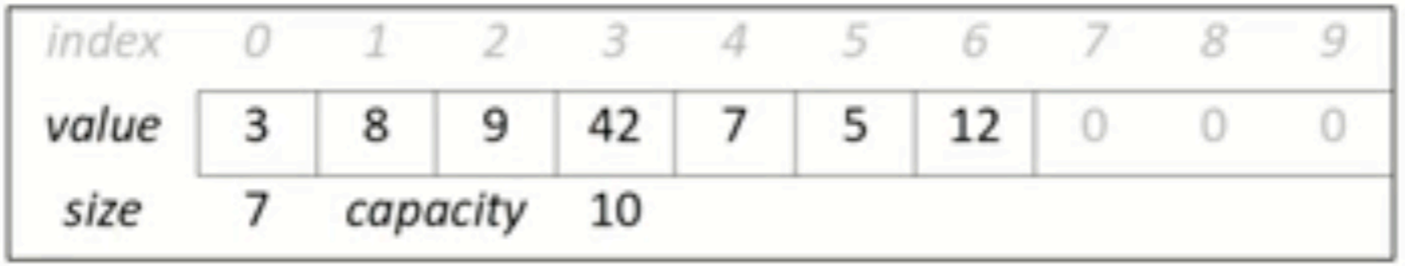
Vector/LL efficiency



Member	Vector	LinkedList
`add(value);`	O(1)	O(1)
`get(i) or [i]`	**O(1)**	**O(N)***
`insert(i, value);`	**O(N)***	**O(N)***
`remove(i);`	**O(N)***	**O(N)***
`set(i, val) or [i]=`	**O(1)**	**O(N)***
`size(), isEmpty()`	O(1)	O(1)
`toString(), cout << v`	**O(N)**	**O(N)**

average case runtime;

Vector = O(1) at end, worst at front;

LinkedList = O(1) at front and end, worst in middle

Abstract data types (ADTs)

- **abstract data type (ADT)**: A specification of a collection of data and the operations that can be performed on it.
  - Describes *what* a collection can do, not *how* it does it.
  - We could say that both `Vector` and `LinkedList` implement the operations of the *abstract data type* called "**list**".
    - other examples of ADTs: stack, queue, set, map, graph

- We don't always know exactly how a given collection is implemented internally, and we don't need to.
  - We just need to understand the idea of the collection and what operations it can perform.

okay so anyway this is an interesting idea this example of edts