Stack. Representations

initEmpty
push **O(1)**
pop
isEmpty

- over array (/ vector)
- over linked-list

Queue. Representations

initEmpty
enqueue
dequeue
isEmpty

O(1)

- over array (/ vector)
- over linked-list

Deque. Representations

initEmpty
push_back
push_front
pop_back
pop_front
isEmpty

O(1)

- over array (/ vector)
- over linked-list

STL: Stack, Queue. Issues

std::stack - container adaptor

template < class T, class Container = deque<T> > class queue;

- the underlying container
  - •back()
  - •push_back()
  - •pop_back()

std::queue - container adaptor

template < class T, class Container = deque<T> > class queue;

- the underlying container
  - •front()
  - •back()
  - •push_back()
  - •pop_front()

STL: deque

Deque:

- Specific libraries may implement deque in different ways
  generally as some form of dynamic array
- with efficient insertion and deletion of elements at the beginning and at its end.

Vector vs. **deque**

provide a very similar interface and can be used for similar purposes

internally can be quite different

Vector: use a single array

Deque: deques are not guaranteed to store all its elements in contiguous storage locations

can be scattered in different chunks of storage

ex.: implemented as a vector of vectors

Java: Stack, Queue, Deque. Issues

Java™ Platform
Standard Ed. 7

Interface :	Queue
Subinterface:	Deque
Implementing Classes:	ArrayDeque
	LinkedList

Java: Stack, Queue, Deque. Issues

Java™ Platform
Standard Ed. 7

public class Stack<E>
extends Vector<E>

- Use Deque instead of Stack

  Deque<Integer> stack = new ArrayDeque<Integer>();

  A more complete and consistent set of LIFO stack operations is provided by the Deque interface and its implementations,

  which ***should be used in preference*** to this class.