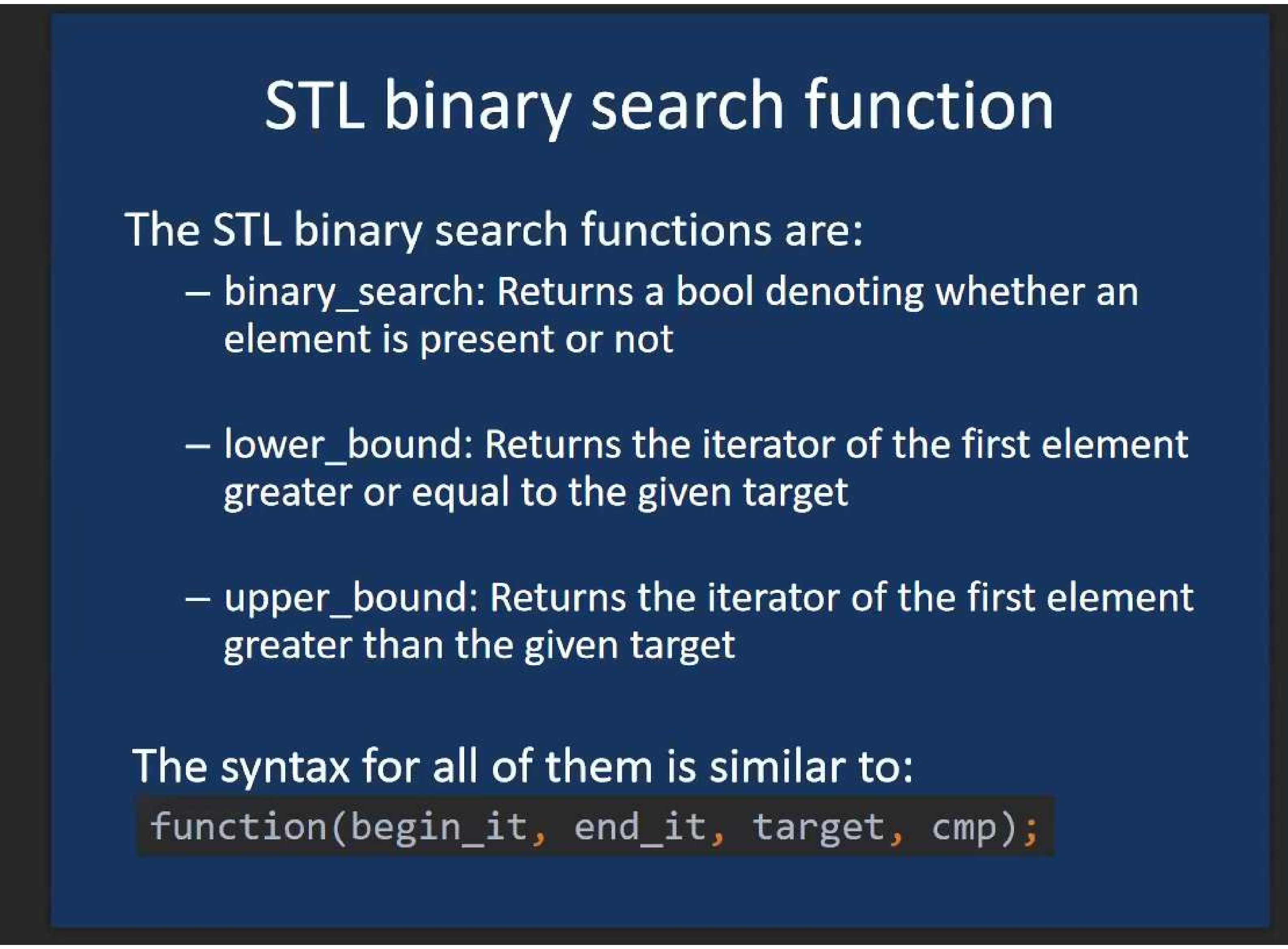
- Default comparator of integers:

```
bool compare(int a, int b) {
    return a < b;
}
```

- Default comparator of pair:

```
bool compare(pair<int, int> a, pair<int, int> b) {
    if (a.first == b.first)
        return a.second < b.second;
    return a.first < b.first;
}
```

