We know that all the objects that we create dynamically are allocated in the application's heap memory. Normally, programmers are responsible for both creating and deleting objects in the program, but programmers often ignore object deletion. This creates OutOfMemoryErrors problem due to insufficient memory for not deleting unwanted objects.

Therefore, we don't have to worry about this problem of freeing the memory of these unused or unwanted objects, because the system always runs in the background and its main purpose is to free heap memory by deleting unreachable objects.

Basically, Garbage Collection is the process of keeping track of all the objects that are still in use and marking the rest of them as garbage. Garbage collection in this example is considered an automatic memory management scheme because the programmer does not have to explicitly locate objects. Garbage collection runs on low priority threads.

The Mark and Scan Algorithm is a basic and original algorithm for Garbage Collection. This algorithm basically performs two main functions: marking and scanning. First, it will monitor and detect inaccessible objects and second, it will release these objects from the heap memory so that the programmer can use them again.

1. walk\_region\_and\_mark - Marking Living Objects

This is the first phase of the algorithm, which detects all objects that are still alive. This is the stage where the Garbage Collection determines which parts of memory are being used and which are not.

In this phase when the condition is fulfilled, its check bit decreasing by 8 until 0 or NULL. We set the ticked for all accessible objects.

Here we can treat each object as a void node and then we access all the objects or nodes that are accessible from this object/node and it repeats until we have reached All region are accessible.

The root is a variable that references a “head” and is directly accessible by a local variable. We'll assume we only have one root to traversing. We can use “ptr” to access the mark bit for an object.

2. sweep - Remove dead objects

The scan phase algorithm "deletes" all inaccessible or unreachable objects, it frees up the stored memory area for all inaccessible objects. Each item with the check value set to free is removed from the stack memory, for every other accessible object we set the value of the is\_marked( bit) to false and clear it.