#ASLI ENGINEERING

Shades of Grey in Garbage Collector

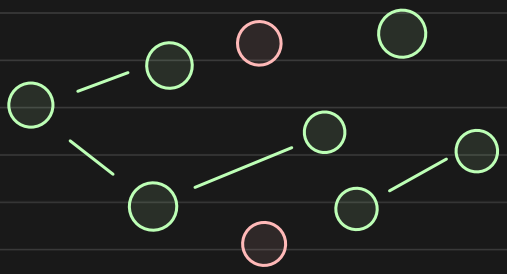
SWIPE

BY

ARPIT BHAYANI

Shades of Grey in Garbage Collection

Mark and Sweep Garbage Collection



Mark: Start from root nodes and continue to mark the reachable objects

whatever's left is dead/garbage

Sweep: iterate to all objects in the heap and clean what's garbage

Each object is in one of the three states

- **live:** object is live and is in use
- **unprocessed:** object is not yet processed
- **processing:** object is seen but its children is yet to be processed

We know **Dijkastra** for his 'shortest path' algorithm, but in 1976/78 he proposed a color abstraction for tracing GC

* Most modern Garbage collectors are concurrent to speed up cleaning, and the Tricolour Abstraction comes in super handy to ensure

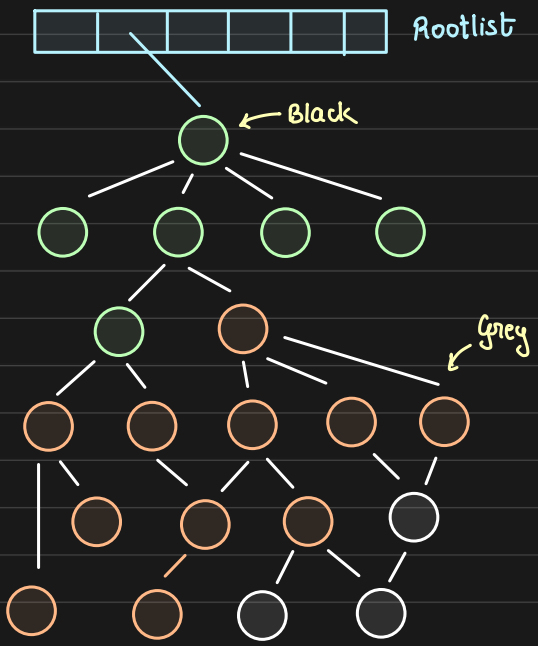
concurrency, smaller pause times, and correctness

Tricolour Abstraction

The object reference graph is partitioned into

BLACK	⟶	object is LIVE
GREY	⟶	object is seen but not yet processed completely
WHITE	⟶	object is not yet seen

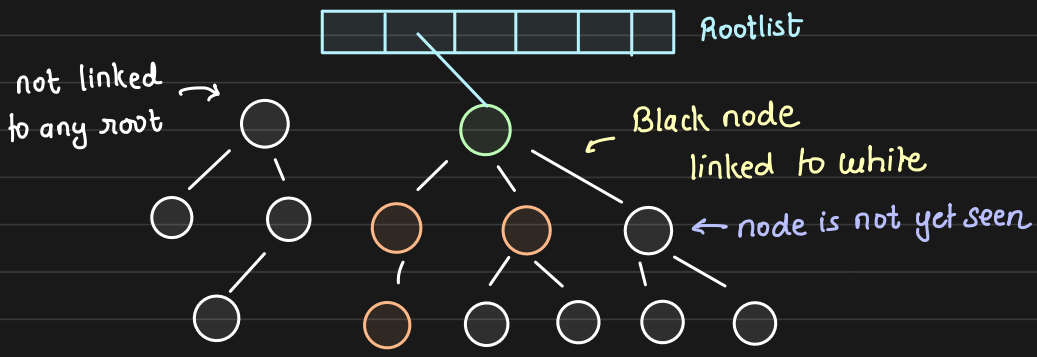
Initially every node is WHITE, when a node is first encountered during tracing it is coloured GREY, when it has been scanned and its children identified it is coloured BLACK



If we visualize tracing based GC we can see it as a wave of Grey nodes moving in the object graph separating the blacks with the whites, untill all nodes turn black

How does tricolour abstraction make thing better?

collector **must** never reclaim the storage of live objects.



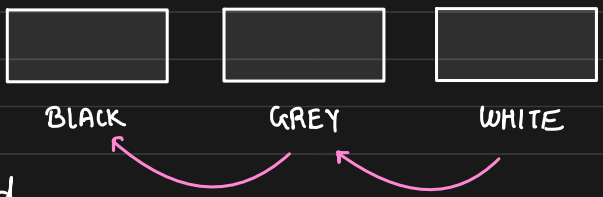
WHITE SET : candidate nodes to be garbage collected

BLACK SET : nodes that have no outgoing edge to any WHITE node
nodes that are definitely live so no cleaning them up

GREY SET : nodes reachable from the root but yet to be scanned for its children. nodes are reachable from the root and eventually will be in the black set

What's left in WHITE is garbage collected

Garbage Collection Flow



Pick object from Grey set and move it to Black set.

Colour each white node referenced as Grey and repeat the process.

Stack of our DFS

Scan is complek when the Grey set is empty

Black objecks are reachable from root

Whik objeks are not and hence cleaned up

Because the objecks are always moved from Whik to Grey & Grey to Black the algorithm preserves an important properly

No black object references white ← Ensures that the white set can be freed once the grey set is emply.

This is called the Tricolour Invariant

But, why did we do this?

Single pass DFS is not fast, but by seggregating it across 3 sets, we can reduce the pause time by putting a lot of threads at work on grey set

Hence, the system is halted for a very short time.

Through this, we make our system reactive

↳ instead of periodically running GC, we can monitor the sets and if items > some threshold we can run a quick cleanup

This method is to collect garbage is "on-the-fly" → Dijkstra, lampart and 3 others

↳ the colour of the nodes are changed by Mutator & not collector and hence the entire GC can be made reactive instead of periodic

and it lays foundation for concurrent GC

where mutator threads runs concurrently with the collector threads.