STACKS

RECALL:

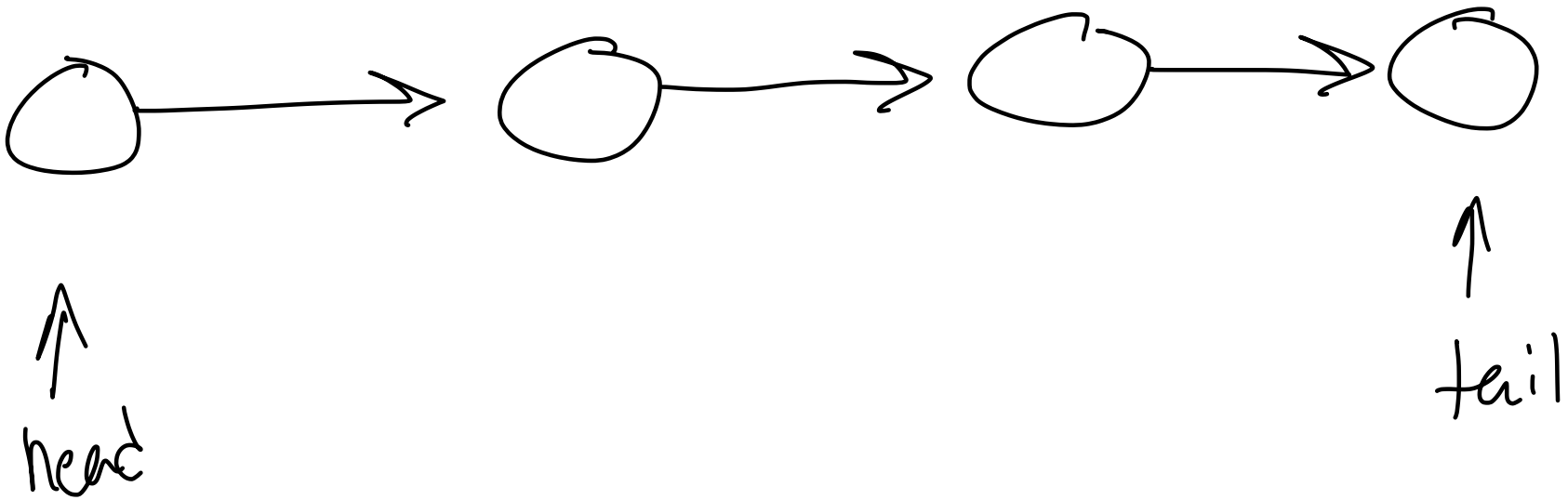
- Collection of objects
- Last in - first out (LIFO)
- Primary operations:
  - push (add to top)
  - pop (remove from top)

IMPELEMNTATIONS:

- Separate from the ADT
- The details of how we create the stack data structure
- Options:
  - Linked list based
  - Array based

LINKED LIST BASED

- Store stack as linked list
- How could we best represent stack using linked list for operations to be as efficient as possible?
  - Can `push` be O(1)?
  - Can `pop` be O(1)?
  - Possible to implement so they are both O(1)?



Stack:

need head

don't need the tail

LINKED LIST BASED

- Think about what operations with linked list were O(1):
  - adding to start
  - adding to end (if there's a `tail`)
  - computing size (if `size` is stored as instance variable)
  - removing from start
- LIFO: want to remove the last one we added

ARRAY BASED

- Store stack using an array
- How can we represent to be as efficient as possible?
  - Can `push` be O(1)?
  - Can `pop` be O(1)?
  - Possible to implement so they are both O(1)?

ARRAY BASED

- Just keep filling elements
  - Keep track of index representing `top`
  - Setting value of element is O(1)
- Problem: everytime we resize need to create new and copy over
  - Don't resize everytime we add
  - Grab a chunk more (typically 2x) when we need to resize
  - O(n), but happens rarely

```
for (int i=0; i<4; i++) {
    int j=0;
    while (j < N) {
        j+=2;
        int k=0 {
        while (k < N/2 * N) {
            k++;
            //some statement   O(1)
        }
    }
}
```

$\frac{n}{2} * n = \frac{n^2}{2}$

$\left(\frac{n^2}{2} + 1\right)\frac{n}{2}\,4$

$O(n)$ $O(n^2)$ $O(n^3)$ $O(n^4)$

$4n^2 - 8n + 6$

$\frac{1}{2}n^2 + 8n + 3$