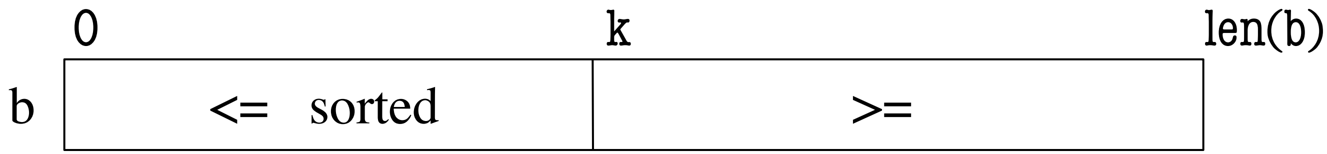
Review 7

Sequence Algorithms

Three Types of Questions

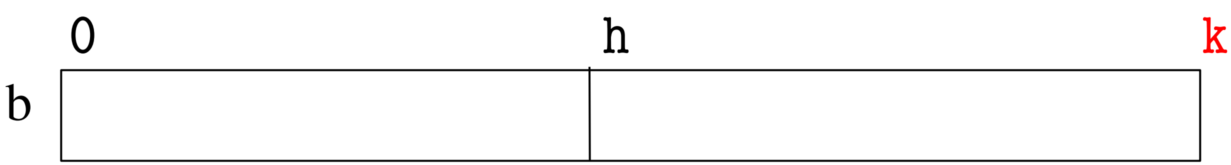
- Write body of a loop to satisfy a given invariant.
  - Exercise 6, Fall 2013 (Final)
  - Exercise 6, Spring 2014 (Final)
- Given an invariant with code, identify all errors.
  - Exercise 6, Spring 2014 (Prelim 2)
  - Exercise 6, Spring 2013 (Final)
- Given an example, rewrite it with new invariant.
  - Lab 13 (the optional one)

Horizontal Notation for Sequences



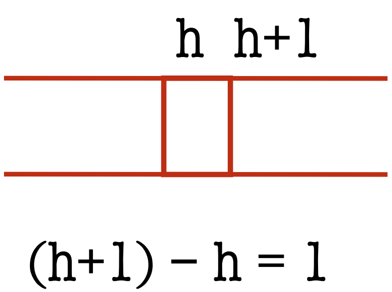
Example of an assertion about an sequence b. It asserts that:

1. b[0..k–1] is sorted (i.e. its values are in ascending order)
2. Everything in b[0..k–1] is ≤ everything in b[k..len(b)–1]



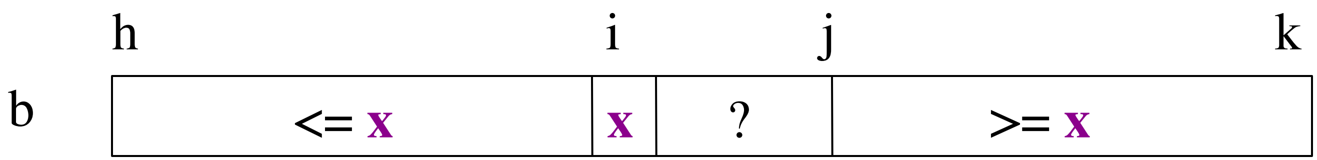
Given index h of the first element of a segment and index k of the element that follows that segment, the number of values in the segment is k – h.

b[h .. k – 1] has k – h elements in it.



DOs and DON'Ts #3

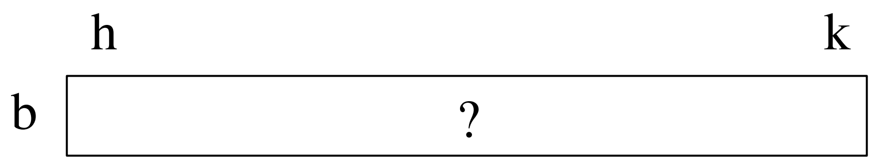
- DON'T put variables directly above vertical line.



  - Where is j?
  - Is it unknown or >= x?

Algorithm Inputs

- We may specify that the list in the algorithm is
  - b[0..len(b)-1] or
  - a segment b[h..k] or
  - a segment b[m..n-1]
- **Work with whatever is given!**



- Remember formula for # of values in an array segment
  - Following – First
  - e.g. the number of values in b[h..k] is k+1–h.

