ADT Dictionaries

- This abstract data type (ADT) operates on sets.
- The operations to be performed are: Insert(x), Delete (x), and Membership(x).
- Remember that for sets we do not allow repeated elements. Therefore Insert(x) when x is in the set will do nothing.

Representation: Unsorted Array

- Represent the set of integers in an unsorted sequential array of size N.
- n tells us how many elements we have currently in the list.
- Note that at all time $0 \leq n \leq N$.
- Example:

0	1	2	3	4				N-1	
3	19	15	12	2		...			
					n				N

- The operations are performed as follows:

Membership(x)

do a sequential search (program discussed before) and return true or false depending whether or not x is in the array.

Time Complexity

- Membership takes $\Omega(1)$ and $O(n)$ time.

Insert(x)

If membership(x) returns false

then { if $n >= N$ then /* No space left */ exit(1)

else add x at position n in the array

and increase n by one.

}

Time Complexity

- Insert takes $\Omega(1)$ and $O(n)$ time.

Delete(x)

If membership(x) returns false then return

do a sequential search (time&space.complexity.2) till you find x, then move all the elements after x one position to the left and decrease the value of n by one.

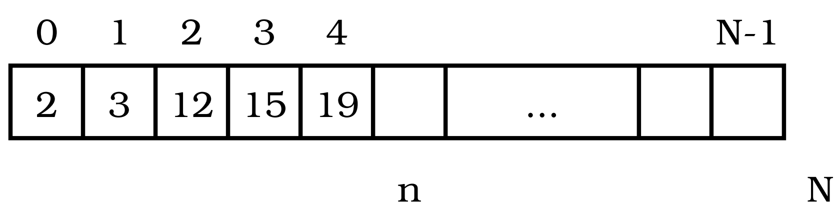
Time Complexity

- Delete takes $\Omega(n)$ and $O(n)$ time.

Actually a “faster” procedure is possible (TC is $\Omega(1)$ and $O(n)$).

Representation: Sorted Array

- Represent the set of integers in a sorted sequential array of size N.
- n tells us how many elements we have currently in the list.
- Note that at all time $0 \leq n \leq N$.
- Example:



- The operations are performed as follows:

Membership(x) is just a binary search (Sec. 3.4 [Sa]).

Time Complexity

- Membership takes $\Omega(1)$ and $O(log\ n)$.

Insert(x)

If membership(x) returns true then return

if $n >= N$ then /* no space left */ exit(1)

do a binary search (Sec. 3.4 [Sa]) and find the first element with value greater than x or the element after the last one if all the element in the list are less than x. Then move all the elements from this position to the end of the list one unit and insert x in the empty position. Increase n by 1.

Time Complexity

- Insert takes $\Omega(1)$ and $O(n)$.

Delete(x)

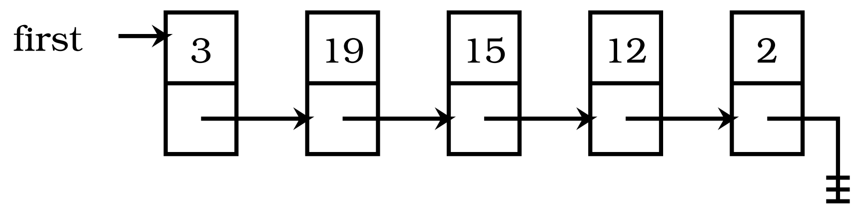
You may use the previous Delete(x), but using binary search instead of sequential search.

Time Complexity

- Delete takes $\Omega(\log n)$ and $O(n)$ time.

Representation: Unsorted Linked

- Represent the set of integers in an unsorted linked list.
- $first$ is either null (list is empty) or points to the first object in the list
- Example:

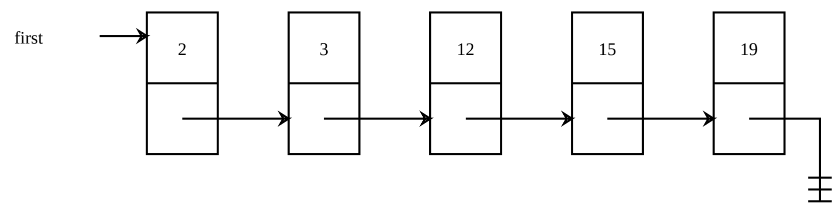


Time Complexity

- Membership takes $\Omega(1)$ and $O(n)$.
- Insert takes $\Omega(1)$ and $O(n)$.
- Delete takes $\Omega(1)$ and $O(n)$ time.

Representation: Sorted Linked

- Represent the set of integers in a sorted linked list.
- $first$ is either null (list is empty) or points to the first object in the list
- Example:



Time Complexity

- Membership takes $\Omega(1)$ and $O(n)$.
- Insert takes $\Omega(1)$ and $O(n)$.
- Delete takes $\Omega(1)$ and $O(n)$ time.

Table 1: Time Comlexity (Representation/Operations)

	Membership	Insert	Delete
Unsorted Array	$\Omega(1), O(n)$	$\Omega(1), O(n)$	$\Omega(n), O(n)$
Sorted Array	$\Omega(1), O(\log n)$	$\Omega(1), O(n)$	$\Omega(\log n), O(n)$
Unsorted Linked List	$\Omega(1), O(n)$	$\Omega(1), O(n)$	$\Omega(1), O(n)$
Sorted Linked List	$\Omega(1), O(n)$	$\Omega(1), O(n)$	$\Omega(1), O(n)$