RECURSION

BINARY SEARCH, MIN-MAX, INSERTION SORT

Reference: Java Documentation http://docs.oracle.com/javase/tutorial/java/

FINDING THE MIN

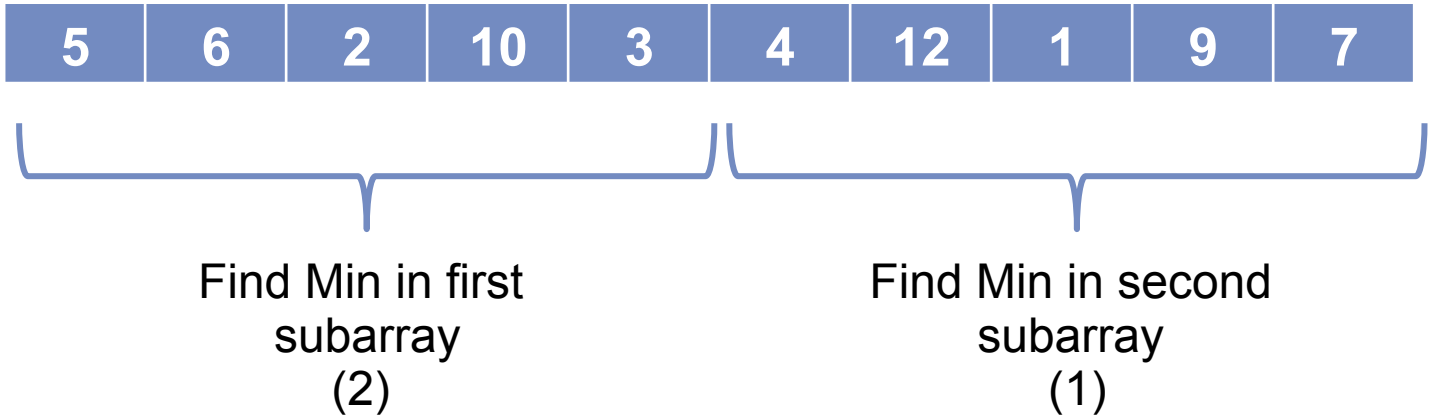
- **Objective: find and return the minimum value in a list**
- **Input: a list (e.g array, ArrayList)**
  - Unsorted
- **output:**
  - Value of the min

FINDING THE MIN - LOOP

```
minIndex = 0;
for (j = 1; j < a.length; j++) {
   if (a[j] < a[minIndex]) {
      minIndex = j;
   }
}
return a[minIndex];
```

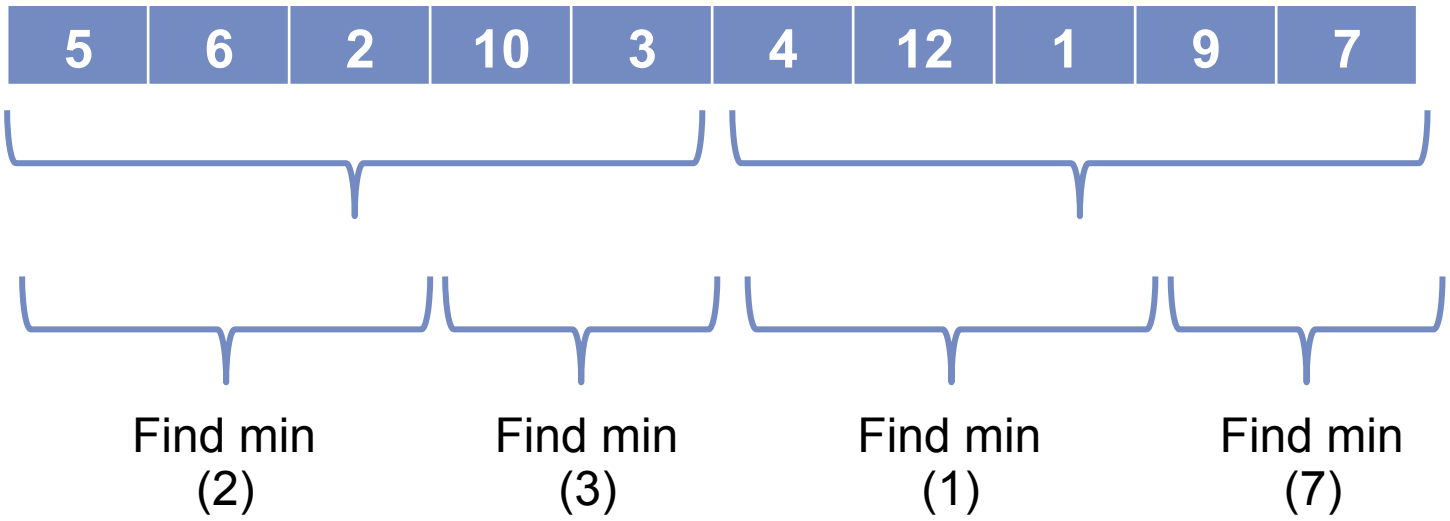
CONVERT TO RECURSION

- **Step 1: break the problem into smaller parts**
  - Find the min in a subarray?



