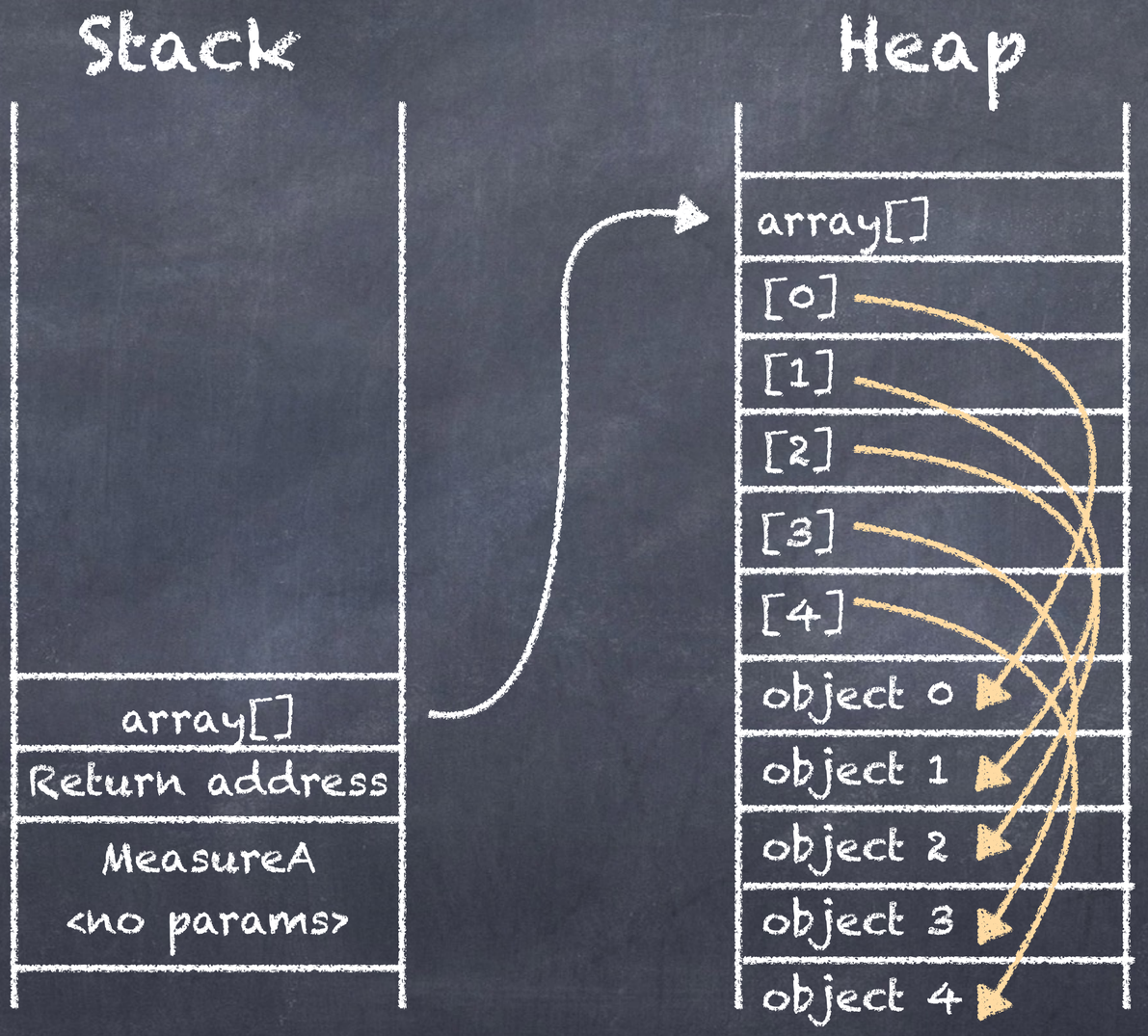
Fast garbage collection
part 1

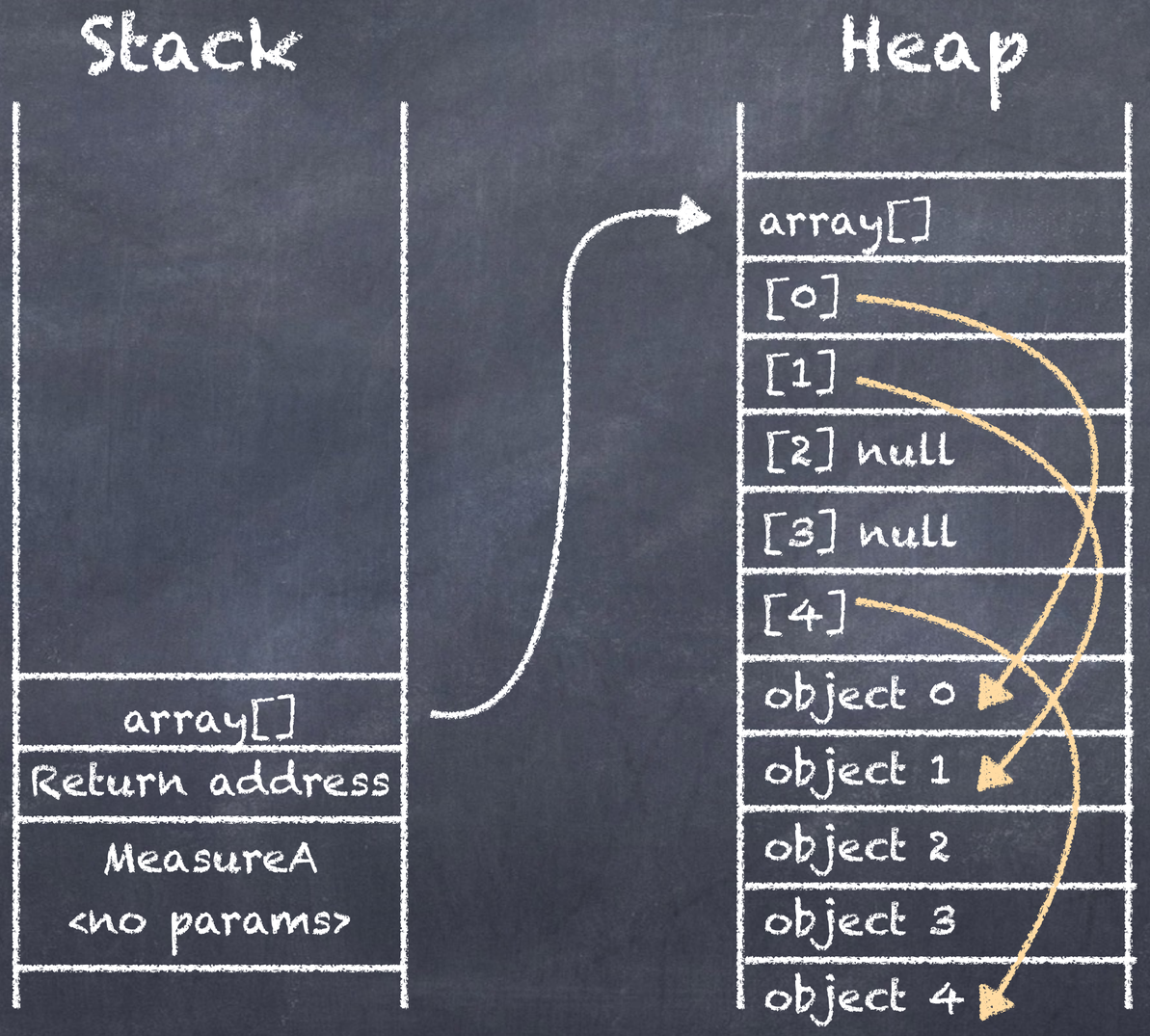
Initial memory layout:

