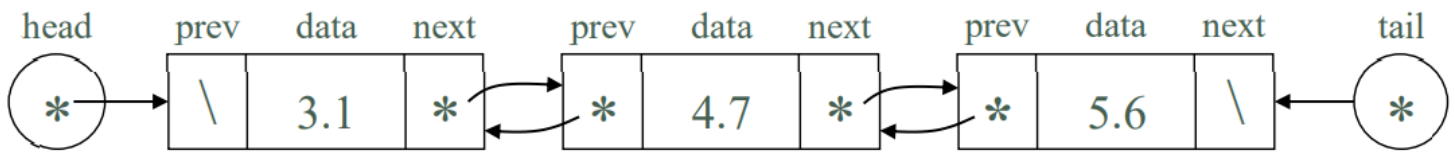
Double and Circular List

Doubly Linked List

Doubly Linked Lists

- Enhance *singly linked lists* by adding pointers to predecessor nodes
  - Allow list traversal from tail to head



Doubly Linked List

- Insertion and deletion are trickier to implement
  - Especially at the start or end of the list, or when the list is, or about to become, empty

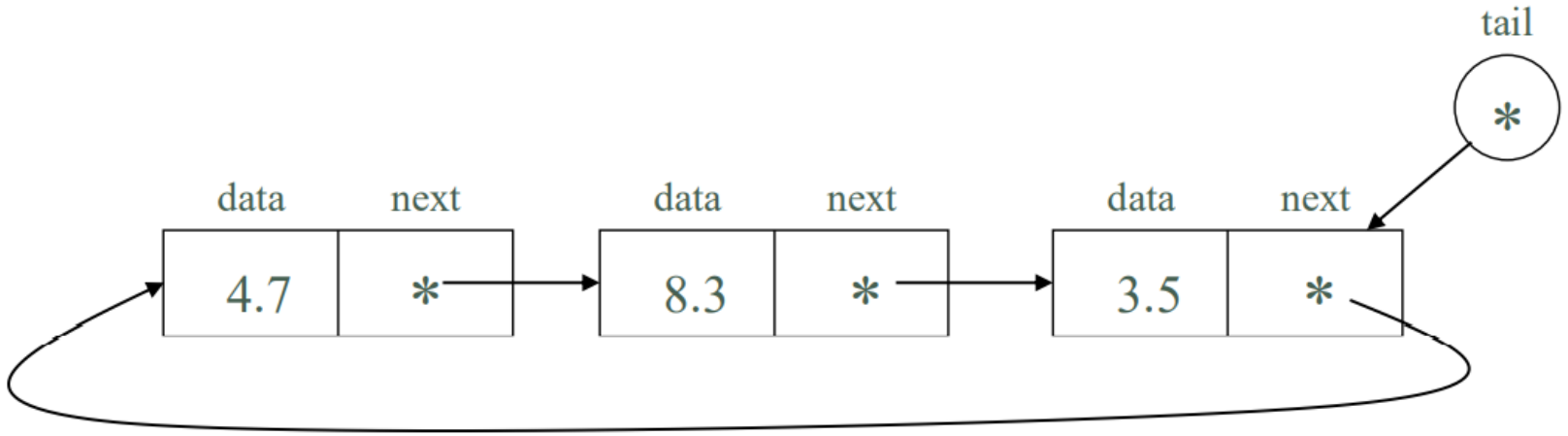
- Java provides a generic implementation with the class java.util.LinkedList

Why Doubly Linked List?

- **Pros**
  - access from either end
  - don’t need to track previous within code
- **Cons**
  - more memory
  - slightly more complex code to maintain extra reference