Three Types of Questions

- Write body of a loop to satisfy a given invariant.
  - Exercise 6, Fall 2013 (Final)
  - Exercise 6, Spring 2014 (Final)
- Given an invariant with code, identify all errors.
  - Exercise 6, Spring 2014 (Prelim 2)
  - Exercise 6, Spring 2013 (Final)
- Given an example, rewrite it with new invariant.
  - Lab 13 (the optional one)

Exercise 6, Fall 2013 Final

	0	k
pre: b	sorted	

	0	h	k
post: b	unchanged	b[0..k] w/o duplicates	

	0	p	h	k
inv: b	unchanged	Unchanged, values all in b[h+1..k]	b[p+1..k] w/o duplicates	

- **Example**:
  - Input [1, 2, 2, 2, 4, 4, 4]
  - Output [1, 2, 2, 2, 1, 2, 4]

Solution to Fall 2013 Final

		0	p	h	k
inv:	b	unchanged	Unchanged, values all in b[h+1..k]	b[p+1..k] w/o duplicates	

```
# Assume 0 <= k, so the list segment has at least one element
p =
h =
# inv: b[h+1..k] is original b[p+1..k] with no duplicates
# b[p+1..h] is unchanged from original list w/ values in b[h+1..k]
# b[0..p] is unchanged from original list
while          :
```

Solution to Fall 2013 Final

		0 p	h	k
inv:	b	unchanged	Unchanged, values all in b[h+1..k]	b[p+1..k] w/o duplicates

```
# Assume 0 <= k, so the list segment has at least one element
p = k-1
h = k-1
# inv: b[h+1..k] is original b[p+1..k] with no duplicates
# b[p+1..h] is unchanged from original list w/ values in b[h+1..k]
# b[0..p] is unchanged from original list
while          :
```

Solution to Fall 2013 Final

	0	p	h	k
inv: b	unchanged		Unchanged, values all in b[h+1..k]	b[p+1..k] w/o duplicates

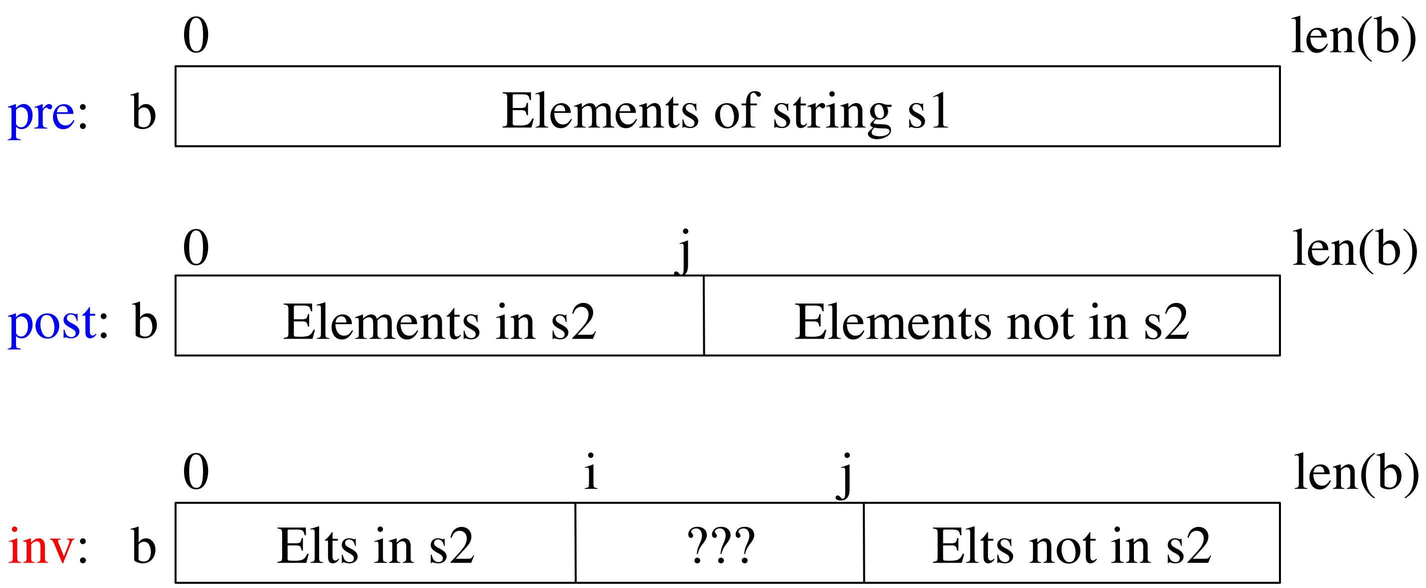
```
# Assume 0 <= k, so the list segment has at least one element
p = k-1
h = k-1
# inv: b[h+1..k] is original b[p+1..k] with no duplicates
# b[p+1..h] is unchanged from original list w/ values in b[h+1..k]
# b[0..p] is unchanged from original list
while 0 <= p:
```

