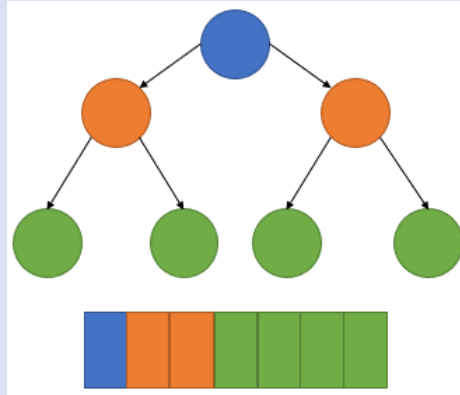


TAD Heap <T,V>



{inv: The values of each node are less than or equal to the values of its children}

Primitive operations:

- father: int ->int
- left: int ->int
- right: int ->int
- addHeapNode: T,V ->boolean
- heapify: int ->HeapNode<T,V>[]
- buildHeap:
- getArraySize: ->int
- getArray: ->HeapNode<T,V>[]

father

“return the floor of $n/2$ of the father node”

{pre: the father node must exist}

{post: floor of $n/2$ }

Analyzer

left

“return the floor of $n/2$ of the left node”

{pre: the left node must exist}

{post: floor of $n/2$ }

Analyzer

right

“return the floor of $n/2$ of the right node”

{pre: the right node must exist}

{post: floor of $n/2$ }

Analyzer

addHeapNode

“add a node to the heap”

{pre: receives the parameters T y V}

{post: node has been added}

Modifier

heapify

“the node with the highest K becomes the father node”

{pre: receives the position of the largest K}

{post: the heapify has been completed}

Builder

buildHeap

“build a heap”

{pre: TRUE}

{post: Heap has been created}

Builder

getArraySize

“return the array size of the heap”

{pre: TRUE}

{post: Heap size}

Analyzer

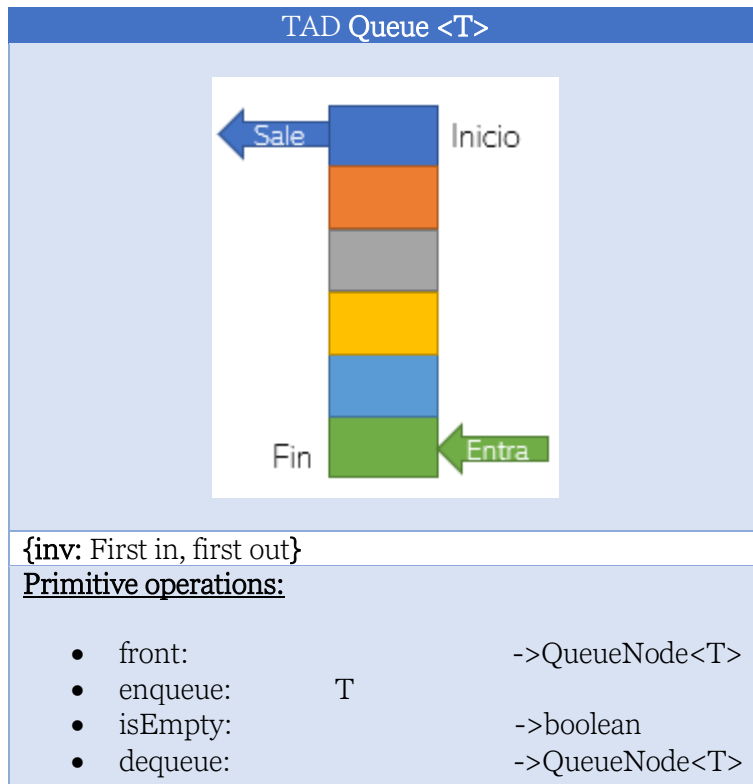
getArray

“return the array of the heap”

{pre: TRUE}

{post: Array of the heap}

Analyzer



front

“return the node of queue that is in the first position”

{pre: TRUE}

{post: Queue node in the first position}

Analyzer

enqueue

“add an element T to the end of the Queue”

{pre: receives a T element}

{post: The element has been added}

Builder

isEmpty

“check that the queue is empty”