Time compexity of sorting is O(NlogN * TC(comparator))



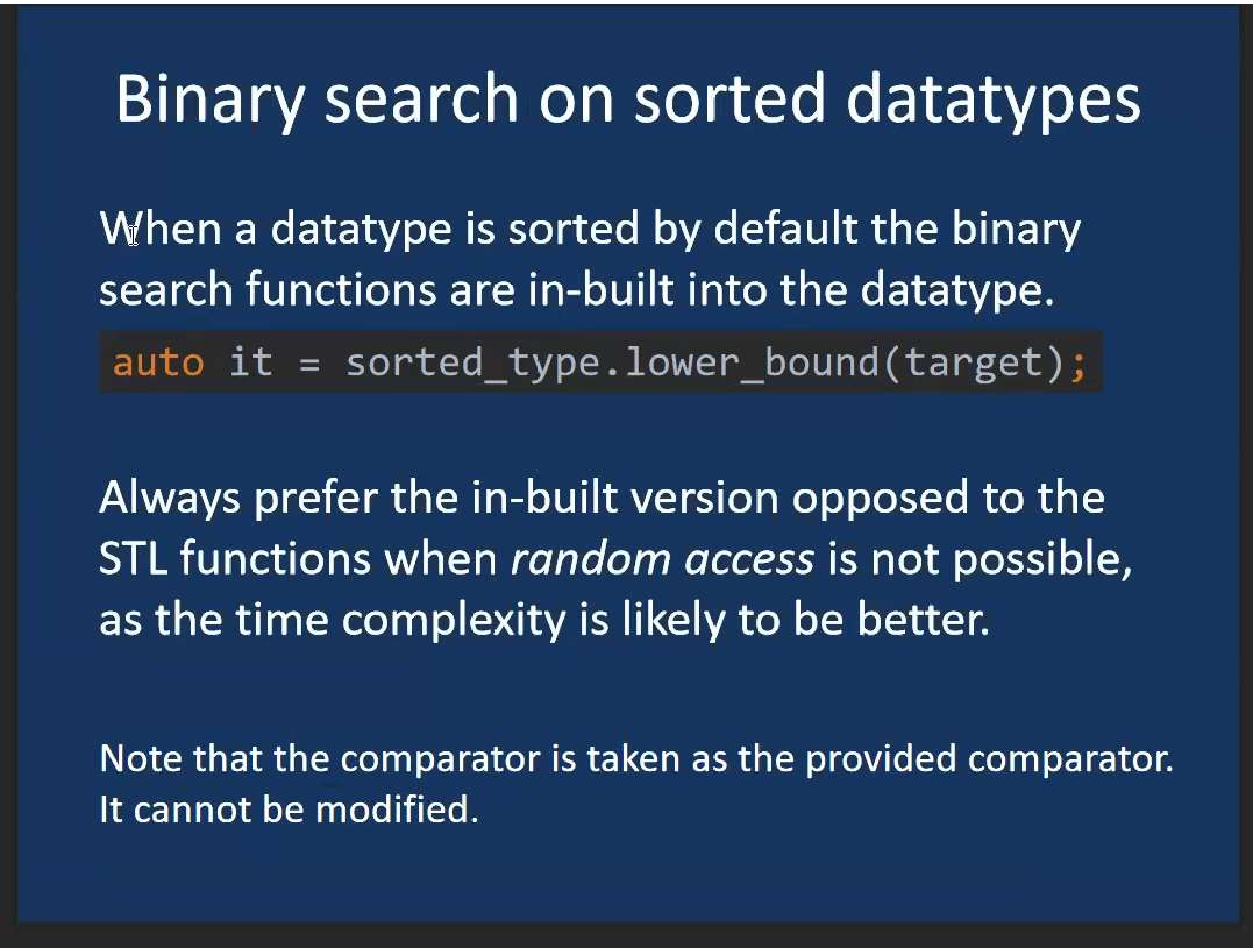
STL binary search function

The STL binary search functions are:

- binary_search: Returns a bool denoting whether an element is present or not
- lower_bound: Returns the iterator of the first element greater or equal to the given target
- upper_bound: Returns the iterator of the first element greater than the given target

The syntax for all of them is similar to:

```
function(begin_it, end_it, target, cmp);
```



