JavaOne℠

The Ghost in the Virtual Machine
A Reference to References

Bob Lee
Google Inc.

Goals

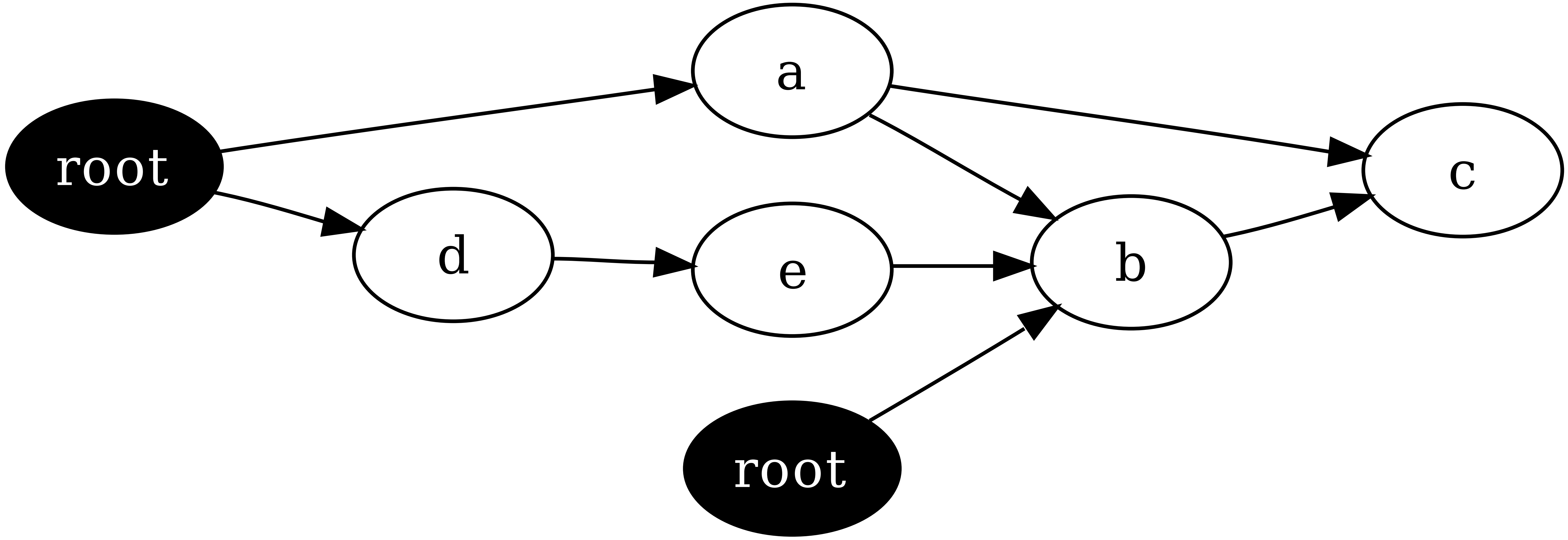
> Perform manual cleanup the right way.

> Take the mystery out of garbage collection.

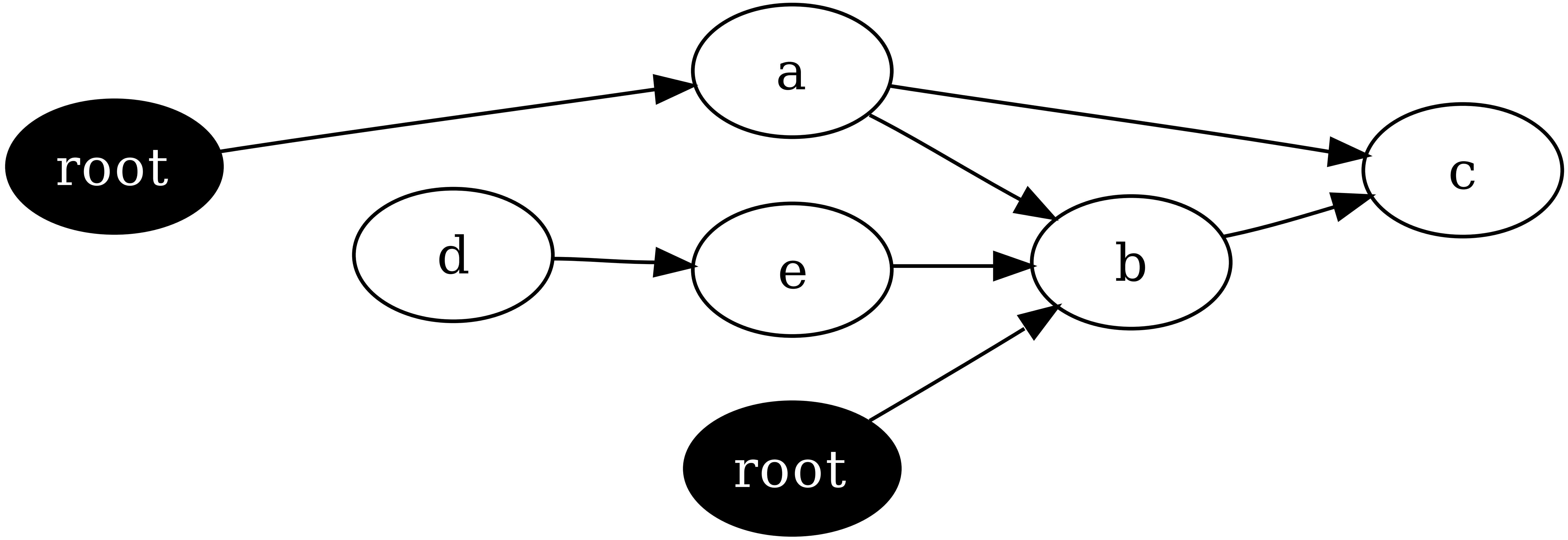
> Learn enough about references to implement our own garbage collectors.

> Become honorary VM sanitation engineers.

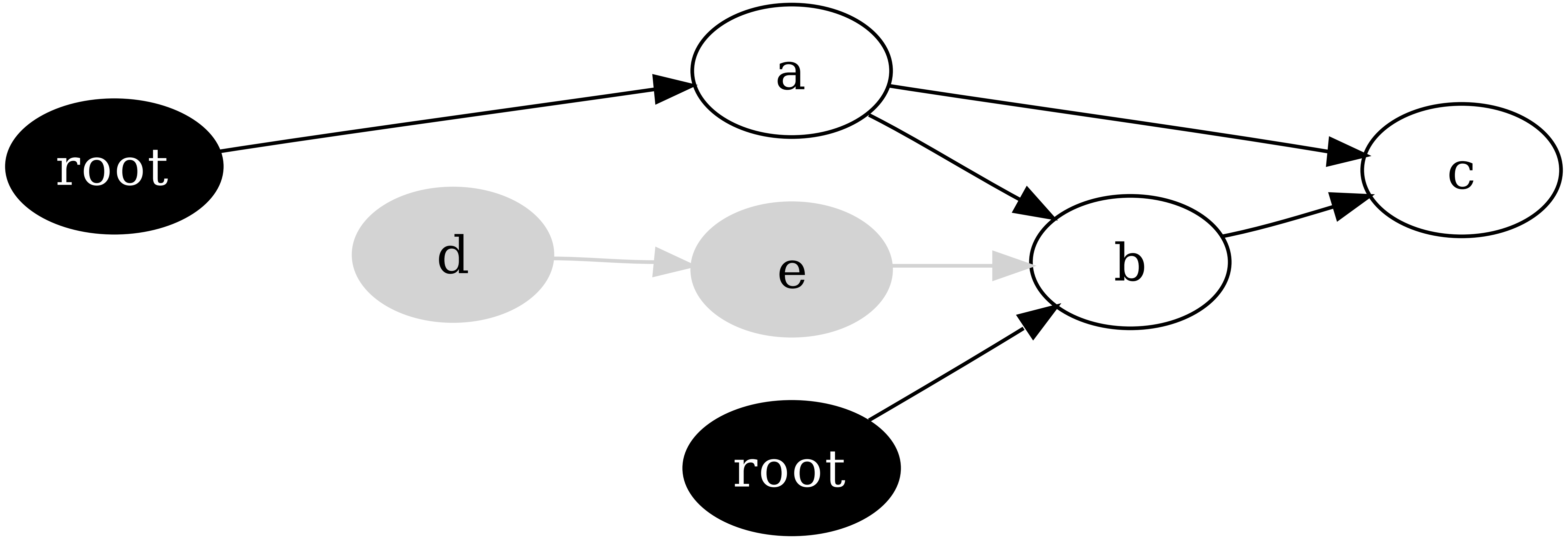
How does garbage collection work?



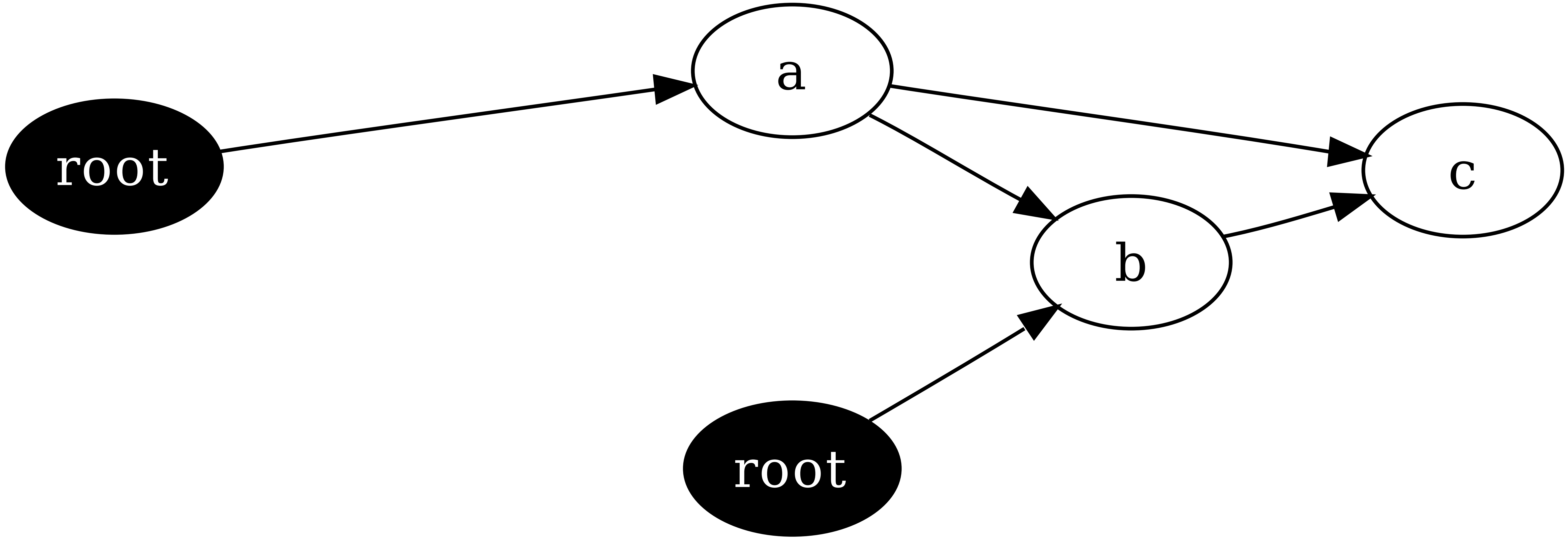
If the reference to D goes away...



We can no longer reach D or E.



So the collector reclaims them.



Reachability

> An object is *reachable* if a live thread can access it.

> Examples of heap roots:

- System classes (which have static state)
- Thread stacks
- In-flight exceptions
- JNI global references
- The finalizer queue
- Interned strings
- etc. (VM-dependent)

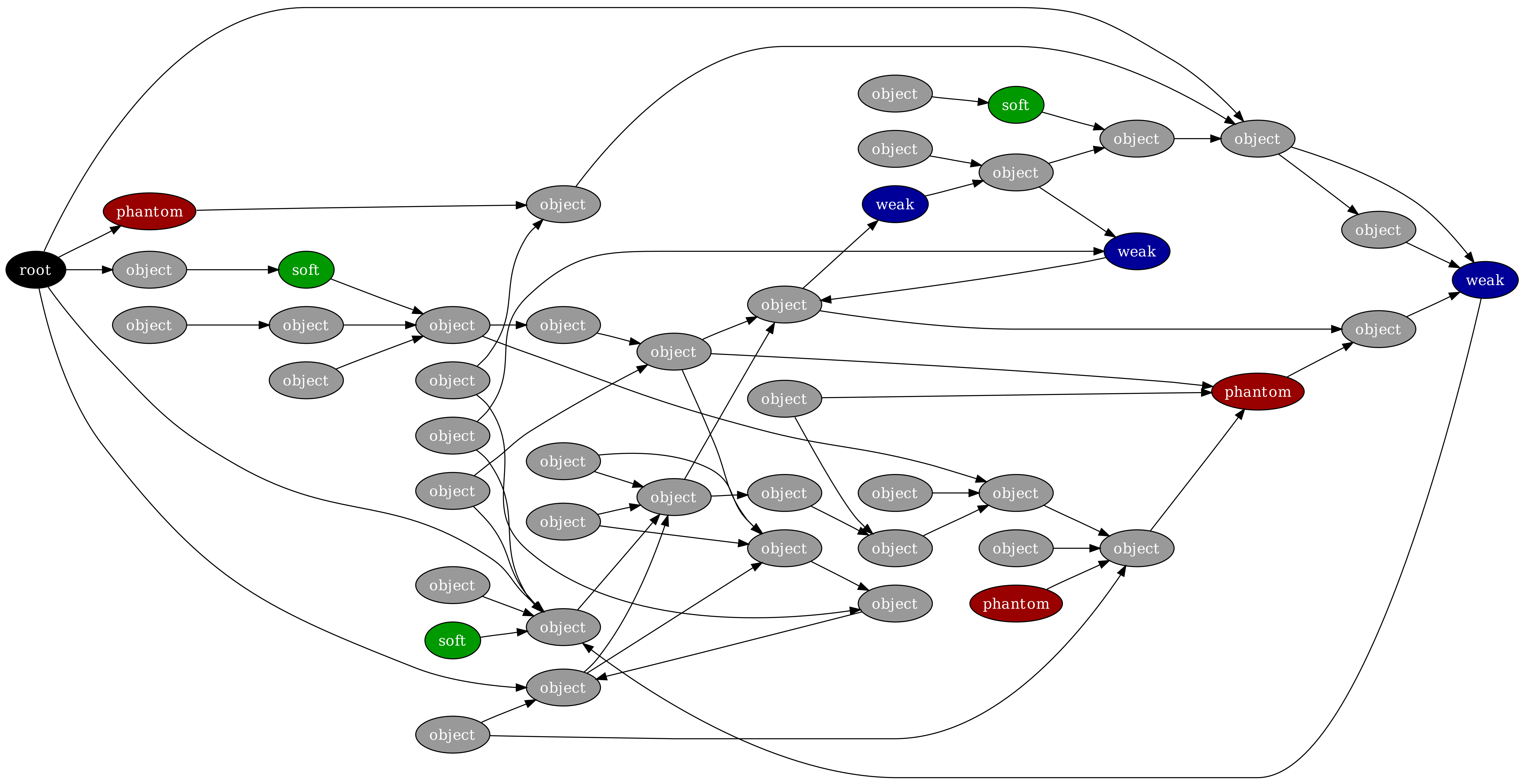
In the beginning, there was the finalizer...

```
public class Foo extends Bar {
  @Override protected void finalize() throws Throwable {
    try {
      ... // Clean up Foo.
    } finally {
      super.finalize(); // Clean up Bar.
    }
  }
}
```