{pre: TRUE}

{post: boolean indicating whether the queue is empty}

Analyzer

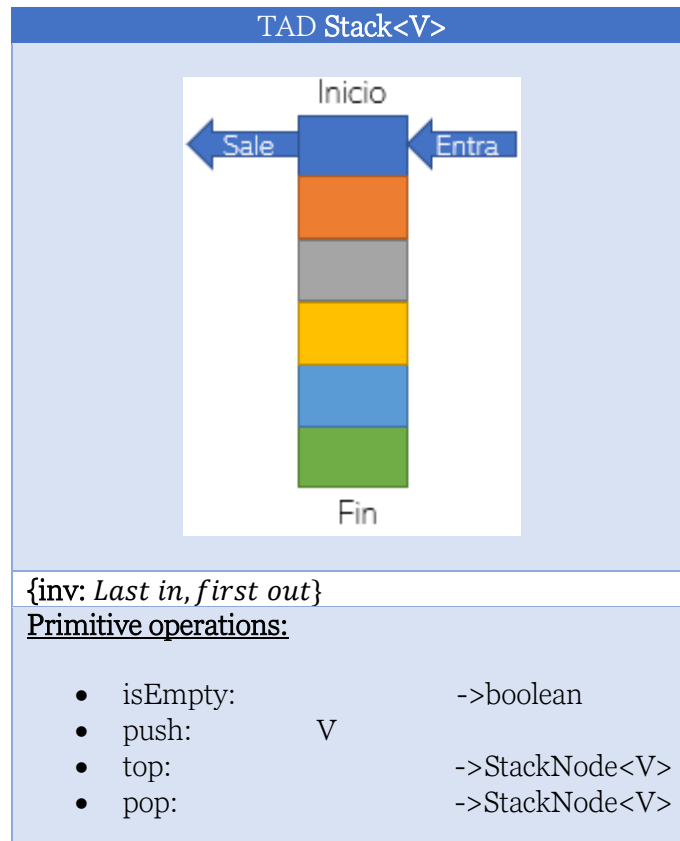
dequeue

“remove the first element of the queue”

{pre: TRUE}

{post: return element T removed}

Modifier



$\{\text{inv: Last in, first out}\}$

- isEmpty: ->boolean
- push: V
- top: ->StackNode<V>
- pop: ->StackNode<V>

isEmpty

```
{pre: TRUE}
```

Analyzer

push

```
{pre: receives the value V}
```

Builder

[top](#)

```
{pre: TRUE}
```

Builder

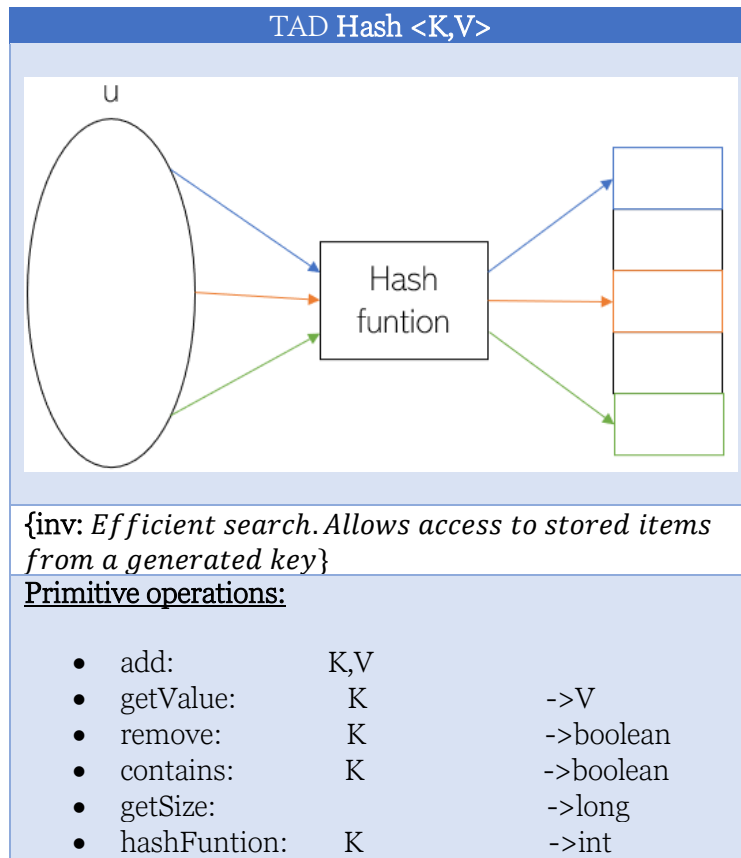
pop

“remove the first value of the Stack”

{pre: TRUE}

{post: return value V removed}

Modifier



add

“add hash node”

{pre: receives a key K and a value V }

{post: hash node has been added}

Builder

getValue

“return the value V of a give key K ”

{pre: receives a key K }

{post: return V corresponding to K }

Analyzer

remove

“remove the value of a given key”

{pre: receives a key K }

{post: boolean indicating that V corresponding to K has been removed}

Modifier

contains

“verifies the existence of the given key K”

{pre: receives a key K}

{post: boolean indicating if the key K exists}

Analyzer

getSize

“return the size of the hash table”

{pre: TRUE}

{post: Hash table size}

Analyzer

hashFunction

“optimize data search in the hash table”

{pre: receives a key K}

{post: position to which the value was added}

Analyzer