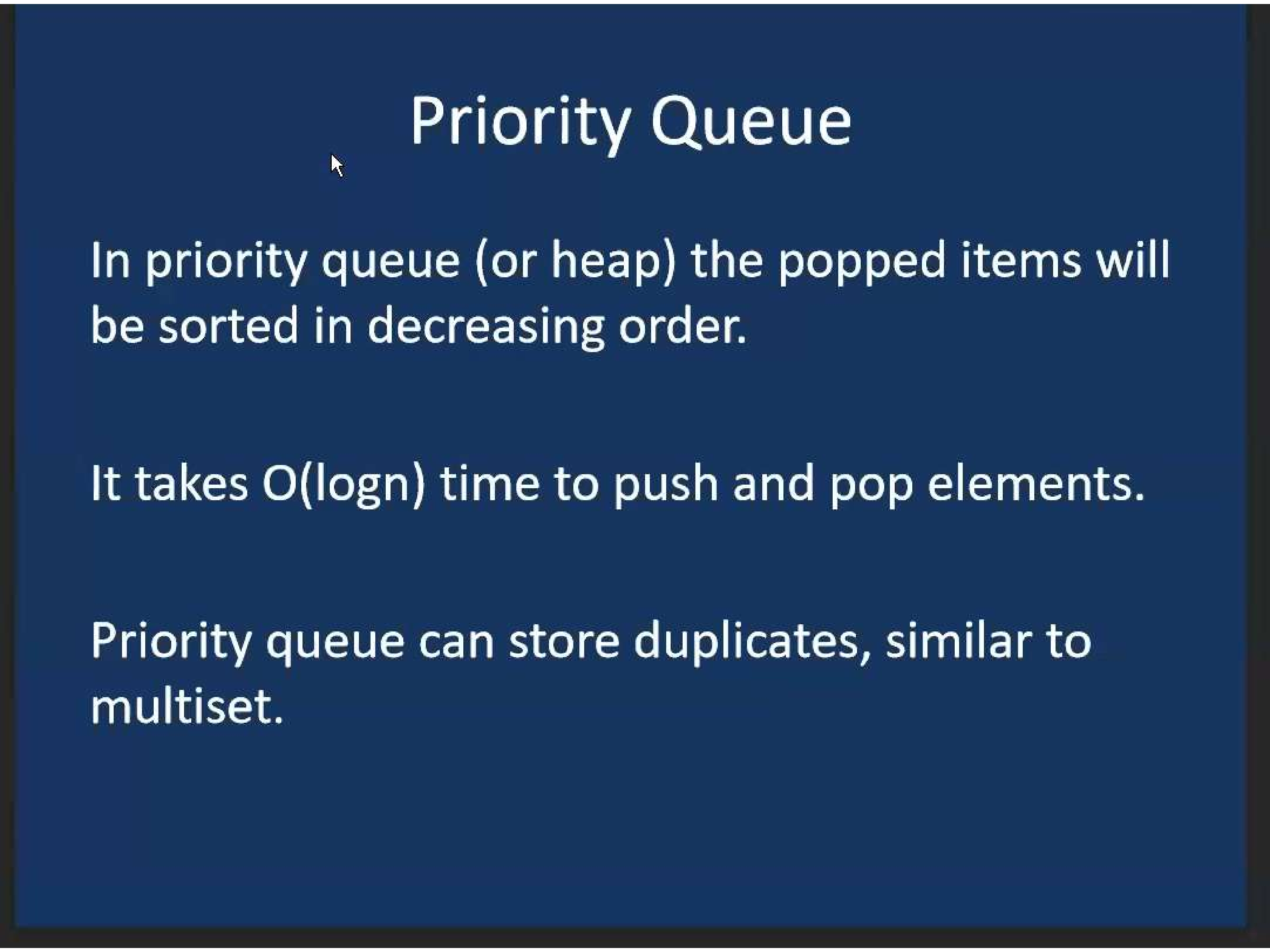
Binary search on sorted datatypes

When a datatype is sorted by default the binary search functions are in-built into the datatype.

```
auto it = sorted_type.lower_bound(target);
```

Always prefer the in-built version opposed to the STL functions when *random access* is not possible, as the time complexity is likely to be better.

