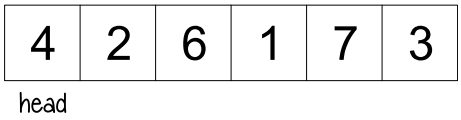
Cola de valores enteros

∗ Escribe una clase `IntegerQueue` que represente una cola de valores enteros ∗



Implementa, *al menos*, los siguientes **métodos**:

```
def __init__(self, *, max_size: int = 10):
```

- Crea los atributos: `items` y `max_size`.
- Ten en cuenta que `items` almacenará los elementos de la cola.

```
def enqueue(self, item: int) -> bool:
```

- Añade `item` a la cola `self`.
- Si la cola está llena habrá que retornar `False`. En otro caso retornar `True`.

```
def dequeue(self) -> int:
```

- Extrae (y devuelve) el elemento que está en el ***head*** de la cola `self`.

```
def head(self) -> int:
```

- Devuelve el elemento que está en el ***head*** de la cola `self`.

```
def is_empty(self) -> bool:
```

- Indica si la cola `self` está vacía.

```
def is_full(self) -> bool:
```

- Indica si la cola `self` está llena.

```
def expand(self, factor: int = 2) -> None:
```

- Expande el tamaño máximo de la cola `self` en el factor indicado.
- Por ejemplo, si el tamaño máximo es 20 y el factor es 3, el tamaño máximo pasa a ser $20 \times 3 = \mathbf{60}$.

```
def dump_to_file(self, path: str) -> None:
```

- Vuelca la cola `self` a un fichero con ruta `path`.
- Todos los elementos de la cola estarán en una misma línea separados por comas.
- El primer elemento del fichero coincide con el elemento del ***head*** de la cola.

`def load_from_file(cls, path: str) -> IntegerQueue:`

- Es un **método de clase**
- Construye (y devuelve) una cola desde el fichero con ruta `path`.
- Todos los elementos de la cola estarán en una misma línea separados por comas.
- El primer elemento del fichero coincide con el elemento del ***head*** de la cola.
- Si la cola se llena al ir añadiendo elementos habrá que expandir con los valores por defecto.

`def __getitem__(self, index: int) -> int:`

- Devuelve el elemento de la cola `self` que ocupa la posición (*índice*) `index`.

`def __setitem__(self, index: int, item: int) -> None:`

- Asigna el elemento `item` en la posición (*índice*) `index` de la cola `self`.

`def __add__(self, other: IntegerQueue) -> IntegerQueue:`

- Suma las colas `self` y `other`.
- La segunda cola `other` va "detrás" de la primera `self`.
- El tamaño máximo de la cola resultante es la suma de los tamaños máximos de cada cola.
- Devuelve la cola resultante.

`def __iter__(self) -> IntegerQueueIterator:`

- Devuelve un objeto iterador de tipo `IntegerQueueIterator`.

∗ Escribe una clase `IntegerQueueIterator` que pueda iterar sobre `IntegerQueue` ∗

Implementa, *al menos*, los siguientes **métodos**:

`def __init__(self, queue: IntegerQueue):`

`def __next__(self) -> int:`