CONVERT TO RECURSION

- **Step 1: break the problem into smaller parts**
  - Find the min in a subarray?



CONVERT TO RECURSION

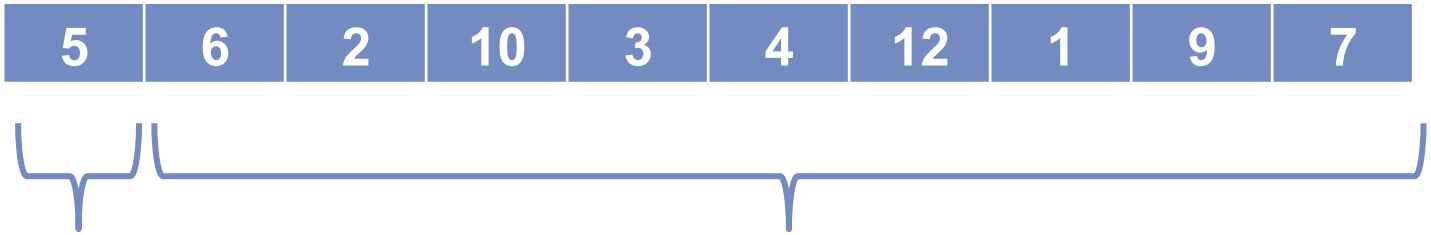
- **Step 1: break the problem into smaller parts**
  - Find the min in a subarray?

```
RecursiveMin (array, lo, hi)
      mid = (lo + hi)/2
      min1= RecursiveMin(array, lo, mid);
      min2= RecursiveMin(array, mid+1, hi);

      ….
```

CONVERT TO RECURSION

- **Step 1: break the problem into smaller parts**
  - Find the min in a subarray?

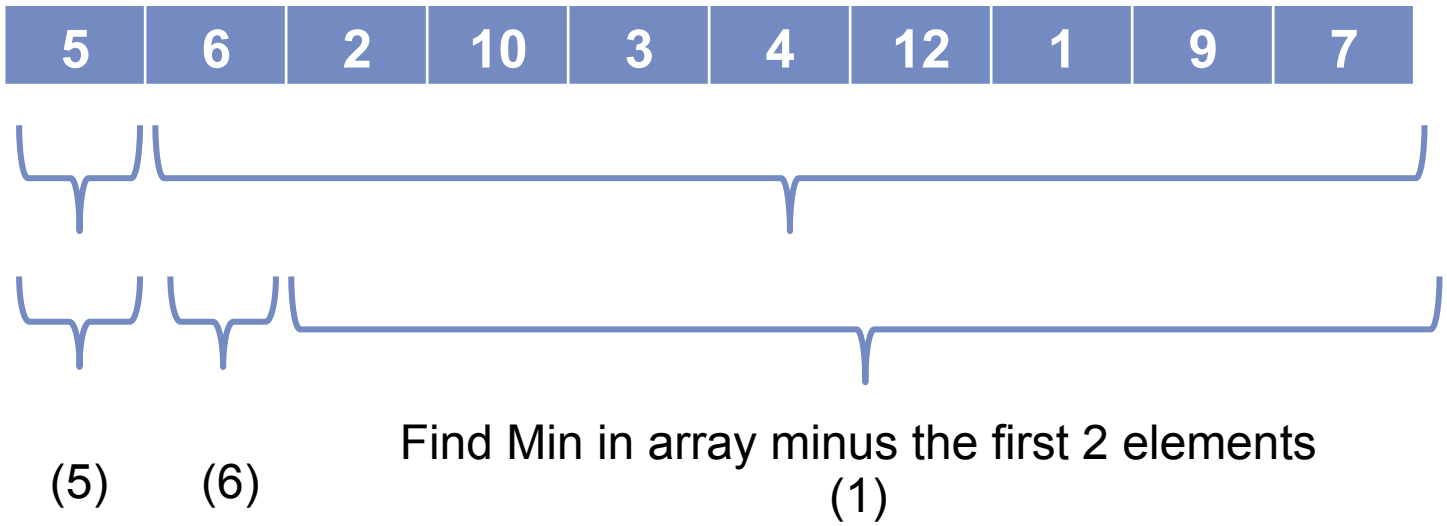


Find Min in array minus the first element

(5) (1)

CONVERT TO RECURSION

- **Step 1: break the problem into smaller parts**
  - Find the min in a subarray?