Solution to Fall 2013 Final

		0	p	h	k
inv:	b	unchanged	Unchanged, values all in b[h+1..k]	b[p+1..k] w/o duplicates	

```
# Assume 0 <= k, so the list segment has at least one element
p = k-1
h = k-1
# inv: b[h+1..k] is original b[p+1..k] with no duplicates
# b[p+1..h] is unchanged from original list w/ values in b[h+1..k]
# b[0..p] is unchanged from original list
while 0 <= p:
    if b[p] != b[p+1]:
        b[h] = b[p]
        h = h-1
    p = p-1
```

Exercise 6, Spring 2014 Final



- **Example**:
- Input s1 = `'abracadabra'`, s2 = `'abc'`
- Output `'abacaabardr'` (or `'aaaabbcrdr'`)

Solution to Spring 2014 Final

```
# convert to a list b
b = list(s1)
# initialize counters



# inv: b[0..i-1] in s2; b[j+1..n-1] not in s2
while          :








# post: b[0..j] in s2; b[i+1..n-1] not in s2
# convert b back to a string
```

Solution to Spring 2014 Final

```
# convert to a list b
b = list(s1)
# initialize counters
i = 0
j = len(b) - 1
# inv: b[0..i-1] in s2; b[j+1..n-1] not in s2
while            :



# post: b[0..j] in s2; b[i+1..n-1] not in s2
# convert b back to a string
```

	0	i	j	len(b)
Inv:	Elts in s2	???	Elts not in s2	

Solution to Spring 2014 Final

```
# convert to a list b
b = list(s1)
# initialize counters
i = 0
j = len(b) - 1
# inv: b[0..i-1] in s2; b[j+1..n-1] not in s2
while j != i - 1:



# post: b[0..j] in s2; b[i+1..n-1] not in s2
# convert b back to a string
```

	0	i	j	len(b)
Inv:	Elts in s2	???	Elts not in s2	

Solution to Spring 2014 Final

```
# convert to a list b
b = list(s1)
# initialize counters
i = 0
j = len(b) - 1
# inv: b[0..i-1] in s2; b[j+1..n-1] not in s2
while j != i - 1:
    if b[i] in s2:
        i = i + 1
    else:
        b[i], b[j] = b[j], b[i]   # Fancy swap syntax in python
        j = j − 1
# post: b[0..j] in s2; b[i+1..n-1] not in s2
# convert b back to a string
```

	0	i	j	len(b)
Inv:	Elts in s2	???	Elts not in s2	

Solution to Spring 2014 Final

```
# convert to a list b
b = list(s1)
# initialize counters
i = 0
j = len(b) - 1
# inv: b[0..i-1] in s2; b[j+1..n-1] not in s2
while j != i - 1:
    if b[i] in s2:
        i = i + 1
    else:
        b[i], b[j] = b[j], b[i]   # Fancy swap syntax in python
        j = j – 1
# post: b[0..j] in s2; b[i+1..n-1] not in s2
# convert b back to a string
result = ''.join(b)
```

	0	i	j	len(b)
Inv:	Elts in s2	???	Elts not in s2	

Three Types of Questions

- Write body of a loop to satisfy a given invariant.
  - Exercise 6, Fall 2013 (Final)
  - Exercise 6, Spring 2014 (Final)
- Given an invariant with code, identify all errors.
  - Exercise 6, Spring 2014 (Prelim 2)
  - Exercise 6, Spring 2013 (Final)
- Given an example, rewrite it with new invariant.
  - Lab 13 (the optional one)

Exercise 6, Spring 2014 Prelim 2

```
def partition(b, z):
    i = 1
    k = len(b)
    # inv: b[0..i-1] <= z and b[k..] > z
    while i != k:
        if b[i] <= z:
            i = i + 1
        else:
            k = k−1
            b[i], b[k] = b[k], b[i]    # python swap
    # post: b[0..k-1] <= z and b[k..] > z
    return k
```

	0	i	k	len(b)
inv: b	<= **z**	???	>= **z**	

Exercise 6, Spring 2014 Prelim 2

```
def partition(b, z):
    i = 1     i = 0
    k = len(b)
    # inv: b[0..i-1] <= z and b[k..] > z
    while i != k:
        if b[i] <= z:
            i = i + 1
        else:
            k = k−1
            b[i], b[k] = b[k], b[i]     # python swap
    # post: b[0..k-1] <= z and b[k..] > z
    return k
```