1 object array with 5 elements



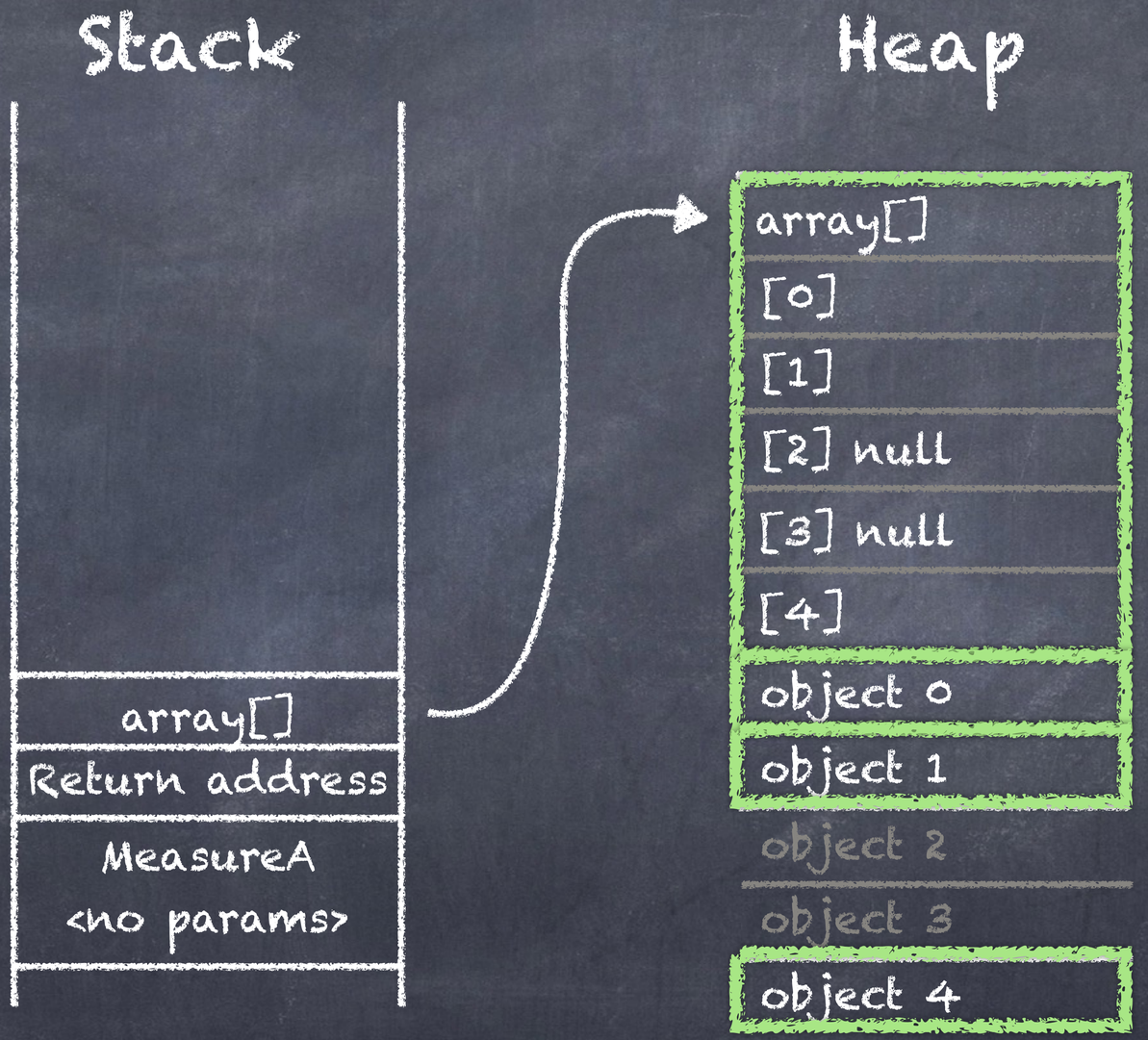
Now we set array elements 2 and 3 to null.

