```
void dequeue() {
  while (true) {
    head = Head;
    next = head->next;
    // ...
    if(CAS(Head,head,next)){


      return;
}}}
```

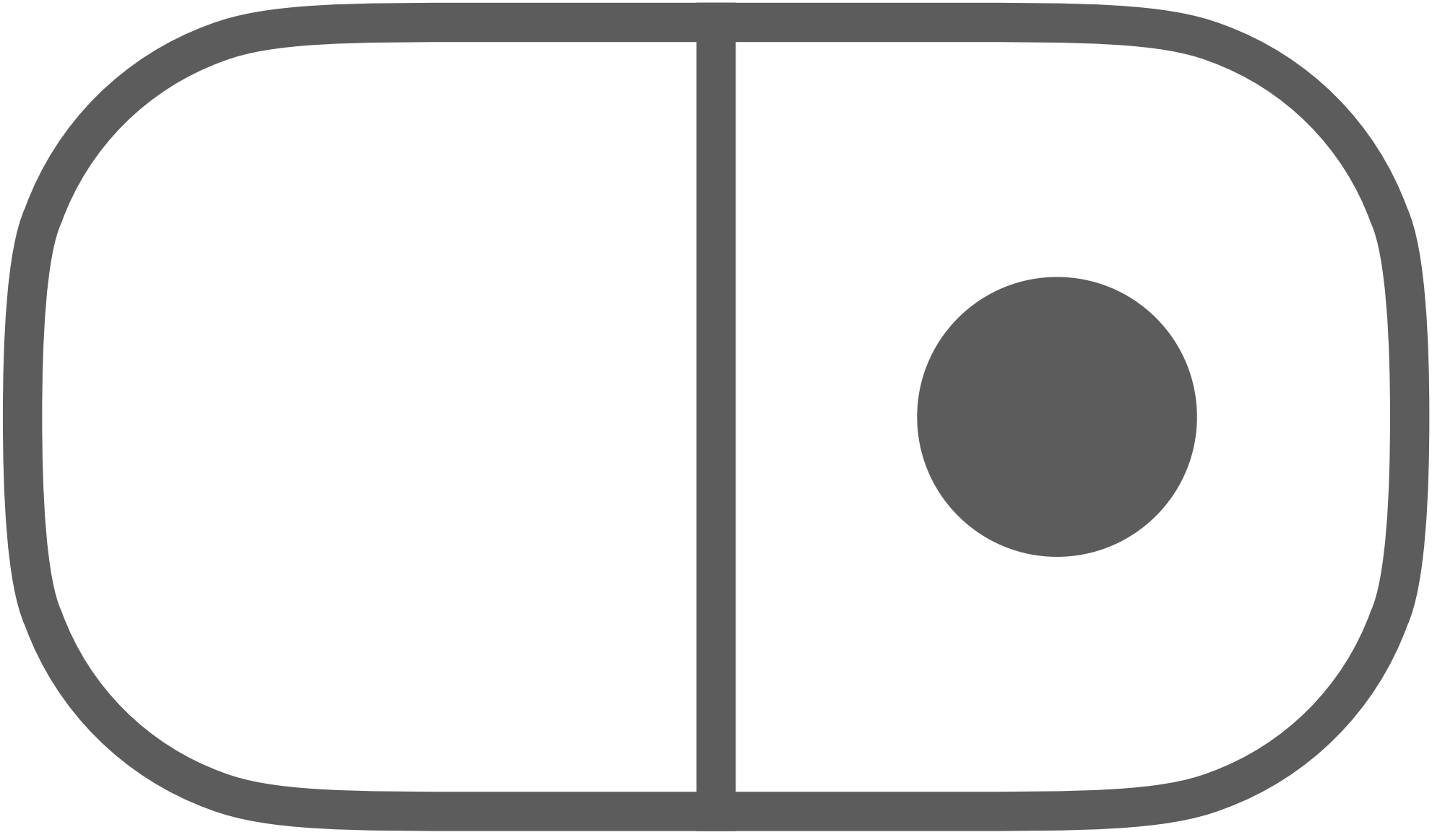
Non-blocking Queue (Michael&Scott)