Circular List

- Are linked lists where the last node points to the first node

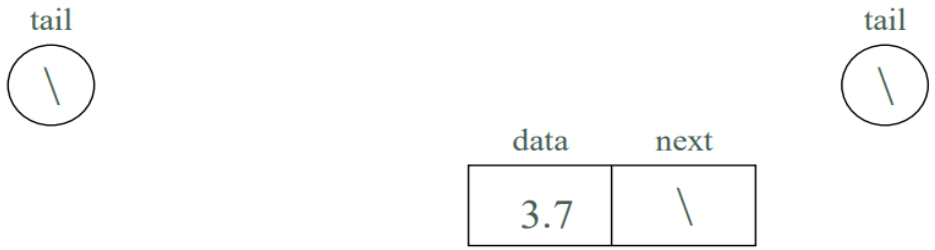


Circular List

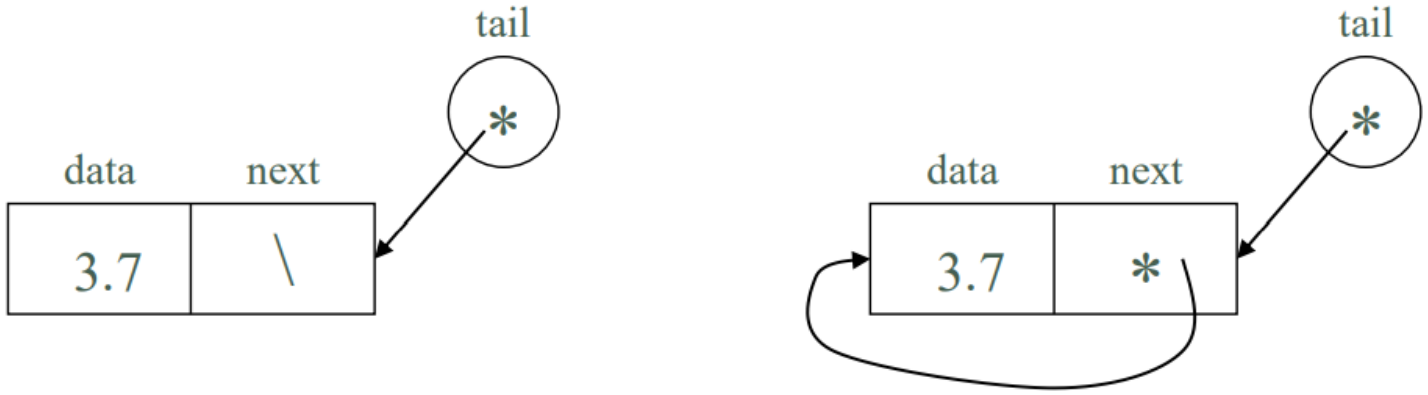
- Only a tail pointer is needed, since the head is pointed to by `tail.getNext()`
- To insert into an empty list:
  - Allocate a new node
    - Set the *data* field to the desired value
  - Assign the node's address to the tail pointer
  - Set the node's *next* field to the same value (a self reference)

