Understanding the Java Virtual Machine: Memory Management

Introduction

Kevin Jones

@kevinrjones

Why GC?

create and forget:
no need to remember
to delete

```
Account acc = new Account();
```

use and forget:
no need to ask
“Should I delete?”

```
Account acc = getAccount();
```

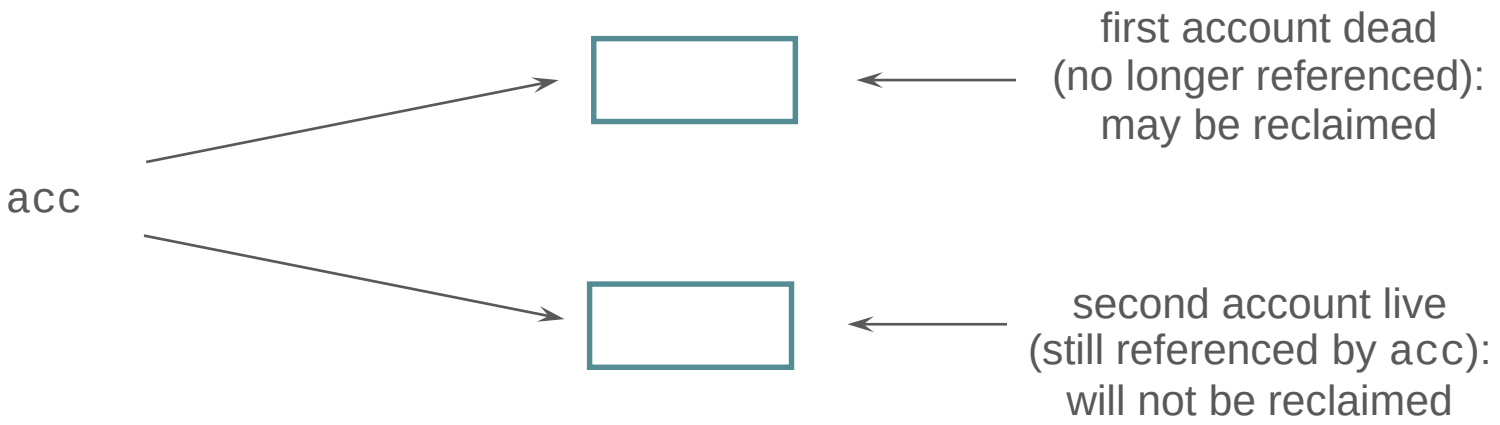
use with confidence:
objects will not
vanish or become corrupt
behind your back

```
acc.increment(amount);
```

The GC promise

- Claim no live objects
  - no promises about dead objects

```
Account acc = new Account();
acc = new Account();
```