Objects 2 and 3 are dereferenced



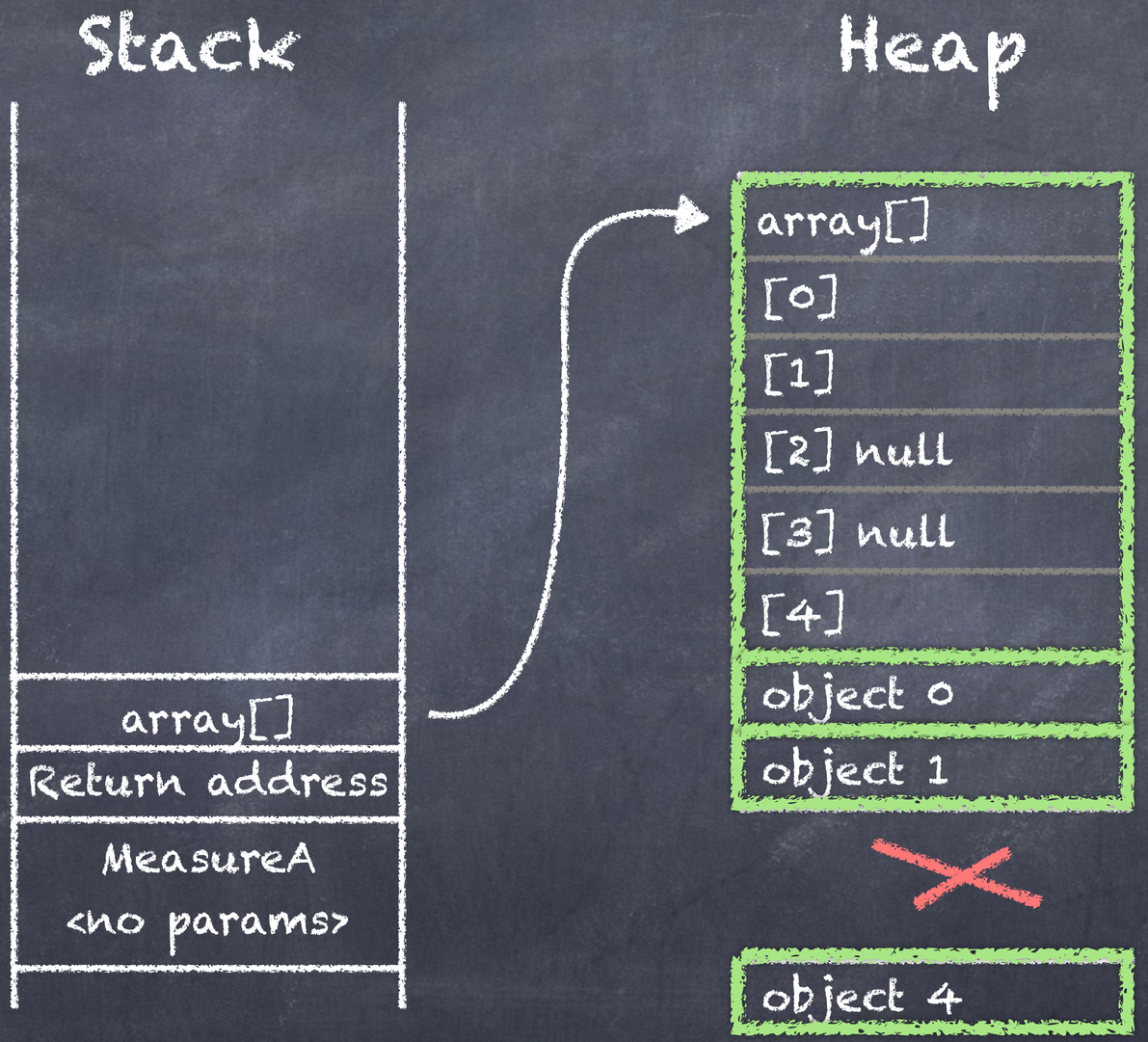
Garbage collection runs:

