Tania Akter
V00810640

8.17

a.
The best option is Array-based sorted list
The element insertion can be done in sorted order in the beginning since the contents are known. Since the container is first created and not dot change frequently, it means we do not have to worry too often about growing the container to insert more values.
We can also search for particular value from already sorted list by using a binary search.

Since we also have to iterate over the container in sorted order, we iterate over easily since it is already sorted.

b.
Node-Based sorted list
Node based sorted list can be used to insert and remove at a constant time, hence modifying the container frequently is not costly. We can search for a particular value using binary search in log(n) time, since the list is already sorted. Moreover, iterating over the elements in sorted order is going to be a linear time since the list is already sorted.

c.
Node-based heap.

We need heap because we are always removing the largest element from the container. Since the content of the container is frequently updated, we can take advantage of having node-based heap and do the insert and remove operations in logarithmic time.

8.18
If we want to initialize a big container size, such as an array, constructing it will construct each of the elements in it, which is costly. We want to only allocate memory for the object in this case without constructing every element in it, hence we want the memory allocation and construction to be separate. Similarly, we may not want to both destroy objects and deallocate memory at the same time. One might want to free the memories to be used later, buExplain why one might want to separate the operations of memory allocation and constructiont if we destroy the object that lives inside, we might be destroying objects that the user has passed.

8.19

Advantages:

1. Elements are stored contiguously in memory
   It provides advantages on caching the elements.

2. No per-element storage overhead

When storing elements with node, we need an iterator which is an overhead for each element insertion. With array-based containers, we can get rid of the extra overhead

8.23

a.
Process has a substantial amount of state, and are neither movable, hence they cannot be stored in nonintrusive containers. We need intrusive container to store the processes.

b
We need intrusive container, because the nonintrusive container provide stable references, and for the mutexes to be used, we need to have stable references.

C
We need intrusive container because the mutexes only need to exist inside the container.