Forms of Garbage Collection

Do Nothing

Reference Counting

Mark and Sweep

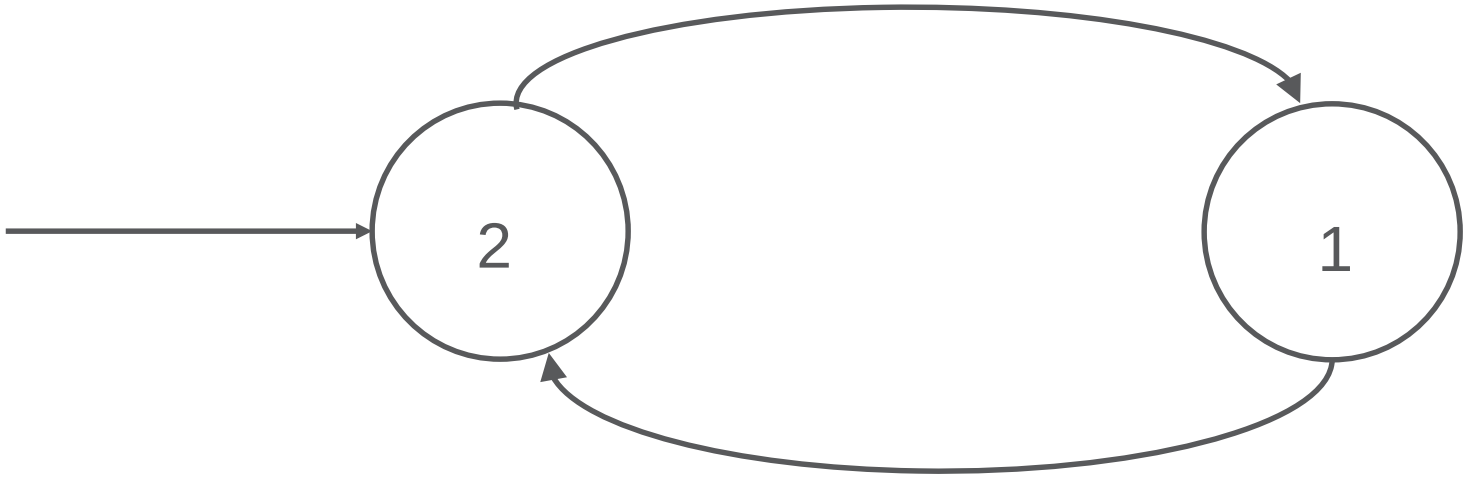
Copying

Generational

Incremental

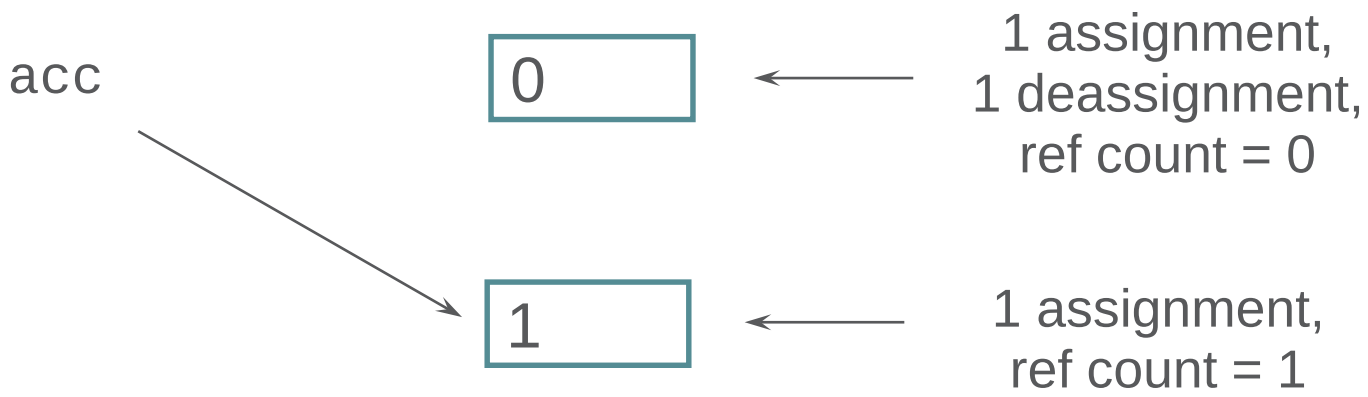
Reference Counting

- Onus on client to call methods when allocating/freeing memory
  - COM for example had AddRef and Release calls for objects
  - When count hits zero object can be freed
  - Problems with circular references



Reference Counting

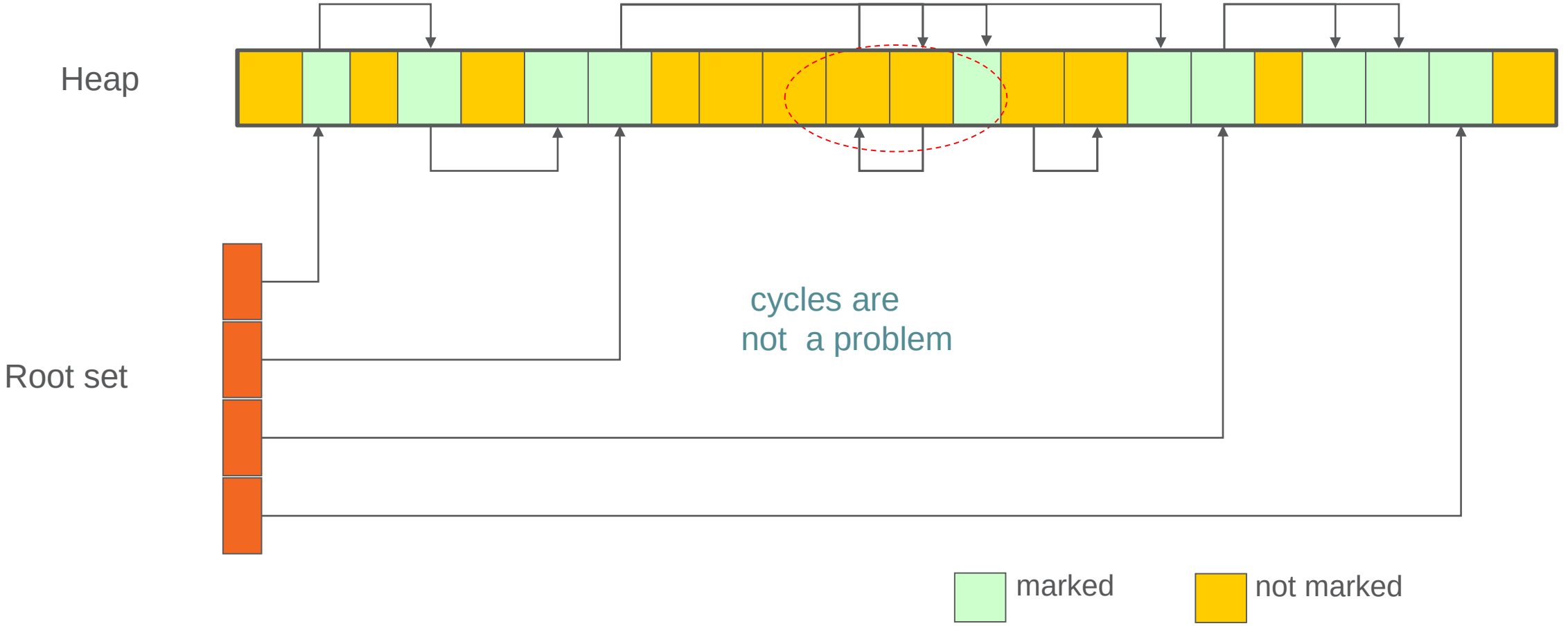
```
Account acc = new Account();
acc = new Account();
```