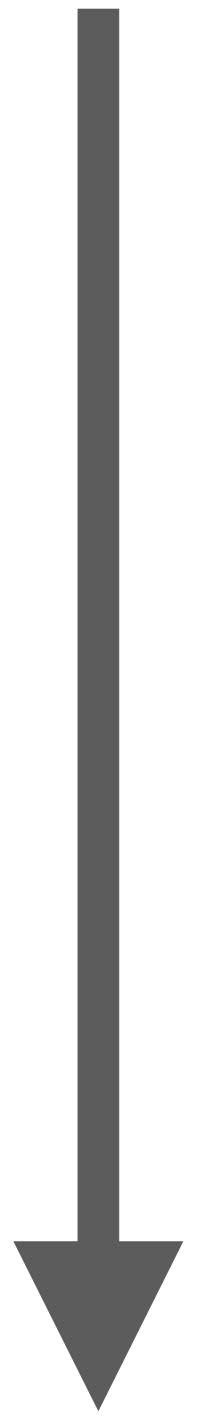

Head

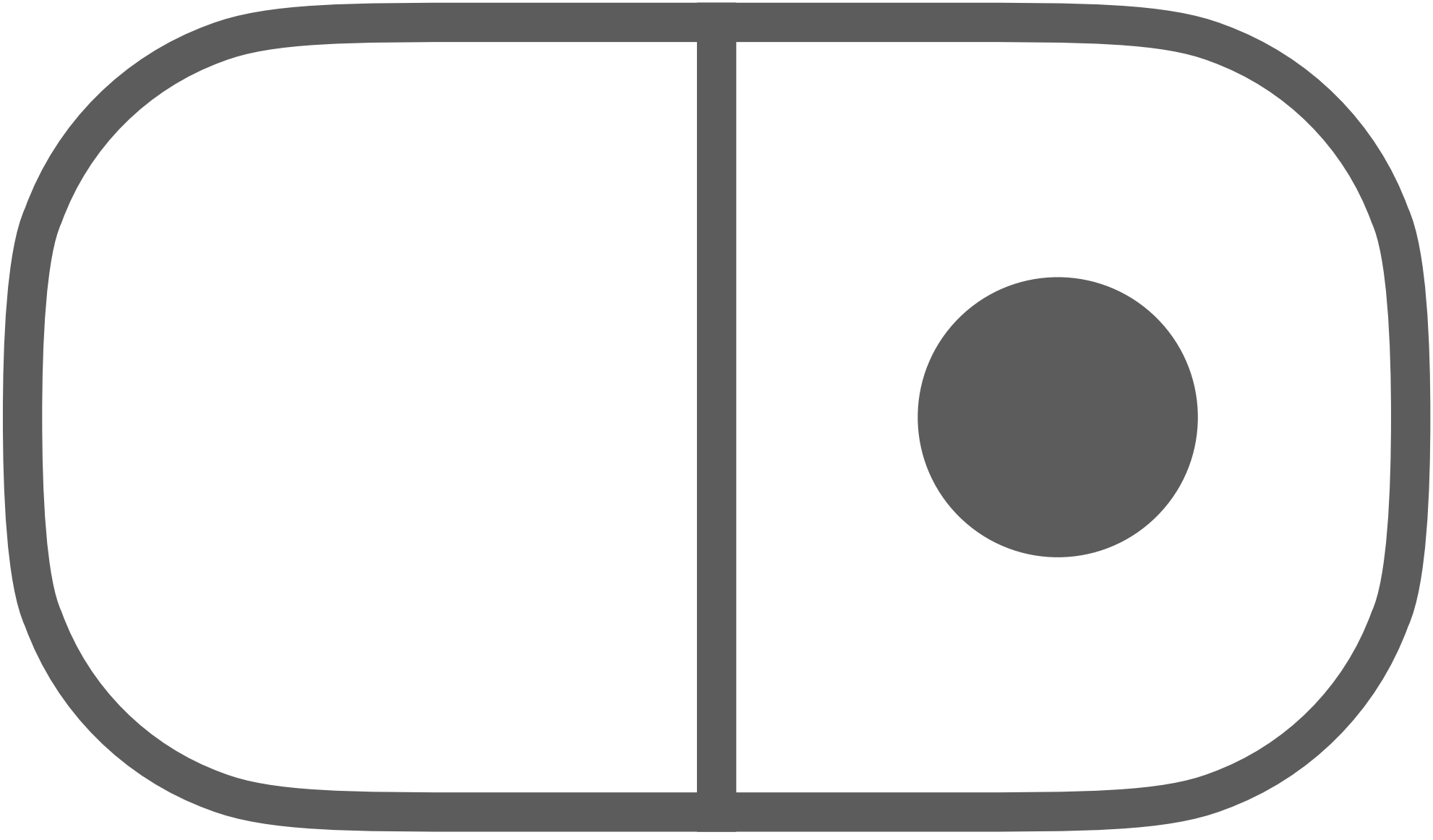

head2

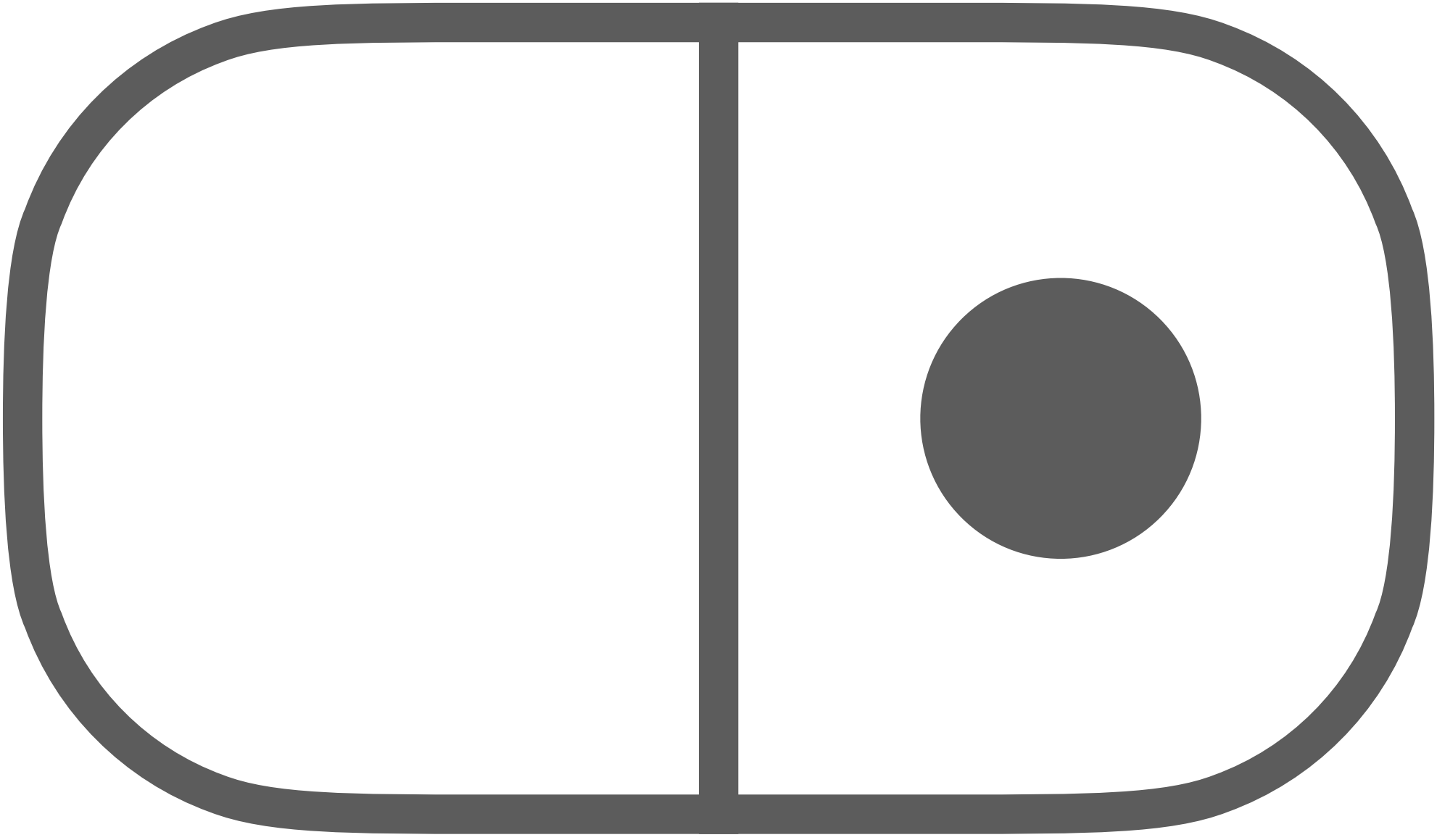

head1

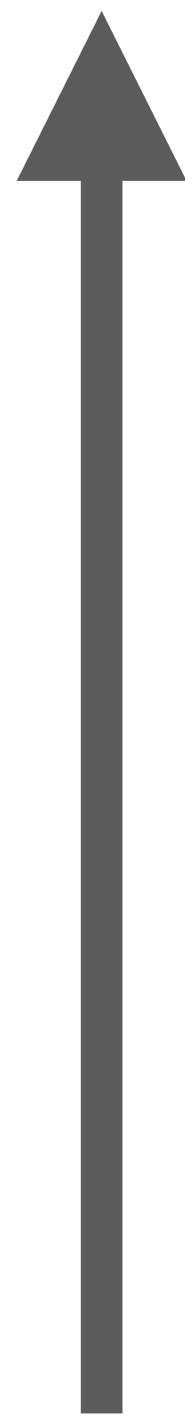

next1

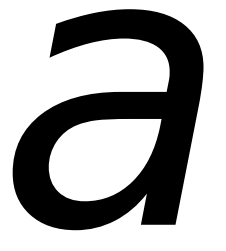


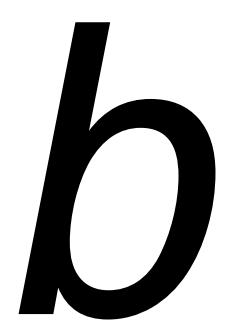