CONVERT TO RECURSION

- **Step 1: break the problem into smaller parts**
  - Find the min in a subarray?

```
RecursiveMin (array, index)
      min1 = array[index];
      min2 = RecursiveMin(array, index+1);
      ….
```

CONVERT TO RECURSION

- **Step 2: Find the base case**
  - When is it straightforward?
  - Example: When there is only 1 element, return it's value

```
RecursiveMin (array, lo, hi)
   if (lo == hi) return array[lo];
   else
      mid = (lo + hi)/2
      min1= RecursiveMin(array, lo, mid);
      min2= RecursiveMin(array, mid+1, hi);
      ….
```

```
RecursiveMin (array, index)
   if (index == array.length -1)
      return array[index];
   else
      min1= array[index];
      min2= RecursiveMin(array, index+1);
```

CONVERT TO RECURSION

- **Step 3: combine solutions**

  return the minimum value, compare min1 and min2

```
min1 = ..
min2= ..

if (min1<min2)
   return min1;
else
   return min2;
```

SEARCHING

- **Objective: Answer the question does a list contain a specific object?**
- **Input: a list (e.g array, ArrayList)**
  - Sorted
  - Unsorted
- **output:**
  - true (found) or false (not found)
  - Index of the object if found, -1 otherwise.

SEQUENTIAL SEARCH

- **Input (any list - sorted or unsorted, target value)**
- **method: exhaustively search from beginning to end**

Begin search

Loop (elements in the array from 0 to array size-1){

if (current element = target)

Return true (or the current index)

(else keep searching)

}

End of while loop

Target element not found, return false (or -1)

BINARY SEARCH

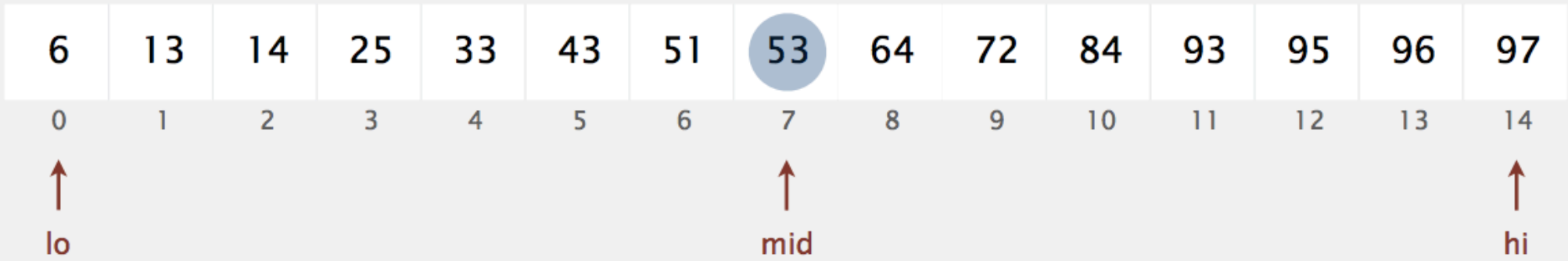
- **Input (sorted list, target value)**
- **method: repeatedly half the search space until target is found**

lowIndex=0, highIndex=arraySize-1

```
BinarySearch (Array a, target k, low, high)
        if (high < low ) return false; // base case
        mid= (low+ high)/2
        if (a[mid]==k) Return true
        if (k < a[mid])
                return BinarySearch (a, k, low, mid-1)
        else
                return BinarySearch (a, k, mid+1, high)
```