Note that the comparator is taken as the provided comparator. It cannot be modified.

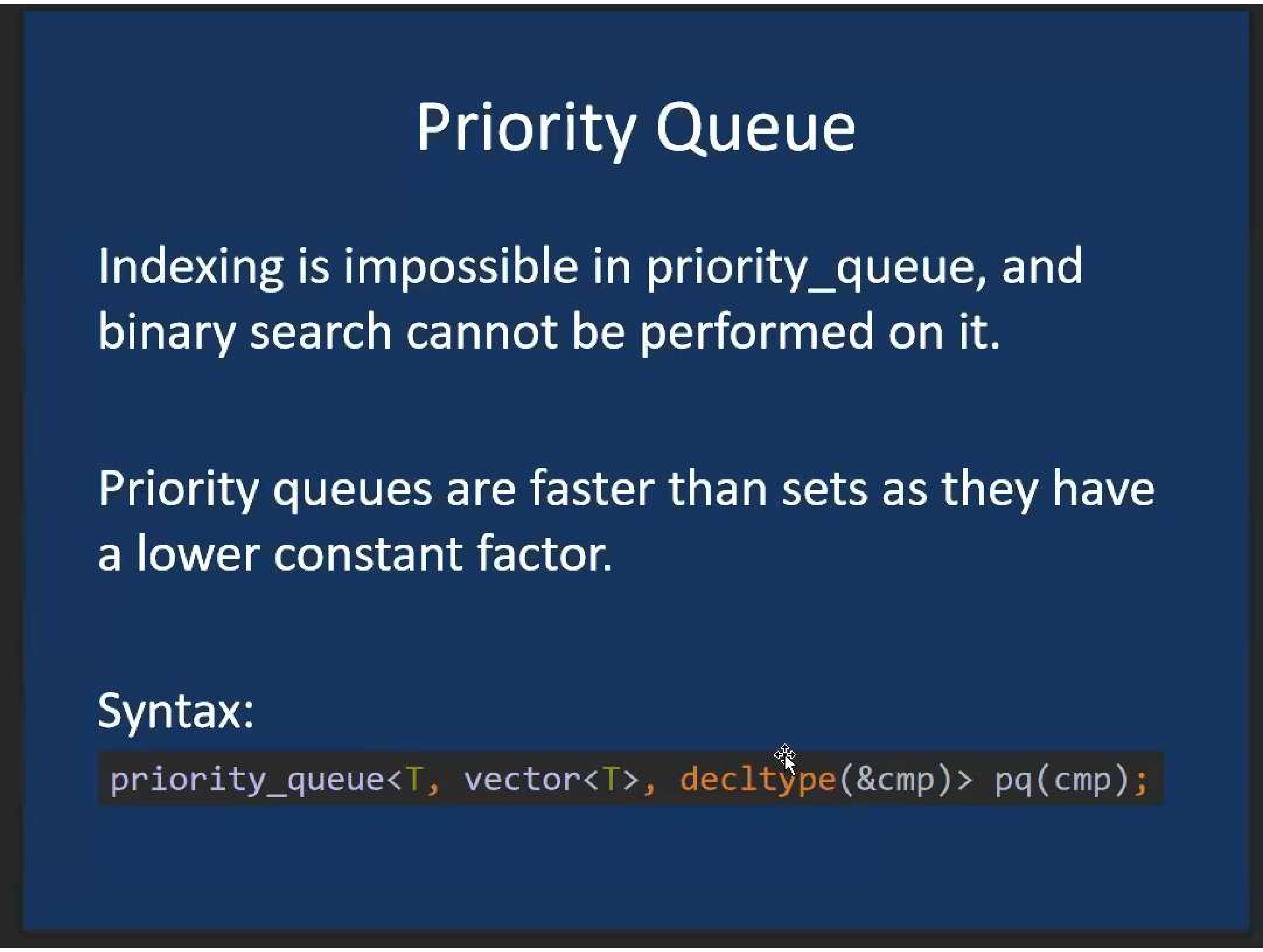


Priority Queue

In priority queue (or heap) the popped items will be sorted in decreasing order.

It takes O(logn) time to push and pop elements.

Priority queue can store duplicates, similar to multiset.