All live objects are marked



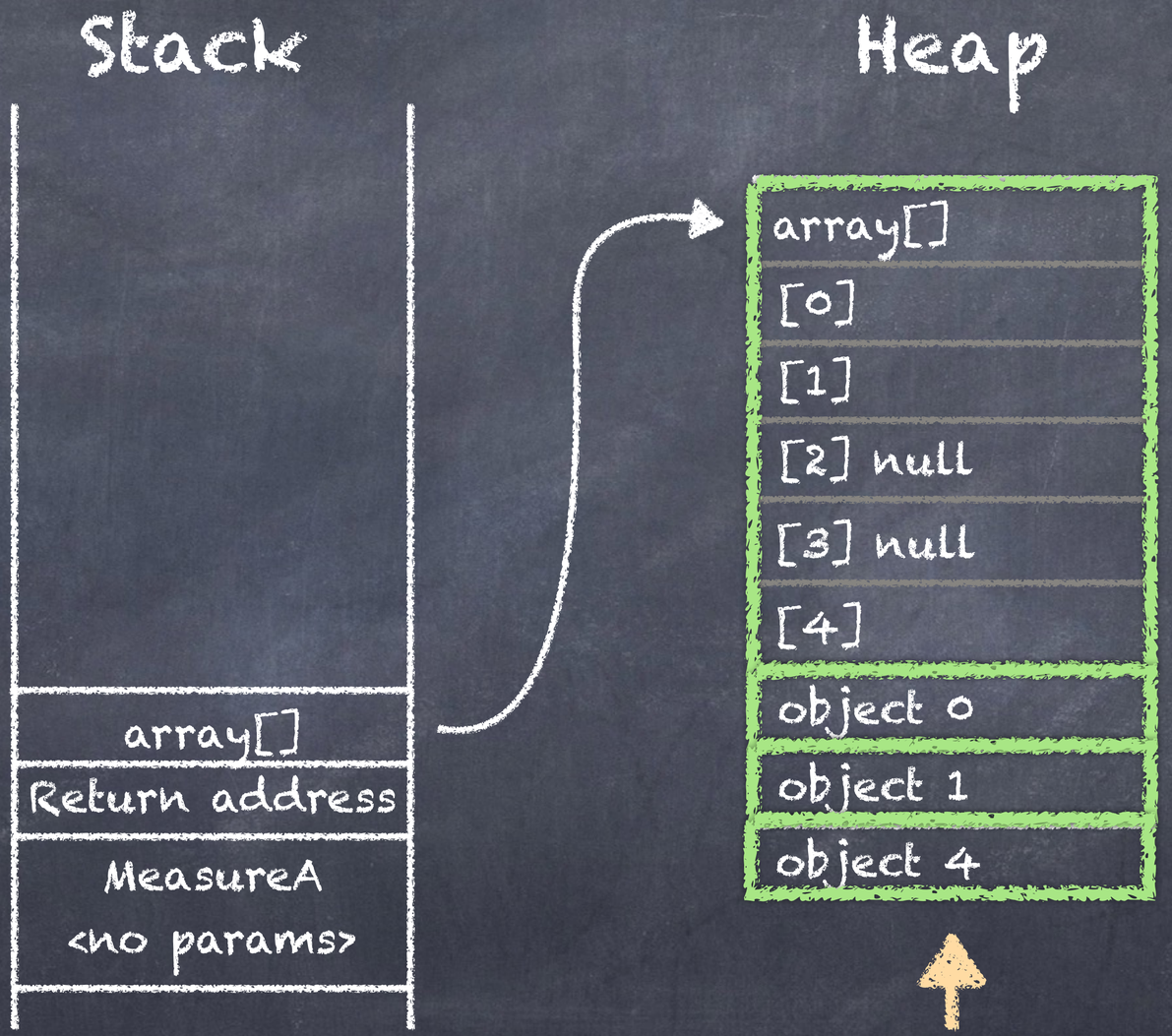
Garbage collection runs:

All dead objects are swept



Garbage collection runs:

The heap is compacted



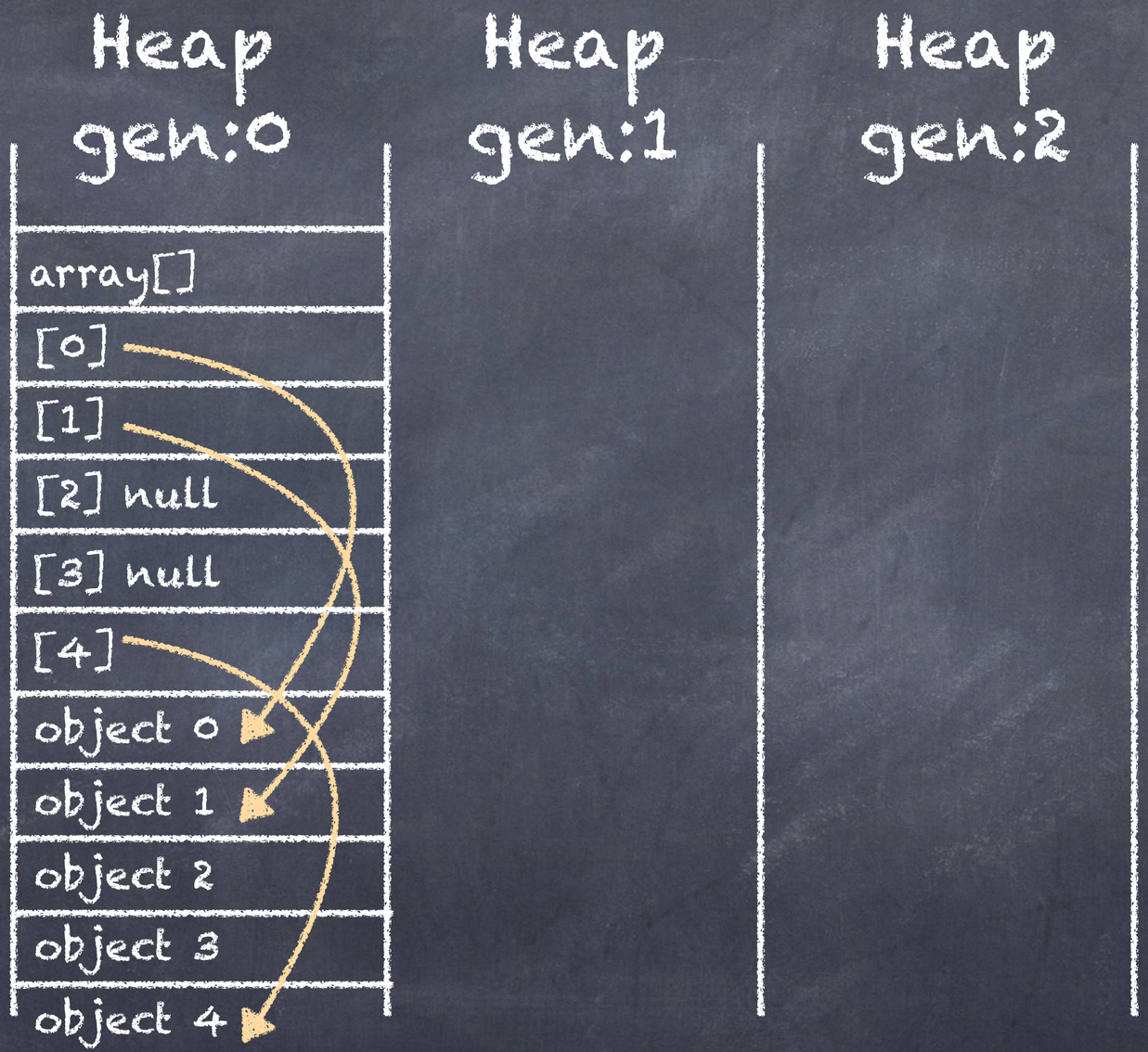
First problem

- A mark and sweep Garbage Collector has to mark all live objects on the heap during each collection cycle.
- This can lead to long program freezes while the collection is running
- Also: repeatedly marking the same objects over and over in each cycle is very inefficient

Solution: Generational Garbage Collector

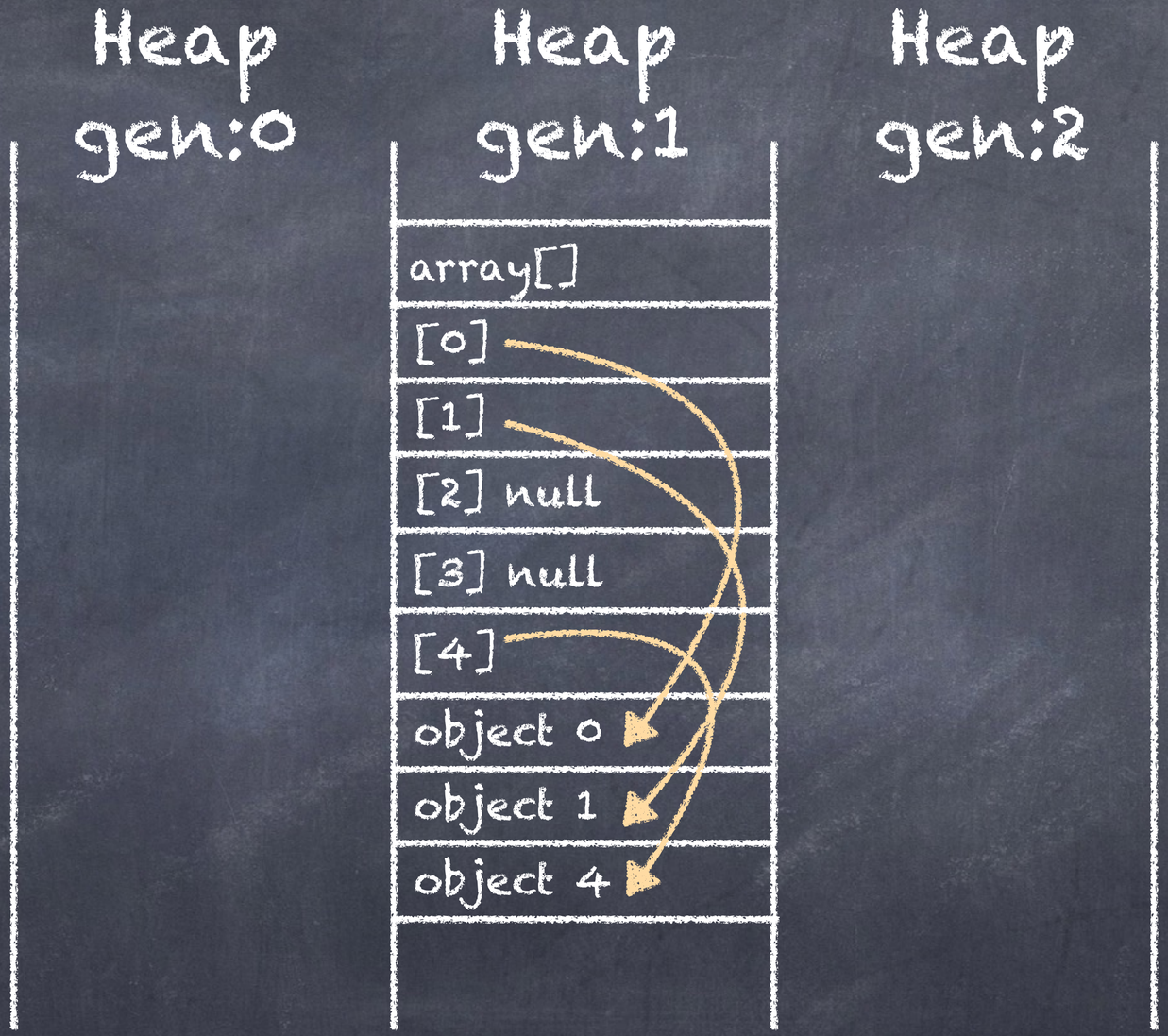
Initial memory layout:

All objects are in gen 0

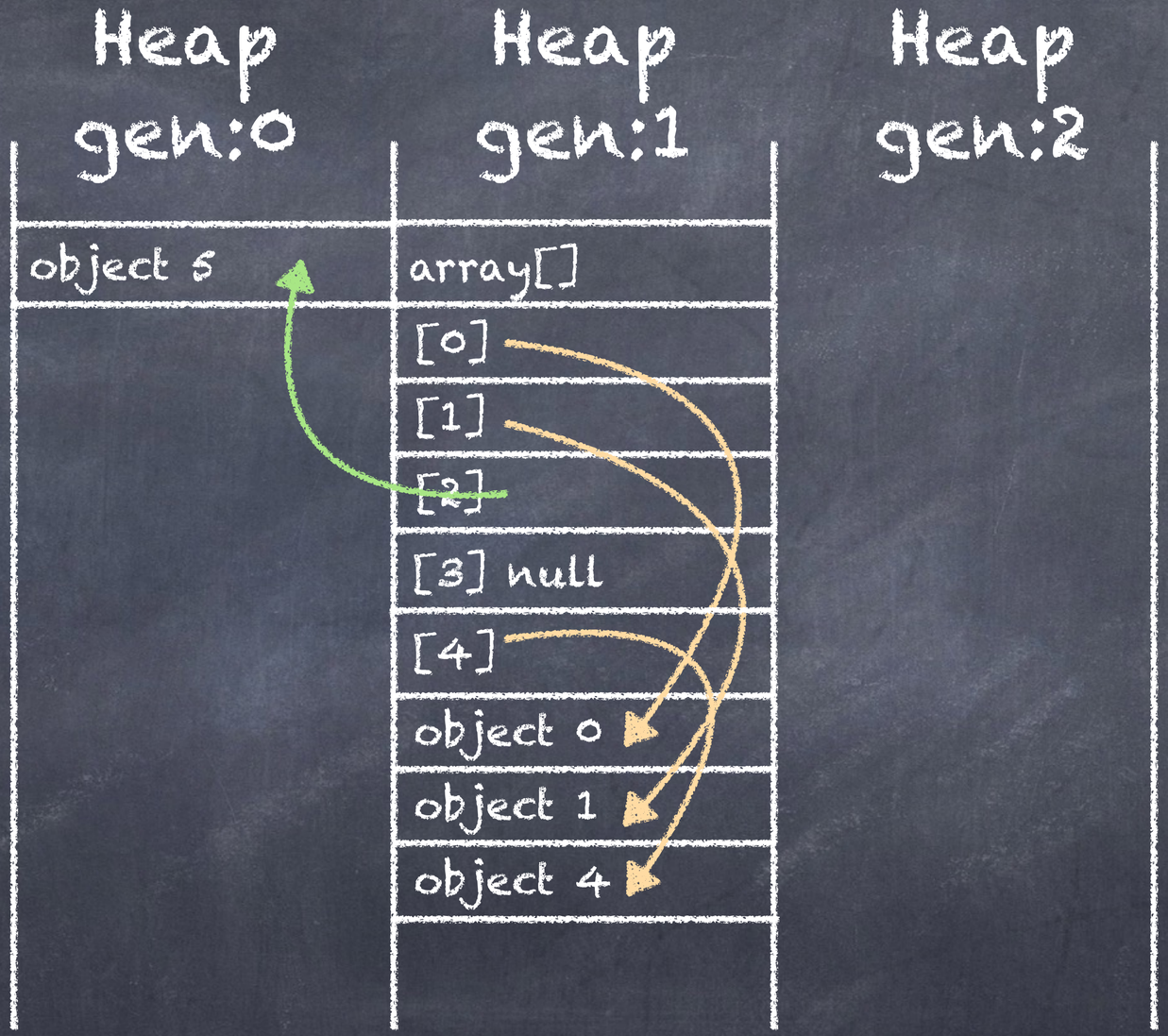


After the first collection cycle:

All remaining live objects move to gen 1



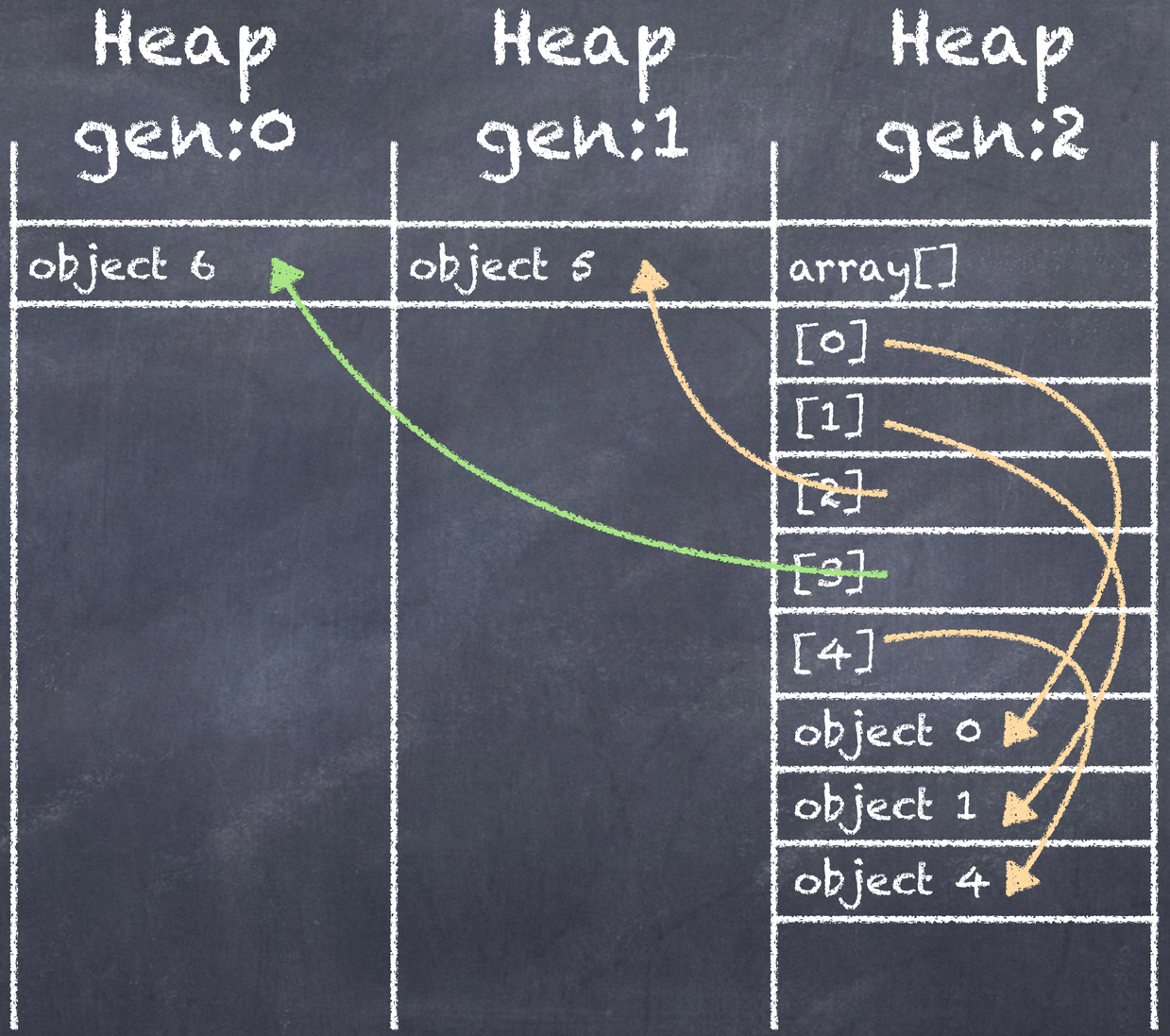
New allocations go to gen 0



After the second collection cycle:

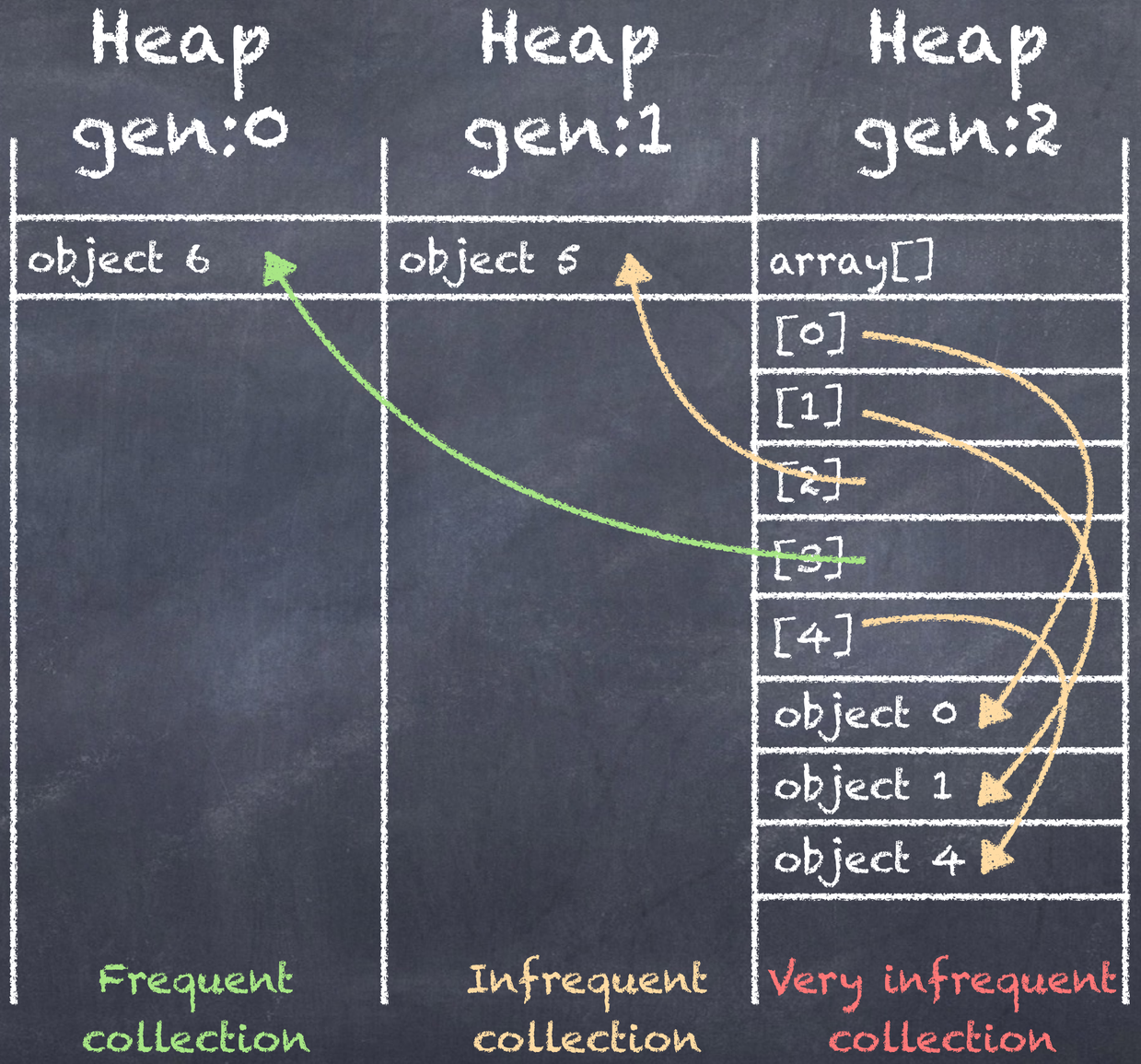
Gen 1 moves to gen 2 and gen 0 moves to gen 1

