Lec 2

Interface (API / ADT) vs. Data Structure

- specification	- representation
- what data can store	- how to store data
- what operations are supported & what they mean	- how to support operations
- problem	- solution

2 main interface	2 main DS approaches
- set	- array
- sequence	- pointer based

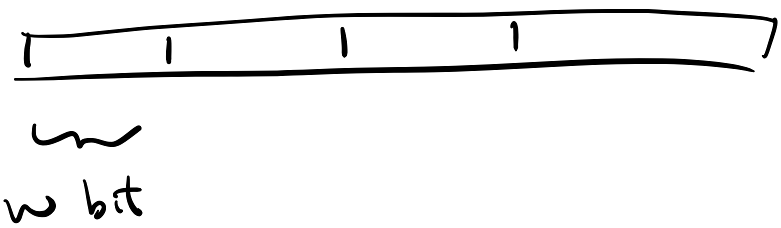
Static sequence interface: maintain a sequence of items $x_0, x_1, \ldots x_{n-1}$ subject to these operations:

- build(x): make new DS for items in x
- len(x): return n
- iter_seq(): output $x_0, x_1, \ldots, x_{n-1}$ in sequence order
- get_at(i): return x_i (index i)
- set_at(i, x): set x_i to x

Solution (natural): Static array

<u>Key</u>: Word RAM model of Computation

- memory: array of w-bit words



- "array" = Consecutive chunck of memory

$\Rightarrow$ array[i] $\equiv$ memory[address(array) + i]

array access is $O(1)$ time

- $O(1)$ per get_at / set_at / len
- $O(n)$ per build / iter_seq

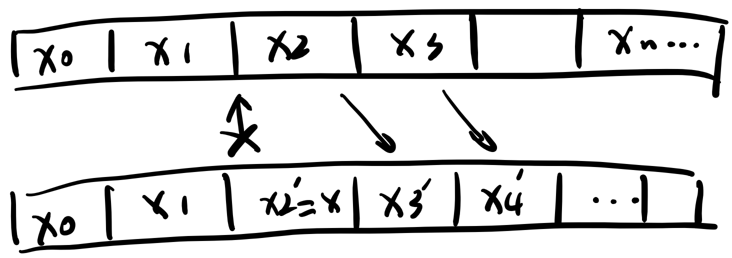
Memory allocation model: allocate array of size n in $\Theta(n)$ time

$\Rightarrow$ space $= O(\text{time})$

Dynamic sequence interface

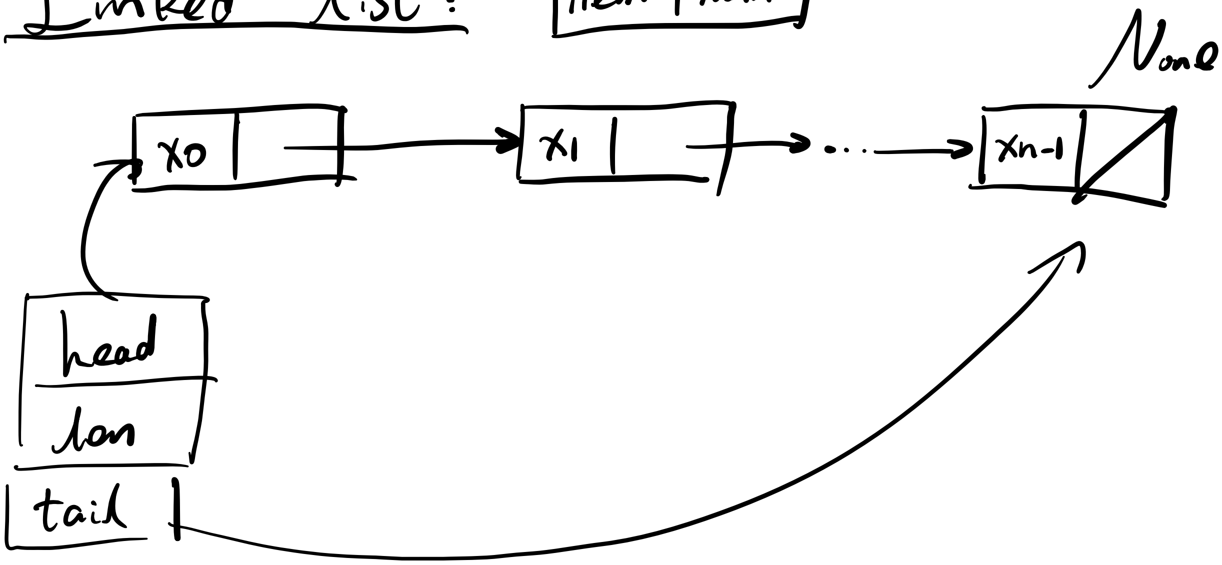
static sequence plus:

- insert_at(i, x): make x the new x_i, shifting $x_i \to x_{i+1} \to x_{i+2} \to \cdots \to x_{n-1} \to x_{\underbrace{n'}_{=n+1}-1}$
- delete_at(i, x): shift $x_i \leftarrow x_{i+1} \leftarrow \cdots \leftarrow x_{\underbrace{n'}_{n-1}-1} \leftarrow x_{n-1}$



- insert / delete_ first / last (x) / ( )

Linked list:



Dynamic seq. ops.

static array

- insert / delete - at( ) cost $\Theta(n)$

① shifting

② allocation / copying

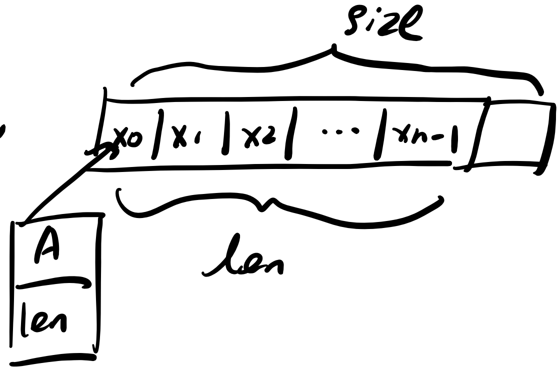
linked list

- insert / delete_first( ) $\Theta(1)$ time
- get / set_at need $\Theta(i)$ time
  ($\Theta(n)$ worst case)

Dynamic arrays (Python lists)

- relax constraint
- enforce size $= \Theta(n)$ & $\geq n$
- maintain $A[i] = x_i$
- insert_last(x)

$$\begin{cases} A[len] = x \\ len += 1 \end{cases}$$



- if $n = size$:

  allocate new array of $2 \cdot size$

- n insert_last from empty array

  resize at $n = 1, 2, 4, 8, \ldots$

$\Rightarrow$ resize cost $= \Theta(1+2+4+8+16+\cdots+n)$

$$= \Theta\left(\sum_{i=1}^{\lg n} 2^i\right)$$

$$= \Theta\left(2^{\lg n}\right) = \Theta(n)$$

Amortization

operation takes $T(n)$ amortized time if

any k operations take $\leq k \cdot T(n)$ time

(averaging over operation sequence)