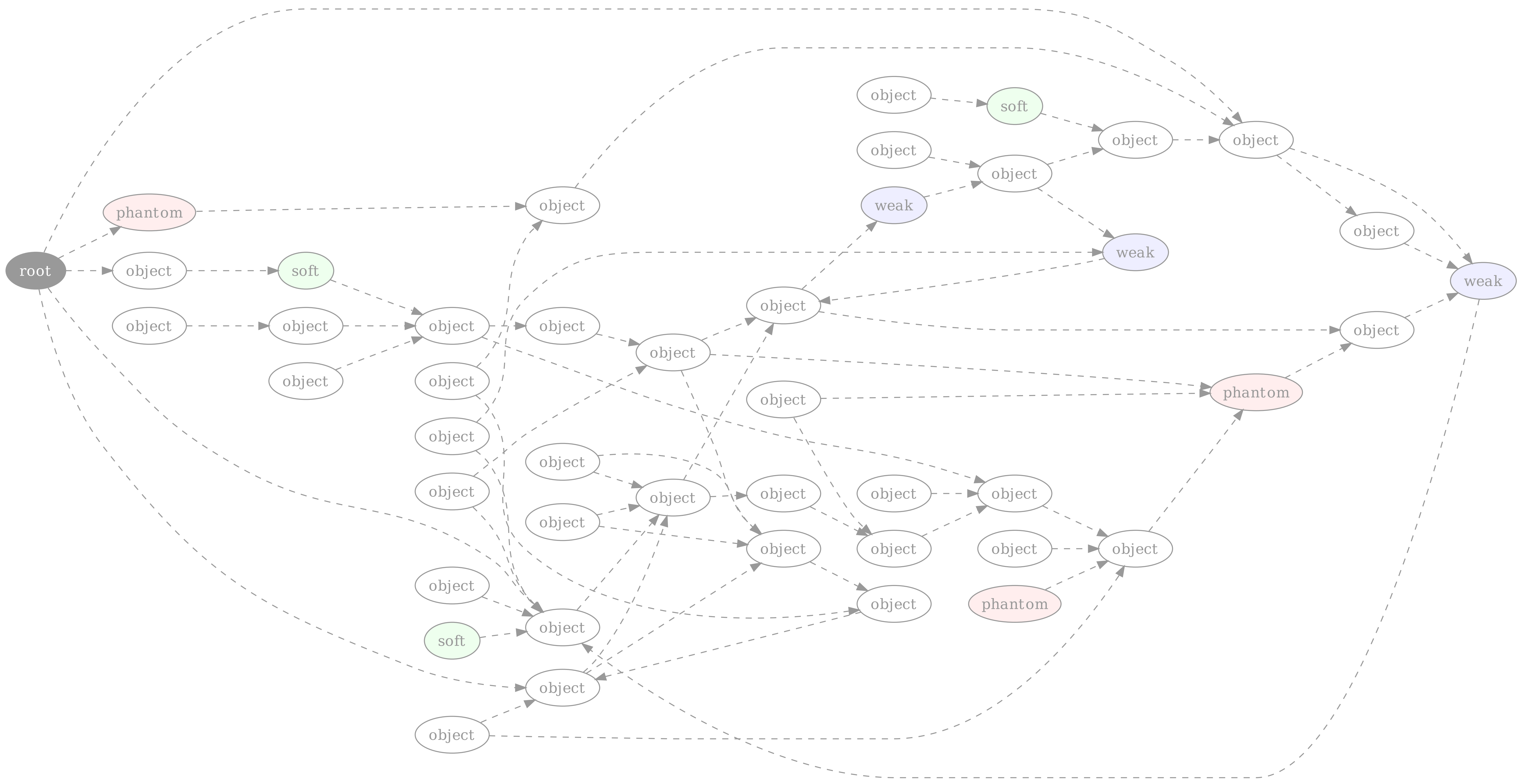
In the beginning, there was the finalizer...

```
public class Foo extends Bar {
  @Override protected void finalize() throws Throwable {
    try {
      ... // Clean up Foo.
    } finally {
      super.finalize(); // Clean up Bar.
    }
  }
}
```

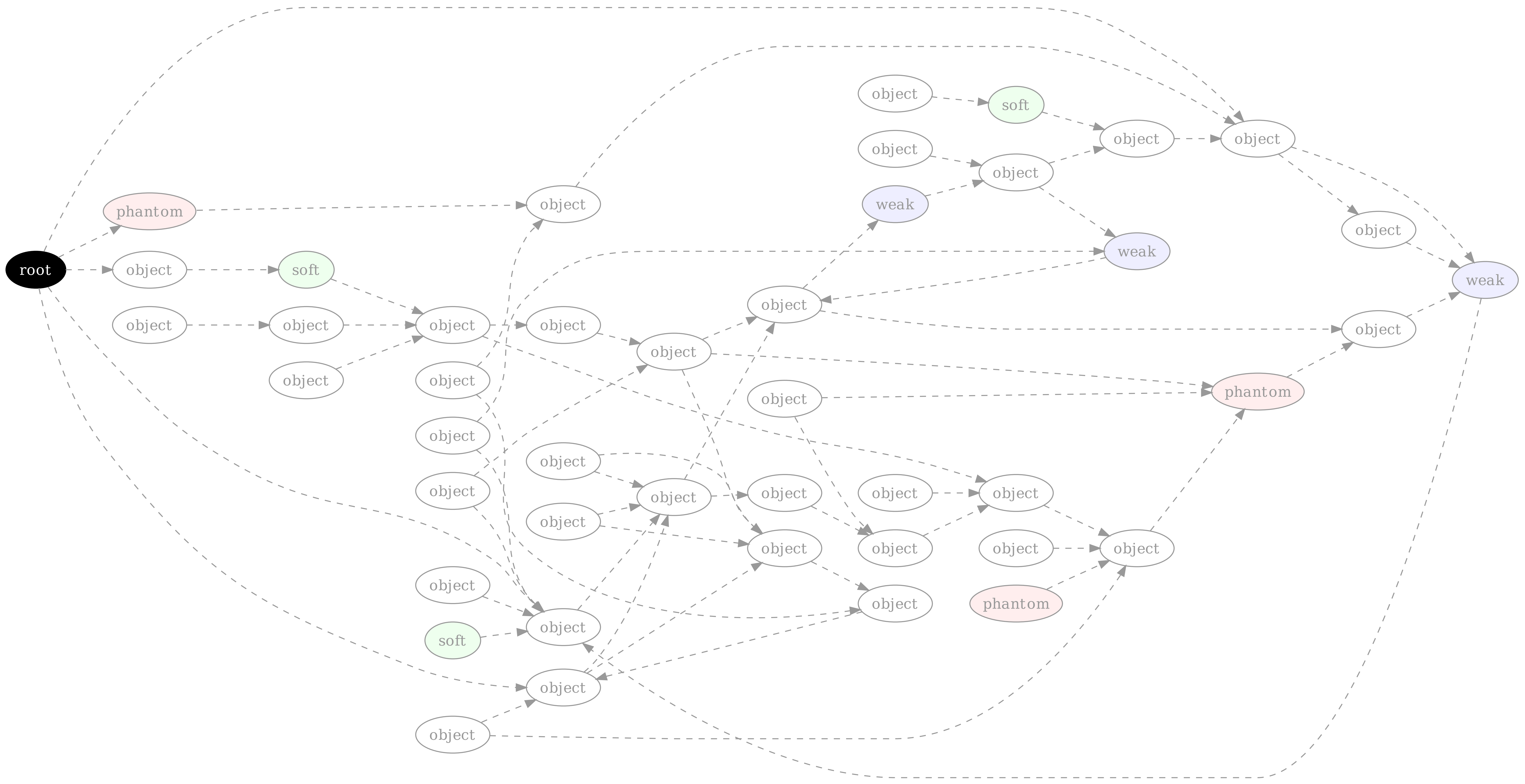
Let's mark and sweep a heap!



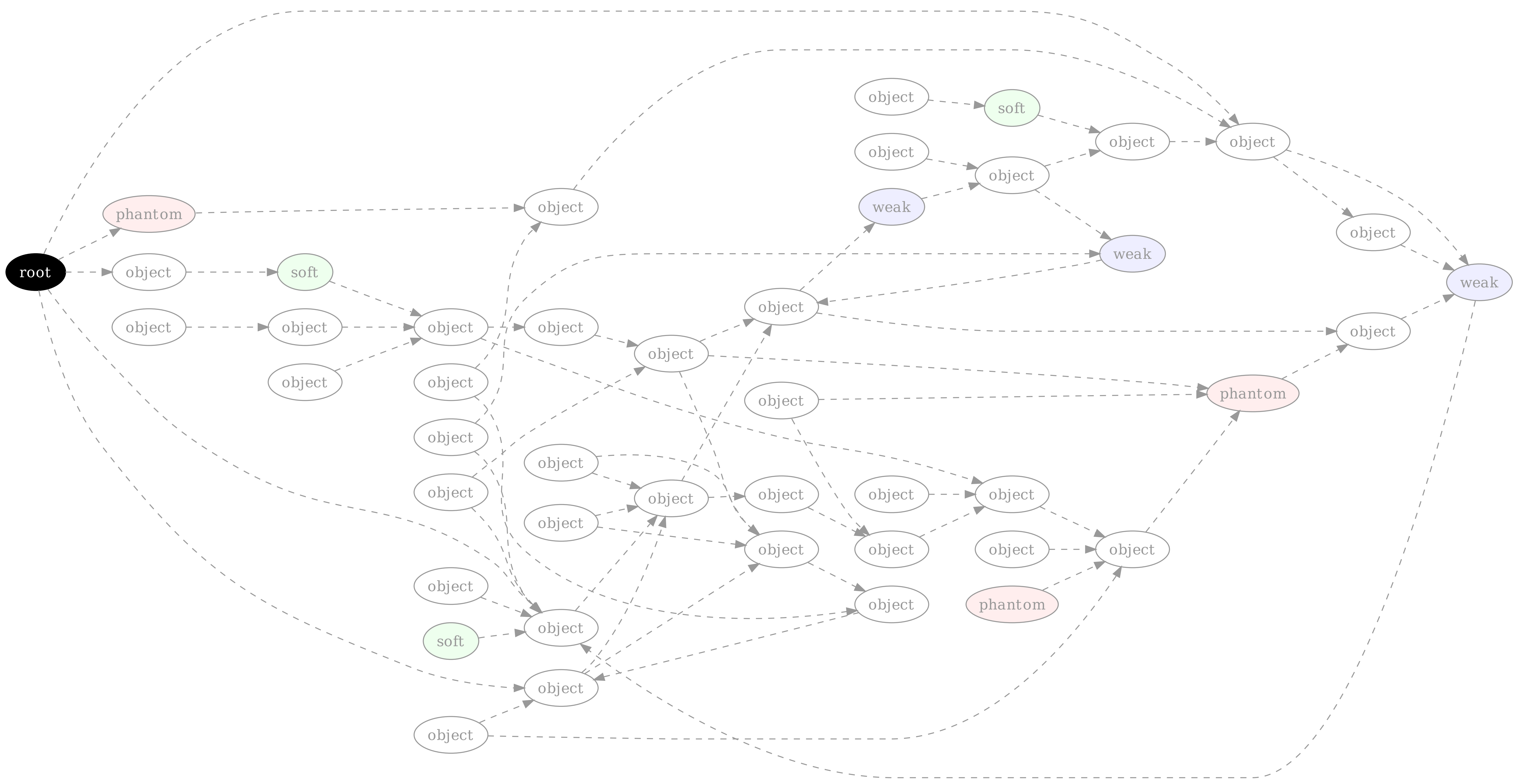
No objects are marked at first.