	0	i	k	len(b)
inv: b	<= **z**	???	>= **z**	

Exercise 6, Spring 2014 Prelim 2

```
def partition(b, z):
    i = -1
    k = len(b)
    # inv: b[0..i] <= z and b[k..] > z
    while i != k:
        if b[i+1] <= z:
            i = i + 1
        else:
            b[i+1], b[k−1] = b[k−1], b[i+1]    # python swap
            k = k−1
    # post: b[0..k-1] <= z and b[k..] > z
    return k
```

	0	i	k	len(b)
inv: b	<= **z**	???	>= **z**	

Exercise 6, Spring 2014 Prelim 2

	0	i	k	len(b)
inv: b	<= **z**	???	>= **z**	

```python
def partition(b, z):
    i = -1
    k = len(b)
    # inv: b[0..i] <= z and b[k..] > z
    while ~~i != k~~:   i != k-1:
        if b[i+1] <= z:
            i = i + 1
        else:
            b[i+1], b[k-1] = b[k-1], b[i+1]    # python swap
            k = k-1
    # post: b[0..k-1] <= z and b[k..] > z
    return k
```

Exercise 6, Spring 2013 Final

```
def num_space_runs(s):
    """The number of runs of spaces in the string s.
    Examples: ' a f g ' is 4 'a f g' is 2 ' a bc d' is 3.
    Precondition: len(s) >= 1"""
    i = 1
    n = 1 if s[0] == ' ' else 0
    # inv: s[0..i] contains n runs of spaces
    while i != len(s):
        if s[i] == ' ' and s[i-1] != ' ':
            n = n+1
        i = i+1
    # post: s[0..len(s)-1] contains n runs of spaces return n
    return n
```

Exercise 6, Spring 2013 Final

```
def num_space_runs(s):
    """The number of runs of spaces in the string s.
    Examples: ' a f g ' is 4 'a f g' is 2 ' a bc d' is 3.
    Precondition: len(s) >= 1"""
    ~~i = 1~~    i = 0
    n = 1 if s[0] == ' ' else 0
    # inv: s[0..i] contains n runs of spaces
    while i != len(s):
        if s[i] == ' ' and s[i-1] != ' ':
            n = n+1
        i = i+1
    # post: s[0..len(s)-1] contains n runs of spaces return n
    return n
```

Exercise 6, Spring 2013 Final

```
def num_space_runs(s):
    """The number of runs of spaces in the string s.
    Examples: ' a f g ' is 4 'a f g' is 2 ' a bc d' is 3.
    Precondition: len(s) >= 1"""
    ~~i = 1~~    i = 0
    n = 1 if s[0] == ' ' else 0
    # inv: s[0..i] contains n runs of spaces
    while i ~~!= len(s)~~):  i != len(s)-1
        if s[i] == ' ' and s[i-1] != ' ':
            n = n+1
        i = i+1
    # post: s[0..len(s)-1] contains n runs of spaces return n
    return n
```

Exercise 6, Spring 2013 Final

```
def num_space_runs(s):
    """The number of runs of spaces in the string s.
    Examples: ' a f g ' is 4 'a f g' is 2 ' a bc d' is 3.
    Precondition: len(s) >= 1"""
    i = 1    i = 0
    n = 1 if s[0] == ' ' else 0
    # inv: s[0..i] contains n runs of spaces
    while i != len(s):  i != len(s)-1
        if s[i] == ' ' and s[i-1] != ' ':   s[i+1] == ' ' and s[i] != ' ':
            n = n+1
        i = i+1
    # post: s[0..len(s)-1] contains n runs of spaces return n
    return n
```

Three Types of Questions

- Write body of a loop to satisfy a given invariant.
  - Exercise 6, Fall 2013 (Final)
  - Exercise 6, Spring 2014 (Final)
- Given an invariant with code, identify all errors.
  - Exercise 6, Spring 2014 (Prelim 2)
  - Exercise 6, Spring 2013 (Final)
- Given an example, rewrite it with new invariant.
  - Lab 13 (the optional one)

Partition Example

```
# Make invariant true at start
j = h
t = k+1
# inv: b[h..j–1] <= x = b[j] <= b[t..k]
while j < t–1:
    if b[j+1] <= b[j]:
        swap b[j] and b[j+1]
        j = j+1
    else:
        swap b[j+1] and b[t-1]
        t=t–1
# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

```
# Make invariant true at start
j =
q =
# inv: b[h..j–1] <= x = b[j] <= b[q+1..k]
while      :

# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j		t k
inv: b	<= x	x	???	>= x