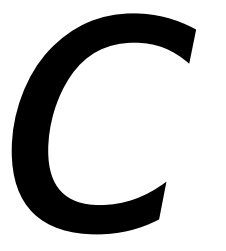













next2


head3

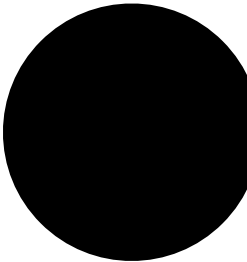
```
// leak head?
```

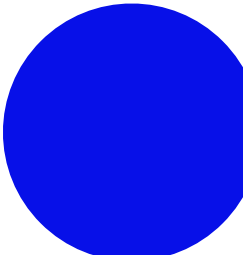

1

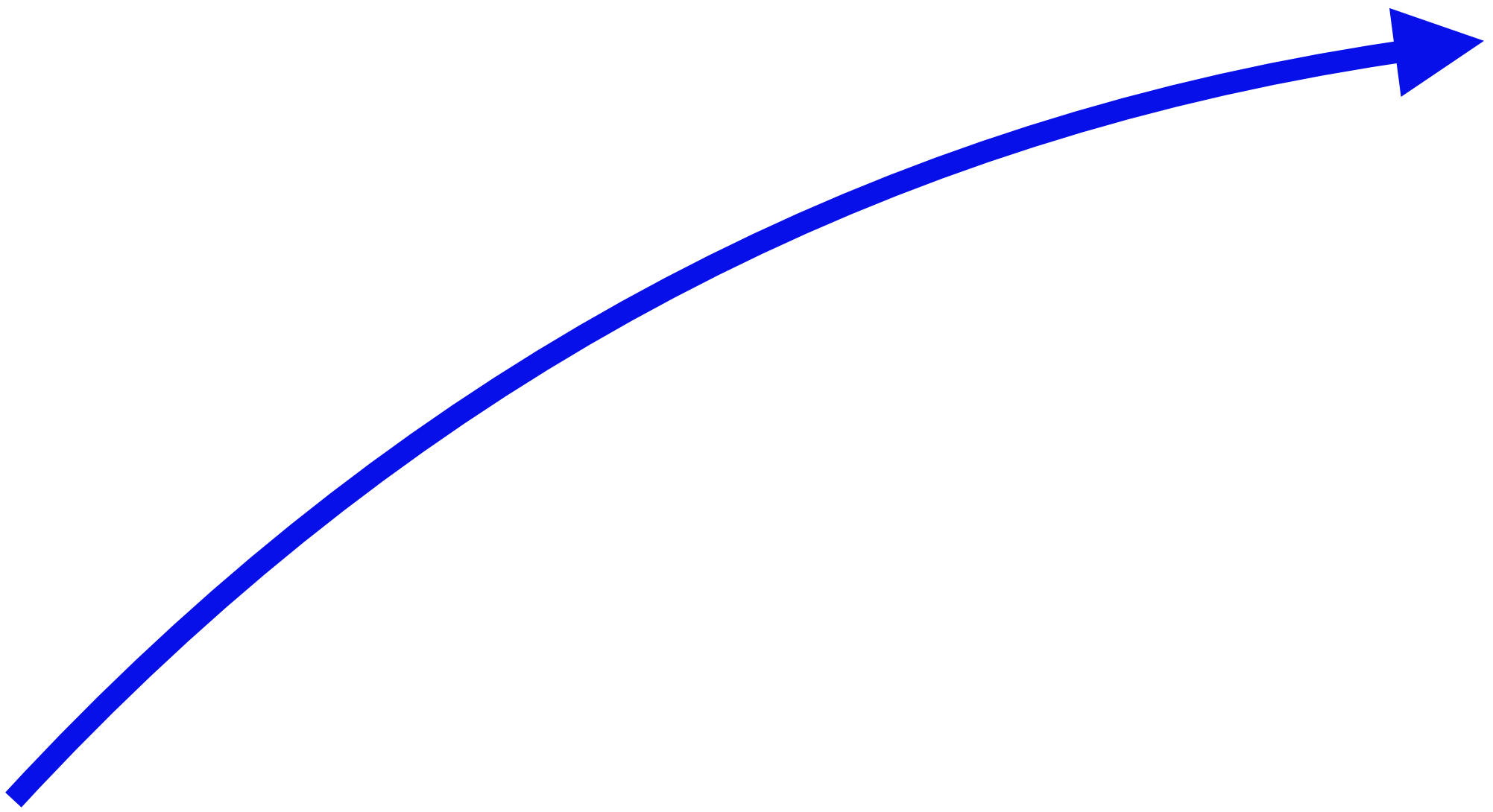

2

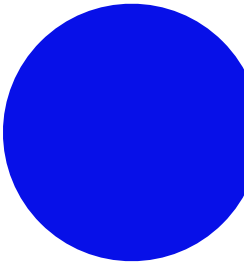

