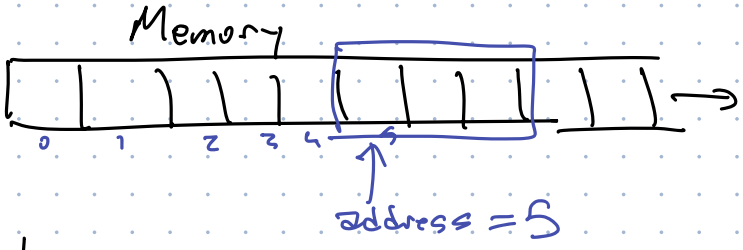
Please Fill out the background survey!
HW0 due Tues (completion only)

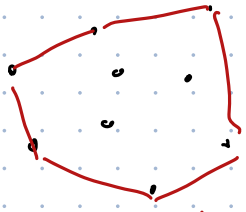
Fundamental Data Types + Data Structures
problem — what
algorithm — how

Store a set of integers
Support INSERT(x)
NewSET()

~~nil~~
NULL ptr!

Primitive data type
Something fits in O(1) space
Booleans T/F
Characters ASCII/Unicode
Theoretical abstraction: integer, real #
exact arithmetic, <> = ≤ ≥ in O(1) time
dangerous outside this class




Convex hull

pointer/reference

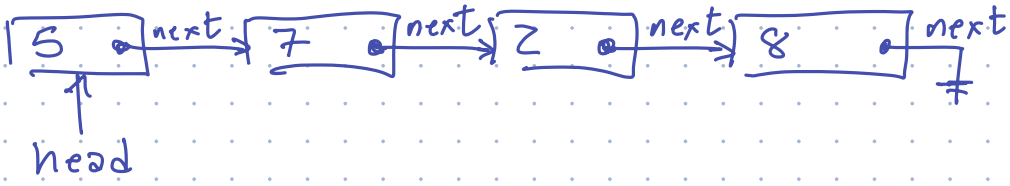
Aggregates – contiguous blocks of memory

Records
fields ref by name
maybe diff types

Arrays
indexed sequence A[l..r]
size r-l+1
each element A[i] has same type
$(a_1, a_2, a_3, \ldots a_n)$

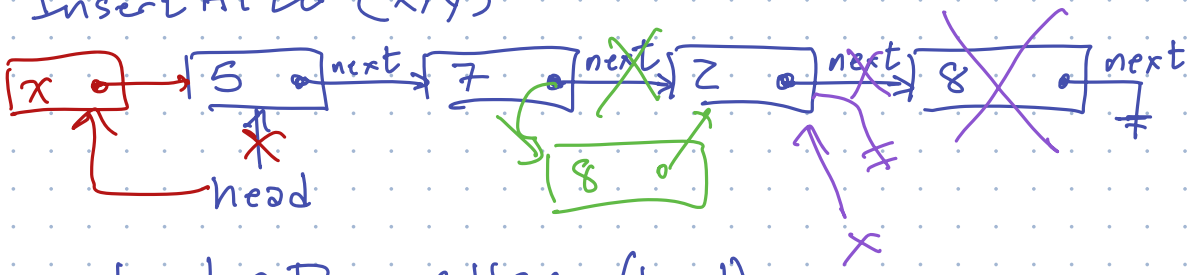
read + write components in O(1) time
size + types of comps are fixed

Linked List



sequence of records with next ptrs

- new head = INSERT BEFORE (head, x)
- Insert After (x, y)

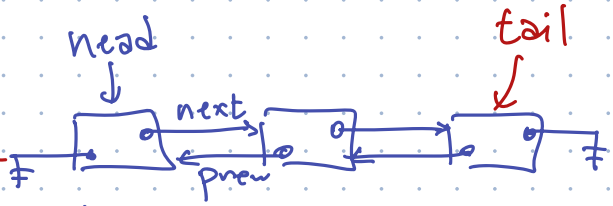


O(1) time

- new head ← DELETE HEAD (head)
- DELETE AFTER (x)

VARIANTS

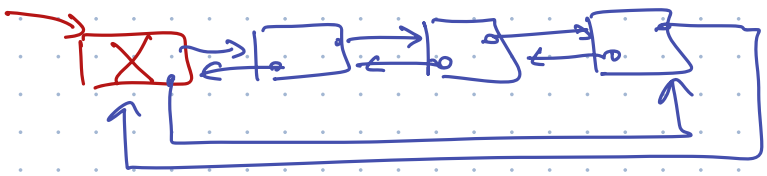
Doubly-linked



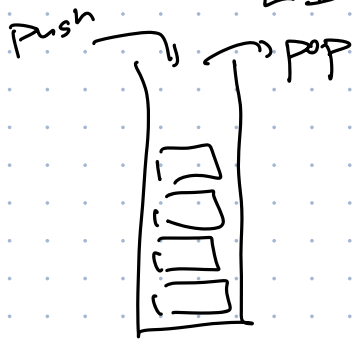
v.prev = w ⇓ w.next = v

circular:

sentinel:

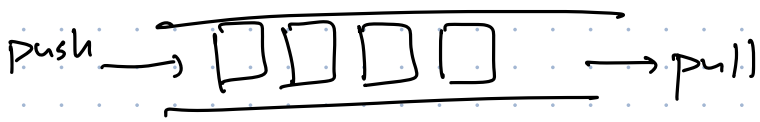


Stack (FILO) LIFO



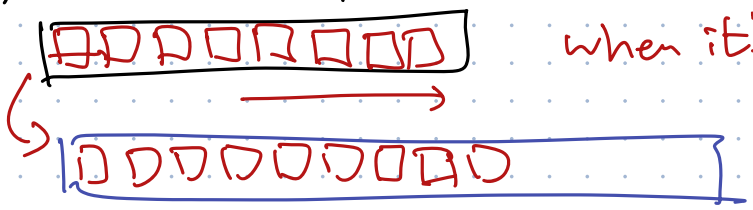
singly LL

Queue (FIFO)



doubly LL

Dynamic array



when it's full, double it

push takes $O(n)$ time in worst case BUT

Total time for N pushes (incl. resizing) is $O(N)$.

$\Rightarrow O(1)$ amortized time per push.

Similarly when array is ¼ full, halve it.

Dyn arrays → deque double-ended queue

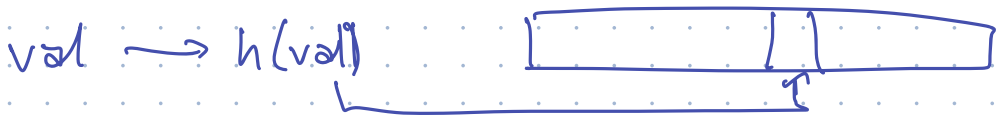
qnack / steque
no poke no pull

Dictionary Key-value store Associative array

Maintain records

Insert (key, val)
val ← Find (key)

→ Hash table "O(1) time" — fiction —



Ordered Dictionary

val ← Pred (key)
val ← Succ (key)

Sorted array
+binary search
$O(\log)$ time

Dynamic: Binary Search Tree

$O(n)$ worst case $O(\log n)$ balanced