8.24 In each of the following scenarios, identify the type of container that should be used to store the collection in question, and justify your choice, being careful to state any important assumptions made. (The justification of your choice is critically important, since more than one correct answer is possible.)

(a) A program needs to maintain a large set of integers using a container. The complete contents of the container are known at the time that the container is first created and do not change subsequently. The container is very frequently searched for particular values. The ability to iterate over the container elements in sorted order is frequently needed.

Binary search tree(multiset ADT) can be used in this case. Since the binary search tree itself mantains some properties, so we can iterate its elements in sorted order. The time complexity for searching is O(log(n)) or O(h) where h is height of the tree.

(b) A program needs to maintain a large set of integers using a container. The container contents are modified very frequently during program execution by inserting and removing elements. The values to be inserted in, and removed from, the container follow no particular pattern. The container is frequently searched for particular values. The ability to iterate over the container elements in sorted order is frequently needed.

Assume the element contains duplicate values, multiset ADT or multimap ADT can be used in this case. They are implemented using balanced binary search tree, and both inserting and removing take O(log(n)).

(c) A program needs to maintain a large set of real numbers using a container. The maximum size of this set is known at the time that the container is created. The container contents are modified very frequently during program execution by inserting and removing elements. While the values inserted into the container follow no particular pattern, each element removed from the container is always the one with the largest value.

Since the question mentions that removed from the container is always the largest value, here we can use priority queue to storing data.

8.26 Explain why one might want to separate the operations of memory allocation and construction. Similarly, explain why one might want to separate the operations of destruction and memory deallocation.

In some situation, we may want to pre-allocate memory first, and then construct object later when needed. Since constructing objects during memory allocation is somehow wasteful if we never use all of them. For the operations of destruction and memory deallocation, we could just want to destruct the object, and reuse the allocated memory later, it is more effieient if we can separate them.

8.27 Identify two advantages that array-based containers have over node-based containers.

1. elements stored contiguously in memory.

2. no per-element storage overhead.

8.29 In each of the following scenarios, indicate whether an intrusive or nonintrusive container should be used to store each collection in question, and justify your choice, being careful to state any important assumptions made.

(a) A operating system kernel represents a process (i.e., a thread plus the resources needed for its execution) using a type called `process`. The `process` type has a very substantial amount of state (and is neither movable nor copyable). The operating system must maintain two containers. The first holds all existing processes (i.e., the process list). The second holds all processes that are ready to run (i.e., the scheduling queue), which is used for process scheduling.

From the question, the `process` type neither movable nor copyable, and the process list and the scheduling queue seems storing same objects, so probably should using intrusive cintainer here.

> (b) A program needs to maintain a collection of mutexes of type `std::mutex` in a container. Due to other design constraints, none of the nonintrusive containers that are available for use provide stable references.

To mantaining mutexes, we may require stronger complexity guarantees and better exception safety guarantees. Therefor, we use intrusive containers to maintain them(no memory allocation or copying performed).

> (c) A program needs to maintain a collection of mutexes of type `std::mutex` in a container. Nonintrusive containers are available that allow in-place construction and provide stable references. The mutexes only need to exist inside the container.

If we guarantee objects are not to be copied or moved subsequently and with the conditions provided in question, we can use nonintrusive containers to maintain these mutexes