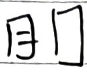
Wk 2 Lecture Note

Abstract Data Types

- defined in terms of its data items & operations
- x on implementation (more like computational φ)
- supported by python (+ many other langs)

Q Abstract?

ex) Driving a car

Interface	implementation
- wheel 	- engine
- pedals 	- electronic chips

Q Why approach with ADT?

↳ easier code to understand

↳ When making choices of implementation, (or experimenting performance trade-offs) the cost is much reduced.

↳ code is reused (from standard library) → faster building

ex) movie theatre reservation system

Q operations needed?

- capacity_available() (all seats) capacity_sold() (seats sold) get_available capaci (seats left)
- customer(x) : check who reserved seat x
- ~~add(x)~~ ~~(reserve(x))~~
- reserve(x, customer) - release(x)

Q Think about edge cases!

- reserve(x, customer) : if x is reserved already?
- release(x) : x isn't reserved?

ADT = specification of the desired behavior (from user perspective)
input → output

Data structure = concrete representation of data (perspective of implementor)

Comp Problem	vs	Algorithm
ADT	vs	Data Structure
implementation	↔	implementation

Q Index-based ADT

- size()
- isEmpty()
- get(i) ← O(1)
- add(i, e) : existing elements with idx ≥ i are shifted up. ← O(n)
- set(i, e) ← O(1)
- remove(i) : remove & return the element at i ← O(n)

  existing elements with idx > i are shifted down.

Q how the operations would work?

if ls = [A, B]

get(1) = B

get(2) = "error"

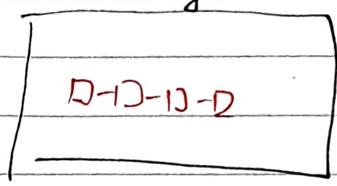
add(2, C) = — ls = [A, B, C]

add(5, D) = "error"

set(3, D) = "error"

add(3, D) = — ls = [A, B, C, D]

memory



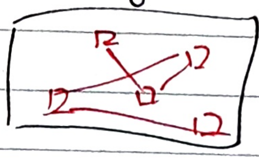
Q Positional Lists

↳ store elements at "positions"

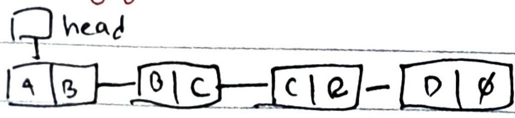
↳ positions not affected even after addition / deletion

- size()
- isEmpty()
- first() : pos of 1st elem (null if empty)
- ~~after~~ last() : pos of last elem (null if empty)
- before(p) : pos of elem before p
- after(p) : pos of elem after p
- insertBefore(p, e) : insert e in front of p
- insertAfter(p, e) : insert e following the elem at p
- remove(p) : remove and return the removed elem

memory

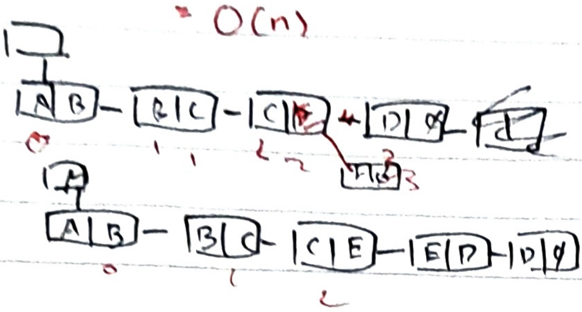


ex) Singly Linked List



insertBefore()   [A, B, C, D] → [A, B, C, E, D]

= O(n)



(3, E)

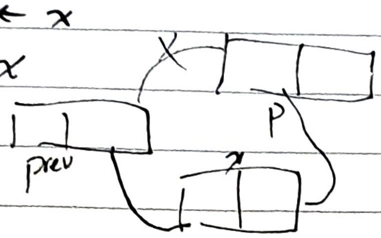
- Start at head and go to pos 3.
- Create a new Node with e: E ~~and linking to p:~~.
- Change prev node to link to the created Node and link the new nod to where prev Node originally pointed to.

ex) Doubly Linked List

insertBefore(p, e)   O(1)

p. prev. next ← x

~~p~~ x . ~~next~~ prev ← x



Q. Iterators

in python: __iter__(self)

" . next() method

for x in ls:
    // process x

=

it = x. __iter() __:

try:
    while True:
        x = it. next()
        // process
except:
    pass

Stacks & Queues

: Restricted forms of List

Stack : last - in - first out (LIFO)

queue : first in - first out (FIFO)

Stack ADT

↳ push(e): inserts an elem (puts plate on top of stack)

↳ pop(): remove + returns the top elem.

push(5) = ___ [5]

push(3) = ___ [5, 3]

pop() = 3 [5]

push(7) = ___ [5, 7]

top() = 7

ex) browsing history (only backwards)

editing doc undo (no redos)

Queue ADT

↳ enqueue(e): inserts e at the end

↳ dequeue: removes the first elem

enqueue(5) = ___ [5]

enqueue(3) = ___ [5, 3]

dequeue: 5 [3]

ex) waiting in a line

buffering pockets in stream (video, audio)