3

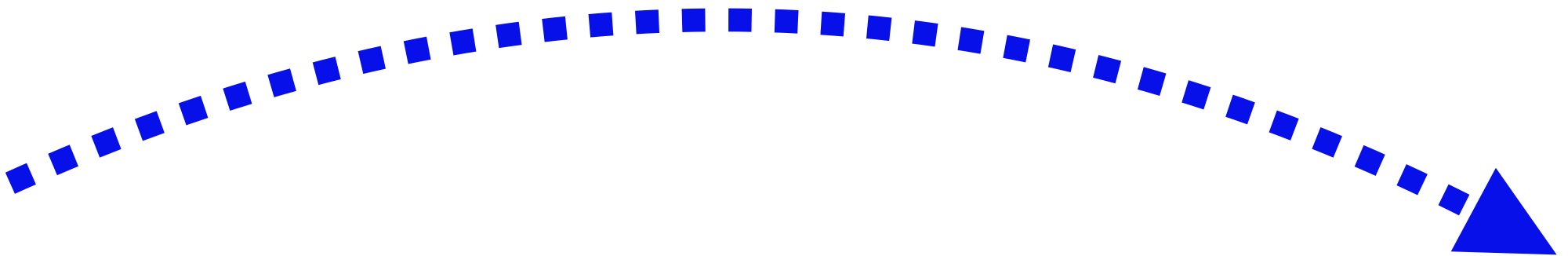
```
delete head;
```

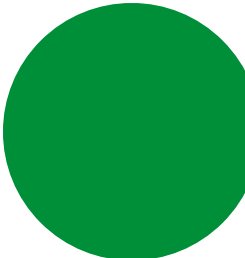


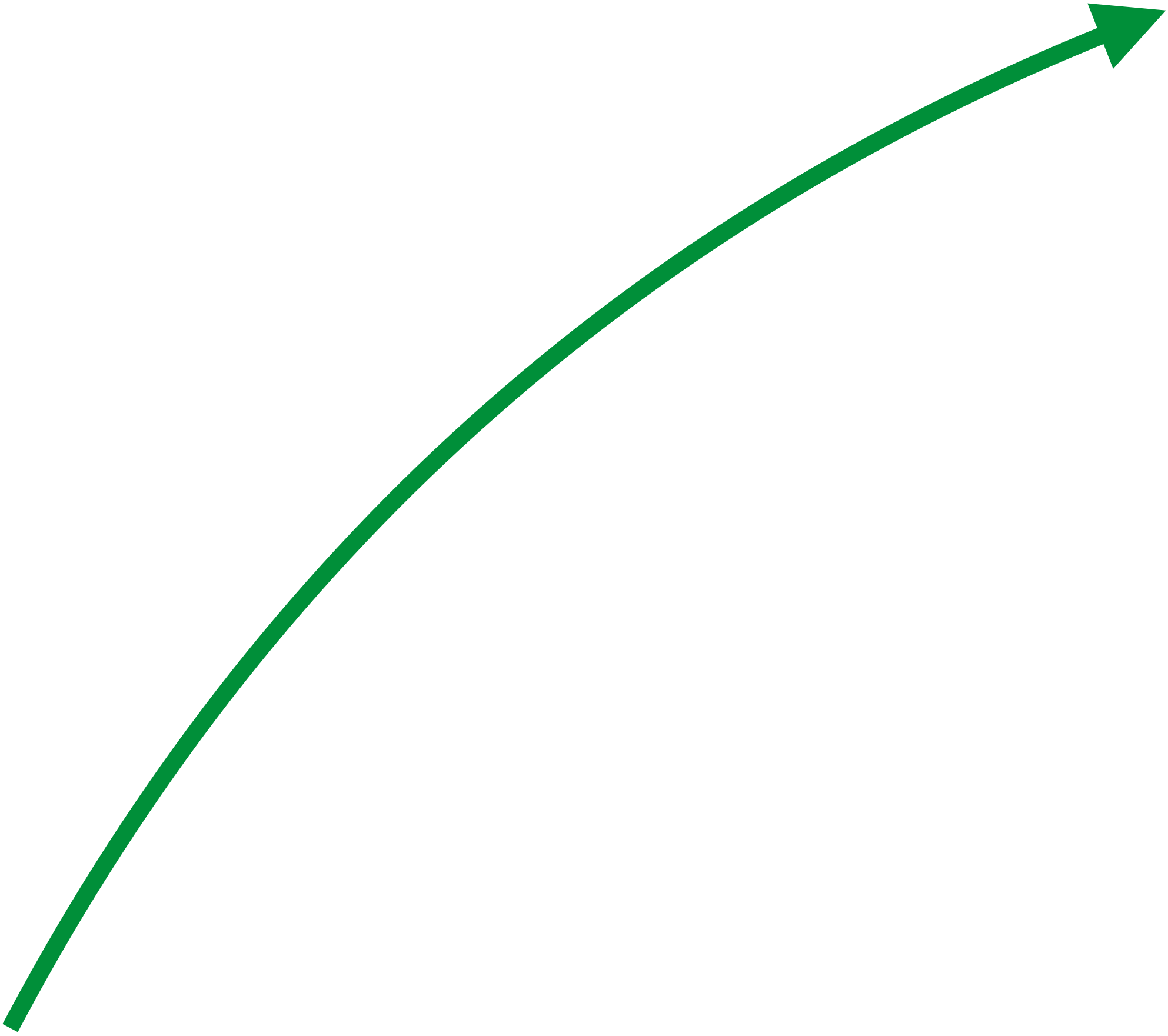


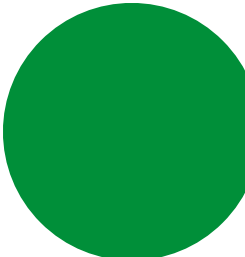


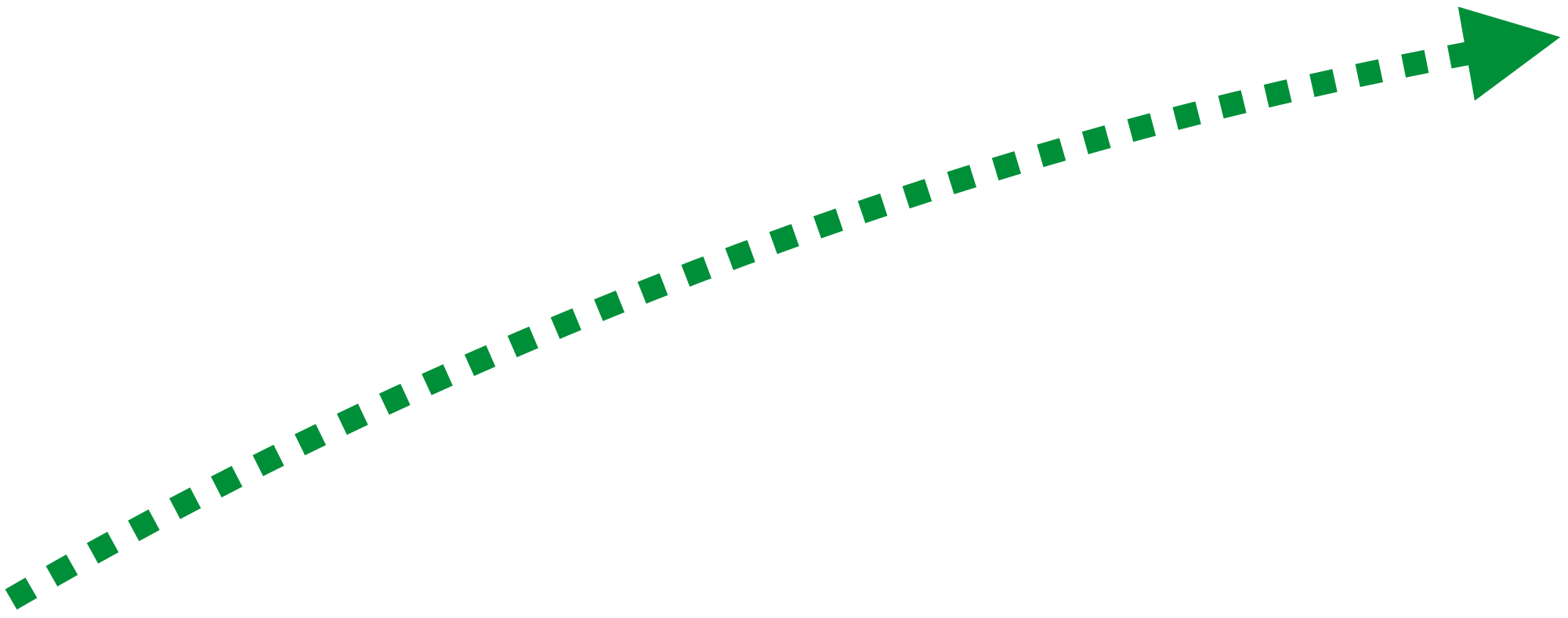


