KAIST CS206

## Searching

Python Lists have a method `index(x)`, which returns the index of the first element that is equal to `x`.

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

```
def linear_search(a, x):
  for i in range(len(a)):
    if a[i] == x:
      return i
  raise ValueError(x)
```

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

KAIST CS206

## Searching a sorted list

Can we do better if the list is sorted?

Given a list $A$ 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, x):
  for i in range(len(a)):
    if a[i] == x:
      return i
    if a[i] > x:
      raise ValueError(x)
  raise ValueError(x)
```

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

KAIST CS206

## Searching a sorted list

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

Given a list $A$ 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 `len(A)`.

```
def sorted_linear_search(a, x):
  for i in range(len(a)):
    if a[i] >= x:
      return i
  return len(a)
```

KAIST CS206

## 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.

## Binary search

```
def binary_search(a, x):
  return find_rec(a, x, 0, len(a) - 1)
```

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

```
def find_rec(a, x, i, j):
  if j < i:
    return i
  mid = (i + j) // 2
  if a[mid] < x:
    return find_rec(a, x, mid+1, j)
  else:
    return find_rec(a, x, 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 $j = i - 1$).

## Binary search — iterative version

```
def binary_search(a, x):
  i = 0
  j = len(a) - 1
  while i <= j:
    # a[k] < x for k < i and a(k) >= x for k > j
    mid = (i + j) // 2
    if a[mid] < x:
      i = mid + 1
    else:
      j = mid - 1
  return i
```

Loop invariant

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