

## Dynamic Sets

Dynamic sets are data structures that support some or all of the following operations.

SEARCH( $S, k$ )	Return $x$ with $key[x] = k$ .
INSERT( $S, x$ )	Insert $x$ into $S$ .
DELETE( $S, x$ )	Delete $x$ from $S$ .
MINIMUM( $S$ )	Return $x$ with minimum $key[x]$ .
MAXIMUM( $S$ )	Return $x$ with maximum $key[x]$ .
SUCCESSOR( $S, x$ )	Return $y$ with next highest key.
PREDECESSOR( $S, x$ )	Return $y$ with next lowest key.

$S$  is a set.  $x$  and  $y$  are elements.  $k$  is a key.  
NIL is returned if operation can't be performed.

## Simple Dynamic Set Data Structures

Dynamic Set	Stack	Queue	Linked List
SEARCH			LIST-SEARCH
INSERT	PUSH	ENQUEUE	LIST-INSERT
DELETE	POP	DEQUEUE	LIST-DELETE

All stack and queue operations are  $\Theta(1)$ .

LIST-INSERT is  $\Theta(1)$  for unsorted lists,

$O(n)$  for sorted lists.

LIST-DELETE is  $\Theta(1)$  for doubly-linked lists,

$O(n)$  for singly-linked lists.

## Average Case Analysis of LIST-SEARCH

LIST-SEARCH( $L, k$ )

$x \leftarrow head[L]$

**while**  $x \neq \text{NIL}$  and  $key[x] \neq k$

**do**  $x \leftarrow next[x]$

**return**  $x$

LIST SEARCH is  $\Theta(n)$  on average.

Suppose there are  $n$  elements.

Each element is equally likely to be searched.

Search for the  $i$ th element uses  $i$  comparisons.

The expected value is the following summation:

$\sum_{i=1}^n$  There are  $n$  elements to search for.

$(1/n)$  Prob. for  $i$ th element is  $1/n$ .

$i$   $i$  comparisons needed to find  $i$ th element.

$$\sum_{i=1}^n \frac{i}{n} = \frac{1}{n} \sum_{i=1}^n i = \frac{n+1}{2}$$

The result is  $\Theta(n)$ .

## Allocating and Freeing Objects

```

ALLOCATE-OBJECT()
  if  $free = \text{NIL}$  then return  $\text{NIL}$ 
   $x \leftarrow free$ 
   $free \leftarrow next[free]$ 
  return  $x$ 

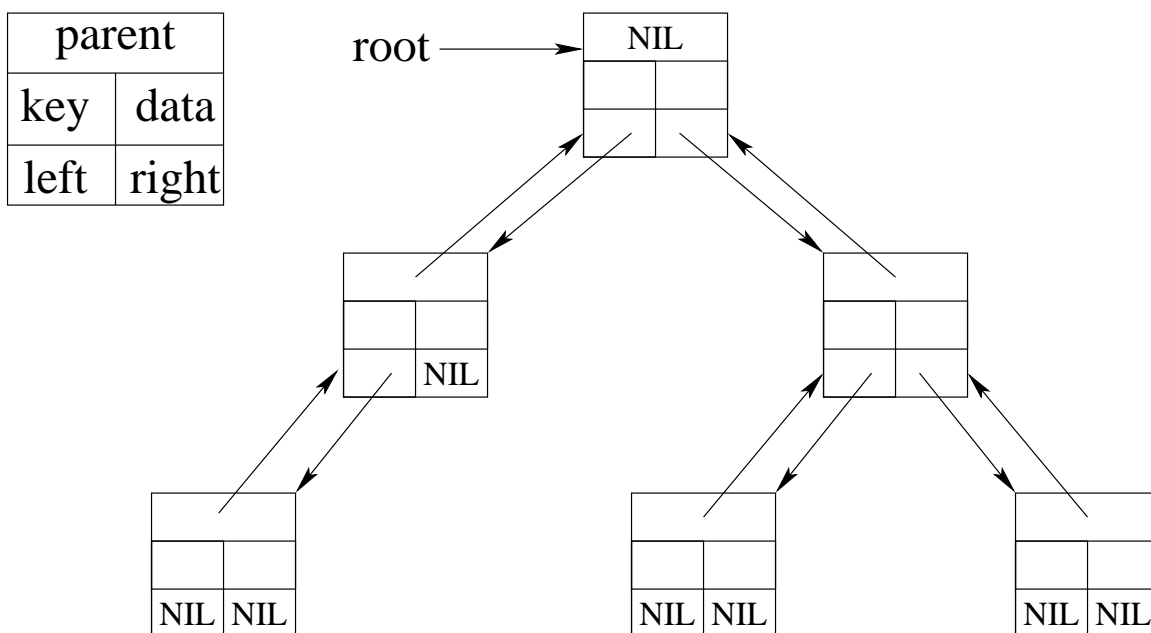
```

```

FREE-OBJECT( $x$ )
   $next[x] \leftarrow free$ 
   $free \leftarrow x$ 

```

## Binary Tree Representation



# Unbounded Tree Representation

