Searching

Scala sequences contain a method `indexOf(el)`, which returns the index of the first element that is equal to `el`.

If the elements of the list are in no known order, we can only use sequential search (linear search):

```
def linear_search(a: Array[Int], x: Int): Int = {
  for (i <- 0 until a.length)
    if (a(i) == x)
      return i
  -1
}
```

What is the running time of linear search?
Best case? Worst case? Average case?

Searching a sorted sequence

If x is not in the list, we get more information: we actually know the index where x needs to be inserted.

Given an array a with a non-decreasing sequence of integers.

5	7	13	13	13	39	59	59	60	75	99	99	197

Given x, find the smallest index i such that $a(i) \geq x$.

If all elements of a are smaller than x, return `a.length`.

```
def sorted_linear_search(a: Array[Int], x: Int) = {
  for (i <- 0 until a.length)
    if (a(i) >= x)
      return i
  a.length
}
```


Searching a sorted sequence

Can we do better if the list is sorted?

Given an array a with a non-decreasing sequence of integers.

5	7	13	13	13	39	59	59	60	75	99	99	197

We can stop as soon as we find an element larger than x:

```
def sorted_linear_search(a: Array[Int], x: Int) = {
  for (i <- 0 until a.length) {
    if (a(i) == x)
      return i
    if (a(i) > x)
      return -1
  }
  -1
}
```

But worst case running time is still $O(n)$.

Binary Search

Binary search: a recursive solution. Compare x with the middle element of a, and recursively search in the left or the right half.

Like searching in a dictionary or telephone book.

```
def find(x: Int, a: Array[Int]): Int =
  find(x, a, 0, a.length-1)
```

Precondition: $a(k) < x$ for $k < i$ and $a(k) \geq x$ for $k > j$.
Output is in $\{i, \ldots j+1\}$

```
def find(x: Int, a: Array[Int], i: Int, j: Int) {
  if (j < i) return i
  val mid = (i + j) / 2
  if (a(mid) < x)
    find(x, a, mid+1, j)
  else
    find(x, a, i, mid-1)
}
```

Can we replace `mid+1` by `mid`?

Note: base case is not a sublist of size 1, but of size 0 (when $j = i - 1$).


```
def find(x: Int, a: Array[Int]): Int = {
  var i = 0
  var j = a.length - 1
  while (i <= j) {
    // a(k) < x for k < i and a(k) >= x for k > j
    val mid = (i + j) / 2
    if (a(mid) < x)
      i = mid + 1
    else
      j = mid - 1
  }
  i
}
```

Loop invariant

Note: It was easy to convert the recursive version because it used tail recursion.