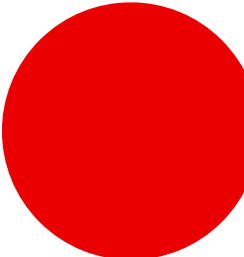


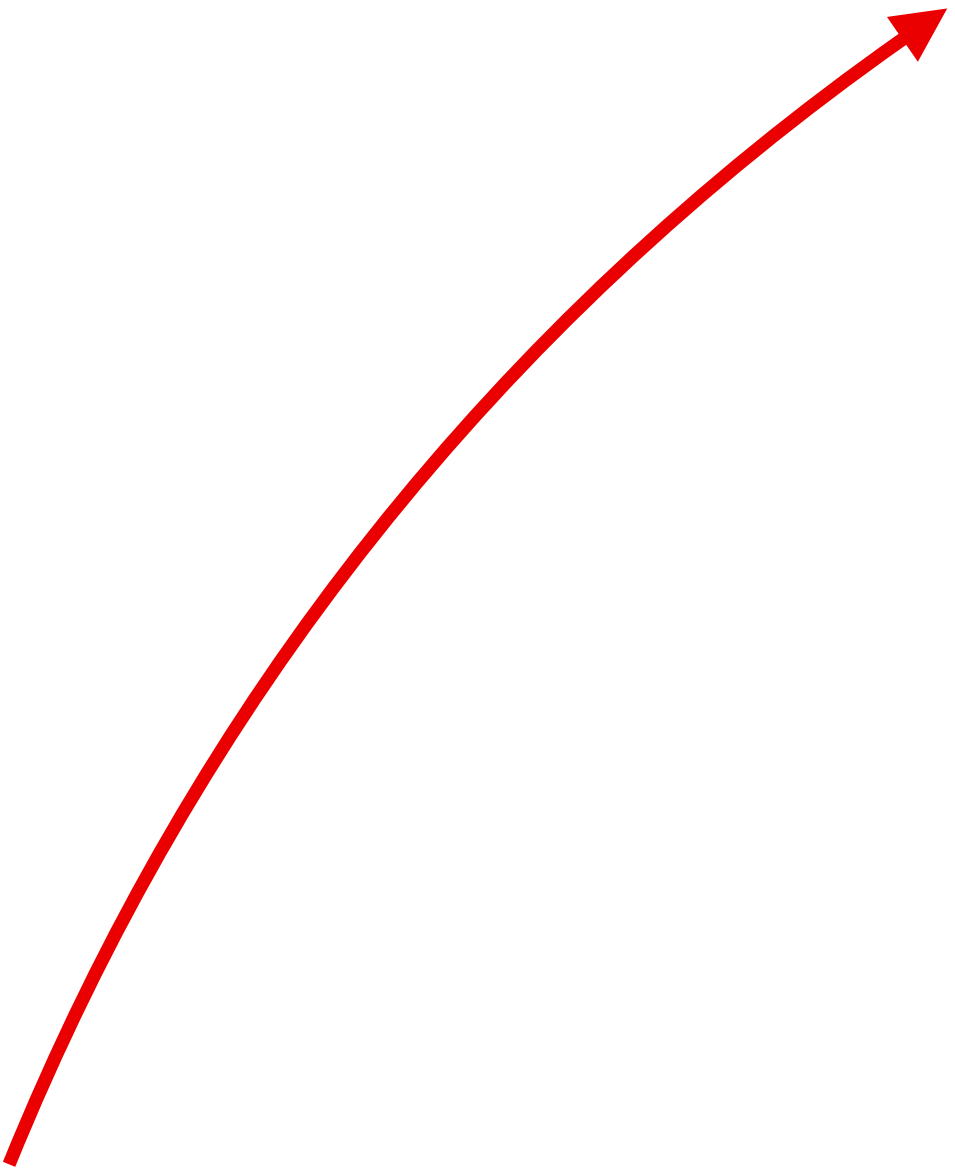


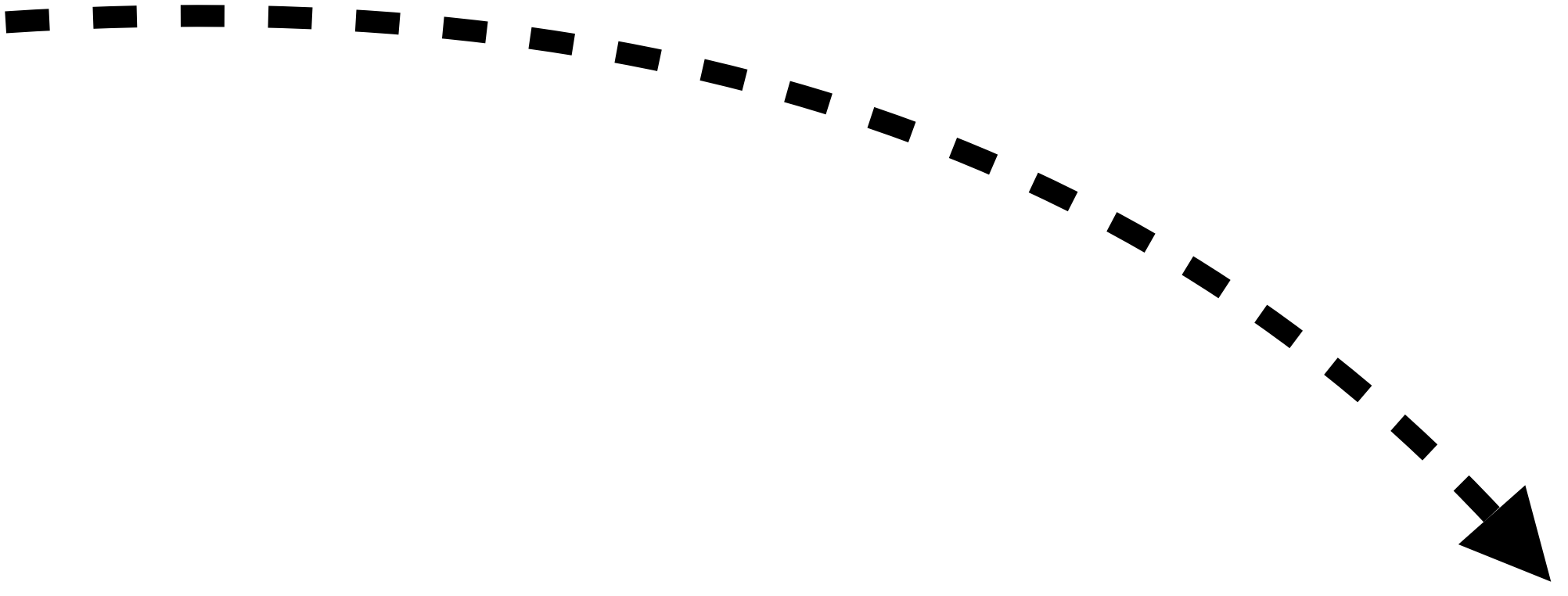


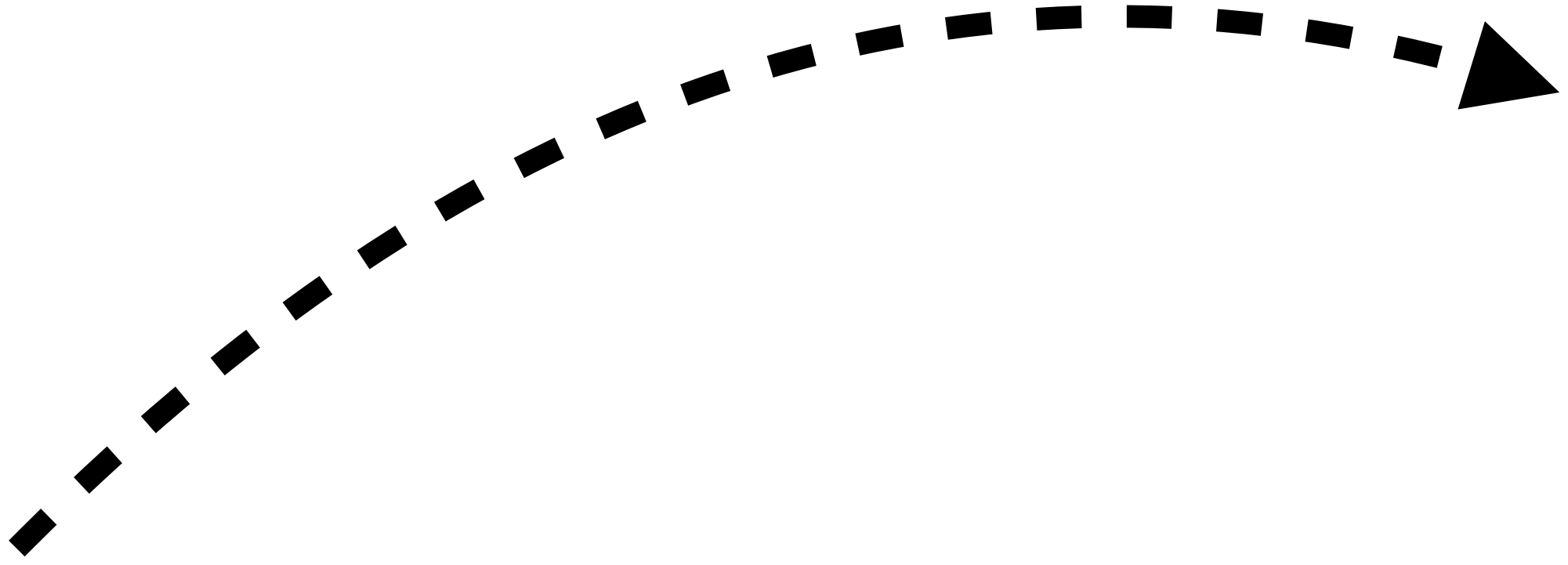


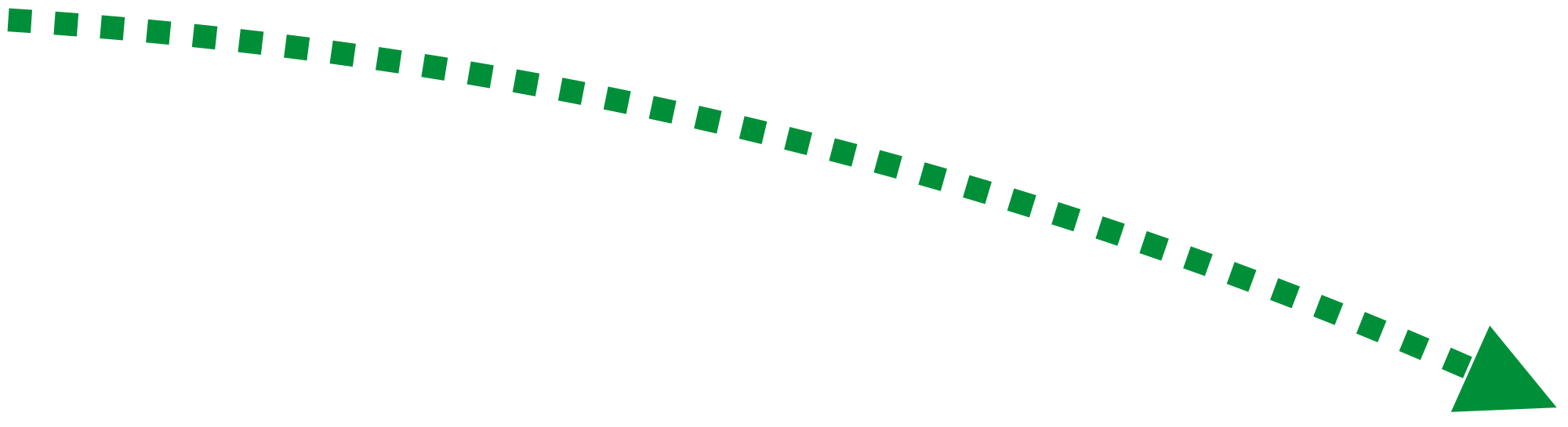


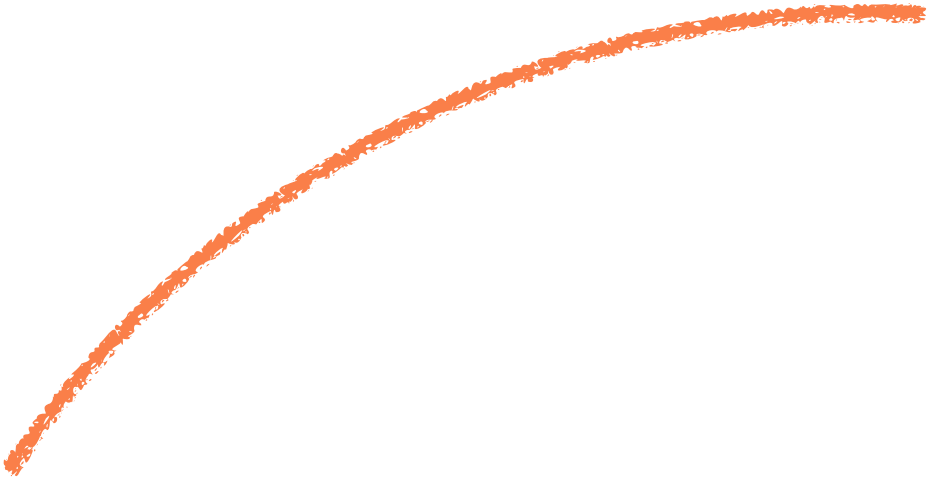


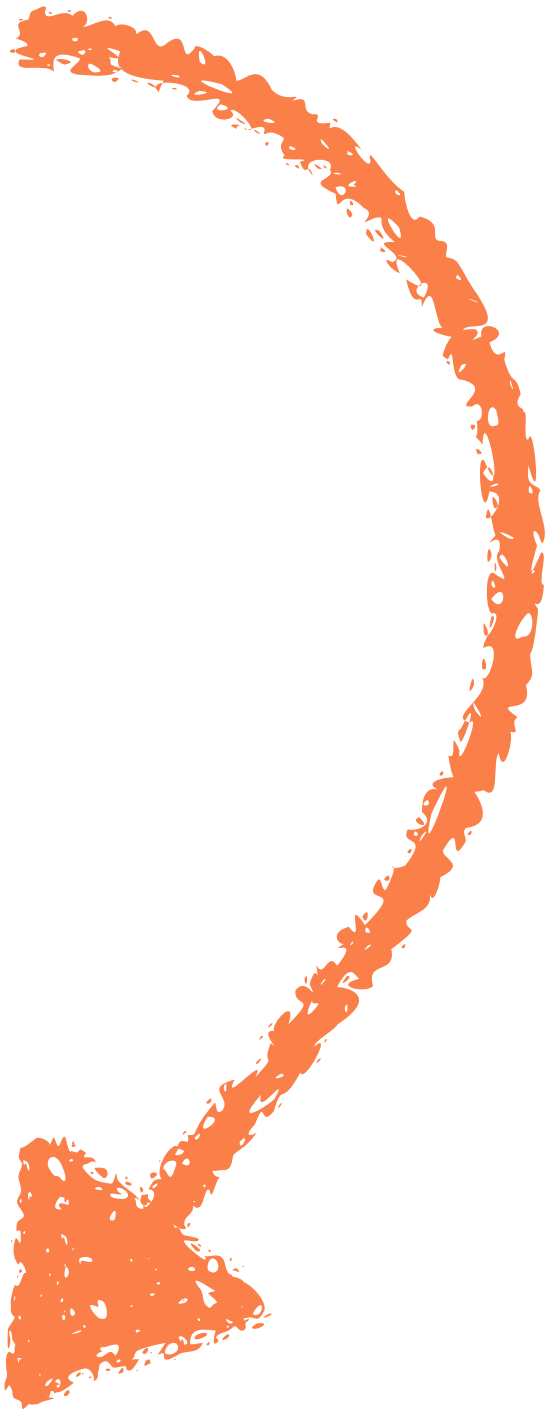