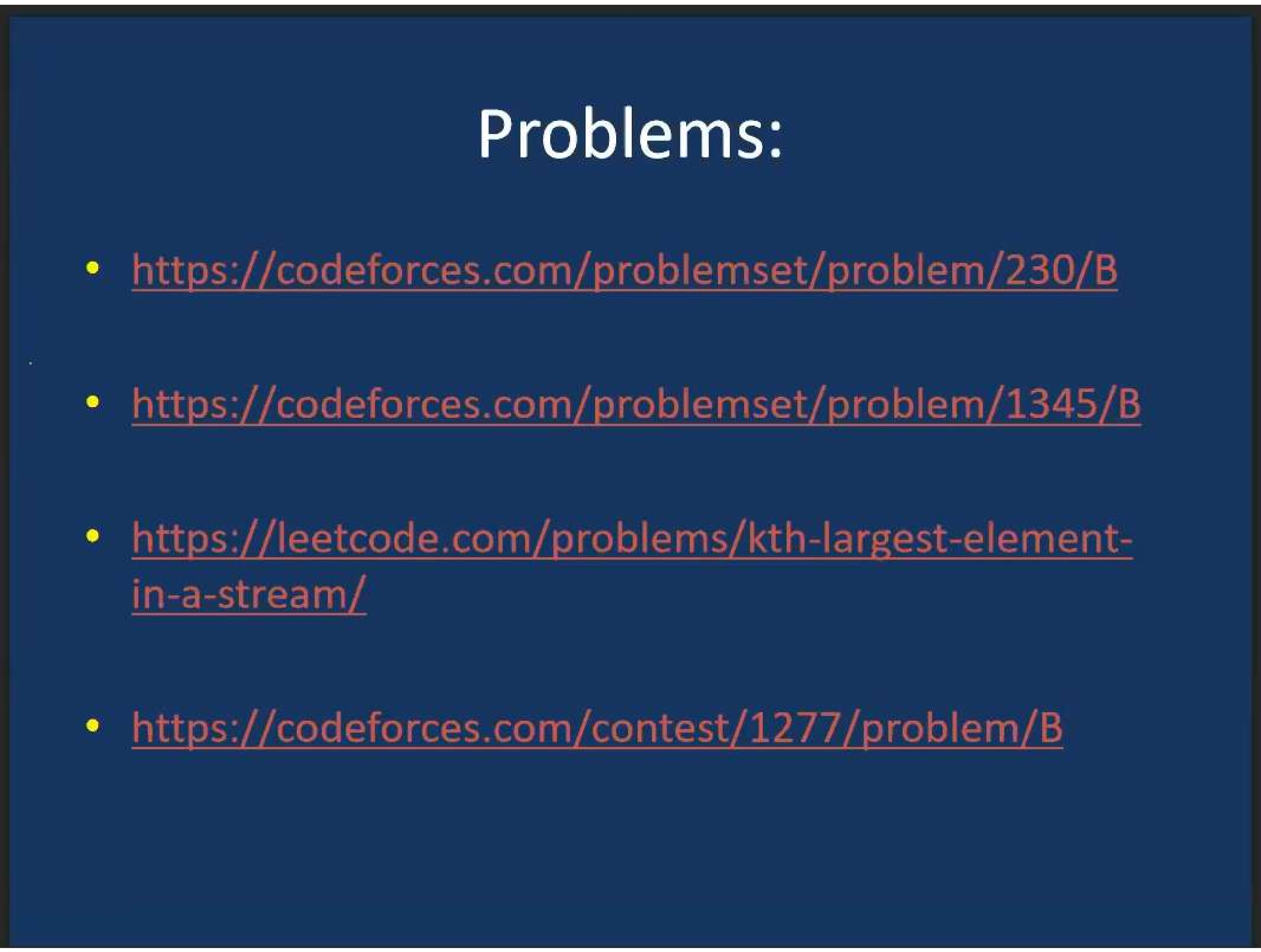
Priority Queue

Indexing is impossible in priority_queue, and binary search cannot be performed on it.

Priority queues are faster than sets as they have a lower constant factor.

