Functionnalities

This list details the functionalities of the SnapshotModule

Name	Function	Description	Implemented [yes, no]	Complexity
Schedule snasphot (in the futur)	scheduleSnapshot	Schedule a snapshot occurring in time after other snapshots	Yes	O(1)
	scheduleSnapshotNotOptimized	Schedule a snapshot at a random place	Yes	O(N) Worst case: O(N) Best case: O(1)
Schedule snasphot (in the past)	-	Schedule a snapshot in the past	No	O(N) Worst case: O(N) Best case: O(1)
Reschedule a snapshot (in the future)	_rescheduleSnapshot	The new time is in the range between the previous snapshot and the next snapshot	Yes	O(1)
	-	The new time can be after or before another existent snapshot	No	O(N) Worst case: O(N) Best case: O(1)
Reschedule a snapshot (in the past)	-	The new time can be in the past	No	-
UnscheduleSnapshot	_unscheduleSnapshot	Unschedule the last snapshot	Yes	O(1)
	_unscheduleNotOptimized	Unschedule a random snapshot in the past	Yes	O(N) Worst case: O(N) Best case: O(1)
	_unscheduleNotOptimized	Unschedule a random snapshot in the future	Yes	O(N) Worst case: O(N) Best case: O(1) The number of elements to moved will be less that for the case where the snasphot is located in the past
Set the current snapshot	_setCurrentSnapshot			Same as _findScheduledMostRecentPastSnapshot
Update a struct Snapshot	_updateSnapshot	Inside a struct Snapshots: - Update the array ids to the current Snapshot time if this one is greater than the snapshot times stored in ids. - Update the value to	Yes	O(1)

		the corresponding value. */		
Update snapshots of the balance of an account	_updateAccountSnapshot	-	Yes	Same as _updateSnapshot
Update snapshots of the total Supply	_updateTotalSupplySnapshot	-	Yes	Same as _updateSnapshot
Get the last snapshot time inside a snapshot ids array	_lastSnapshot	-	Yes	O(1)
Find a snapshot	_findScheduledSnapshotIndex	Find the snapshot index at the specified time	Yes	O(log2(N)) We use a binary search to find the value at the specified time
Find the mot recent past snapshot	_findScheduledMostRecentPastSnapshot	-	Yes	O(1) Worst case: O(N) Best case: O(1) We only have a O(N) complexity if all next scheduled snapshot are situated in the past but no update of the actual snapshot was made.
Update balance and/or total supply snapshots before the values are modified	_beforeTokenTransfer	Call before each transfer. It is very important to have a low complexity because this function is called very often.	Yes	The complexity depends of th functions _setCurrentSnapshot _updateAccountSnapshot _updateTotalSupplySnapshot
Get the next scheduled snapshotd	getNextSnapshots	-	Yes	O(N) Nevertheless, we maintain a pointer on the actual snapshot to avoid loop through past snapshot
Get all snapshot	getAllSnapshots	s-	Yes	O(1) We directly return the array
Get the balance of an owner st the time specified	snapshotBalanceOf	Return the number of tokens owned by the given owner at the time when the snapshot with the given time was created.	Yes	O(log2(N)) We use a binary search to find the value at the snapshot time
Get the totalSupply at the time specified	snapshotTotalSupply	Retrieves the total supply at the specified time.	Yes	O(log2(N)) We use a binary search to find the value at the snapshot time