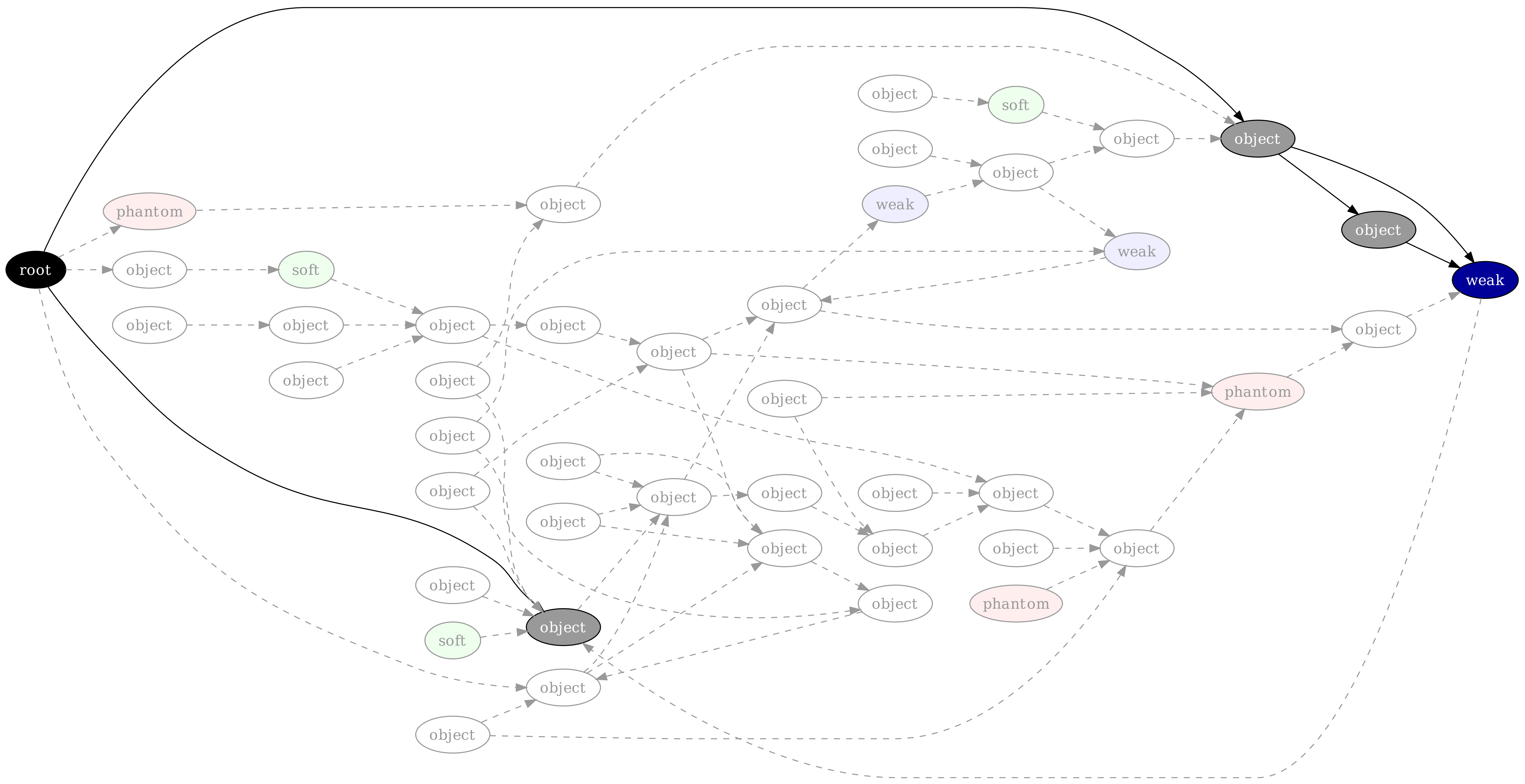
1. Start at a root.



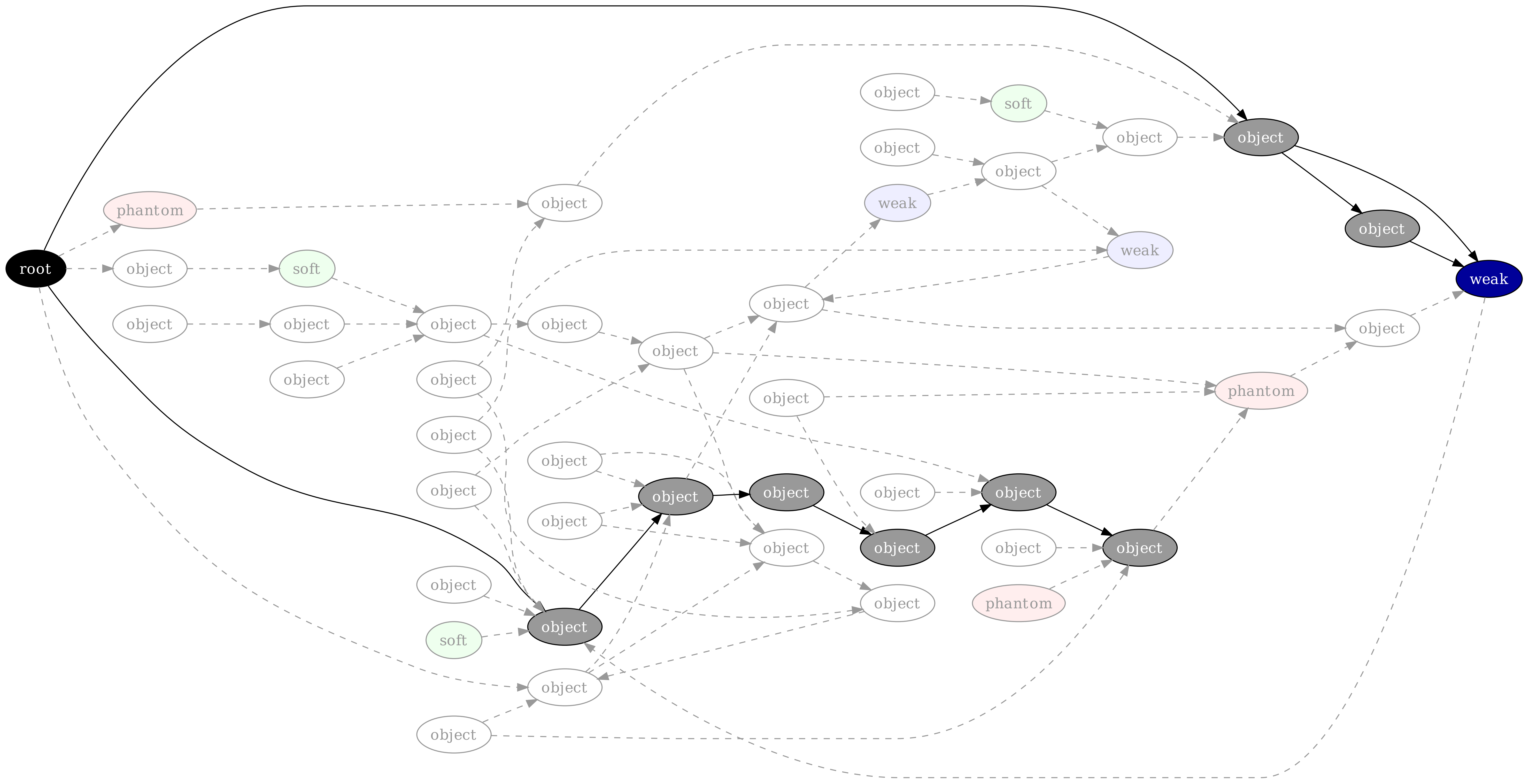
2. Trace and mark strongly-referenced objects.



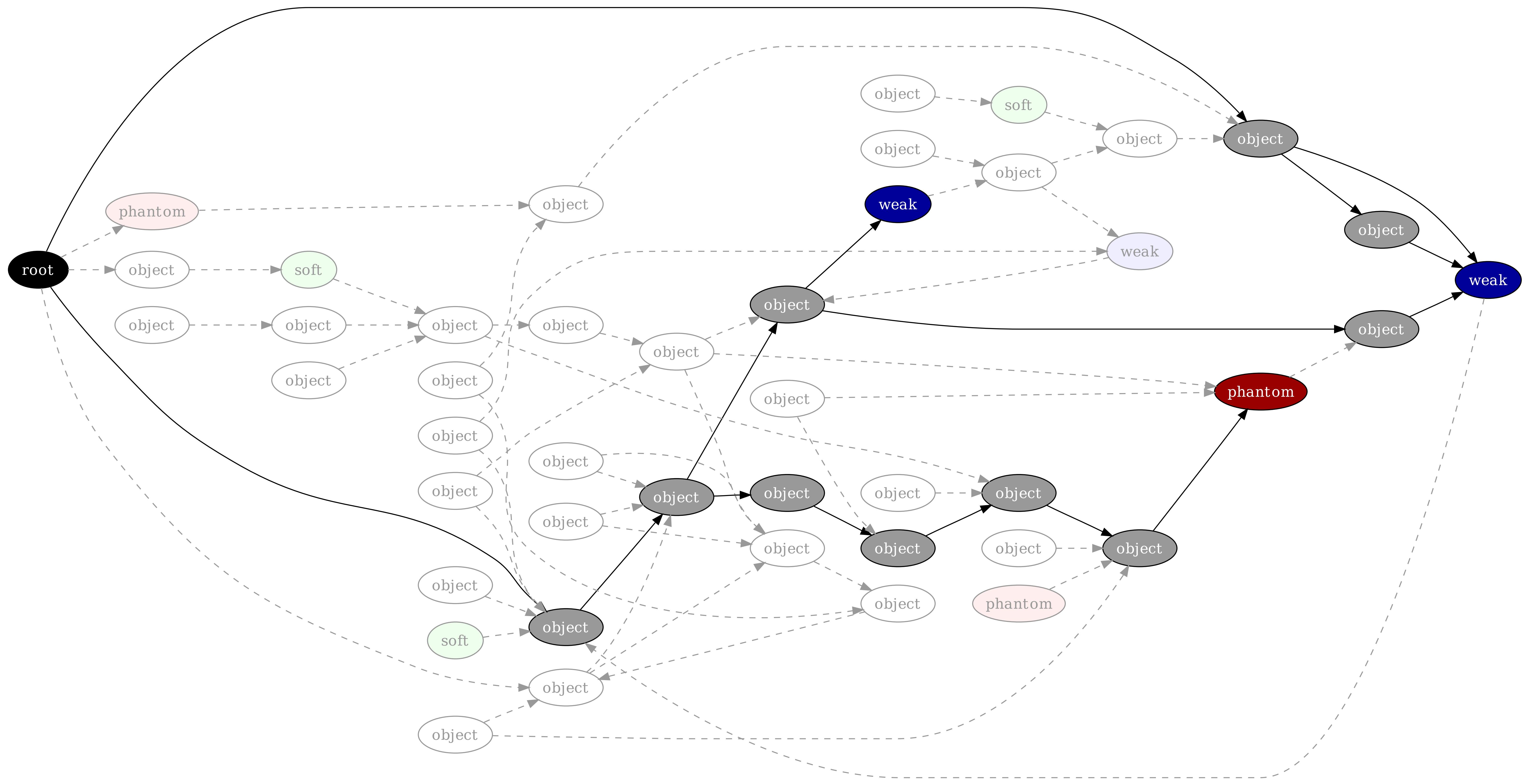
2. Trace and mark strongly-referenced objects.



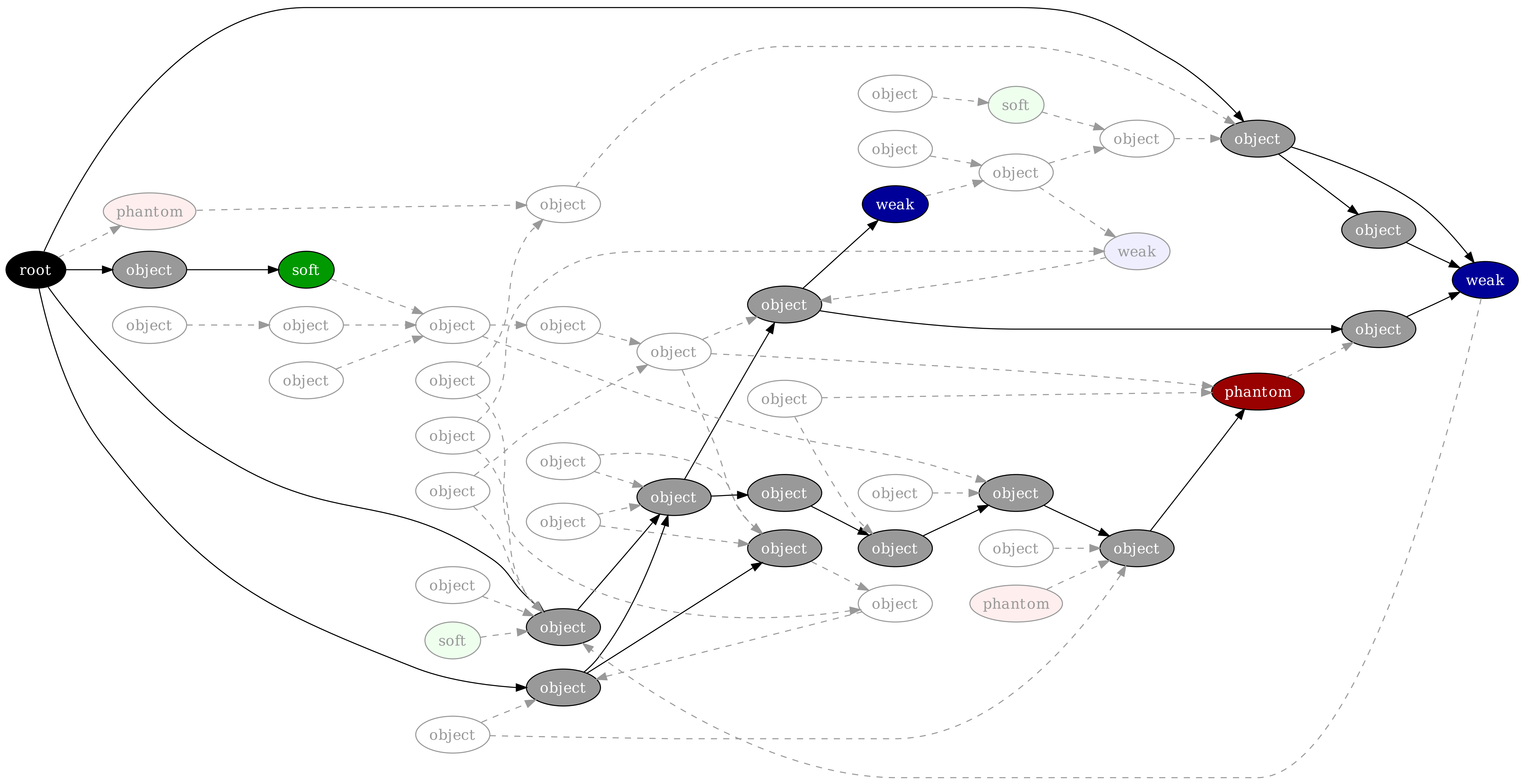
2. Trace and mark strongly-referenced objects.



