DAT 600

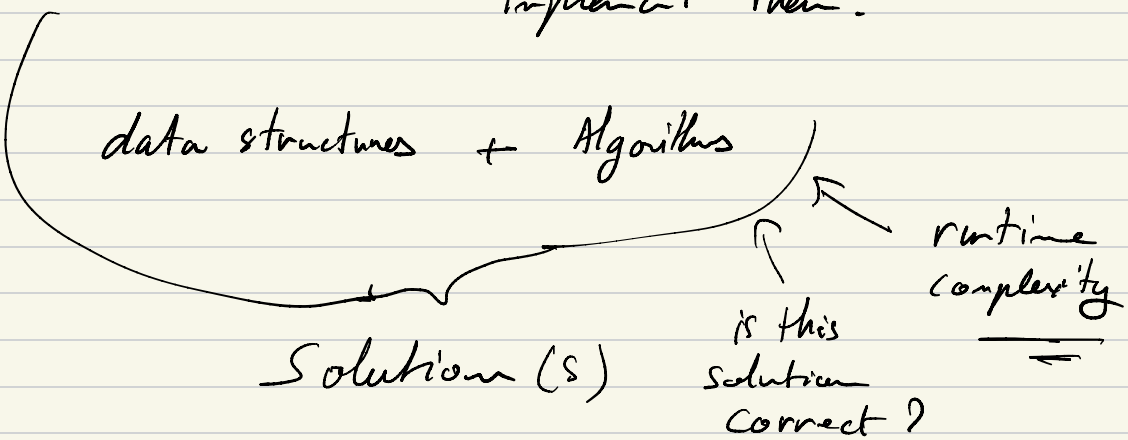
Summary of week 2

Interface or ADT → what are our specifications?

Sequence
Set
← two abstractions

define a set of operations

but they don't say how implement them.



Sequence interface

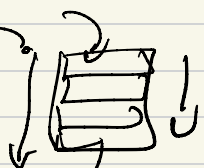
Data Structure	Operation, Worst Case $O(\cdot)$				
	Container	Static	Dynamic		
	build(X)	get_at(i) set_at(i,x)	insert_first(x) delete_first()	insert_last(x) delete_last()	insert_at(i, x) delete_at(i)
Array	n	1	n	n	n
Linked List	n	n	1	n	n
Dynamic Array	n	1	n	$1_{(a)}$	n

$X = (5, 1, 6, 7, 8, 3)$ ← deque

earliest ↑ — FIFO

oldest ↑

LIFO



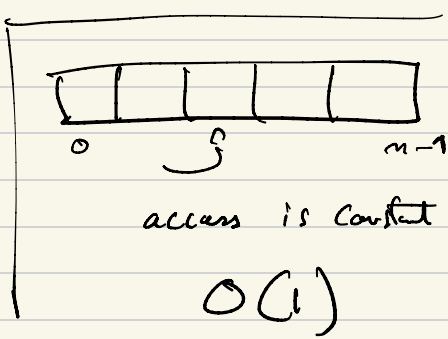
FIFO

e.g.

get_at(i)

T(n) is get_at Time using an array $O(1)$

$T(n) = O(1)$



access is constant

$O(1)$

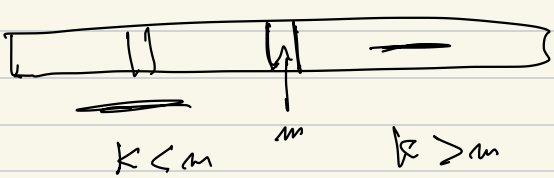
Set interface

Set Data Structure	Operations $O(\cdot)$				
	Container	Static	Dynamic	Order	
	`build(X)`	`find(k)`	`insert(x)` `delete(k)`	`find_min()` `find_max()`	`find_prev(k)` `find_next(k)`
Array	n	n	n	n	n
Sorted Array	$n \log n$	$\log n$	n	1	$\log n$

- But how to construct a sorted array efficiently?

$O(n)$ for all operations with Array data structures

Not sorted array.



a Sorted Set is good for runtime Time Complexity.

Asymptotic effiency:

- O → upper bound
- Ω → lower bound
- Θ → tight bound
- o → strict upper bound
- ω → strict lower bound.

e.g

Find max in a not sorted array

```
Find_max(A)          A has size n
  max = -∞ ------     1   c1
  ∀e ∈ A   ------     n   c2
    if e > max -----  n   c3
      max = e  ----   n   c4
```

$$c_1 \cdot 1 + n c_2 + n c_3 + n c_4$$

$$\overbrace{(c_2 + c_3 + c_4)}^{C} n + c_1 = Cn + c_1$$

$T(n) = c\,n + c_1$

$T(n) \overset{?}{=} O(n)$

$T(n) \overset{?}{=} \Omega(n)$

$T(n) \overset{?}{=} \Theta(n)$

—

we defined O Ω Θ o w