BINARY SEARCH - DEMO

- **Searching for 33**



Reference: http://algs4.cs.princeton.edu/home/

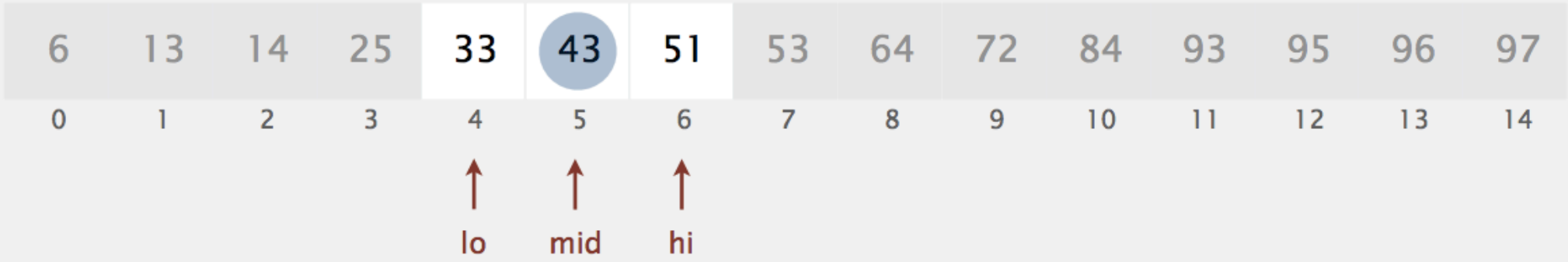
BINARY SEARCH - DEMO

- **Searching for 33**

Reference: http://algs4.cs.princeton.edu/home/

BINARY SEARCH - DEMO

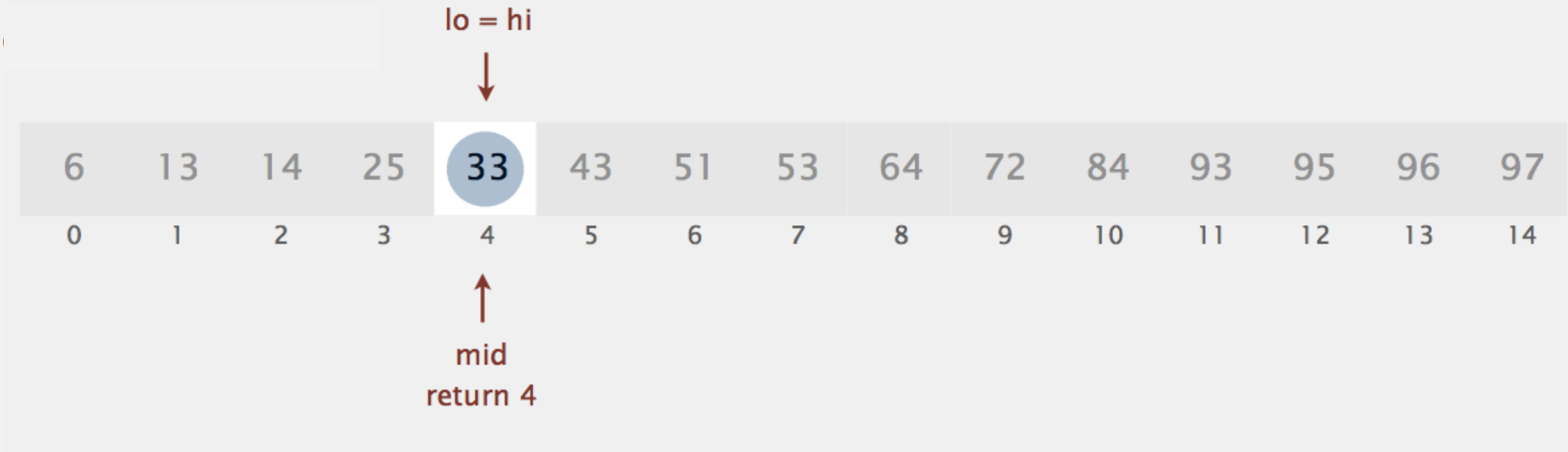
- **Searching for 33**



Reference: http://algs4.cs.princeton.edu/home/

BINARY SEARCH - DEMO

- **Searching for 33**



Reference: http://algs4.cs.princeton.edu/home/

INSERTION SORT

```
void insertionSort(int a[]){
   int temp;
   int size = a.length;
   for (int i = 1; i < size; i++){
     int j = i;
     temp = a[i];
     while (j > 0 && temp < a[j - 1]{
         a[j] = a[j - 1]; \\shift element
         j--;
     }
     a[j] = temp;
   }
}
```

INSERTION SORT

```
//start with index = array.length-1

recursiveInsertion(array, index)
   if (index == 0)
         return;
   else{
         recursiveInsertion(array, index-1);
         //insert element (see previous slide)

   }
```