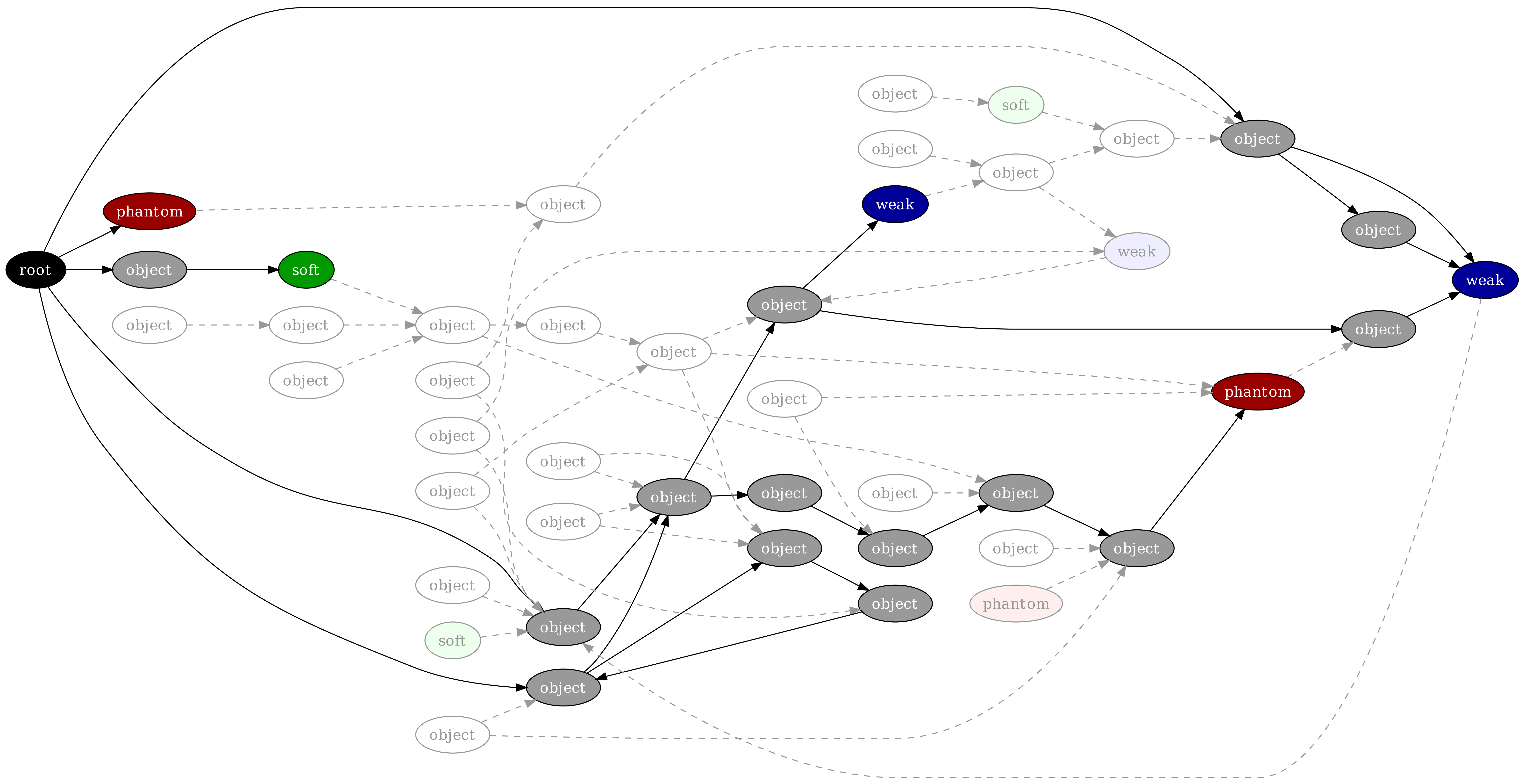
2. Trace and mark strongly-referenced objects.



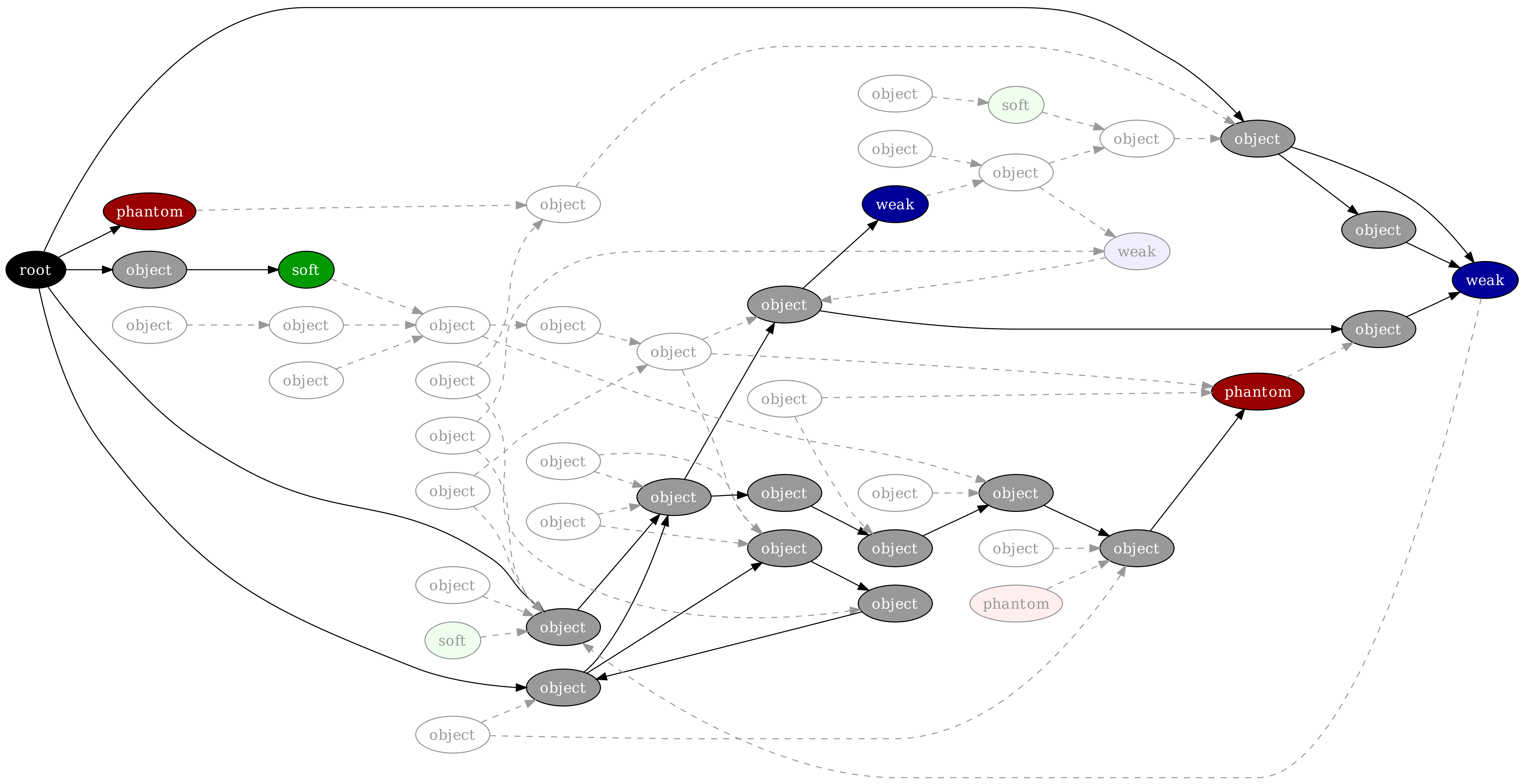
2. Trace and mark strongly-referenced objects.



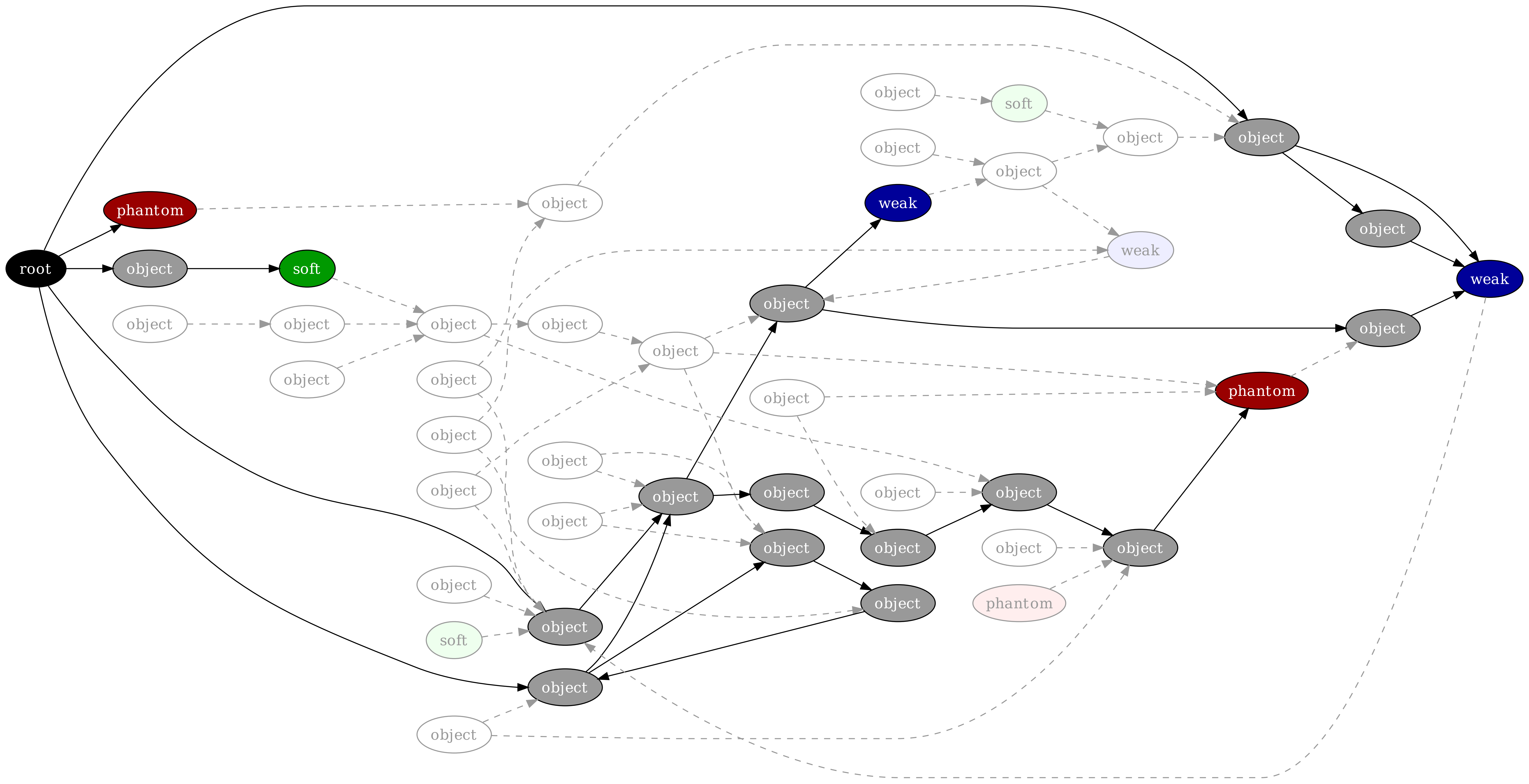
2. Trace and mark strongly-referenced objects.



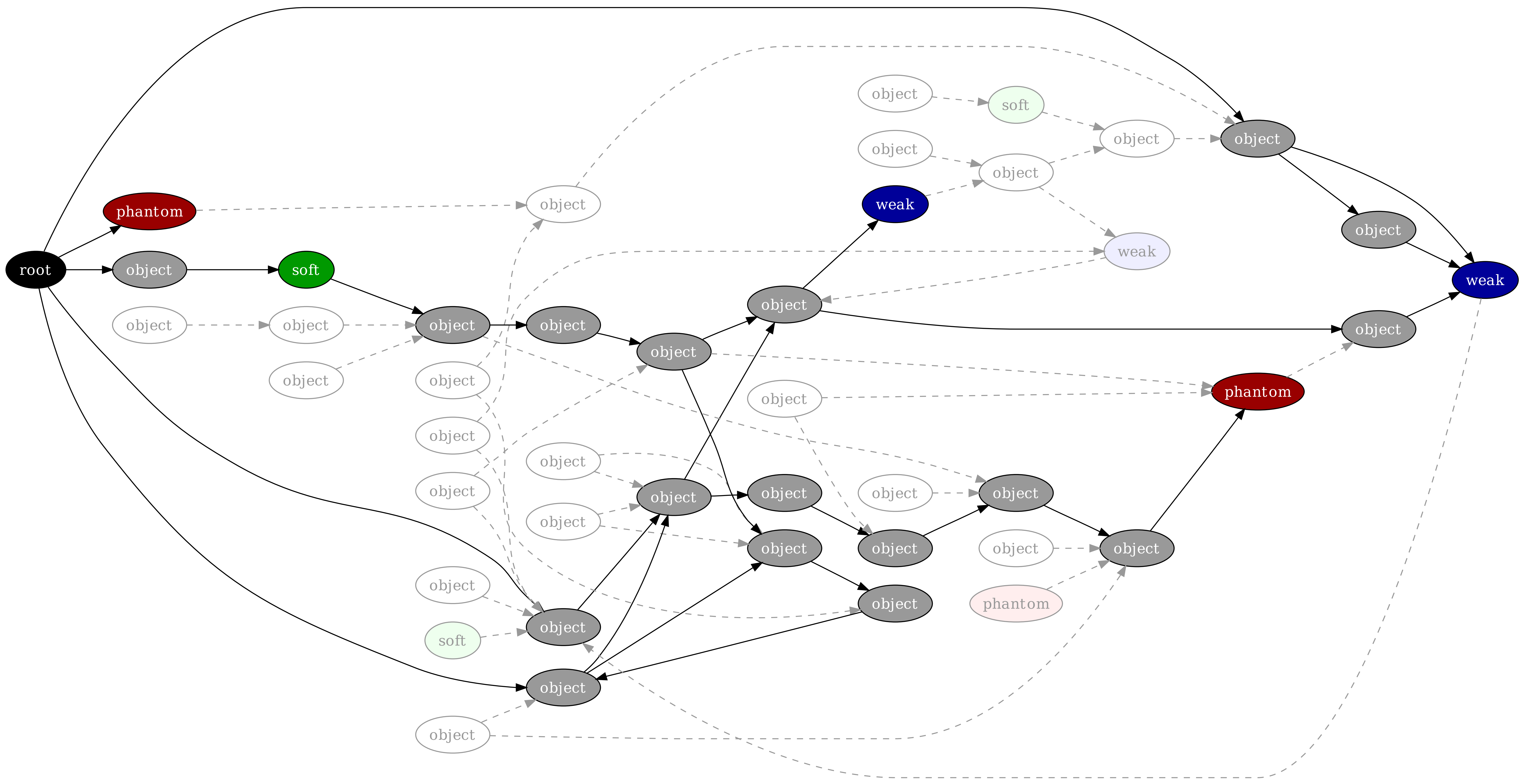
3. Optionally clear soft references.