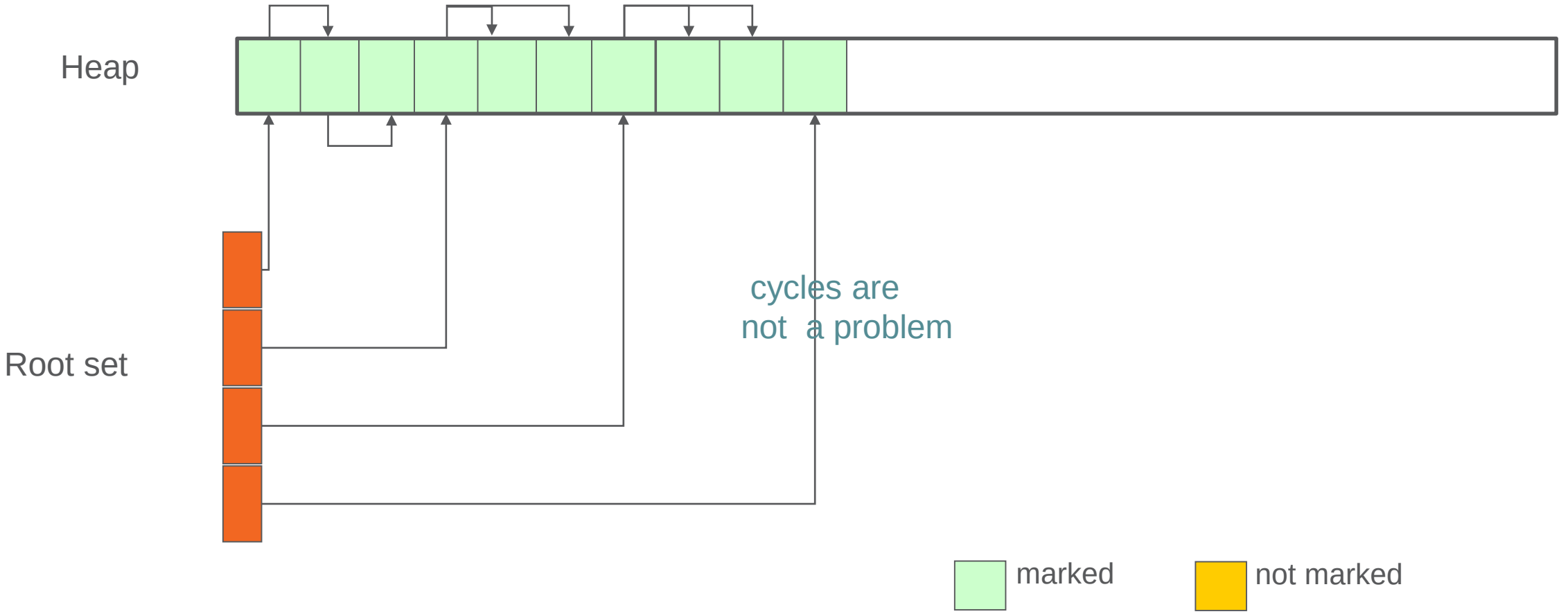
Mark and Sweep

- ‘mark’ phase that identifies the objects that are still in use
- ‘sweep’ phase to remove unused objects
- ‘compact’ phase to compact the memory

