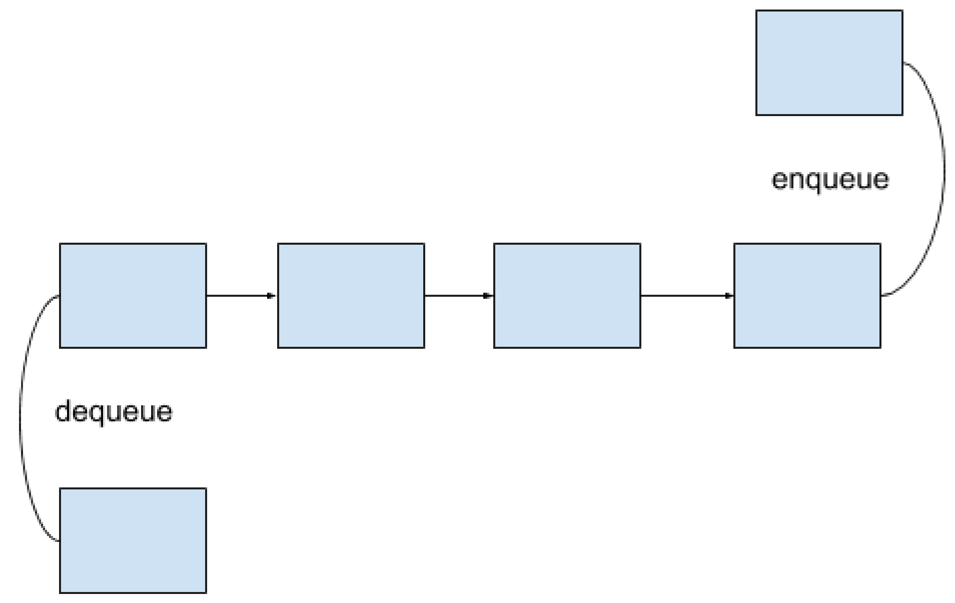
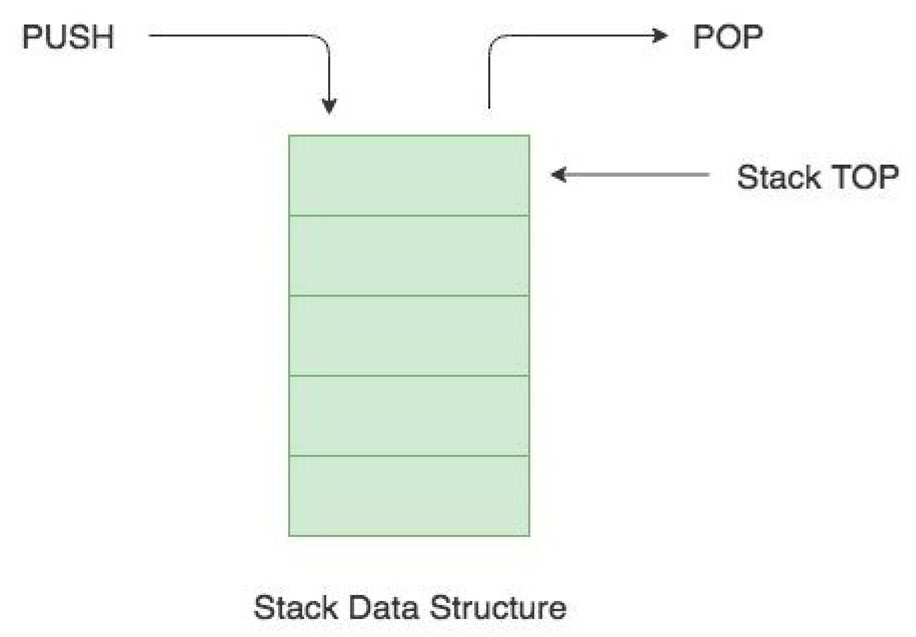
TAD <Queue>

{inv: $0 \leq n \land Size(Queue) = n \land front = e1 \land back = en$ }
Operaciones primitivas: ❏ enqueue: queue x value → queue ❏ dequeue: queue → queue ❏ peek: queue → value ❏ isEmpty: queue → boolean

enqueue
An input is added into the end of the queue.
pre: Having the queue already created and unfilled.
pos: A value was added to the end of the queue.

dequeue
Takes out the first value of the queue and returns it.
pre: Having the queue already created.
pos: Value that was on the top of the queue or an exception.

IsEmpty
Checks if the queue is empty
pre: queue exists.
pos: Boolean with the result of the check.

Peek
Returns the value on top of the queue without touching it.
pre: queue exists.
pos: Value on the top of the queue or exception if it's empty.