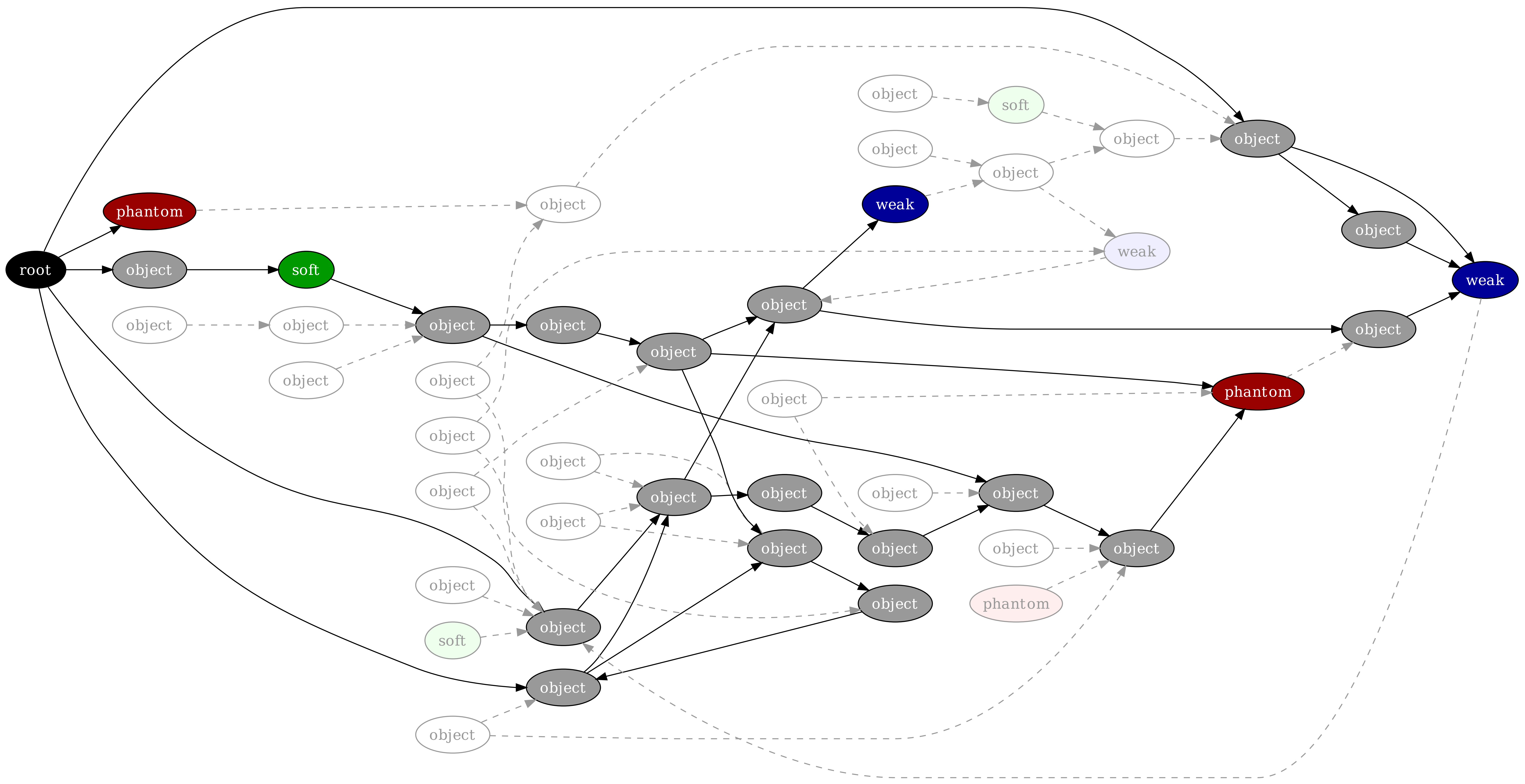
4. Trace and mark softly-referenced objects.



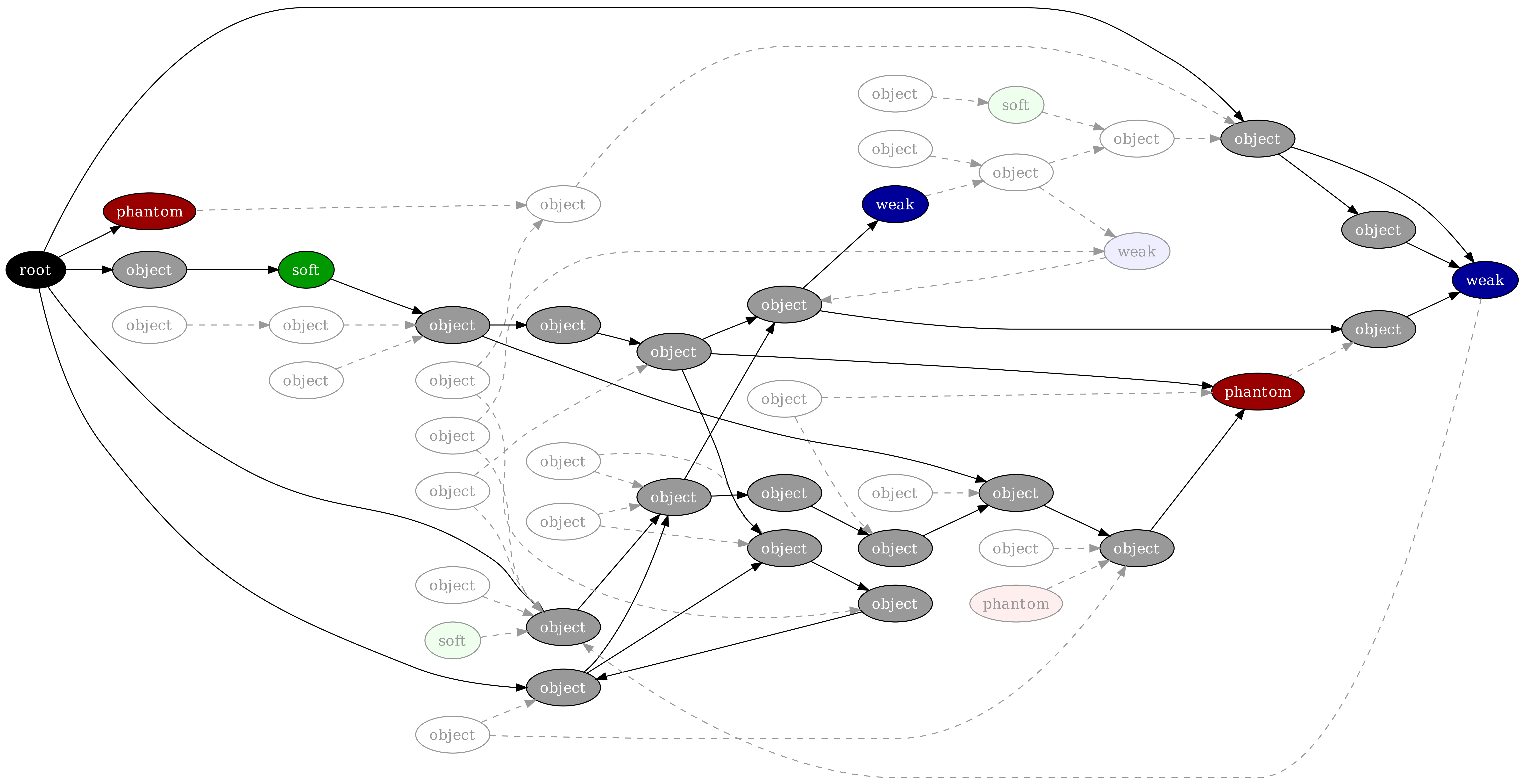
4. Trace and mark softly-referenced objects.



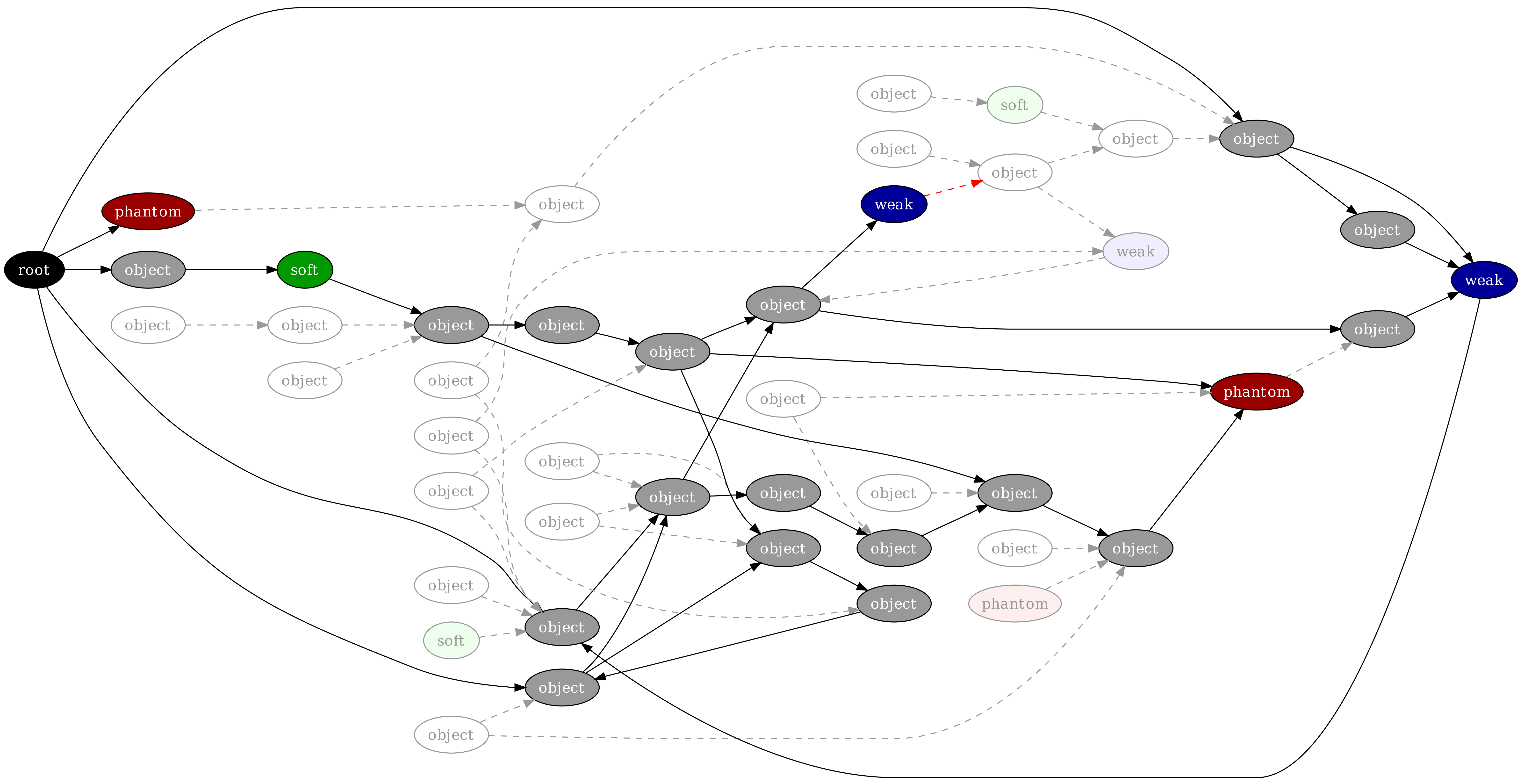
4. Trace and mark softly-referenced objects.



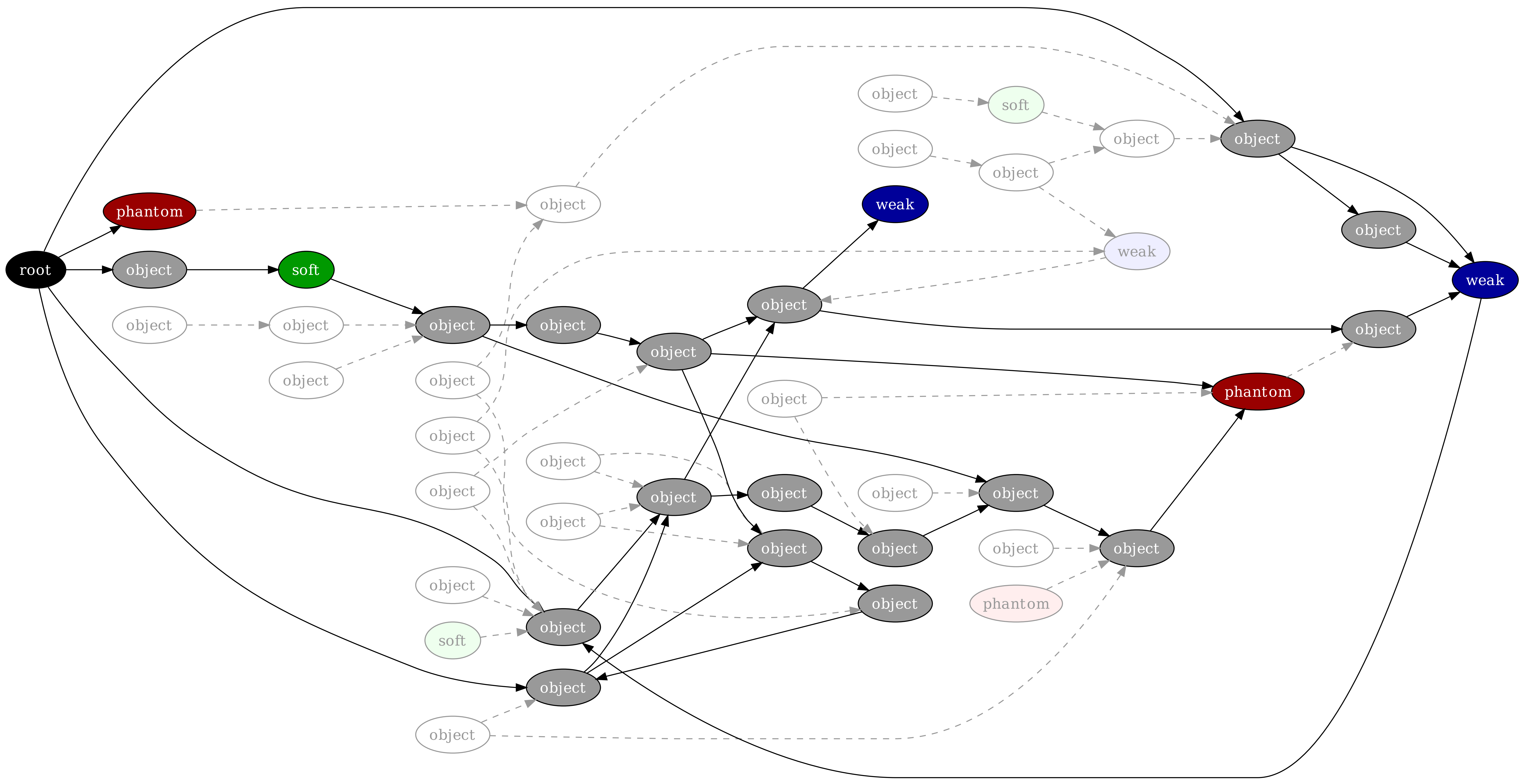
5. Clear weak references.