Sweep Phase



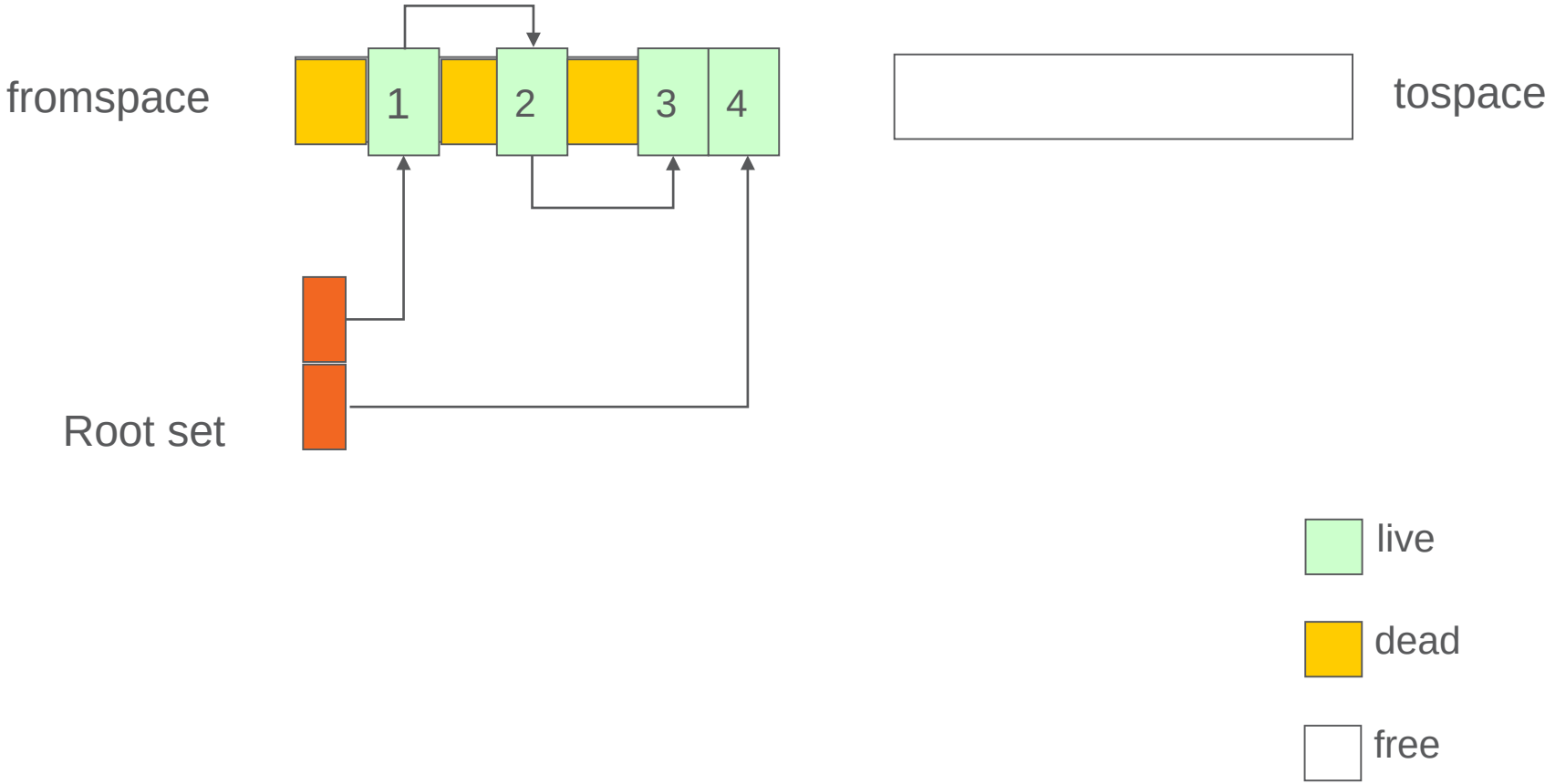
Compact Phase



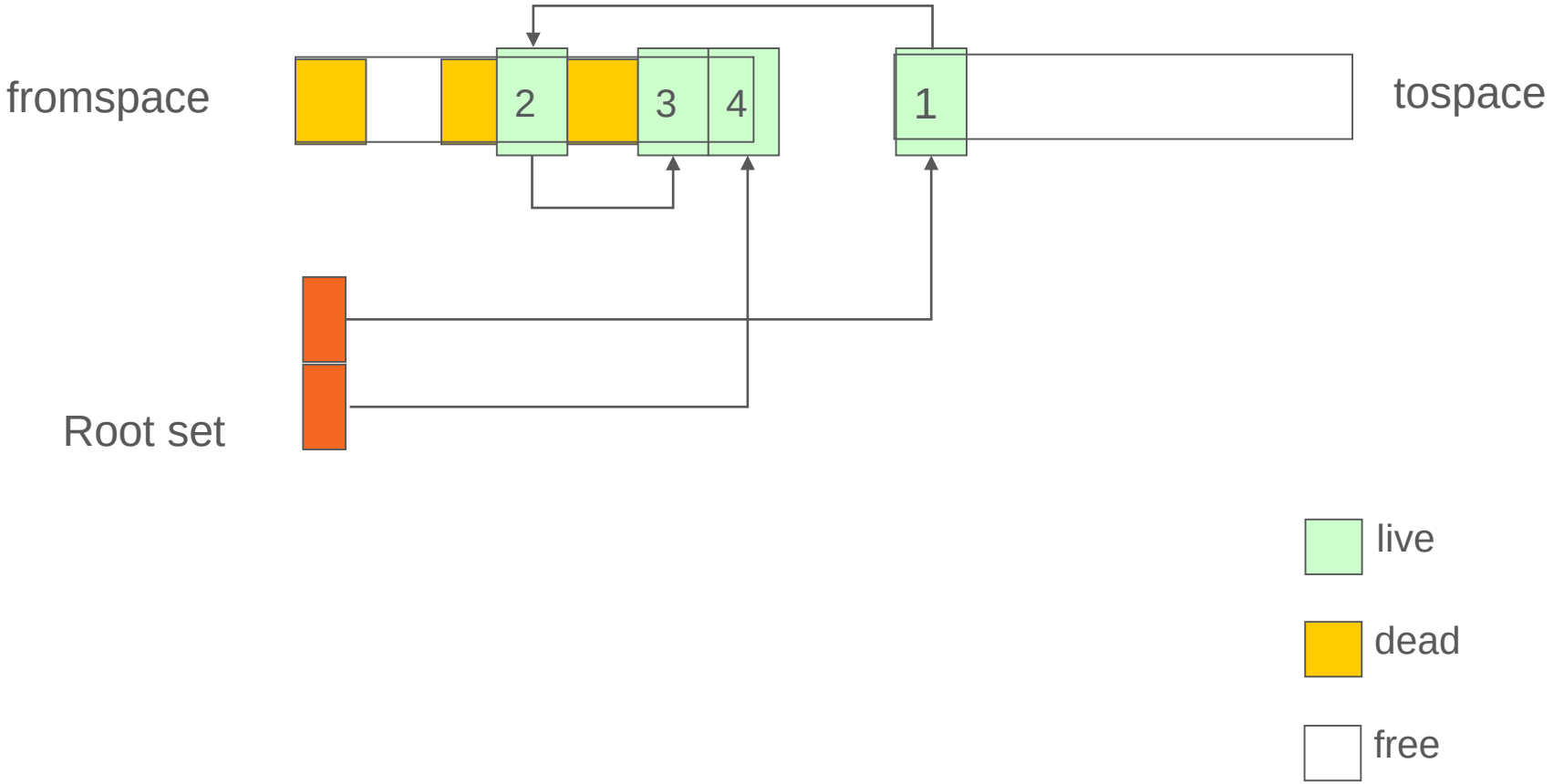
Copying

- Uses different ‘spaces’ to manage memory

