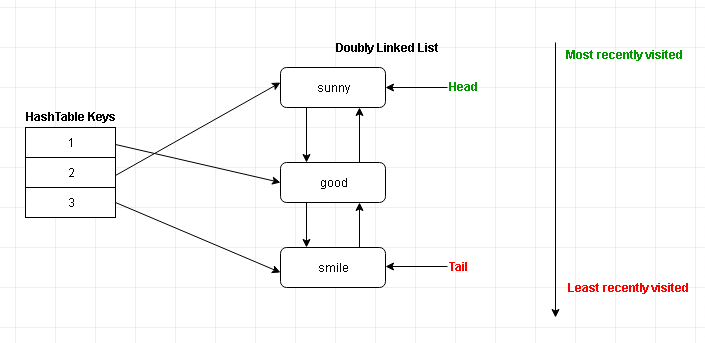
MemoryCache

# Description

MemoryCache is a fast fixed-size thread-safe key-value in memory cache based on Eviction Policy. It supports both the primitive type and user defined type as the key.

# High Level Design

MemoryCache uses **HashTable** and **Doubly Linked List** data structures to achieve **O(1)** operations including cache **eviction**.



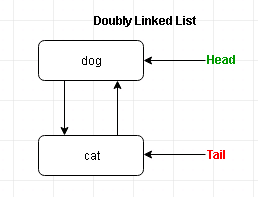
## Doubly Linked List Operation Scenarios

### Scenario 1 for AddOrUpdate’s Add

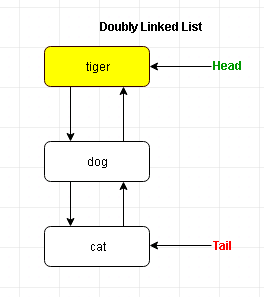
Add a node to the list when the current size is **smaller than the Size Limit**.

* There is no need to remove existing node from the list.
* We just need to add the node to the List add the Head.

The Size limit of the cache is 3 and the list before adding node is:



After adding “tiger” to the list

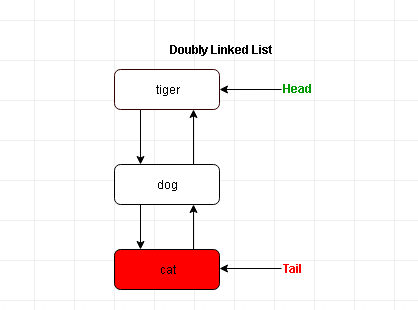


### Scenario 2 AddOrUpdate’s Add:

Add a node to the list when the current size is **equal to the Size Limit**, which means the Memory Cache is **FULL**.

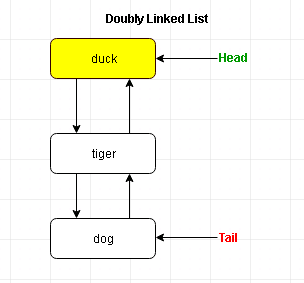
* We need to remove the Least Recently Visited node from the list
* Then add the node to the List add the Head.

The Size limit of the cache is 3 and the list before adding node is:



“**Cat**” is the **Tail** and **least recently** visited node, so it will be **removed** from the list.

After adding “**Duck**” to the list



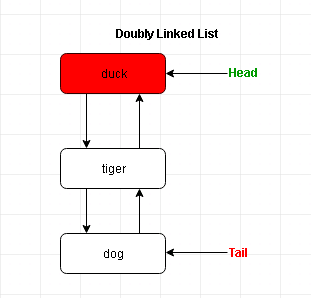
Now “**duck**” becomes the **most recently visited** node and “**dog**” becomes the **least recently visited** node.

### Scenario 3 for AddOrUpdate’s Update:

Update a node in the list when the node is the **Head** of the List.

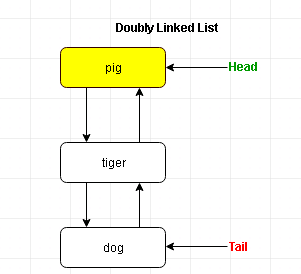
* We just need to update the value of the node.

The Size limit of the cache is 3 and the list before adding node is:



“**duck**” is the node for the **given KEY** and it is the **head** of the list, so we just need to update its value.

After updating “duck” to “pig”



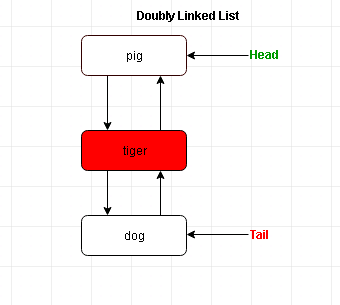
“**pig**” is still the **Head** of the list and “**dog**” is still the least **recently visited node**.