Circular List

- E.g.
```
tail = new Node(3.7, null);
tail.setNext(tail);
```



Circular List

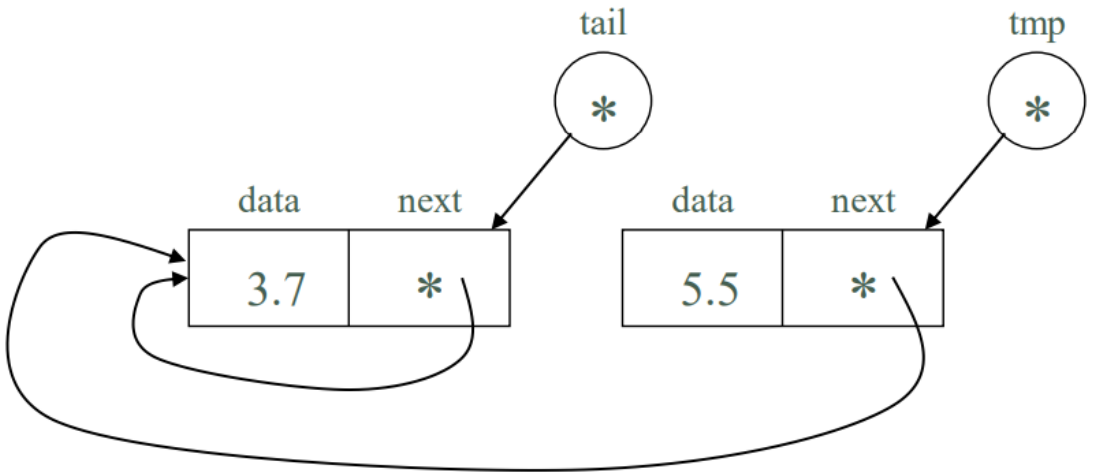


Circular List

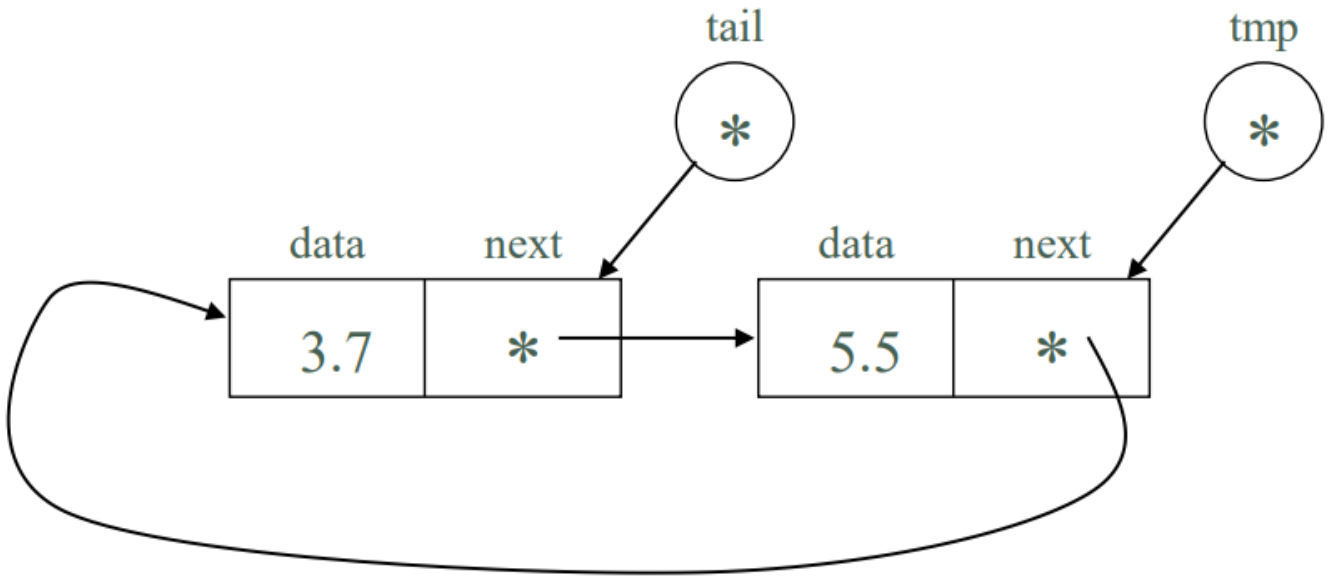
- To insert into the end of the list:
  - Allocate a new node
    - Use a temporary pointer to point to it
    - Set the *data* field to the desired value
    - Set the *next* field to tail.next
  - Set tail.next to the temporary variable
  - Set tail to the temporary variable