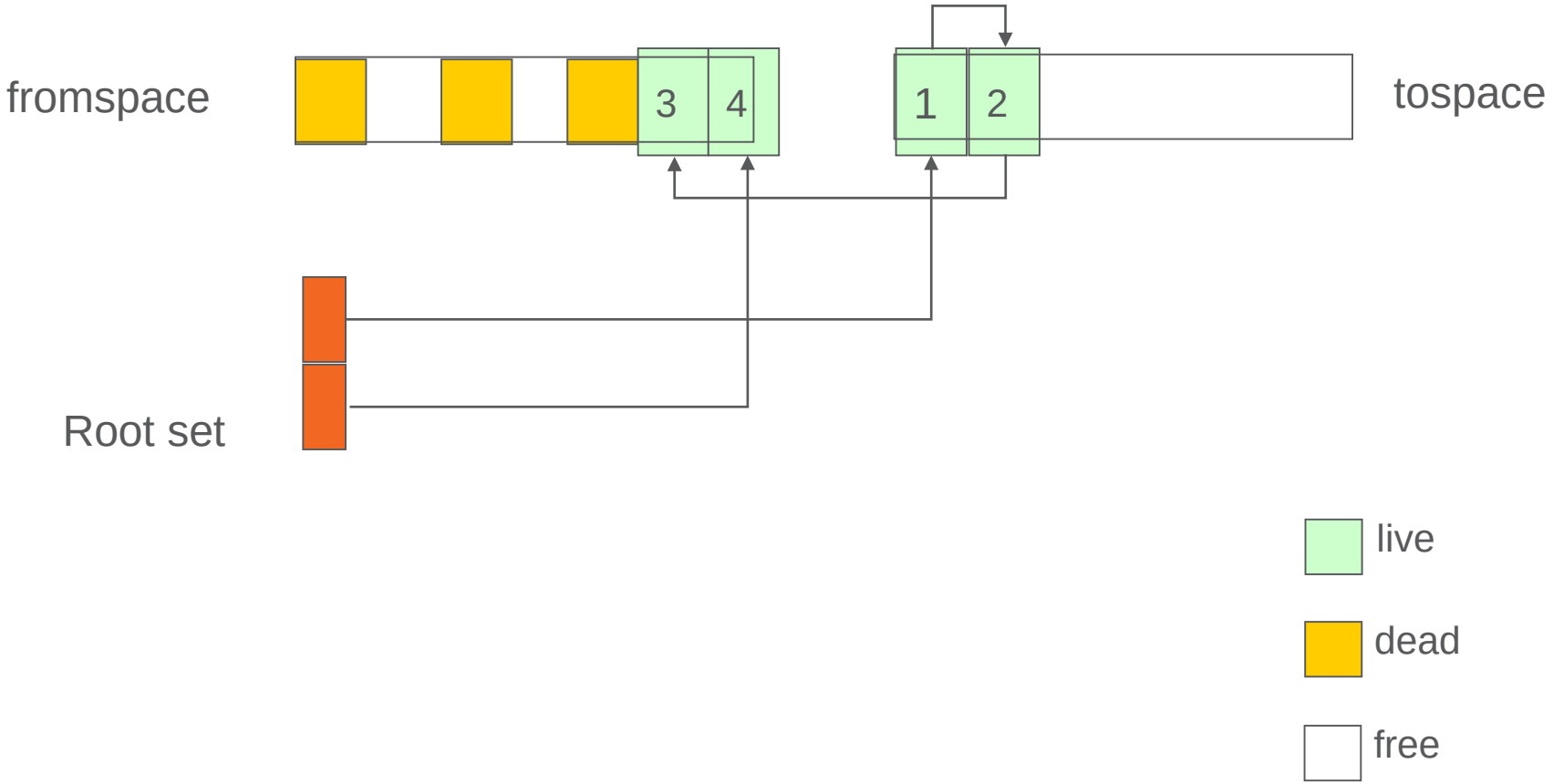
Copying Fromspace to Tospace



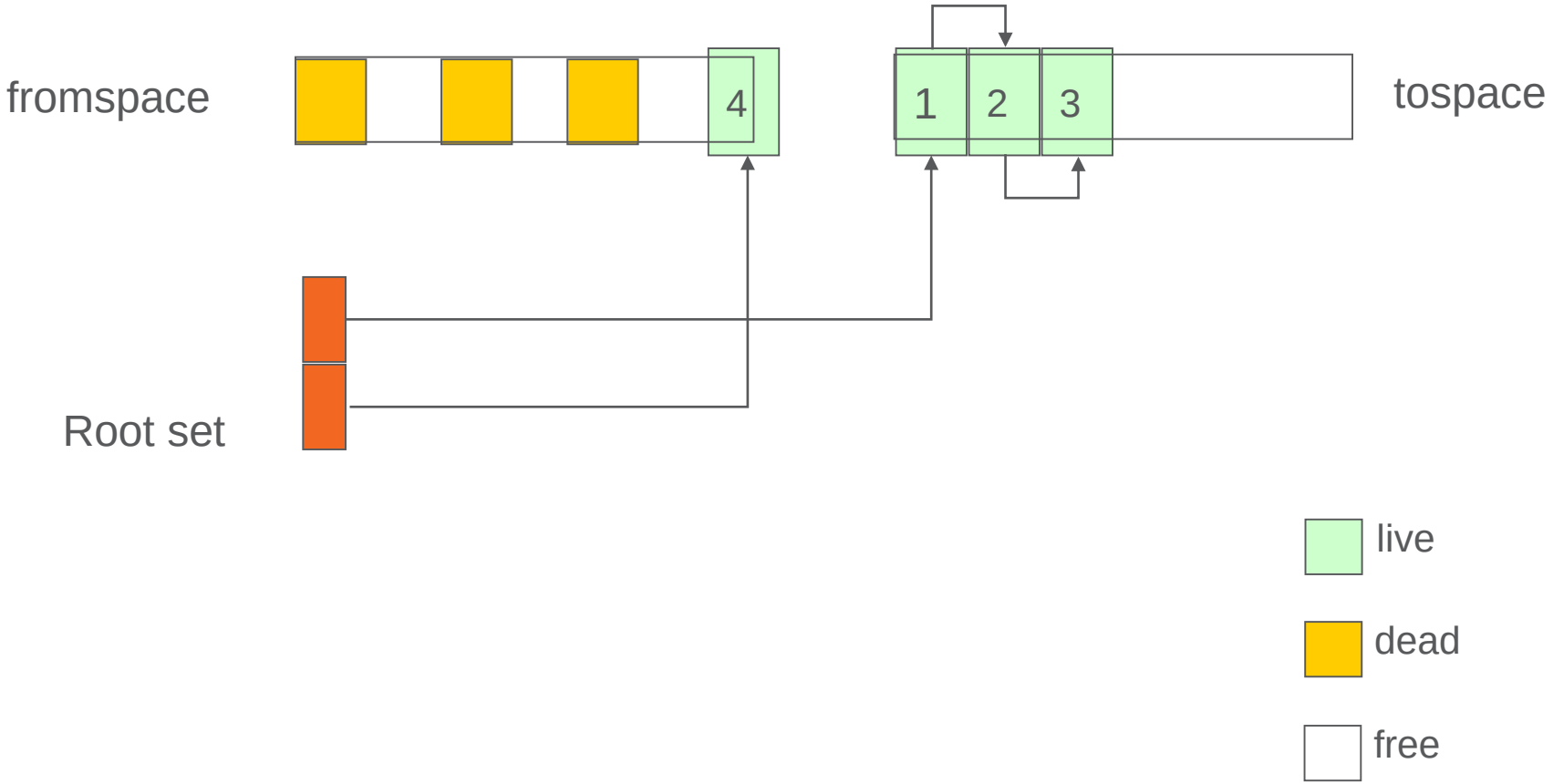
Copying Fromspace to Tospace



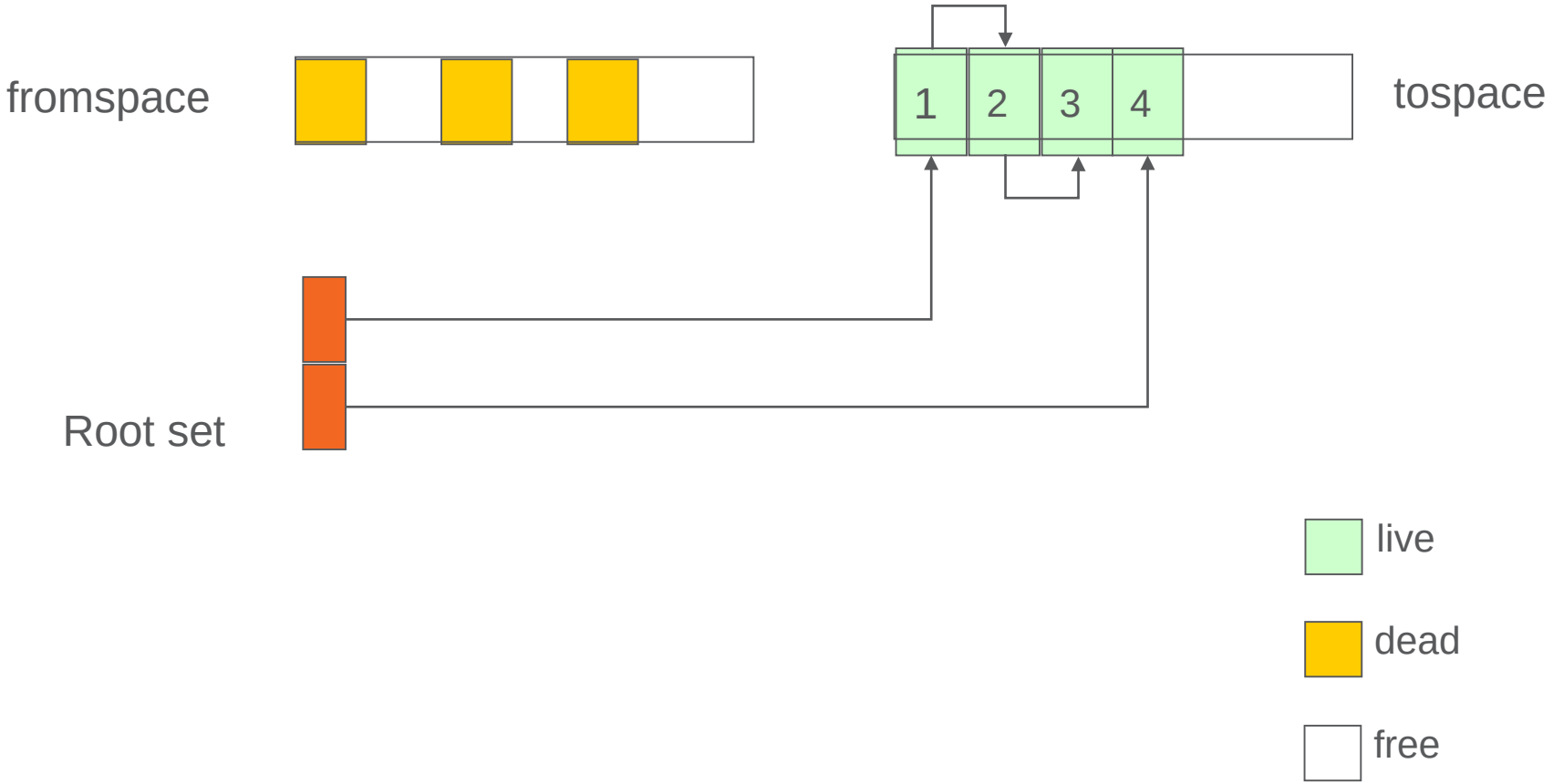