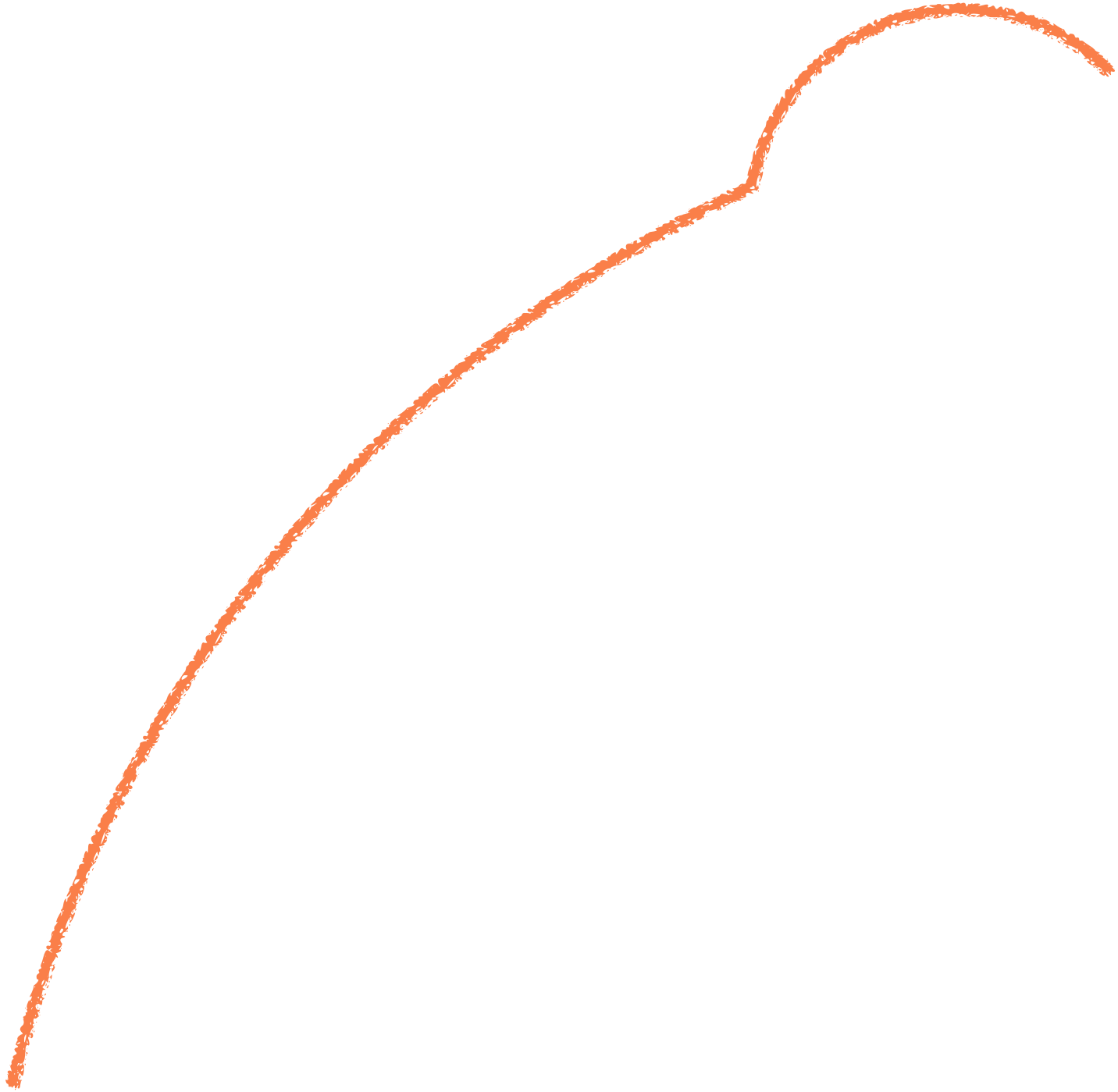












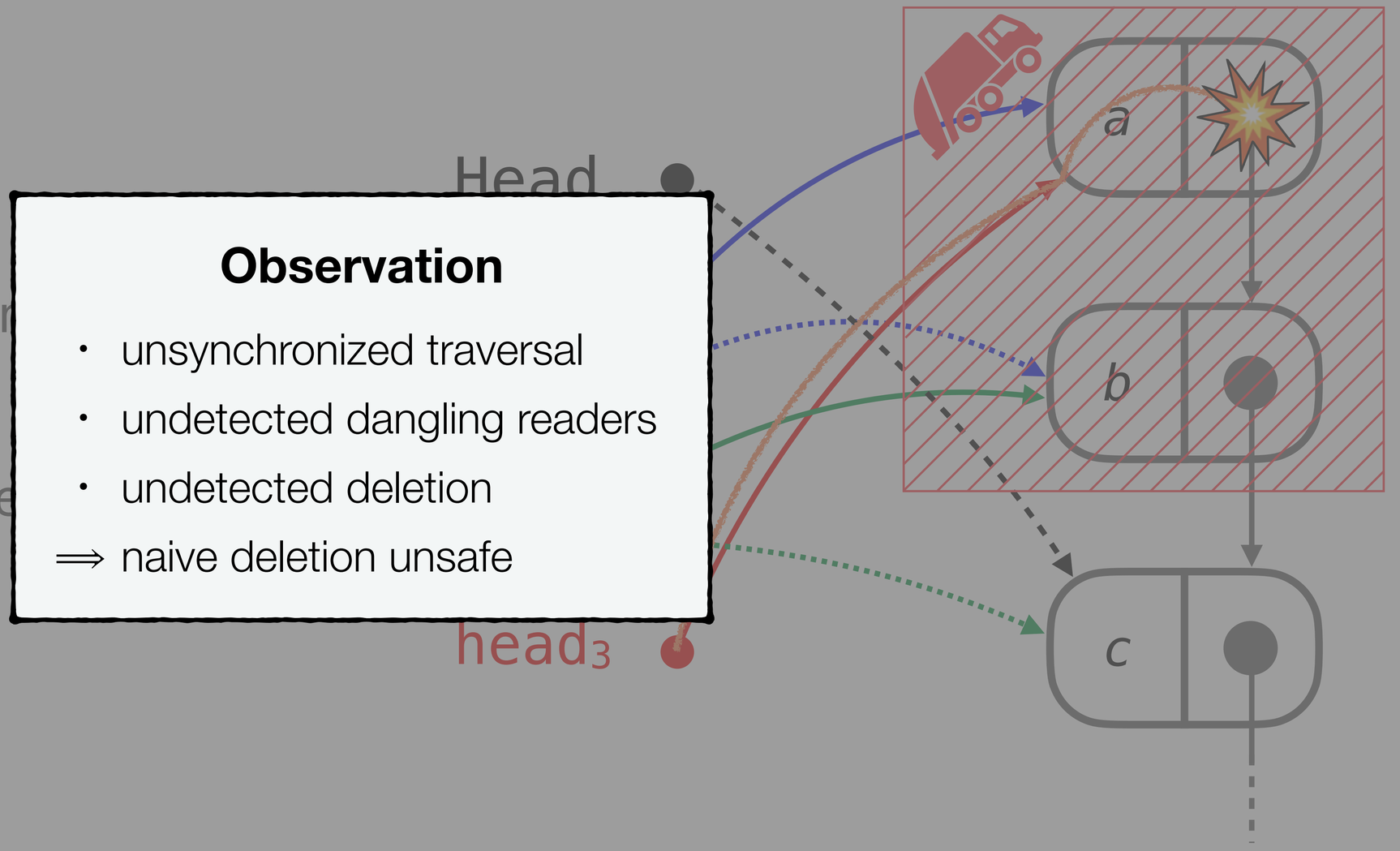
Observation

- unsynchronized traversal
- undetected dangling readers
- undetected deletion

$\Longrightarrow$ naive deletion unsafe

Non-blocking Queue (Michael&Scott)

```
void dequeue() {
  while (true) {
    head = Head;
3   next = head->n
    // ...
    if(CAS(Head,he
      delete head;
1 2   return;
}}}
```



Observation

- unsynchronized traversal
- undetected dangling readers
- undetected deletion

$\Longrightarrow$ naive deletion unsafe

Safe Memory Reclamation (SMR)

```
void dequeue() {
  while (true) {
1 2 3 head = Head;
    protect(head);
    if(head != Head) continue;
    next = head->next;
    // ...
    if(CAS(Head,head,next)){
      retire(head);
      return;
}}}
```