MAKE SCHOOL

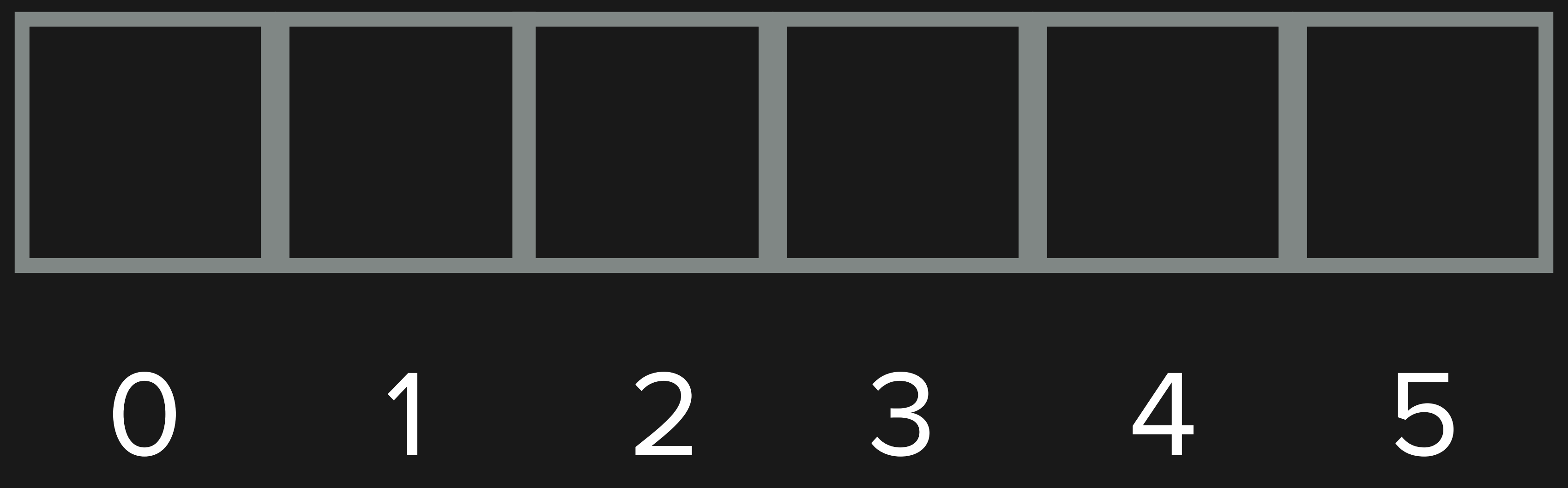
ARRAYS & LINKED LISTS

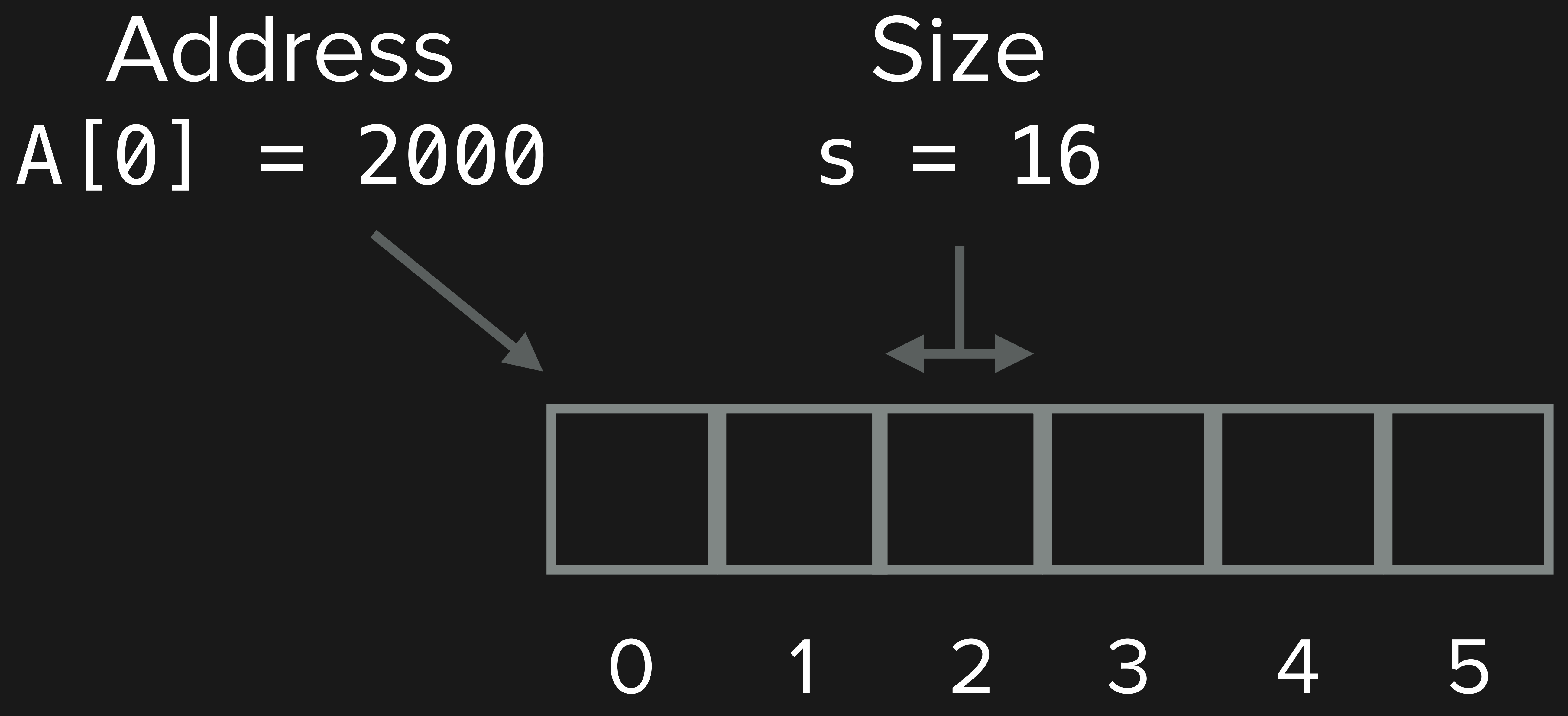
ARRAYS

Contiguous block of memory

Think of an array as a row of mailboxes with each having a unique integer address