TAD <Stack>

{inv: $0 \leq n \wedge Size(Stack) = n \wedge top = e_1$ }
Primitive Operations: ❏ Push Stack x Value → Stack ❏ Pop Stack → Stack ❏ IsEmpty → Boolean ❏ Peek Stack → Value/Text

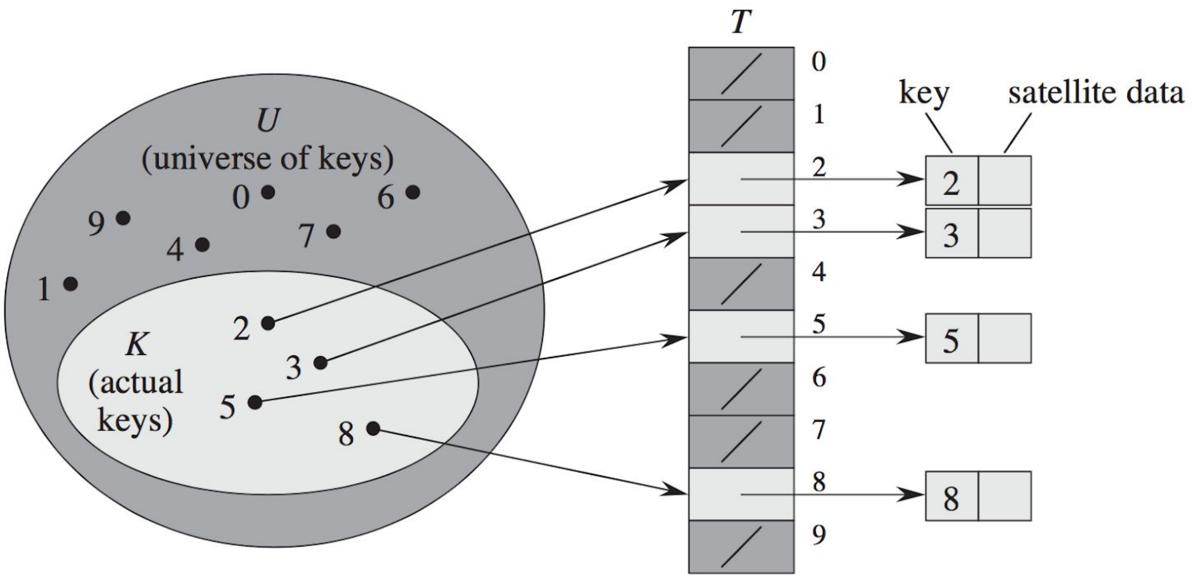
❏ Push	Stack x Value	→ Stack
❏ Pop	Stack	→ Stack
❏ IsEmpty		→ Boolean
❏ Peek	Stack	→ Value/Text

Push
An input is added into the top of the Stack.
pre: Having the Stack already created and unfilled.
pos: A value was added to the top of the Stack.

Pop
Takes out the value on top of the stack and returns it.
pre: Having the Stack already created.
pos: Value that was on the top of the stack or an exception.

IsEmpty
Checks if the Stack is empty
pre: Stack exists.
pos: Boolean with the result of the check.

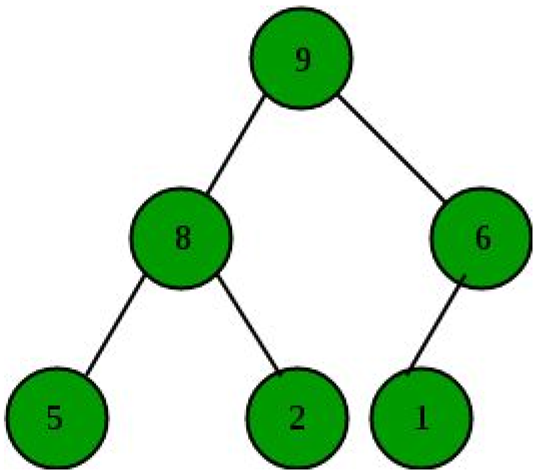
Peek
Returns the value on top of the Stack without touching it.
pre: Stack exists.
pos: Value on the top of the stack or exception if it's empty.

TAD <HashTable>
 T U (universe of keys) K (actual keys) key satellite data 0 1 2 3 4 5 6 7 8 9
{inv: MaxSize <= u}
Primitive Operations: ❏ Add HashTable x Value → HashTable ❏ Remove HashTable x Value → HashTable ❏ Search → Boolean

Add
Adds a value into the HashTable.
pre: HashTable already created.

Remove
Overwrites the position of the hashtable with an Object defining that position as “open”
pre: HashTable exists, index is valid.
pos: Null or the element that was deleted

Search
Searches the given index on the HashTable.
pre: HashTable exists, index is valid.
pos: true if value was found else false.

TAD <MaxHeap>

{inv: ValueOfParent > ValueOfLeftChildren ^ ValueOfParent > ValueOfRightChildren}

Primitive Operations:

❑	Insert	MaxHeap x Value	→ MaxHeap
❑	Heapify	int	→ MaxHeap
❑	BuildMaxHeap		→ MaxHeap
❑	HeapSort		→ MaxHeap

Insert
Adds a value into the heap and calls BuildMaxHeap.
pre: MaxHeap already created.

BuildMaxHeap
Calls the method Heapify on each value of the array until it reaches a null space or it ends
pre: MaxHeap already created.

Heapify
Arranges node i and it’s subtrees to satisfy the heap property.
pre: HashTable exists,

HeapSort
Sorts the heap
pre: MaxHeap exists and is not empty.