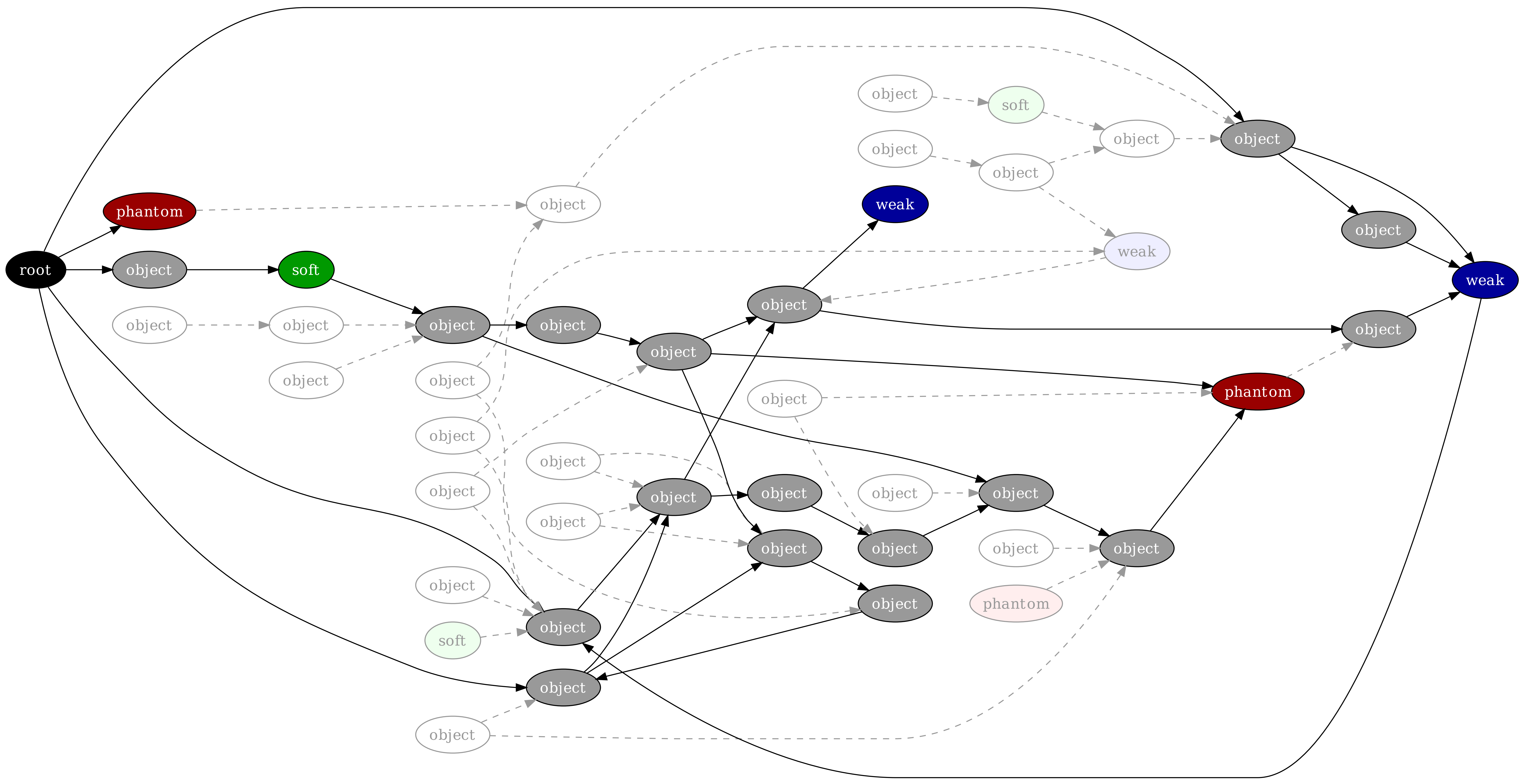
5. Clear weak references.



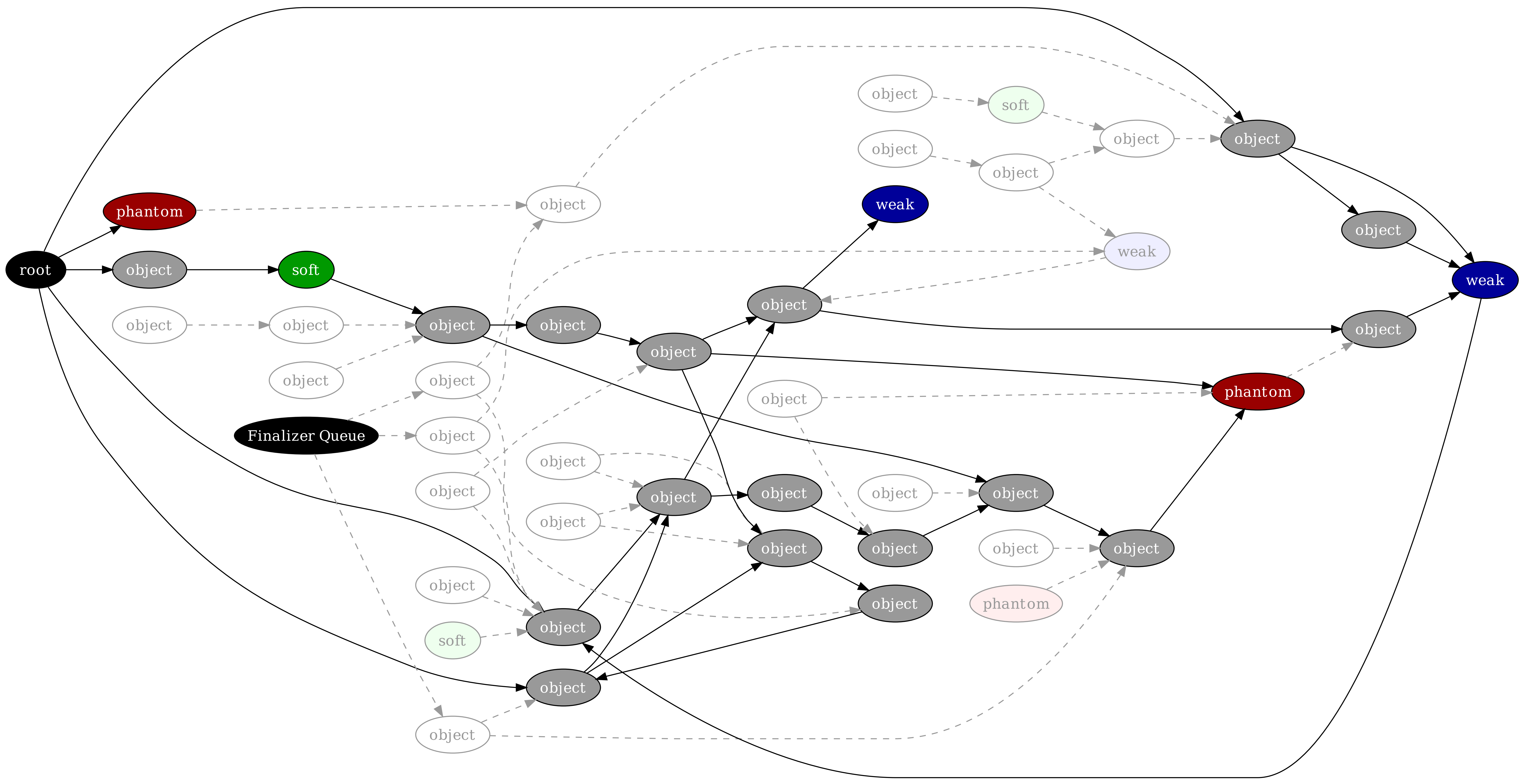
5. Clear weak references.



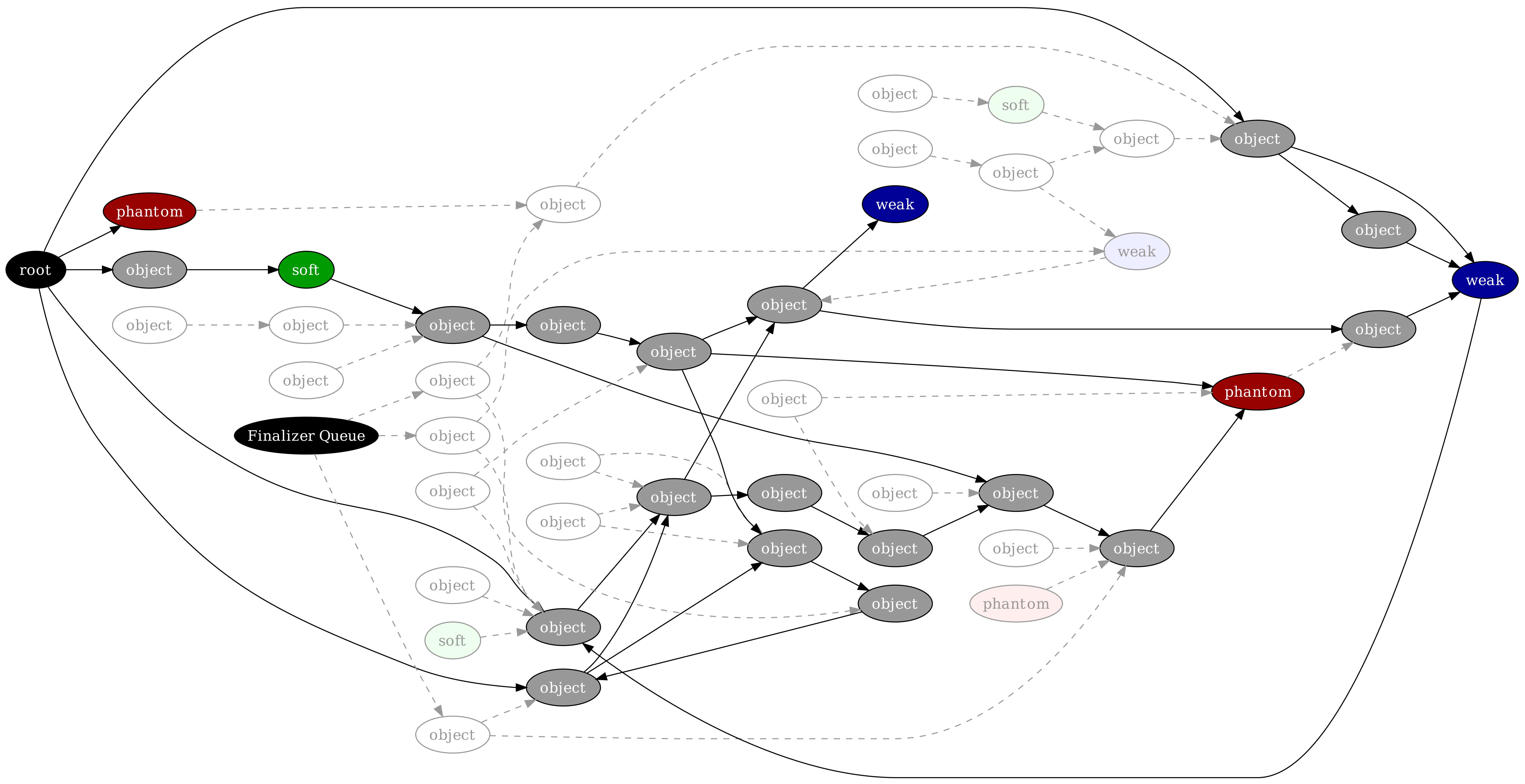
6. Enqueue finalizable objects.



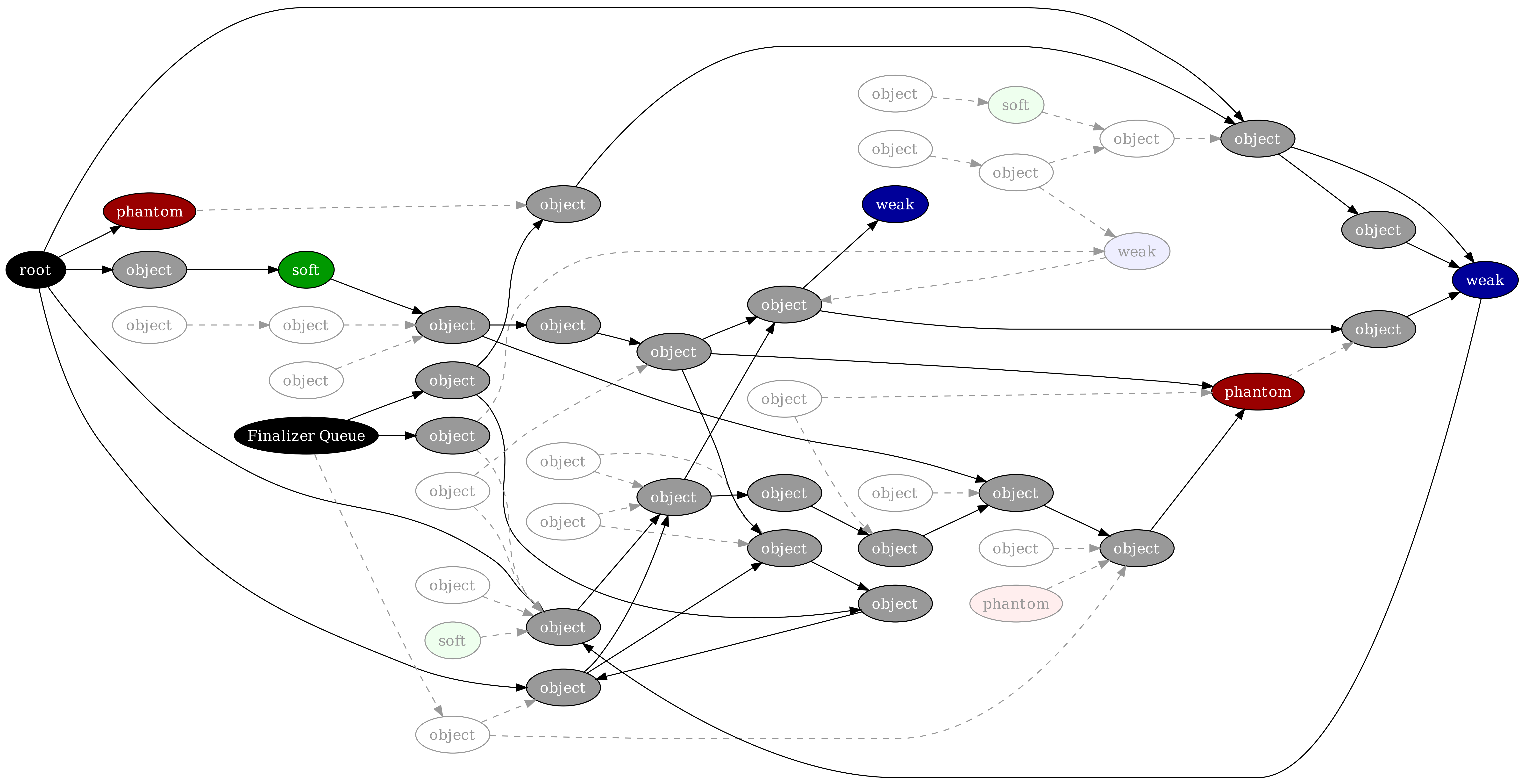
6. Enqueue finalizable objects.