Circular List

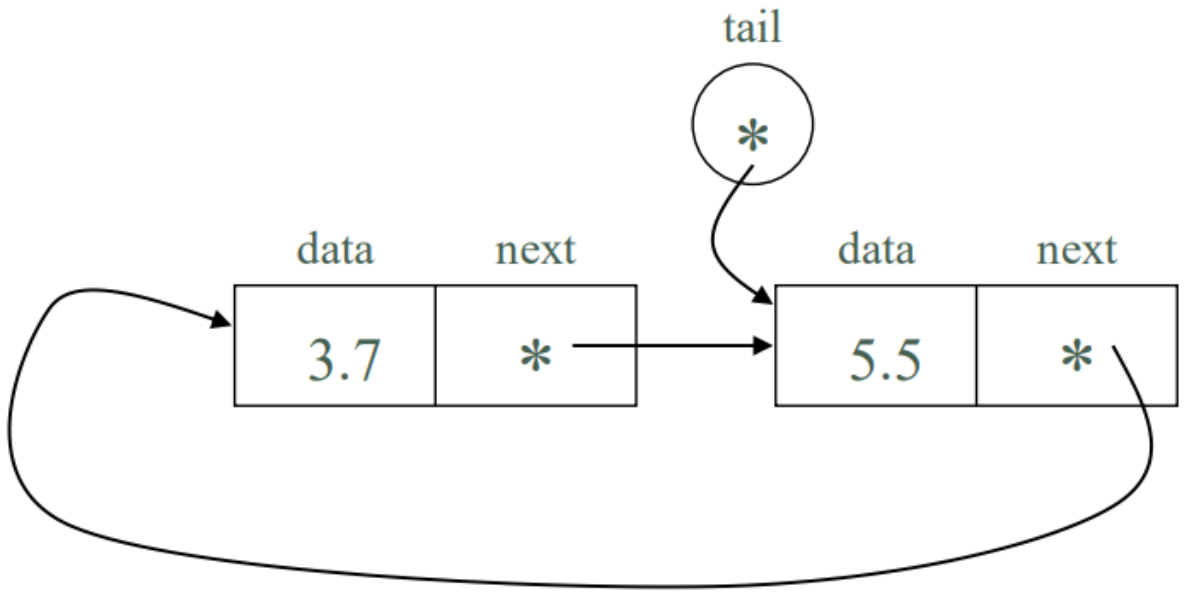
- E.g.
```
tmp = new Node(5.5, tail.getNext());
tail.setNext(tmp);
tail = tmp;
```



Circular List



Circular List



DLLs compared to SLLs

- **Advantages:**
  - Can be traversed in either direction (may be essential for some programs)
  - Some operations, such as deletion and inserting before a node, become easier
- **Disadvantages:**
  - Requires more space
  - List manipulations are slower (because more links must be changed)
  - Greater chance of having bugs (because more links must be manipulated)

Linked List in Games?

- Things for which you don’t want a finite number
- **Role-playing games**:
  - Player can only carry 32 items
  - Give him a 32-item array?
  - What if different types of characters can gradually carry more items as they grow stronger?
  - Let’s use linked lists instead

Linked Lists: Algorithm Comparisons

Algorithm	Array	Singly Linked (no tail)	Singly Linked (tail)	Doubly Linked (tail)
`size()`	O(1)	O(1)	O(1)	O(1)
`isEmpty()`	O(1)	O(1)	O(1)	O(1)
`isFull()`	O(1)	O(1)	O(1)	O(1)
`get(i)`	O(1)	O(n)	O(n)	O(n)
`set(i,e)`	O(1)	O(n)	O(n)	O(n)
`add(i,e)`	O(n)	O(n)	O(n)	O(n)
`remove(i)`	O(n)	O(n)	O(n)	O(n)

Real-World Issues in Games

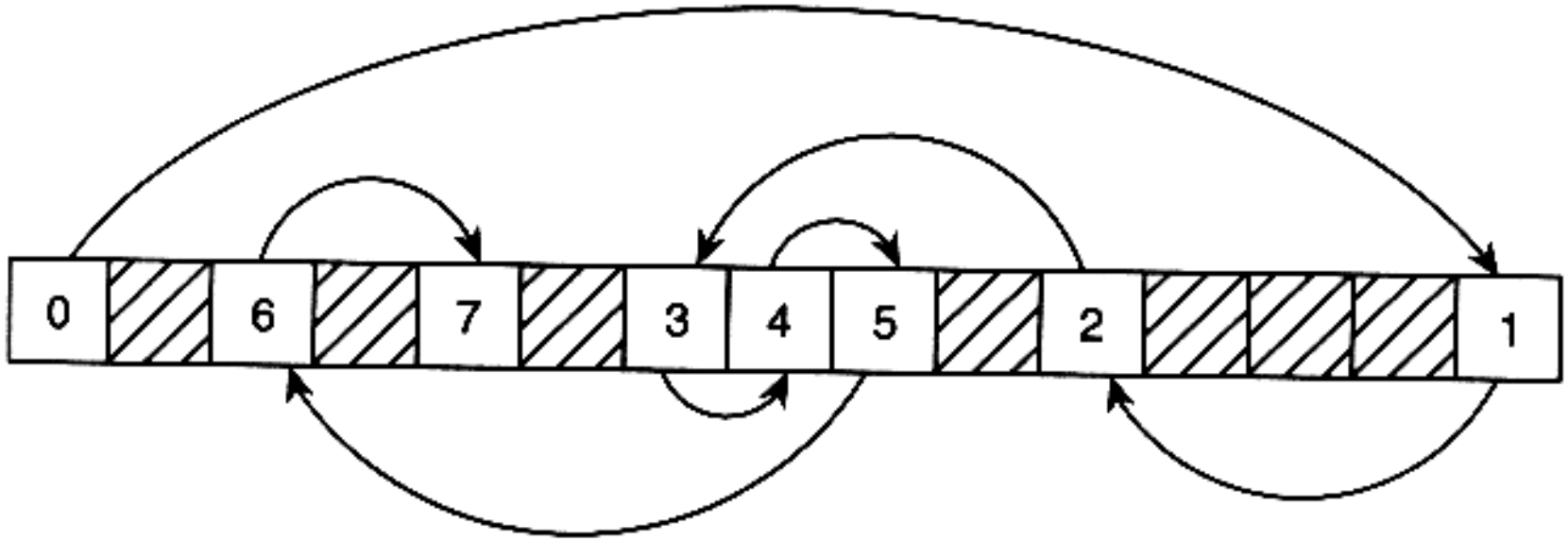
- Linked lists in games don’t have many uses if you want to make your game super-fast…