New allocations go to gen 0



Generations help to reduce the number of objects in gen 0

Gen 1 and 2 are collected much less frequently



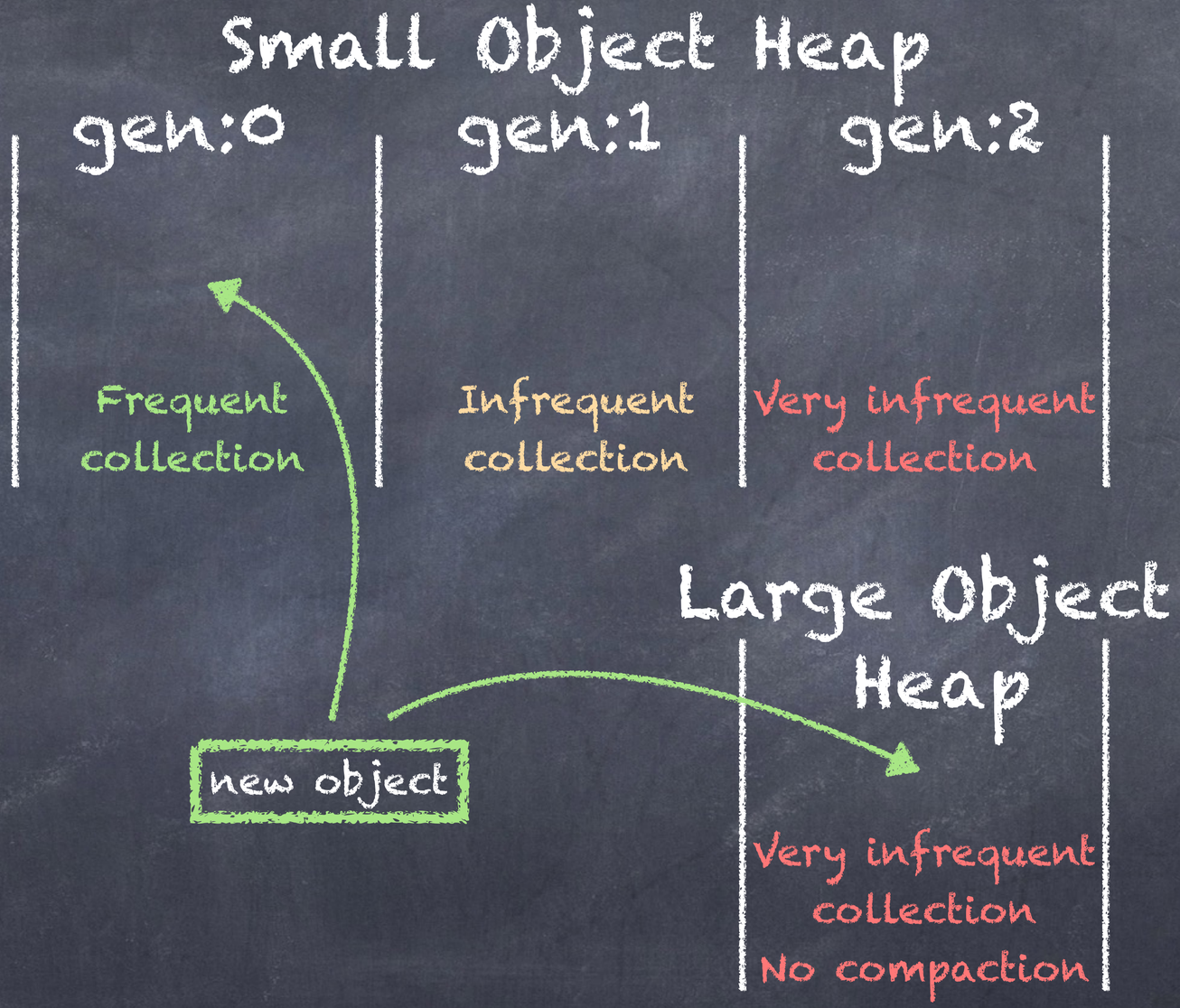
Second problem

- During 2 successive garbage collection cycles a long-living object will be compacted twice and moved twice:
  4 memcopy operations for a single object
- For long-lived very large objects these 4 memory copy operations impact performance

Solution: Small Object and Large Object heaps

Objects larger than 85k are allocated on the Large Object Heap

The LOH is collected in gen 2 and not compacted



Implicit assumptions

- Objects are either short-lived or long-lived
- Short-lived objects will be allocated and discarded within a single collection cycle
- Objects that survive two collection cycles are long-lived
- 90% of all small objects are short-lived
- All large objects (85K+) are long-lived

- Do not go against these assumptions

What have we learned

- The Garbage Collector uses a mark/sweep/compact cycle
- Small objects are allocated on the Small Object Heap (SOH), Large objects on the Large Object Heap (LOH)
- The Small Object Heap (SOH) has 3 generations. Objects are allocated in gen 0 and progress towards gen 2.
- The LOH has only one gen and does not compact

- .NET assumes 90% of all small objects are short-lived
- .NET assumes all large objects are long-lived