Same item storage size at each index





Address

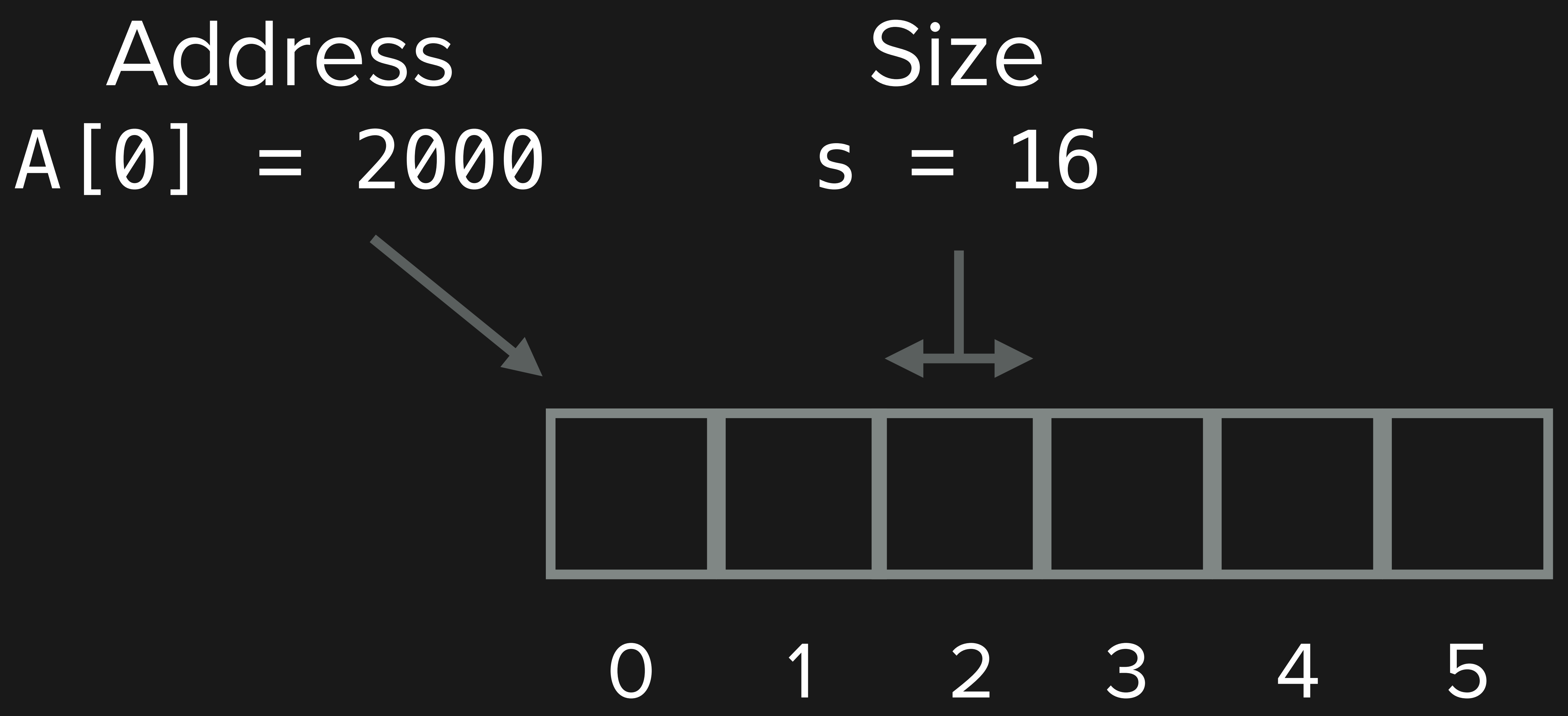
`A[0] = 2000`

Size

`s = 16`

0 1 2 3 4 5

How can we calculate the memory address for index 4?



Address

A[0] = 2000

Size

s = 16

0 1 2 3 4 5

$$A[i] = A[0] + s * i$$

$$A[4] = 2000 + 16 * 4$$

$$A[4] = 2064$$

STATIC ARRAYS

Static arrays are a direct representation of how memory is organized in physical RAM

Can’t change size because their memory is allocated once as a single contiguous block