- Why? Caching!

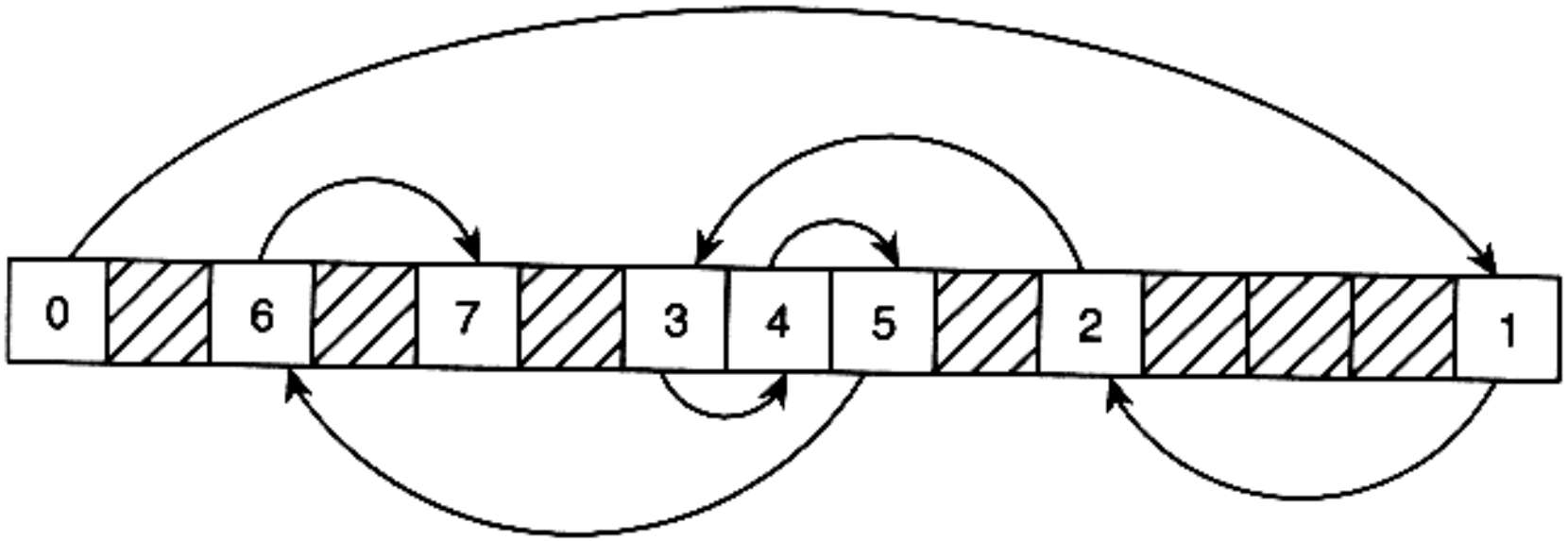
Caches

- Load entire chunks of memory into ultra-fast memory
- It works great with arrays because an array is a chunk of memory
- Linked list is not a chunk of memory → nodes can be anywhere in memory

