Ordered Array

Odd methods

- insertAtFirst()
- insertAtLast()

New methods

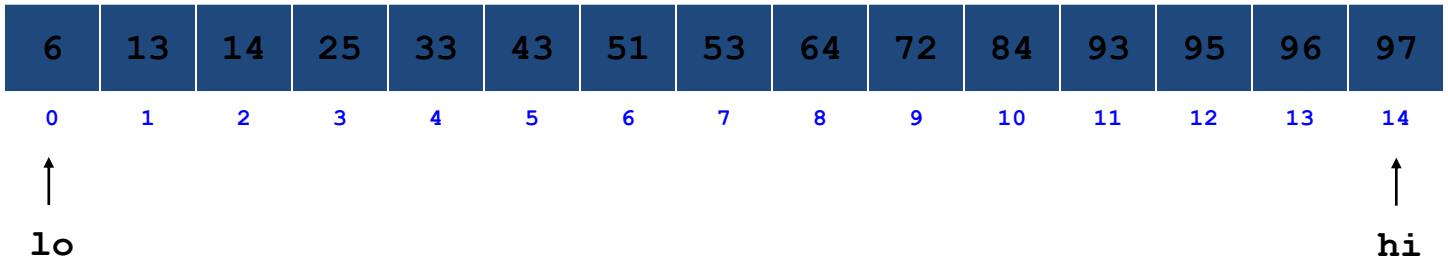
- insertData(Object data)
- deleteData(Object data)
- binarySearch(Object[] a, Object target)

Binary Search

Binary search. Given `target` and sorted array `a[]`, find index `i` such that `a[i] = target`, or report that no such index exists.

Invariant. Algorithm maintains `a[lo]` ≤ `target` ≤ `a[hi]`.

Ex. Binary search for 33.

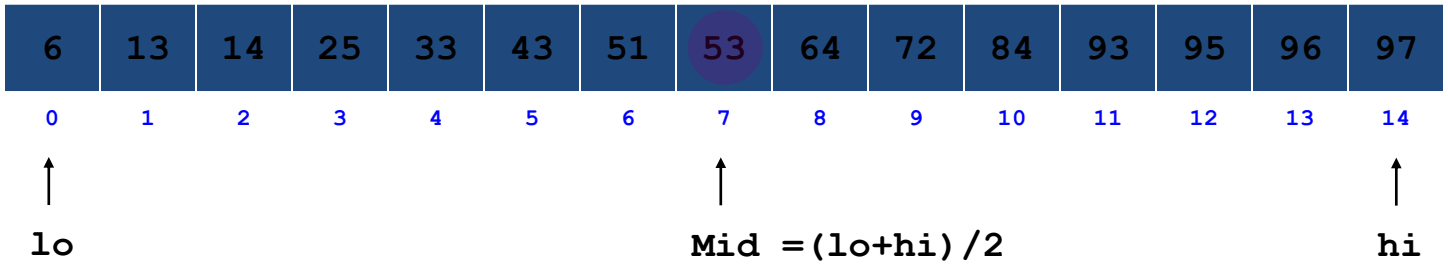


Binary Search

Binary search. Given `target` and sorted array `a[]`, find index `i` such that `a[i]` = `target`, or report that no such index exists.

Invariant. Algorithm maintains `a[lo]` ≤ `target` ≤ `a[hi]`.

Ex. Binary search for 33.

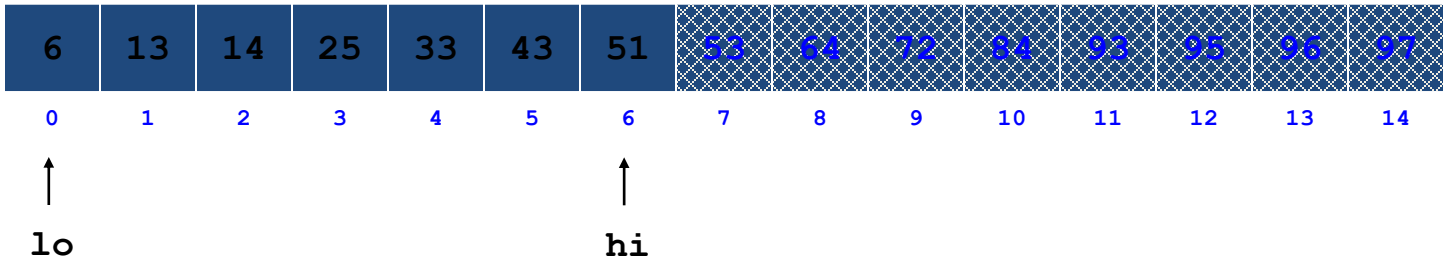


Binary Search

Binary search. Given `target` and sorted array `a[]`, find index `i` such that `a[i]` = `target`, or report that no such index exists.