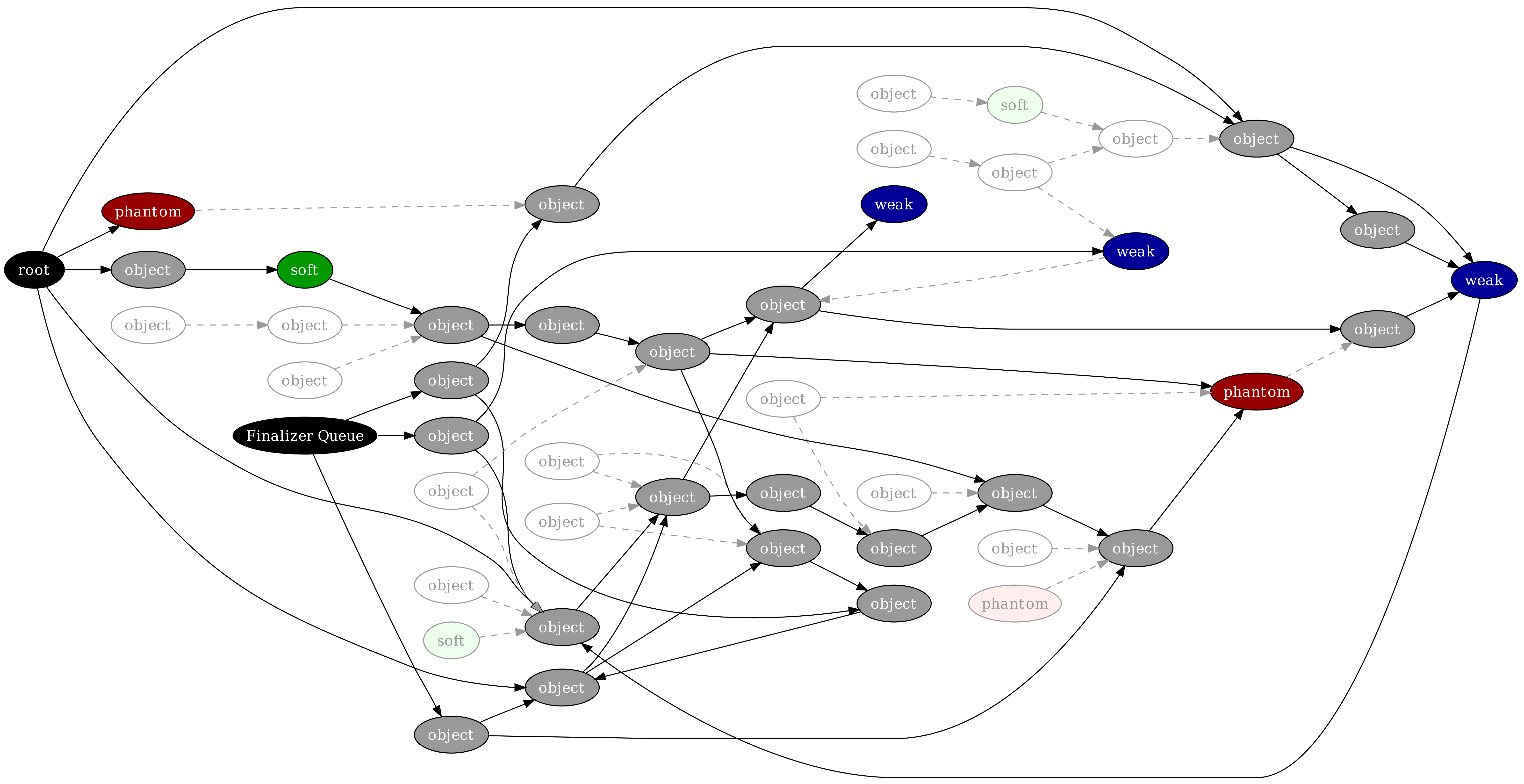
7. Repeat steps 1 through 5 for the queue.



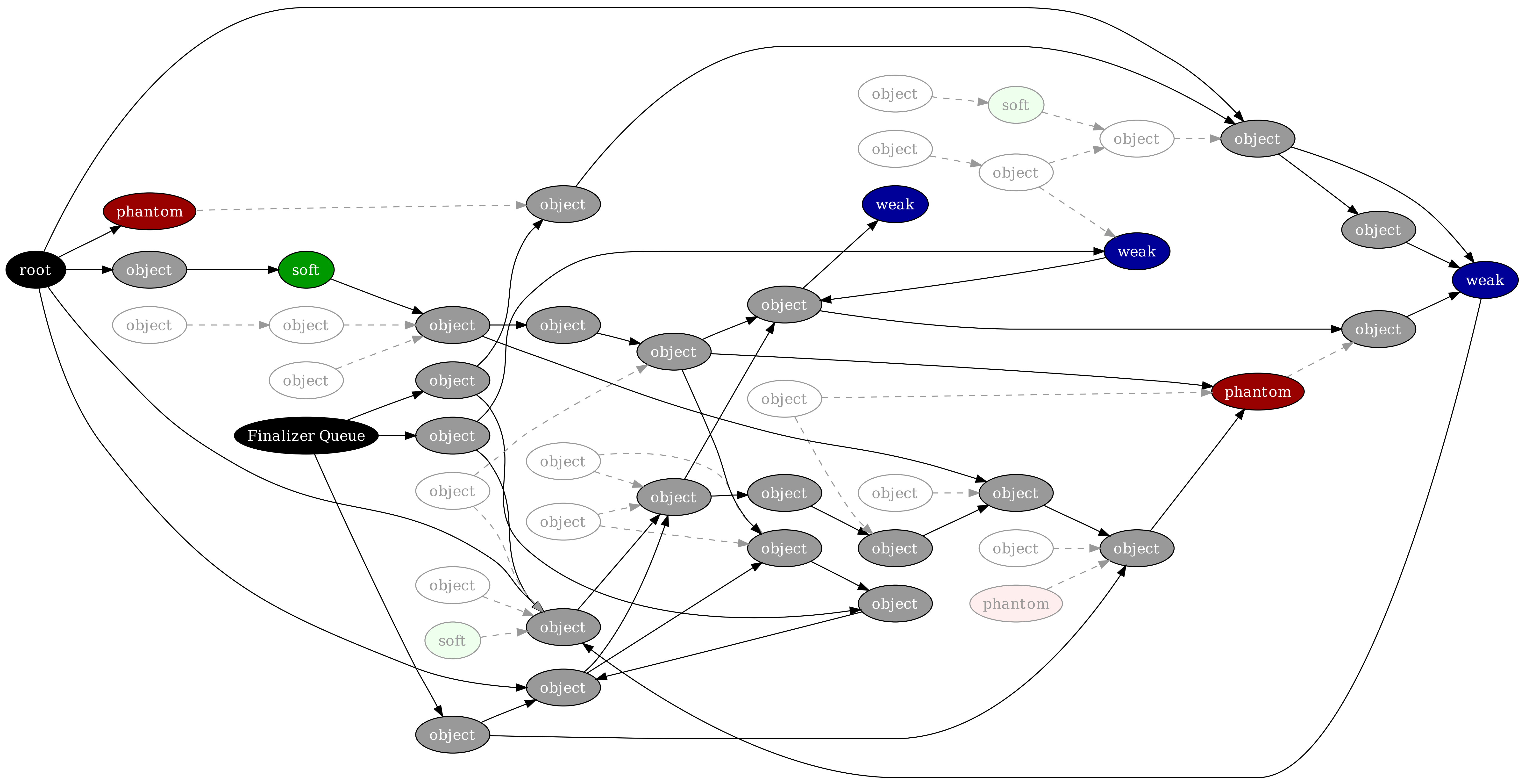
7. Repeat steps 1 through 5 for the queue.



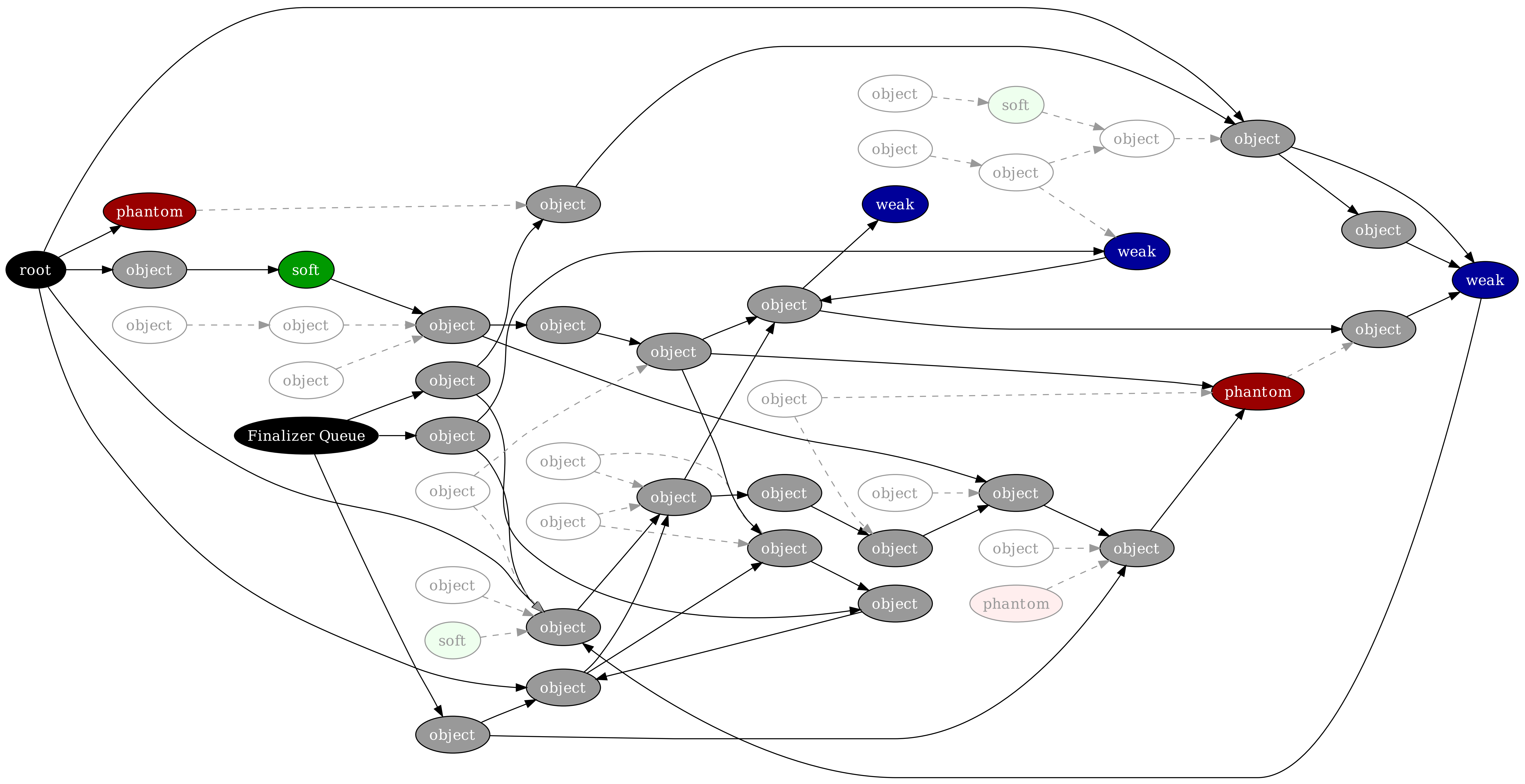
7. Repeat steps 1 through 5 for the queue.