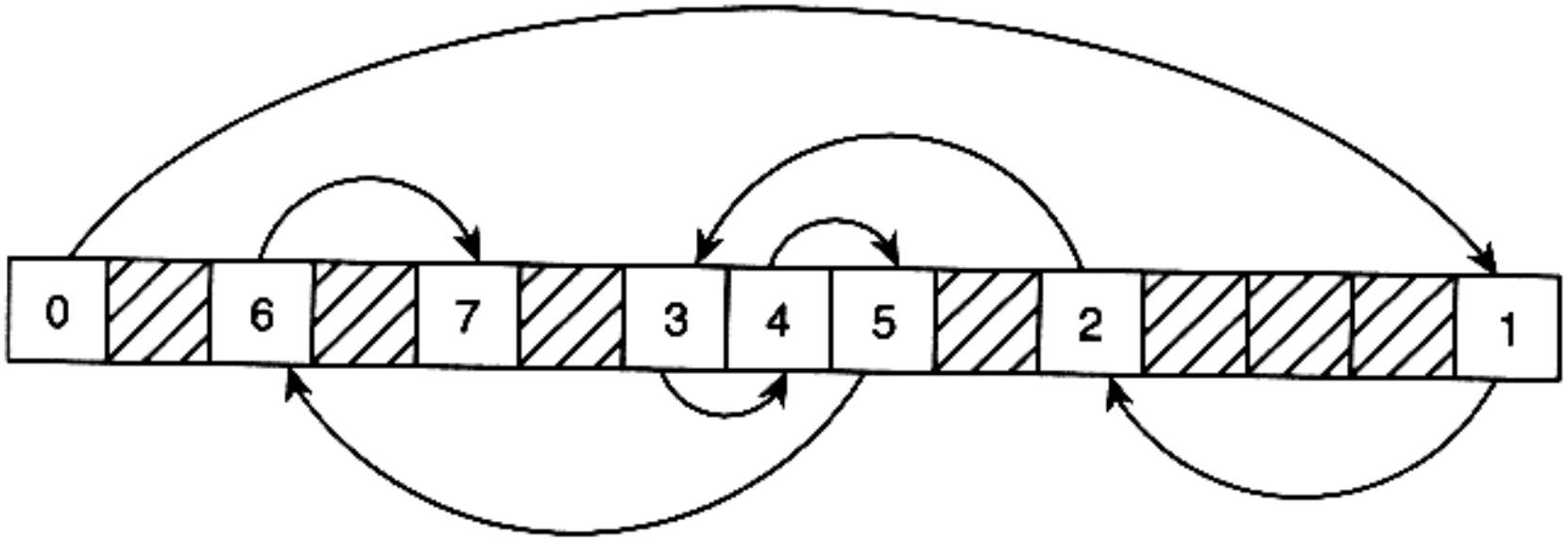
Linked list nodes in memory



- Each block → one node
- Shaded blocks → memory positions separating the nodes in memory



- Nodes jump all over the place in memory
- This isn't like an array at all!



- Result → cache is constantly swapping blocks of memory in and out
- This is the same effect of randomly accessing elements in an array!

Bottom Line

- Lists are generally slower than arrays when performing small algorithms on every item in the list

- A large algorithm that does a lot of work on each node in the list, then the overhead of the cache swapping is diminished greatly

Bottom Line

- Remember that every time you create a new node, you tell the computer to allocate more memory, which is slow

- End result?

  Don’t use linked lists for things that require little processing or things that will be created and destroyed quickly

Example

- Maintaining information about the number of bullets flying around in a game at any time

Example

- Maintaining information about the number of bullets flying around in a game at any time
- Linked lists? NO!!
- Life time of a bullet = a few frames at most (1/10 of a second) → you would have to delete it almost immediately after creating it!
- Create a large array to store all these bullets instead