Super quick review of how they work / everything!

The important concepts for all of them:

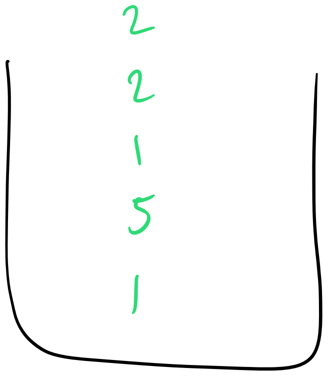
- Insertion / Deletion
- Lookup
- Complexity
- Respect the Interface! !!!!!!!!!!!!!!

STACKS

LIFO: Last In, First Out

↳ The last element pushed onto a stack is the first element popped off.

1, 5, 1, 2, 2

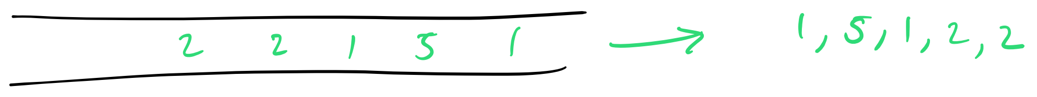


2, 2, 1, 5, 1

Queues

FIFO: First In, First Out

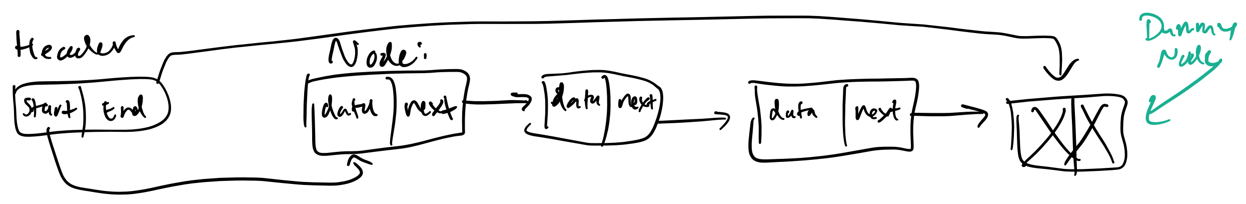
↳ First element enqueued onto the queue is the first element dequeued.



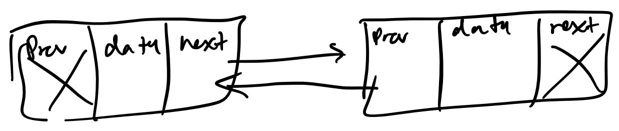
1,5,1,2,2

LINKED LISTS

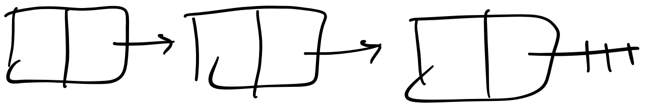
Super Flexible! Used to implement others!



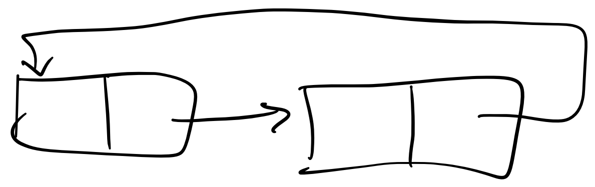
We can also have doubly linked:



OR
null terminated!



OR
circular...

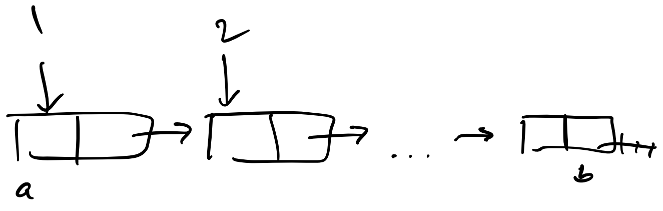


O(1) to insert/delete at the ends

O(n) to find a given node

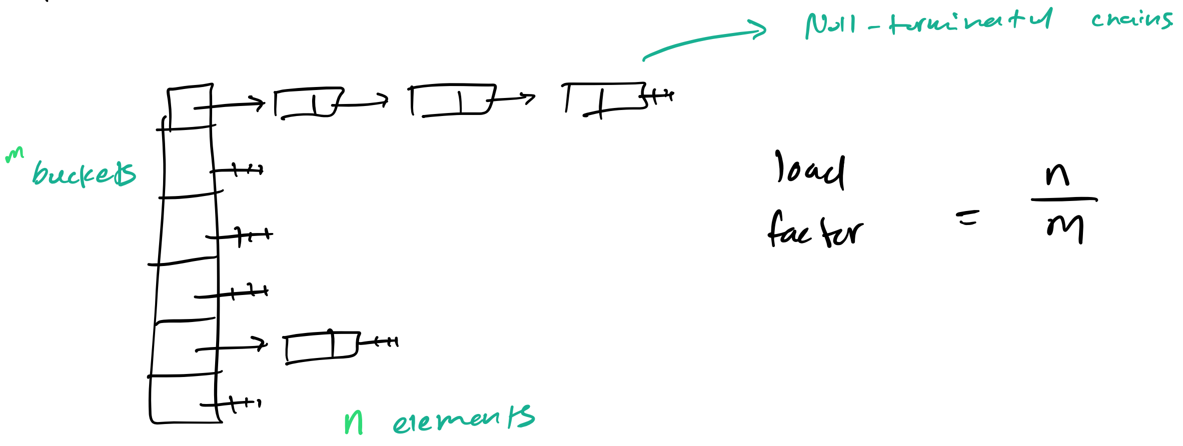
unbounded size :)

is_segment (a, b) :
↳ checks if you can get from a to b



is_acyclic (a) :
↳ checks if there are no cycles
↳ tortoise and hare algo.

HASH TABLES / DICTIONARIES



load factor $= \frac{n}{m}$

Insertion / Lookup:

hash (K) $O(1)$

Find bucket $O(1)$

Search chain for duplicate $O(1)$ amortized, $O(n)$ worst case

Insert $O(1)$

With a good hash function, we can spread the data across the buckets $\Rightarrow$ avg. of $\sim n/m$ nodes/bucket

If we are resizing (ubas),

load factor $(n/m) \approx$ constant $\Longrightarrow O(n/m) \Rightarrow O(1)$ Amortized

Else:

load factor (n/m) not constant $\Longrightarrow O(n/m) \Rightarrow O(n)$