Copying Fromspace to Tospace



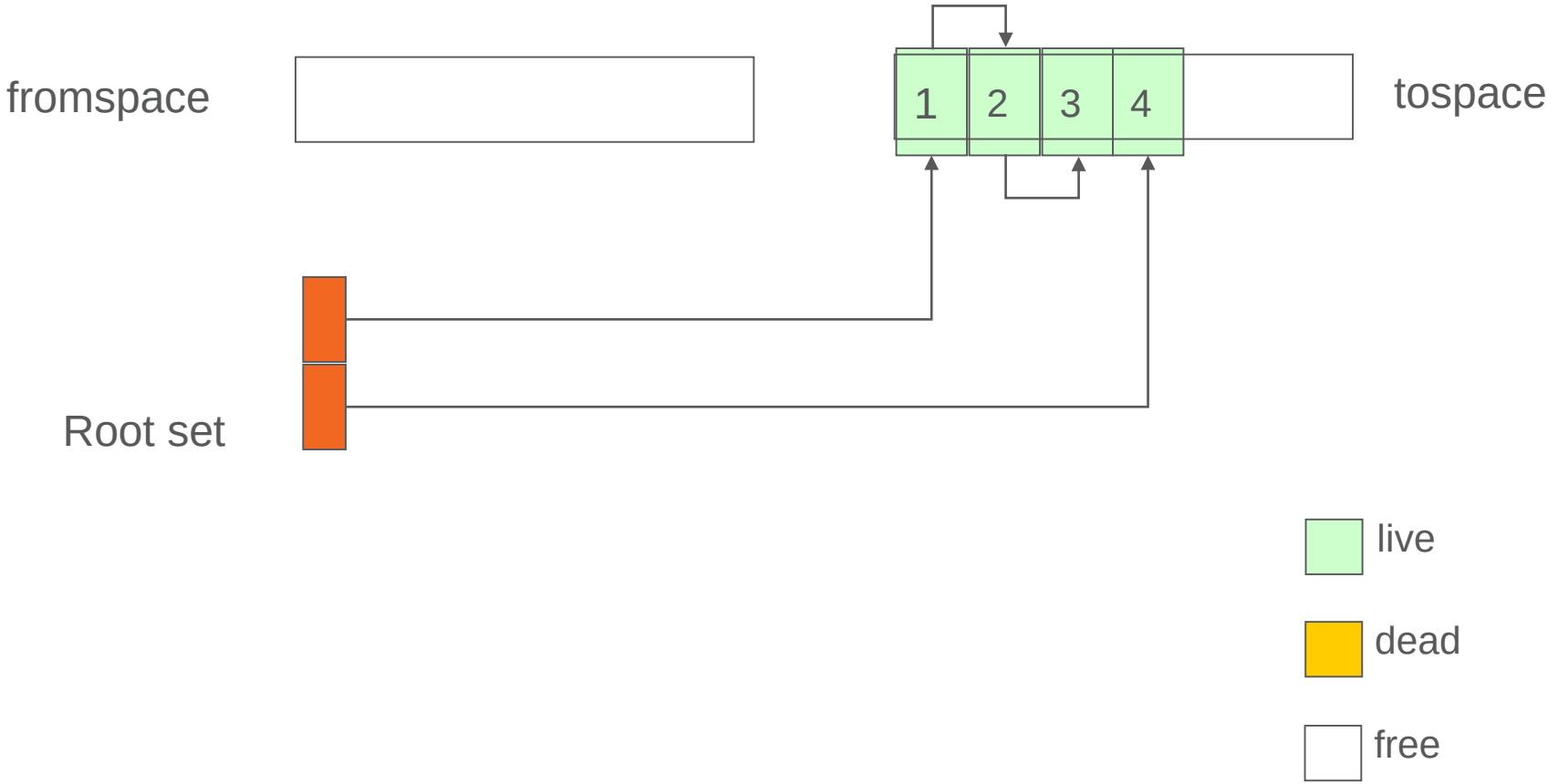
Copying Fromspace to Tospace



Copying Fromspace to Tospace



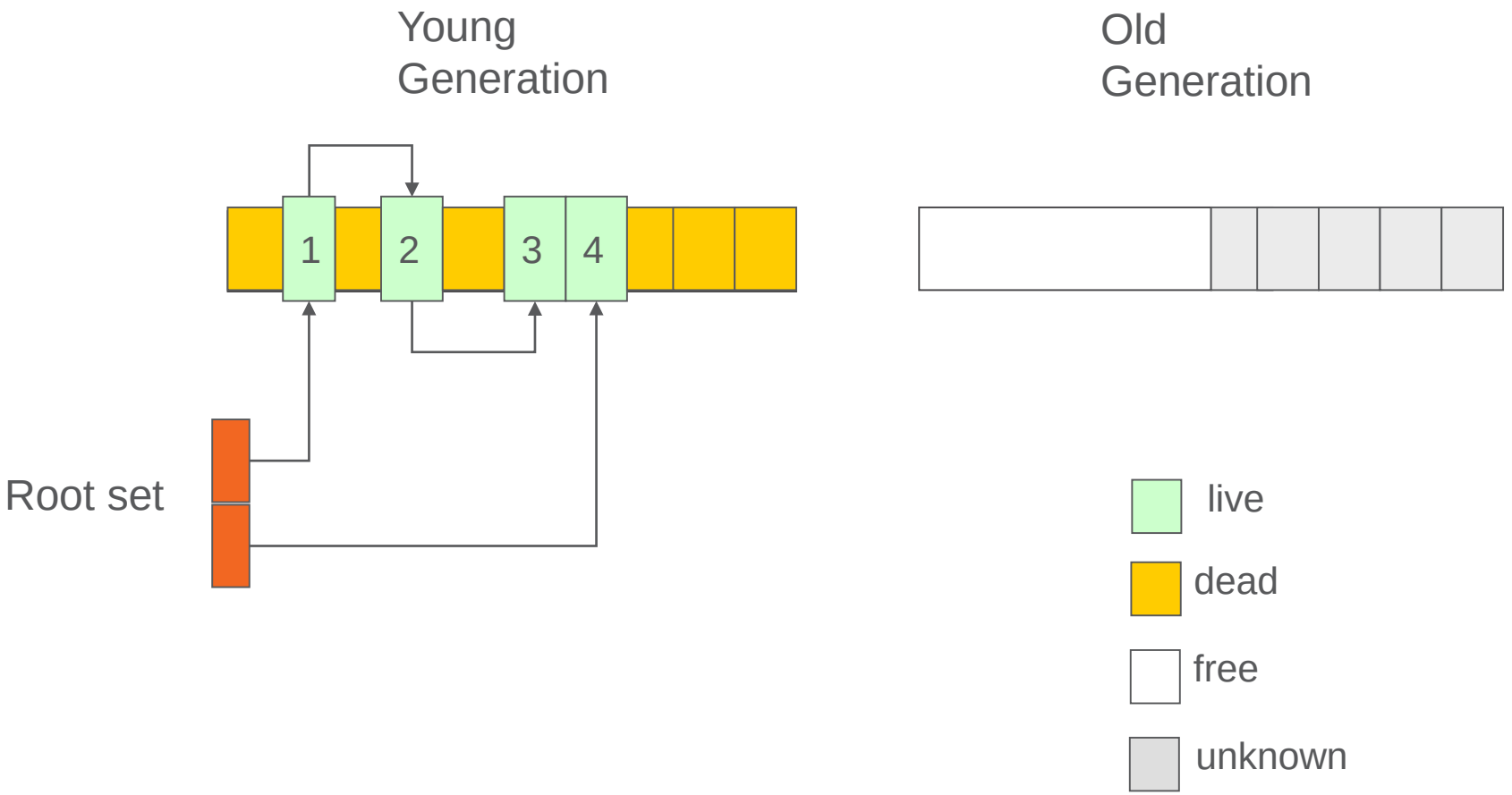