However, we often do not know or cannot predict how many items we need to store…

DYNAMIC ARRAYS

Dynamic arrays can change size but still have to store their items in a static array of fixed size – indexes are marked as occupied or available

When the static array is out of space we need to **allocate** a larger one and **copy** all existing items into it before we can append a new item

ARRAY RUNTIME

Access item via index

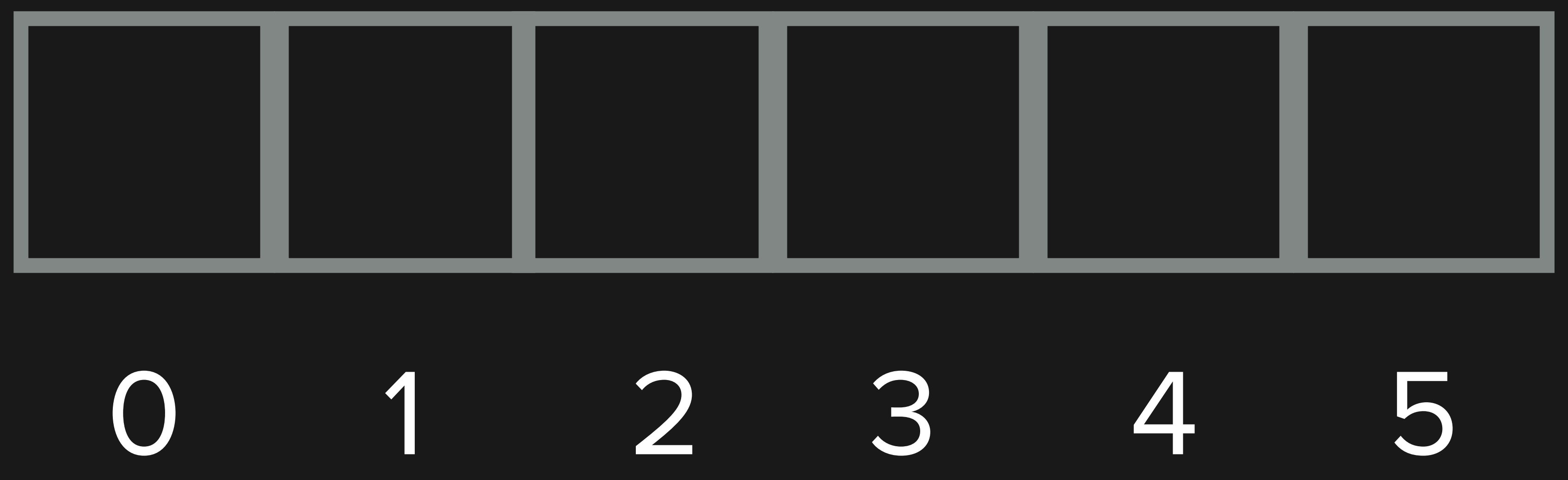
O(1)

