```cpp
template<typename T>
class drow_queue_v1
{
public:
    drow_queue_v1 (size_t capacity_)
        : capacity (capacity_)
    {}

    bool try_push (const T&);
    bool try_pop (T&);

private:
    size_t capacity = 0;
    std::vector<T> data { std::vector<T> (capacity) };
    std::atomic<size_t> head { 0 }, tail { 0 };

    size_t next_index (size_t current) const
    {
        return (current + 1) % capacity;
    }
};
```

```
bool try_push (const T& v)
{
    size_t current_tail = tail;
    size_t next_tail = next_index (current_tail);

    if (next_tail == head)
        return false;

    data[current_tail] = v;
    tail = next_tail;
    return true;
}
```

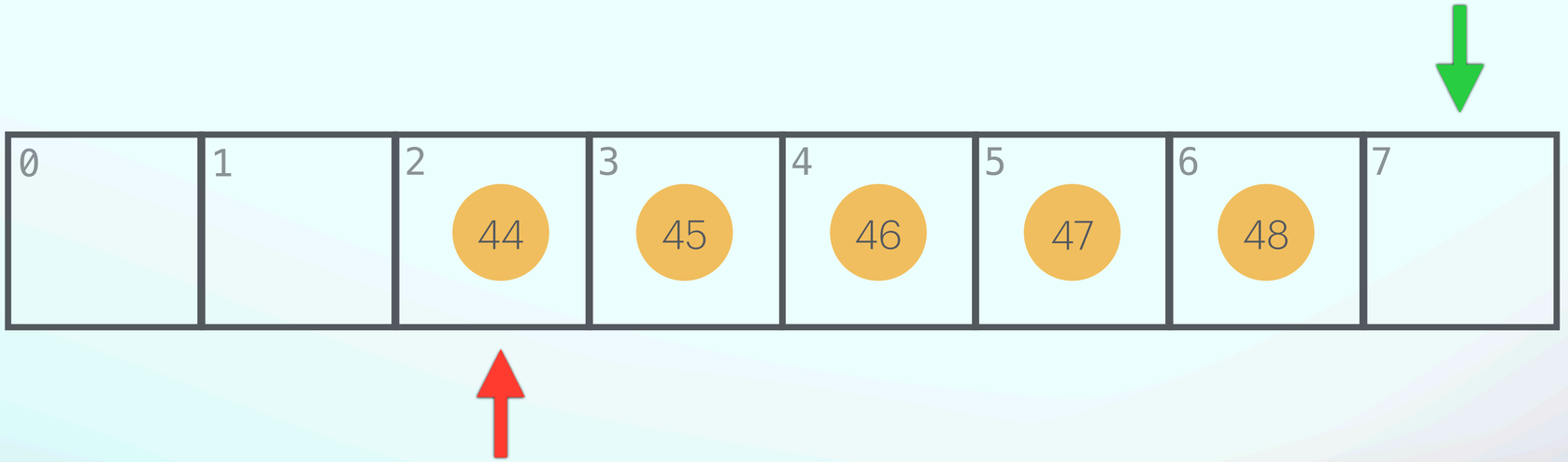
```
bool try_pop (T& v)
{
    size_t current_head = head;

    if (current_head == tail)
        return false;

    v = data[current_head];
    head = next_index (current_head);
    return true;
}
```

```cpp
template<typename T>
class drow_queue_v1
{
public:
    drow_queue_v1 (size_t capacity_)
        : capacity (capacity_)
    {}

    bool try_push (const T&);
    bool try_pop (T&);

private:
    size_t capacity = 0;
    std::vector<T> data { std::vector<T> (capacity) };
    std::atomic<size_t> head { 0 }, tail { 0 };

    size_t next_index (size_t current) const
    {
        return (current + 1) % capacity;
    }
};
```

```cpp
bool try_push (const T& v)
{
    size_t current_tail = tail;
    size_t next_tail = next_index (current_tail);

    if (next_tail == head)
        return false;

    data[current_tail] = v;
    tail = next_tail;
    return true;
}
```

```cpp
bool try_pop (T& v)
{
    size_t current_head = head;

    if (current_head == tail)
        return false;

    v = data[current_head];
    head = next_index (current_head);
    return true;
}
```

tail/
write



head/
read