Copying Fromspace to Tospace



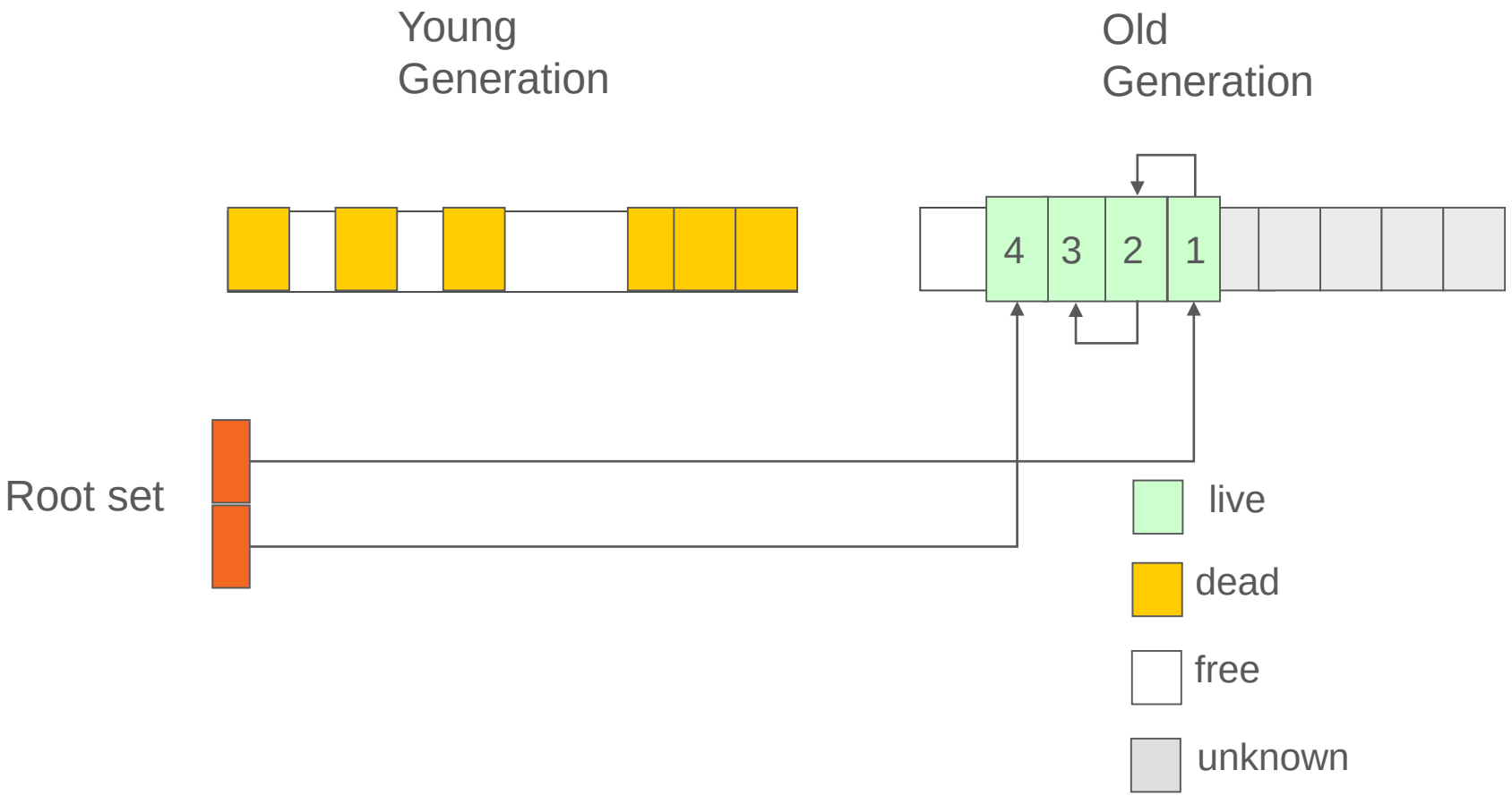
Generational Collectors

- Maintain different generations for memory
  - Long living objects 'promoted' to a different generation
  - For a given definition of 'long'

Before a Generational Minor Collection



After a Generational Minor Collection



Demonstration