Insert or delete item at index

Beginning: O(n)

Middle: O(n)

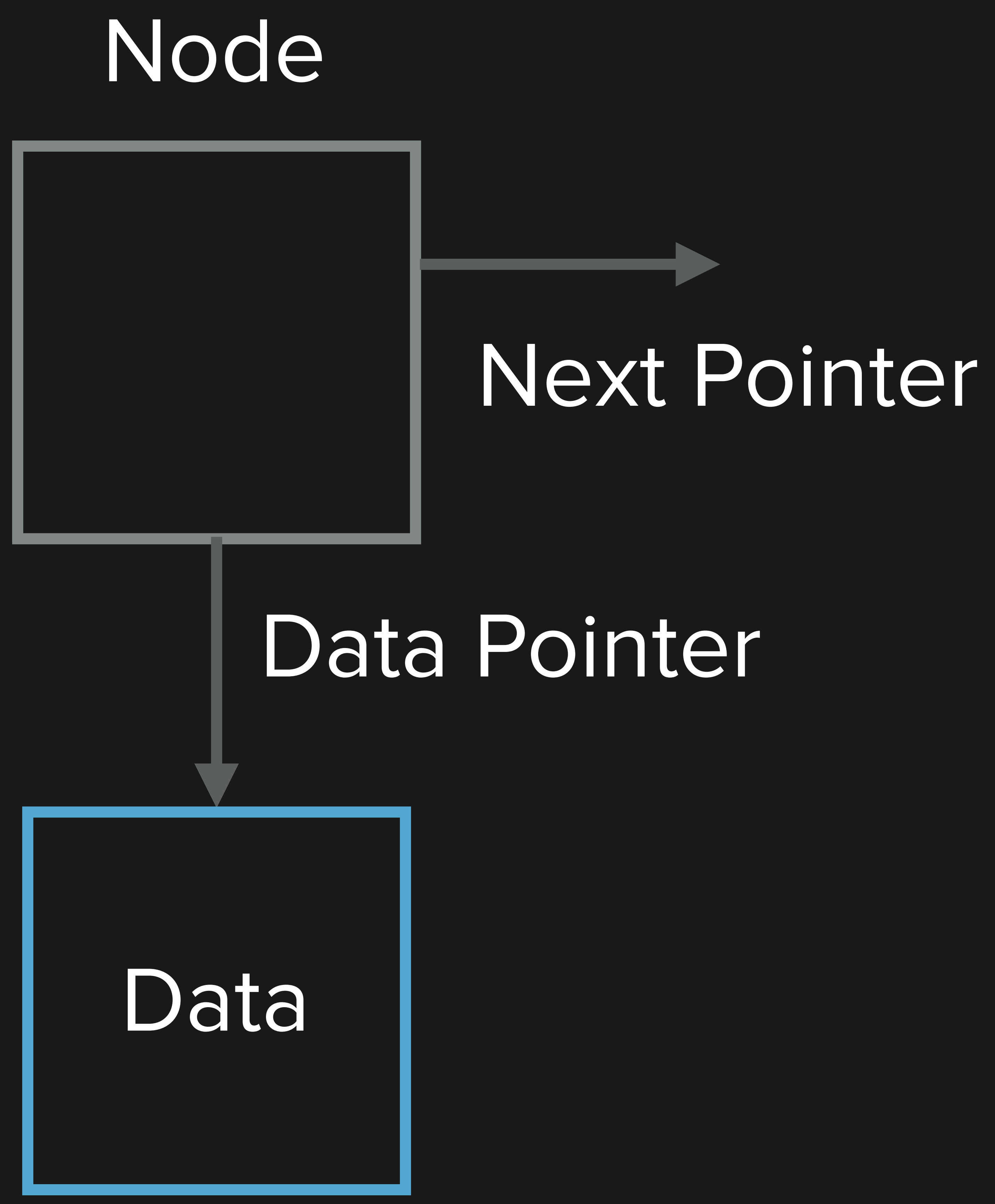
End: O(1)* – *on average*



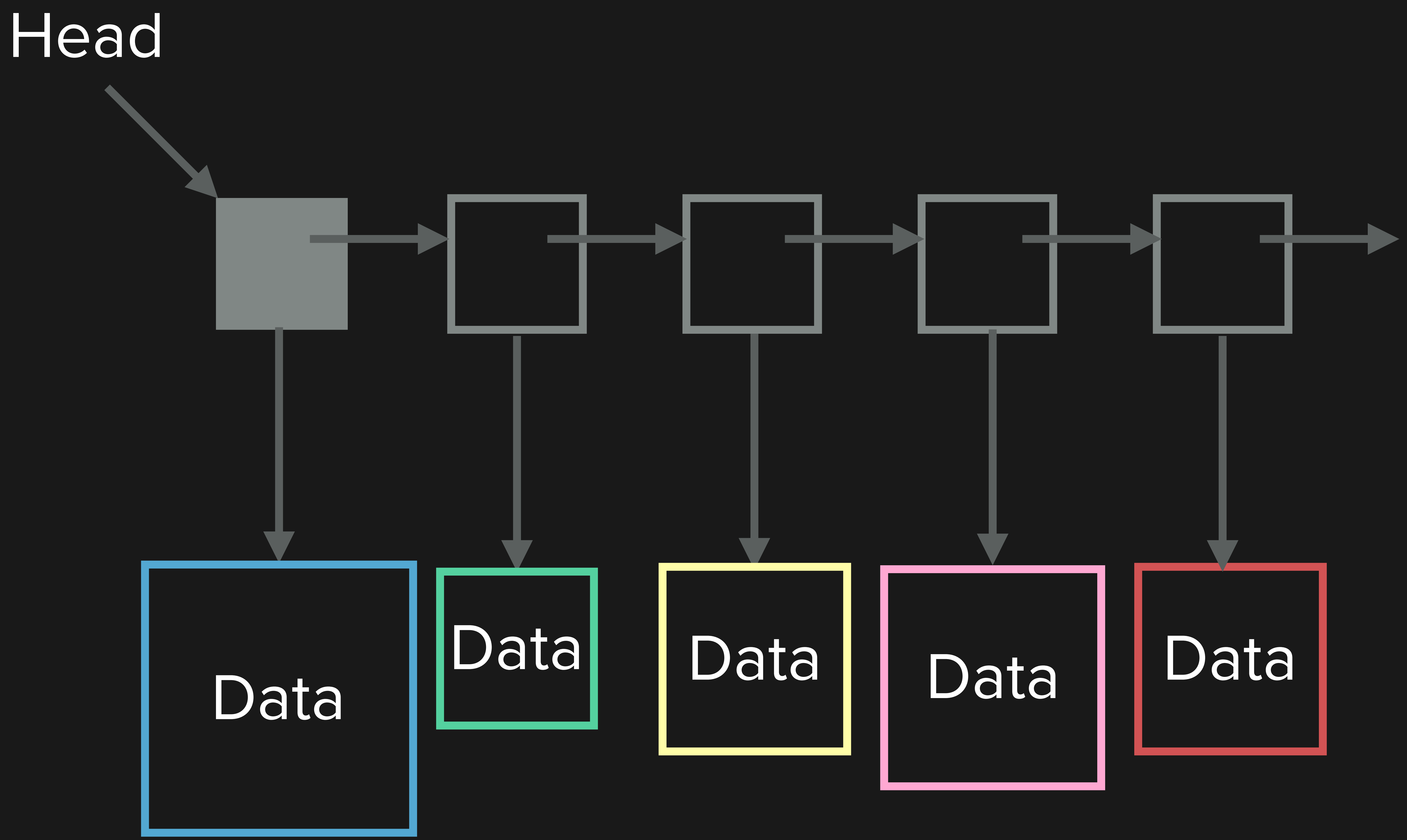
LINKED LISTS

A LINKED LIST IS LIKE A FREIGHT TRAIN

LINKED LISTS



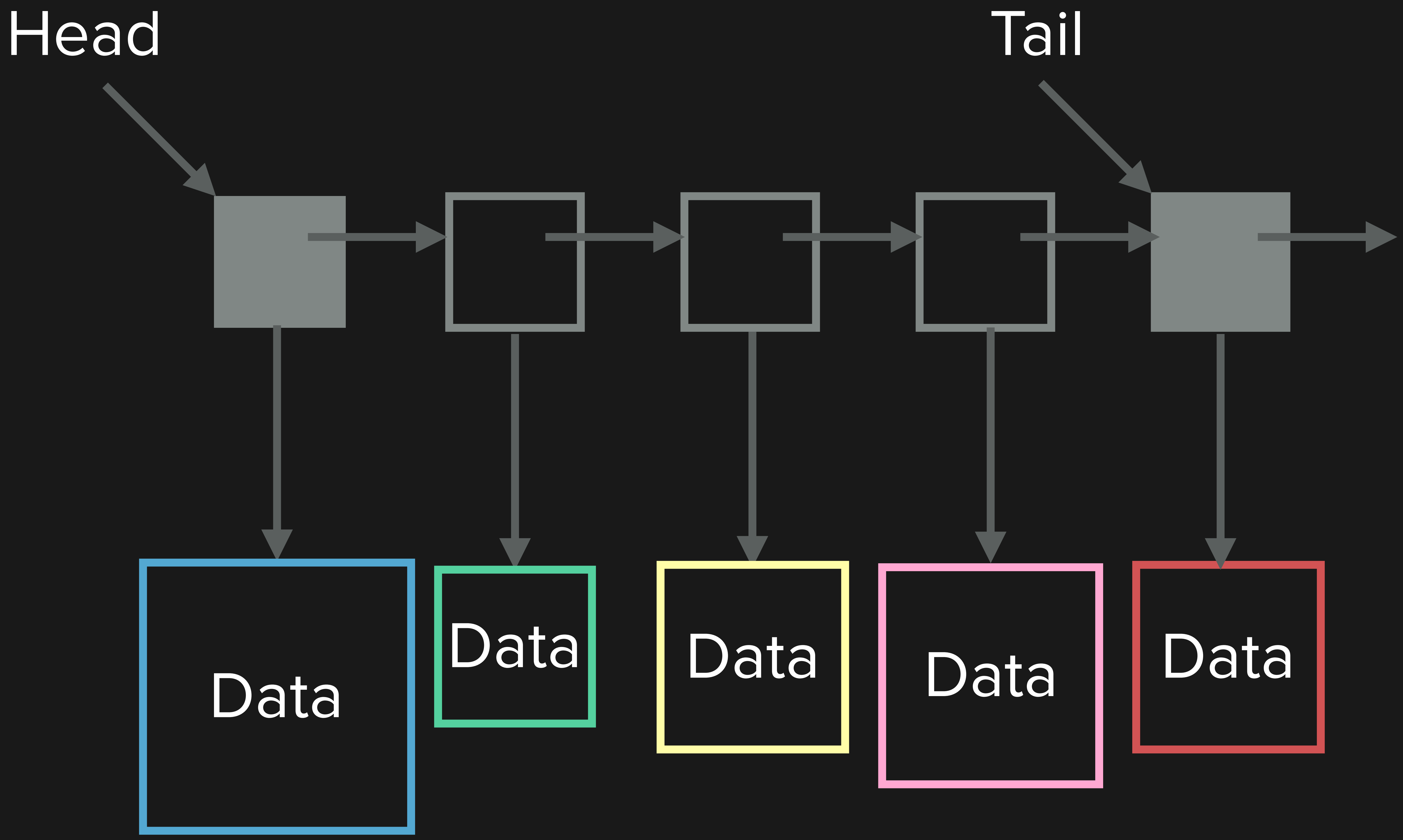
LINKED LISTS



BECOMING A LINKED LIST

- **Interpreter** – Executes 4 linked list operations:
  - `append('cats'),  append('are'),`
    `append('cool'),  prepend('fluffy')`
- **Memory allocator** – Finds a node to store data (a desk with two people to hold data and next)
- **Head pointer** – Tracks the first node in the chain

LINKED LISTS



LINKED LISTS

Not contiguous piece of memory, several small, scattered pieces strung together

Can have different storage size for each item

Dynamic: new piece of memory allocated

Never need to copy all items like an array

LINKED LIST RUNTIME

Access item by searching

O(n)

Insert or delete item

Beginning: O(1)

Middle: O(n)

End: O(1) – *with tail*