Partition Example

```
# Make invariant true at start
j = h
t = k+1
# inv: b[h..j–1] <= x = b[j] <= b[t..k]
while j < t–1:
    if b[j+1] <= b[j]:
        swap b[j] and b[j+1]
        j = j+1
    else:
        swap b[j+1] and b[t-1]
        t=t–1
# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j		t	k
inv: b	<= x	x	???	>= x	

```
# Make invariant true at start
j =
q =
# inv: b[h..j–1] <= x = b[j] <= b[q+1..k]
while       :

# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j		q	k
inv: b	<= x	x	???	>= x	

Partition Example

```
# Make invariant true at start
j = h
t = k+1
# inv: b[h..j–1] <= x = b[j] <= b[t..k]
while j < t–1:
    if b[j+1] <= b[j]:
        swap b[j] and b[j+1]
        j = j+1
    else:
        swap b[j+1] and b[t-1]
        t=t–1
# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j		t k
inv: b	<= x	x	???	>= x

```
# Make invariant true at start
j = h
q = k
# inv: b[h..j–1] <= x = b[j] <= b[q+1..k]
while j < q:


# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j	q	k
inv: b	<= x	x	???	>= x

Partition Example

```
# Make invariant true at start
j = h
t = k+1
# inv: b[h..j–1] <= x = b[j] <= b[t..k]
while j < t–1:
    if b[j+1] <= b[j]:
        swap b[j] and b[j+1]
        j = j+1
    else:
        swap b[j+1] and b[t-1]
        t=t–1
# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j	t	k
inv: b	<= x	x	???	>= x

```
# Make invariant true at start
j = h
q = k
# inv: b[h..j–1] <= x = b[j] <= b[q+1..k]
while j < q:
    if b[j+1] <= b[j]:
        swap b[j] and b[j+1]
        j = j+1
    else:
        swap b[j+1] and b[q]
        q=q–1
# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j	q	k
inv: b	<= x	x	???	>= x

Partition Example

```
# Make invariant true at start
j = h
t = k+1
# inv: b[h..j–1] <= x = b[j] <= b[t..k]
while j < t–1:
    if b[j+1] <= b[j]:
        swap b[j] and b[j+1]
        j = j+1
    else:
        swap b[j+1] and b[t-1]
        t=t–1
# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

```
# Make invariant true at start
j =
m =
# inv: b[h..j–1] <= x = b[j] <= b[j+1..m]
while      :

# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j		t k
inv: b	<= x	x	???	>= x

Partition Example

```
# Make invariant true at start
j = h
t = k+1
# inv: b[h..j–1] <= x = b[j] <= b[t..k]
while j < t–1:
    if b[j+1] <= b[j]:
        swap b[j] and b[j+1]
        j = j+1
    else:
        swap b[j+1] and b[t-1]
        t=t–1
# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

```
# Make invariant true at start
j = h
m = h
# inv: b[h..j–1] <= x = b[j] <= b[j+1..m]
while      :

# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j	t	k
inv: b	<= x	x	???	>= x

	h	j	m	k
inv: b	<= x	x	>= x	???

Partition Example

```
# Make invariant true at start
j = h
t = k+1
# inv: b[h..j–1] <= x = b[j] <= b[t..k]
while j < t–1:
    if b[j+1] <= b[j]:
        swap b[j] and b[j+1]
        j = j+1
    else:
        swap b[j+1] and b[t-1]
        t=t–1
# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j		t k
inv: b	<= x	x	???	>= x

```
# Make invariant true at start
j = h
m = h
# inv: b[h..j–1] <= x = b[j] <= b[j+1..m]
while m < k:

# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j	m	k
inv: b	<= x	x	>= x	???

Partition Example

```
# Make invariant true at start
j = h
t = k+1
# inv: b[h..j–1] <= x = b[j] <= b[t..k]
while j < t–1:
    if b[j+1] <= b[j]:
        swap b[j] and b[j+1]
        j = j+1
    else:
        swap b[j+1] and b[t-1]
        t=t–1
# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j	t	k
inv: b	<= x	x	???	>= x

```
# Make invariant true at start
j = h
m = h
# inv: b[h..j–1] <= x = b[j] <= b[j+1..m]
while m < k:
    if b[m+1] <= b[j]:
        swap b[j] and b[m+1]
        swap b[j+1] and b[m+1]
        m = m+1; j=j+1
    else:
        m = m+1
# post: b[h..j–1] <= x = b[j] <= b[j+1..k]
```

	h	j	m	k
inv: b	<= x	x	>= x	???

Questions?