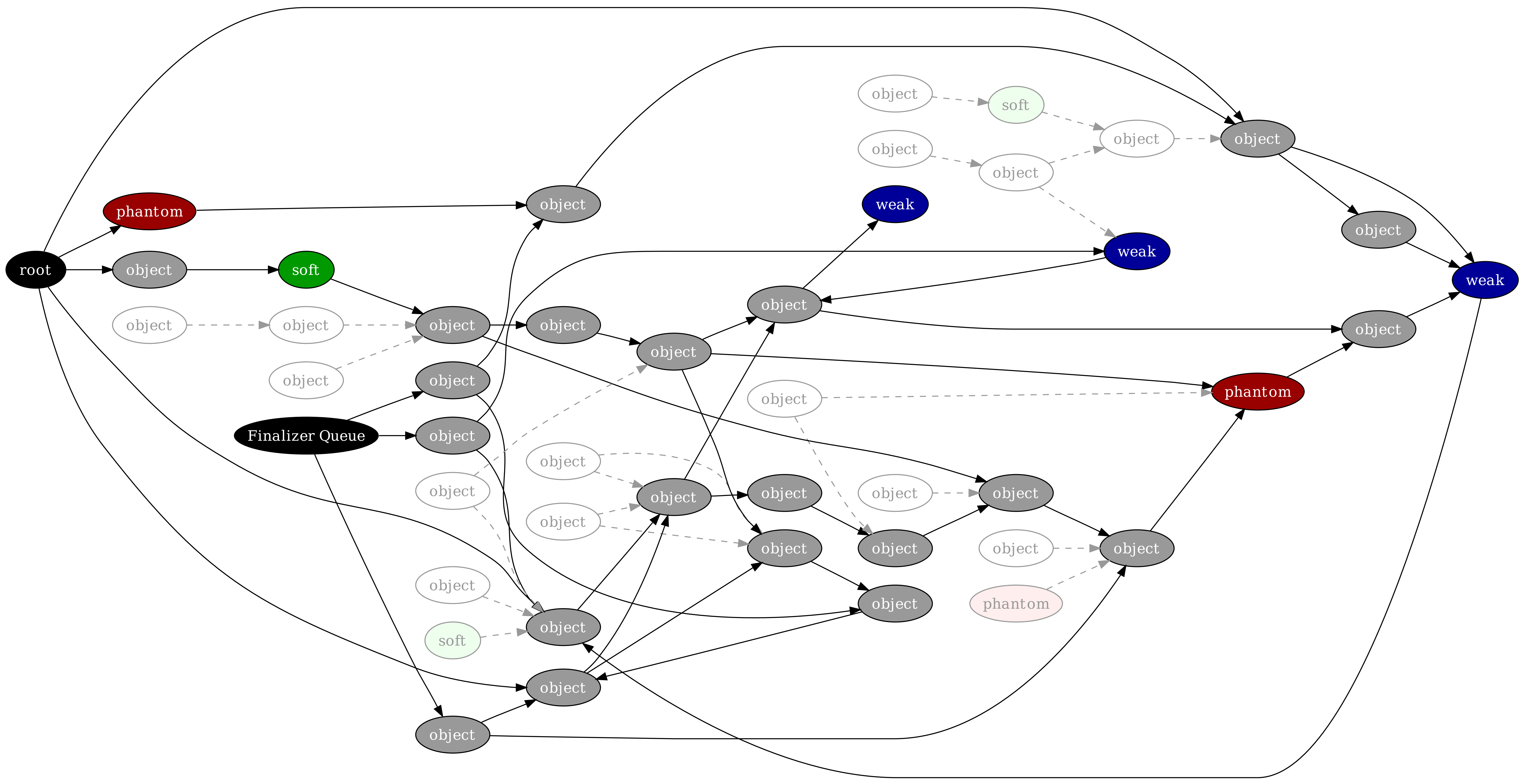
7. Repeat steps 1 through 5 for the queue.



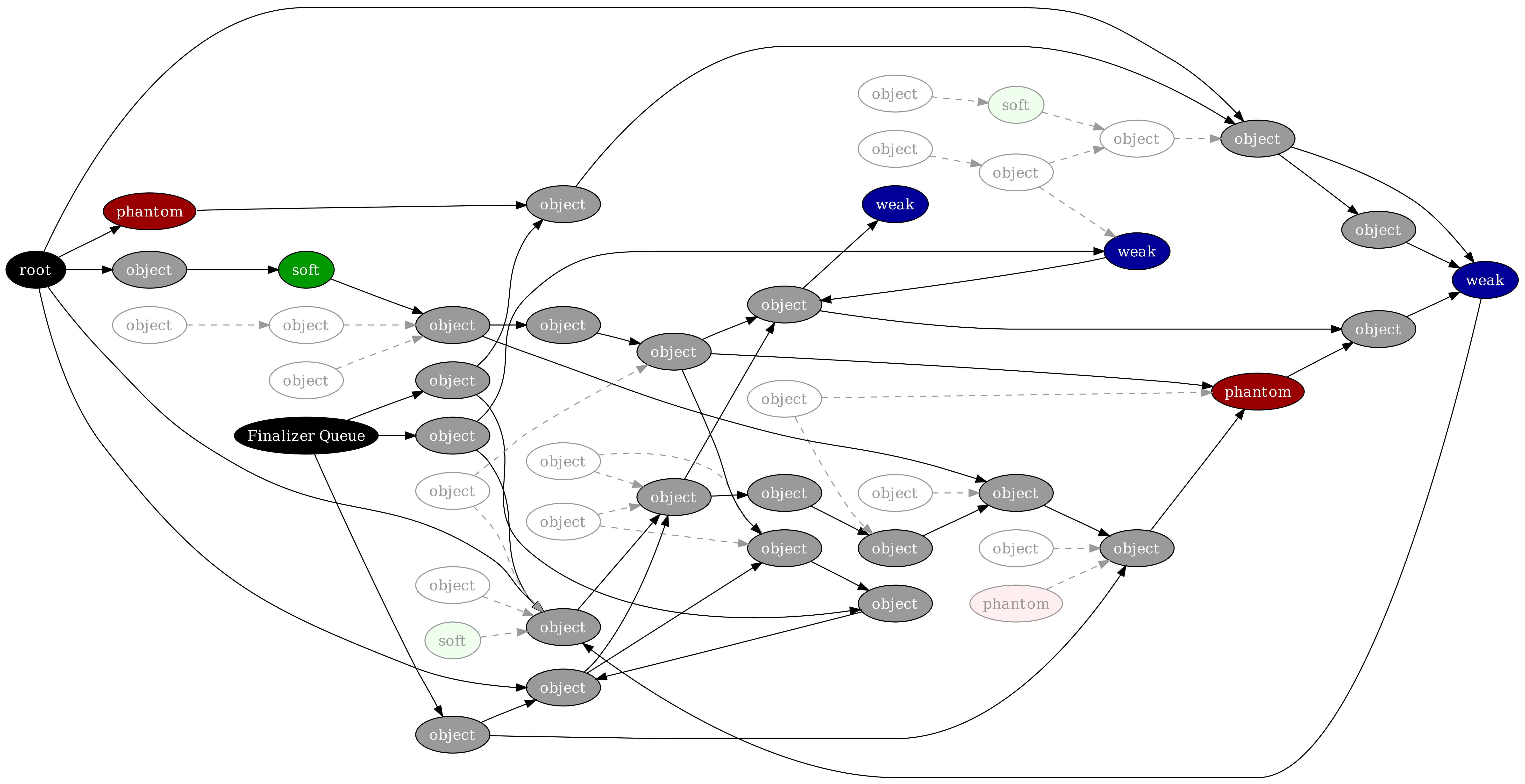
8. Possibly enqueue phantom references.



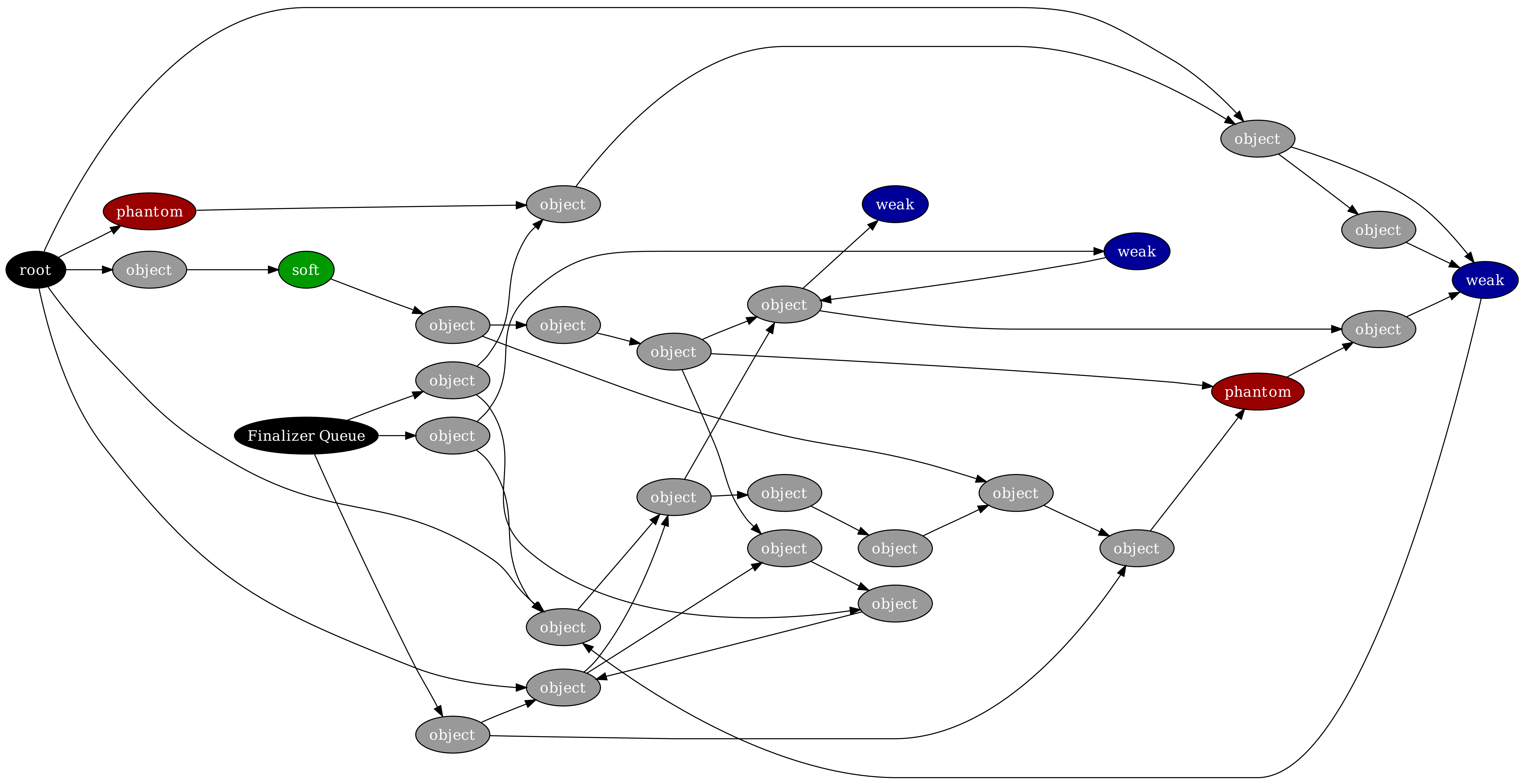
8. Possibly enqueue phantom references.



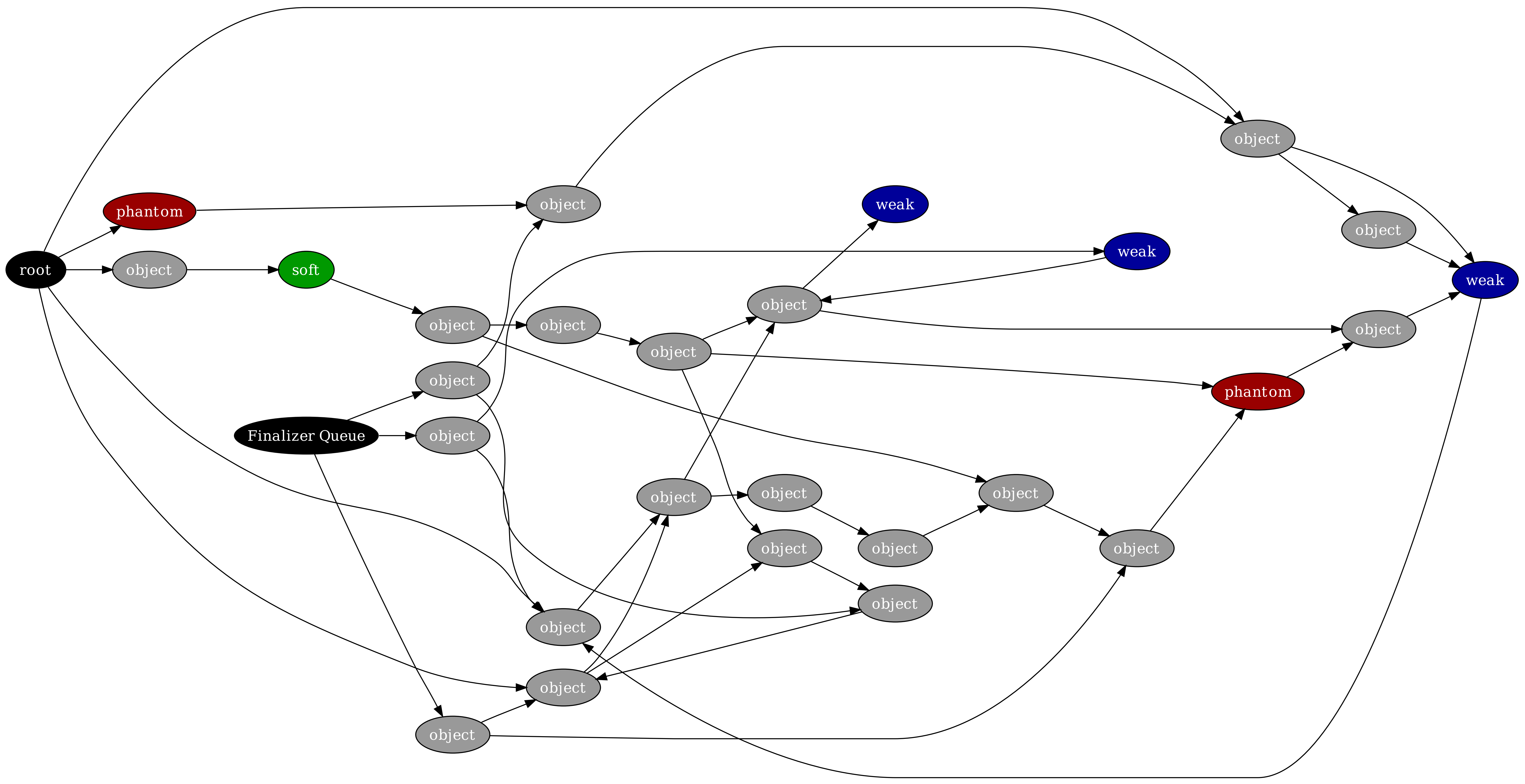
9. The remaining objects are dead.



9. The remaining objects are dead.



10. Repeat.


root
phantom
object
soft
Finalizer Queue
weak
weak
weak
phantom
object

Recap

1. Start at a root.

2. Trace and mark strongly-referenced objects.

3. Optionally clear soft references.

4. Trace and mark softly-referenced objects.

5. Clear weak references.

6. Enqueue finalizable objects.

7. Repeat steps 1 through 5 for the queue.

8. Possibly enqueue phantom references.

9. The remaining objects are dead.

10. Repeat.

Two options for freeing native resources

> Use a finalizer.
  - You must defend against subsequent use!

> Or use a phantom reference.