Invariant. Algorithm maintains `a[lo]` ≤ `target` ≤ `a[hi]`.

Ex. Binary search for 33.

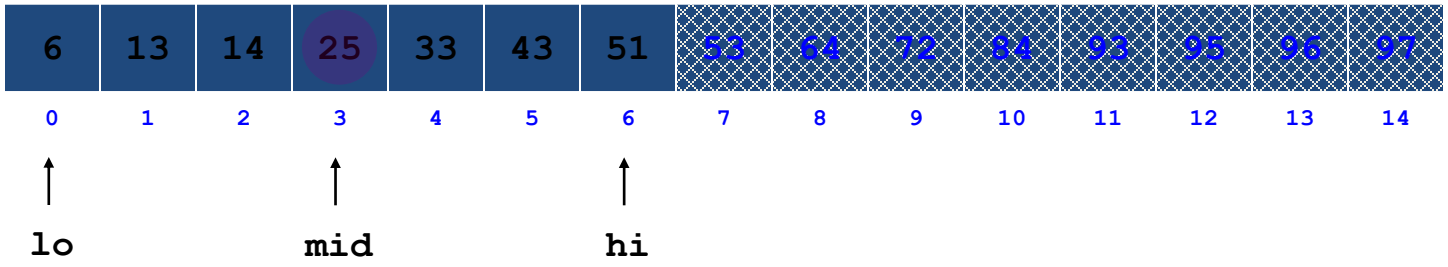


Binary Search

Binary search. Given `target` and sorted array `a[]`, find index `i` such that `a[i]` = `target`, or report that no such index exists.

Invariant. Algorithm maintains `a[lo]` ≤ `target` ≤ `a[hi]`.

Ex. Binary search for 33.

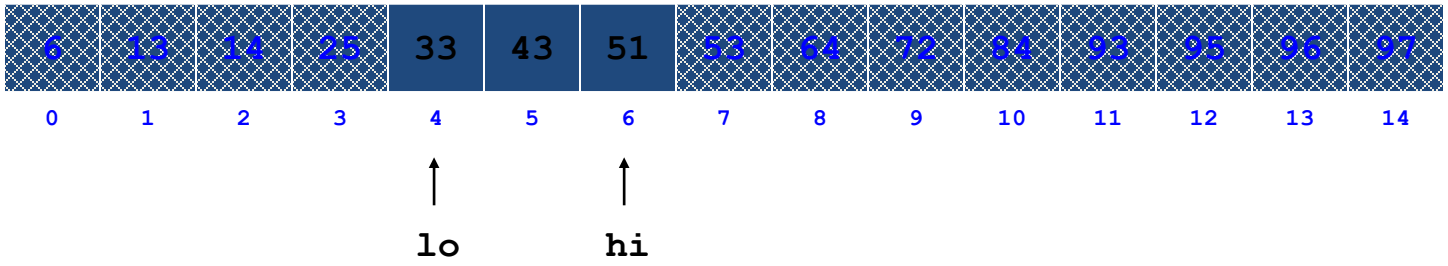


Binary Search

Binary search. Given `target` and sorted array `a[]`, find index `i` such that `a[i]` = `target`, or report that no such index exists.

Invariant. Algorithm maintains `a[lo]` ≤ `target` ≤ `a[hi]`.

Ex. Binary search for 33.