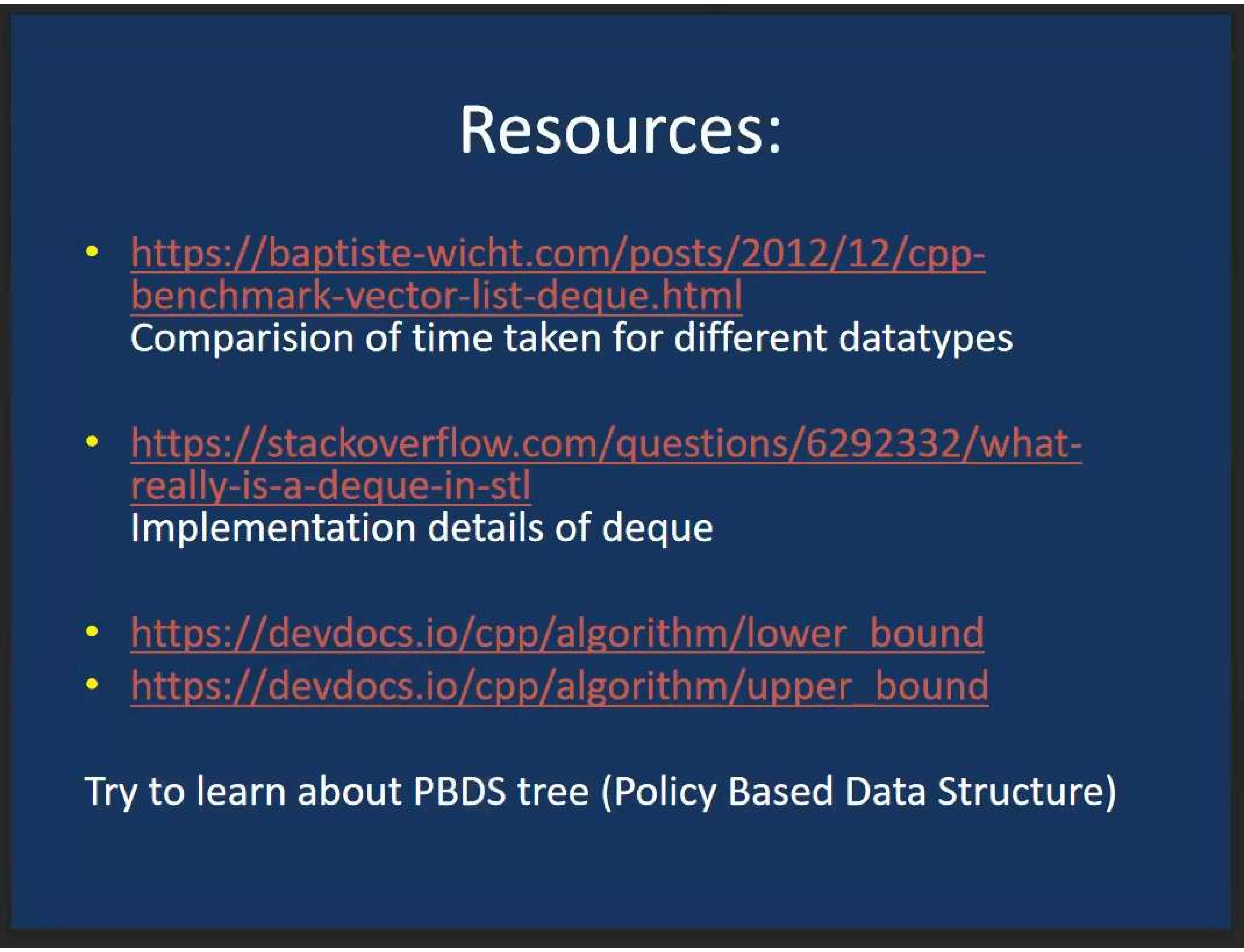
Syntax:

```
priority_queue<T, vector<T>, decltype(&cmp)> pq(cmp);
```



Problems:

- https://codeforces.com/problemset/problem/230/B
- https://codeforces.com/problemset/problem/1345/B
- https://leetcode.com/problems/kth-largest-element-in-a-stream/
- https://codeforces.com/contest/1277/problem/B