### Scenario 4 for AddOrUpdate’s Update:

Update a node in the list when the node is in the **Middle** of the List.

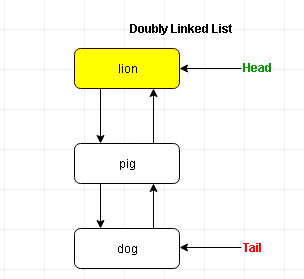
* We needs to update the value of the node first
* Make this node as the new Head of the List.

The Size limit of the cache is 3 and the list before adding node is:



“**tiger**” is the node for the **given KEY** and it is in the **middle** of the list, so we need to **update** its value and make it as the **NEW HEAD**.

After updating “tiger” to “lion”



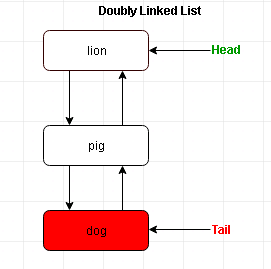
Now “**lion**” becomes the **head** of the list and it is the **most recently visited node**.

### Scenario 5 for AddOrUpdate’s Update:

Update a node in the list when the node is the **Tail** of the List.

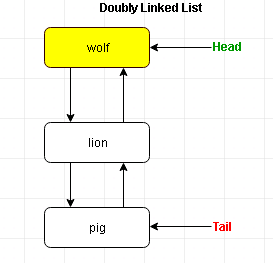
* We need to update the value of the node first
* Make this node as the **new Head** of the List.

The Size limit of the cache is 3 and the list before adding node is:



“**dog**” is the node for the given KEY and it is the **Tail** of the list, so we need to update its value and make it as the **NEW HEAD**.

After updating “dog” to “wolf”



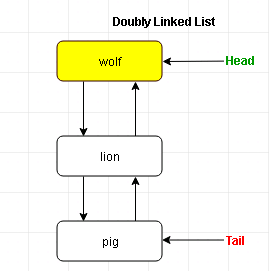
Now “**wolf**” becomes the **head** of the list and it is the **most recently visited** node and “**pig**” becomes the **Tail** of the List and it is the **least recently visited** node.

### Scenario 6 for TryGetValue:

Get a node in the list when the node is the **Head** of the List.

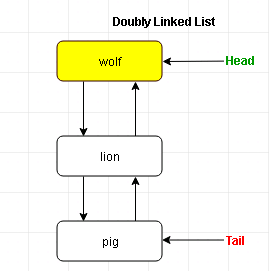
* We just need to return the value of the node.

The Size limit of the cache is 3 and the list before adding node is:



“**wolf**” is the node for the given KEY and it is the **head** of the list, so we just need to return its value.

After getting “wolf”

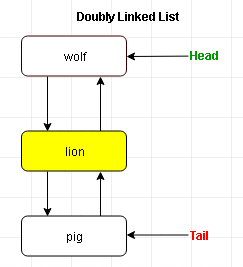


### Scenario 7 for TryGetValue:

Get a node in the list when the node is in the **Middle** of the List.

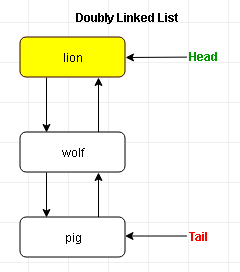
* We need to return the value of the node first
* Make this node as the new Head of the List.

The Size limit of the cache is 3 and the list before adding node is:



“**lion**” is the node for the given KEY and it is in the **middle** of the list, so we need to return its value and make it as the **NEW HEAD**.

After getting “lion”



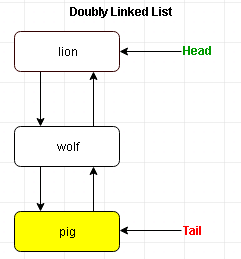
Now “**lion**” becomes the **head** of the list and it is the **most recently visited node**.

### Scenario 8 for TryGetValue:

Get a node in the list when the node is the **Tail** of the List.

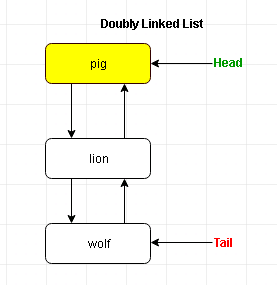
* We need to return the value of the node first
* Make this node as the new Head of the List.

The Size limit of the cache is 3 and the list before adding node is:



“**pig**” is the node for the given KEY and it is the **Tail** of the list, so we need to return its value and make it as the **NEW HEAD**.

After getting “pig”



Now “**pig**” becomes the **head** of the list and it is the **most recently visited** node and “**wolf**” becomes the **Tail** of the List and it is the **least recently visited** node.