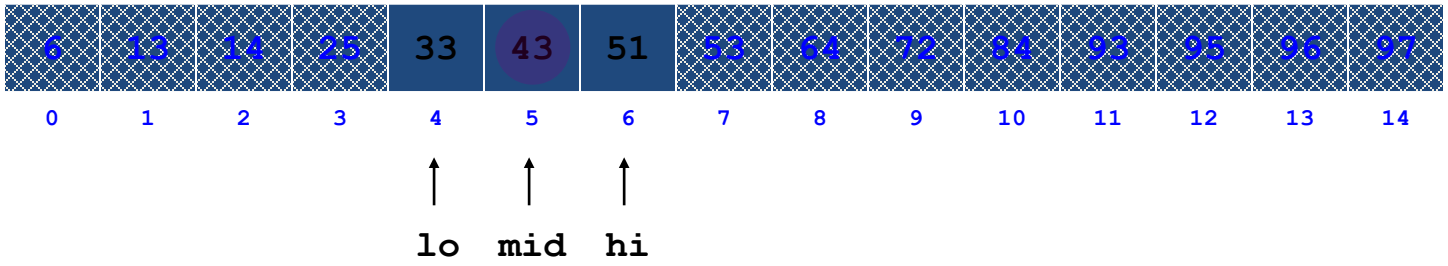


Binary Search

Binary search. Given `target` and sorted array `a[]`, find index `i` such that `a[i]` = `target`, or report that no such index exists.

Invariant. Algorithm maintains `a[lo]` ≤ `target` ≤ `a[hi]`.

Ex. Binary search for 33.

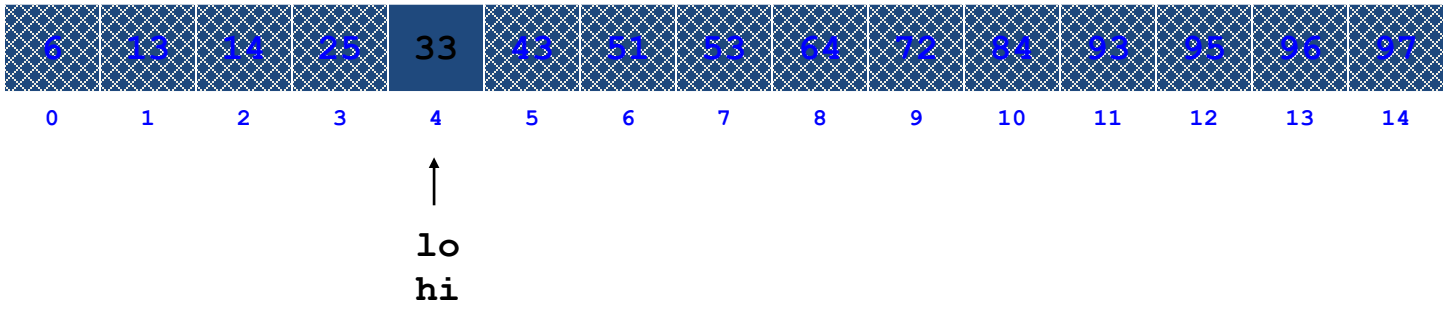


Binary Search

Binary search. Given `target` and sorted array `a[]`, find index `i` such that `a[i]` = `target`, or report that no such index exists.

Invariant. Algorithm maintains `a[lo]` ≤ `target` ≤ `a[hi]`.

Ex. Binary search for 33.

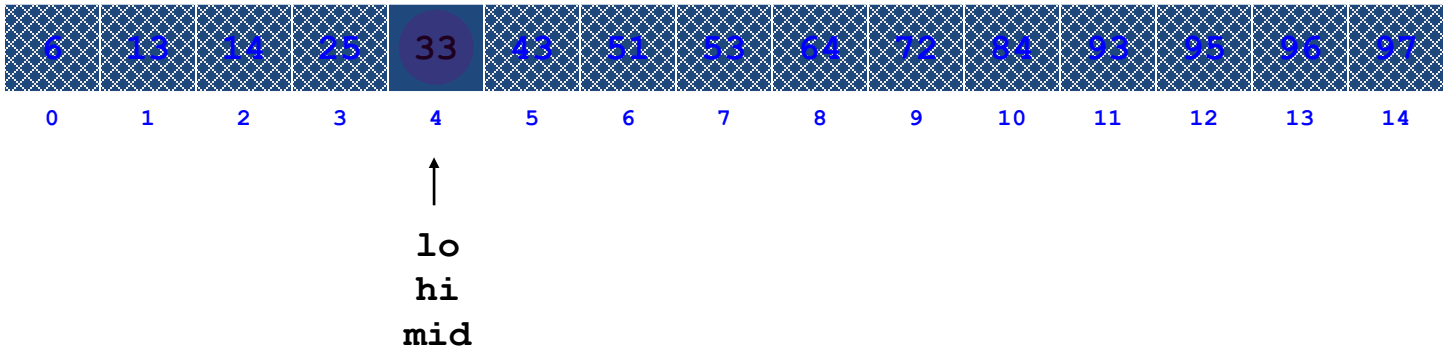


Binary Search

Binary search. Given `target` and sorted array `a[]`, find index `i` such that `a[i]` = `target`, or report that no such index exists.