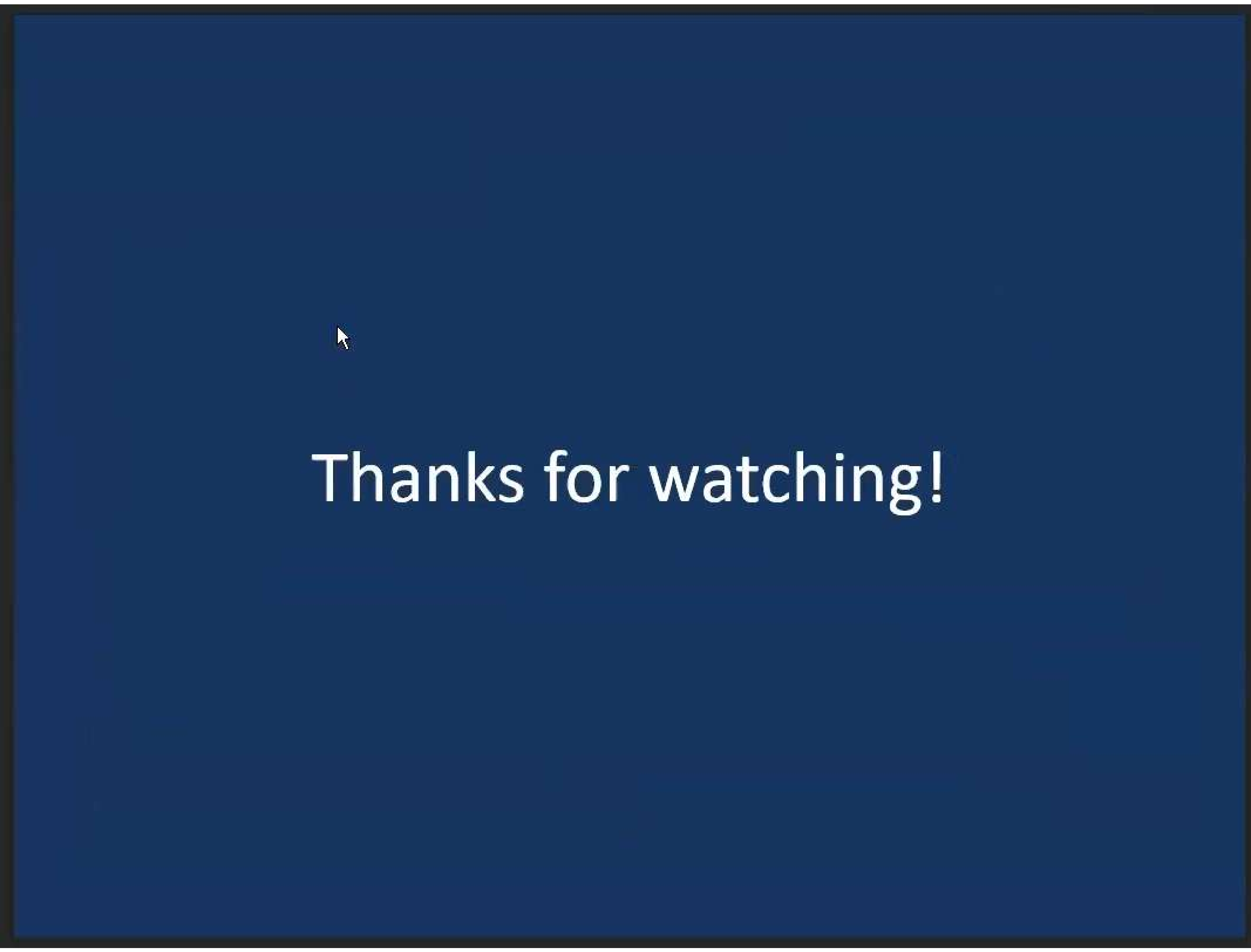
Resources:

- https://baptiste-wicht.com/posts/2012/12/cpp-benchmark-vector-list-deque.html
  Comparision of time taken for different datatypes

- https://stackoverflow.com/questions/6292332/what-really-is-a-deque-in-stl
  Implementation details of deque

- https://devdocs.io/cpp/algorithm/lower_bound
- https://devdocs.io/cpp/algorithm/upper_bound

Try to learn about PBDS tree (Policy Based Data Structure)



Thanks for watching!