Invariant. Algorithm maintains `a[lo]` ≤ `target` ≤ `a[hi]`.

Ex. Binary search for 33.

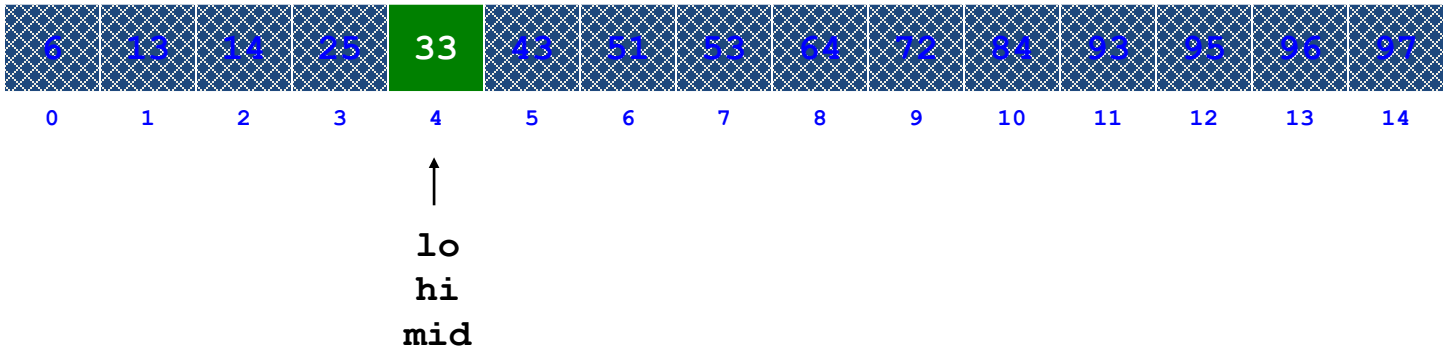


Binary Search

Binary search. Given `target` and sorted array `a[]`, find index `i` such that `a[i]` = `target`, or report that no such index exists.

Invariant. Algorithm maintains `a[lo]` ≤ `target` ≤ `a[hi]`.

Ex. Binary search for 33.



Binary search code

```
// Returns the index of target in array a,
// or -1 if the target is not found.
// Precondition: elements of a are in sorted order

public static int binarySearch(int[] a, int target)
{   int lo = 0;
    int hi = a.length - 1; // or size-1
    int mid;

    while (lo <= hi)
    {   mid = (lo + hi) / 2;
        if (a[mid] < target) {
            lo = mid + 1;
        } else if (a[mid] > target) {
            hi = mid - 1;
        } else {
            return mid;   // target found
        }
    }

    return -1;    // target not found
}
```

Ordered Array

Advantages

Quick search –$O(\log_2 n)$

Disadvantages

Slow insertion - O(n)

Slow deletion - O(n)

Fixed size- need dynamic array

Better performed when searches are more than insert

	Advantages	Disadvantages
Unordered array list	Insertion - O(1)	Search – O(n) Deletion – O(n) Fixed array size
Ordered array list	search –$O(\log_2 n)$	Insertion - O(n) Deletion - O(n) Fixed array size

Iterators

An iterator permits you to examine the elements of a data structure one at a time.

Iterator Methods

Iterator ix = x.iterator();

constructs and initializes an iterator
to examine the elements of x;
constructed iterator is assigned to ix

you must define the method iterator
in the class for x

Iterator Methods

ix.hasNext()

returns true iff x has a next element

ix.next()

throws NoSuchElementException if there is no next element

returns next element otherwise

Optional Iterator Method

ix.remove()

removes last element returned by ix.next()

throws UnsupportedMethodException if method not implemented

throws IllegalStateException if ix.next() not yet called or did not return an element

Using An Iterator

```
Iterator ix = x.iterator();
while (ix.hasNext())
    examine(ix.next());
```

vs

```
for (int i = 0; i < x.size(); i++)
    examine(x.get(i));
```

Java's Array Linear List Class

java.util.ArrayList

A type of our ArrayLinearList With Iterator