http://www.cs.cornell.edu/courses/cs1110/2018sp

Lecture 25: Sequence Algorithms

CS 1110

Introduction to Computing Using Python

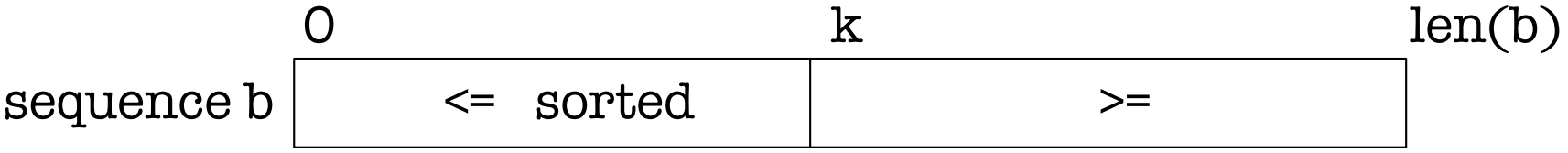
Cornell CIS
COMPUTING AND INFORMATION SCIENCE

[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]

Announcements

- Prelim 2 is graded
- A5 is out!
  - Clarification on webpage about what happens when the deck has too few cards in it
- Final Lab is this week:
  - **Reminder:** labs **cannot** be checked off during Wed consulting hours the week after they are released
- Final Exam:
  - May 17th, 9am-11:30am
  - **Location**: Barton Hall Central and East

Box Notation for Sequences



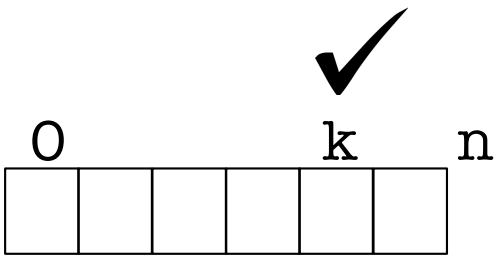
Graphical assertion about sequence b. It asserts that:

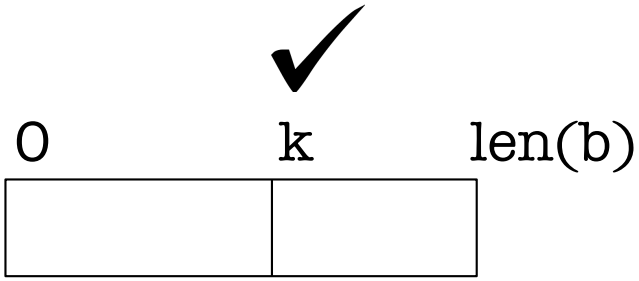
1. b[0..k–1] is sorted (values are in ascending order)
2. all of b[0..k–1] is ≤ all of b[k..len(b)–1]

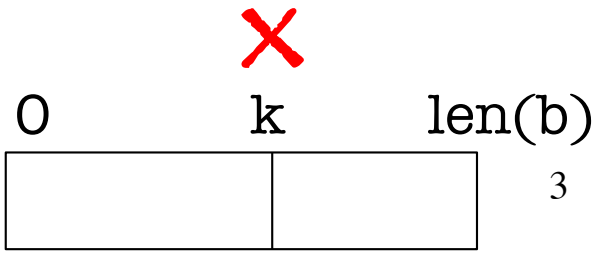
Pro Tip #1:

index always goes *above a box*, never above a line

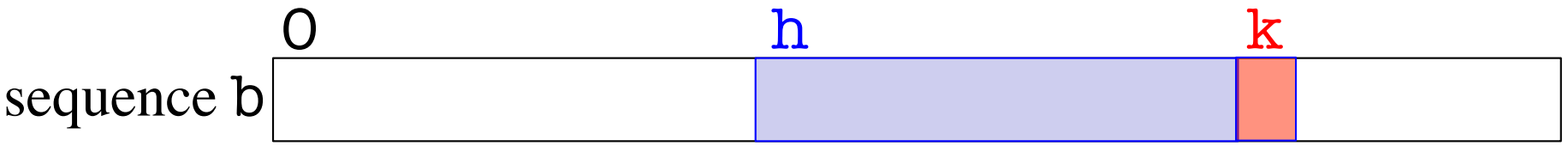
(just like house numbers go on a house not between the houses)







Q: Indices for Box Notation



Given:

- index **h** of the first element of a segment
- index **k** of the element that follows that segment,

Questions:

1. How many values are in segment `b[h .. k - 1]`
2. How many values are in `b[h .. h - 1]` ?
3. How many values are in `b[h .. h + 1]` ?

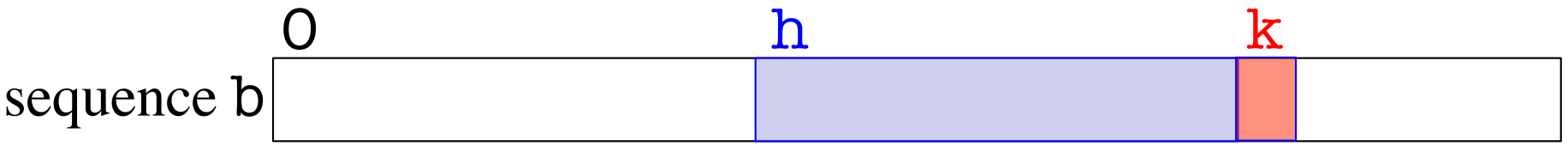
A: `0`
B: `1`
C: `2`
D: `k - h`
E: `k + h`

Pro Tip #2:

Size is "Follower minus First"

Follower: next thing outside the specified range

A: Indices for Box Notation



Given:

- index **h** of the first element of a segment
- index **k** of the element that follows that segment,

Questions:

1. How many values are in segment `b[h .. k - 1]` **D**
2. How many values are in `b[h .. h - 1]` ? **A**
3. How many values are in `b[h .. h + 1]` ? **C**

A: `0`
B: `1`
C: `2`
D: `k - h`
E: `k + h`

count num adjacent equal pairs (1)

Approach #1: compare s[k] to the character in front of it (s[k-1])

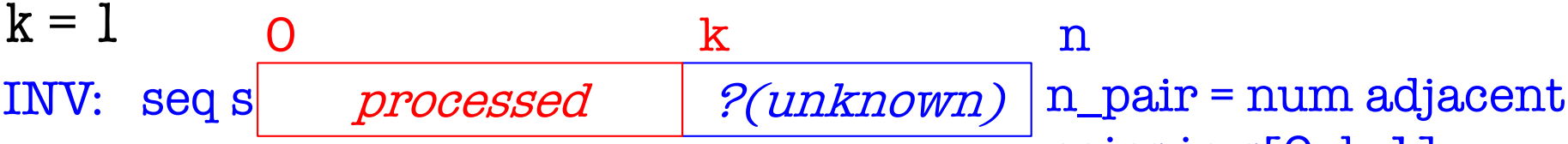
```
# set n_pair to # adjacent equal pairs in s



n_pair = 0
k = 1

while  k < len(s):
    if s[k-1] == s[k]:
        n_pair += 1
    k = k + 1
```

count num adjacent equal pairs (2)

Approach #1: compare s[k] to the character in front of it (s[k-1])

```
# set n_pair to # adjacent equal pairs in s
```


```
n_pair = 0
k = 1
```




```
while  k < len(s):
    if s[k-1] == s[k]:
        n_pair += 1
    k = k + 1
```

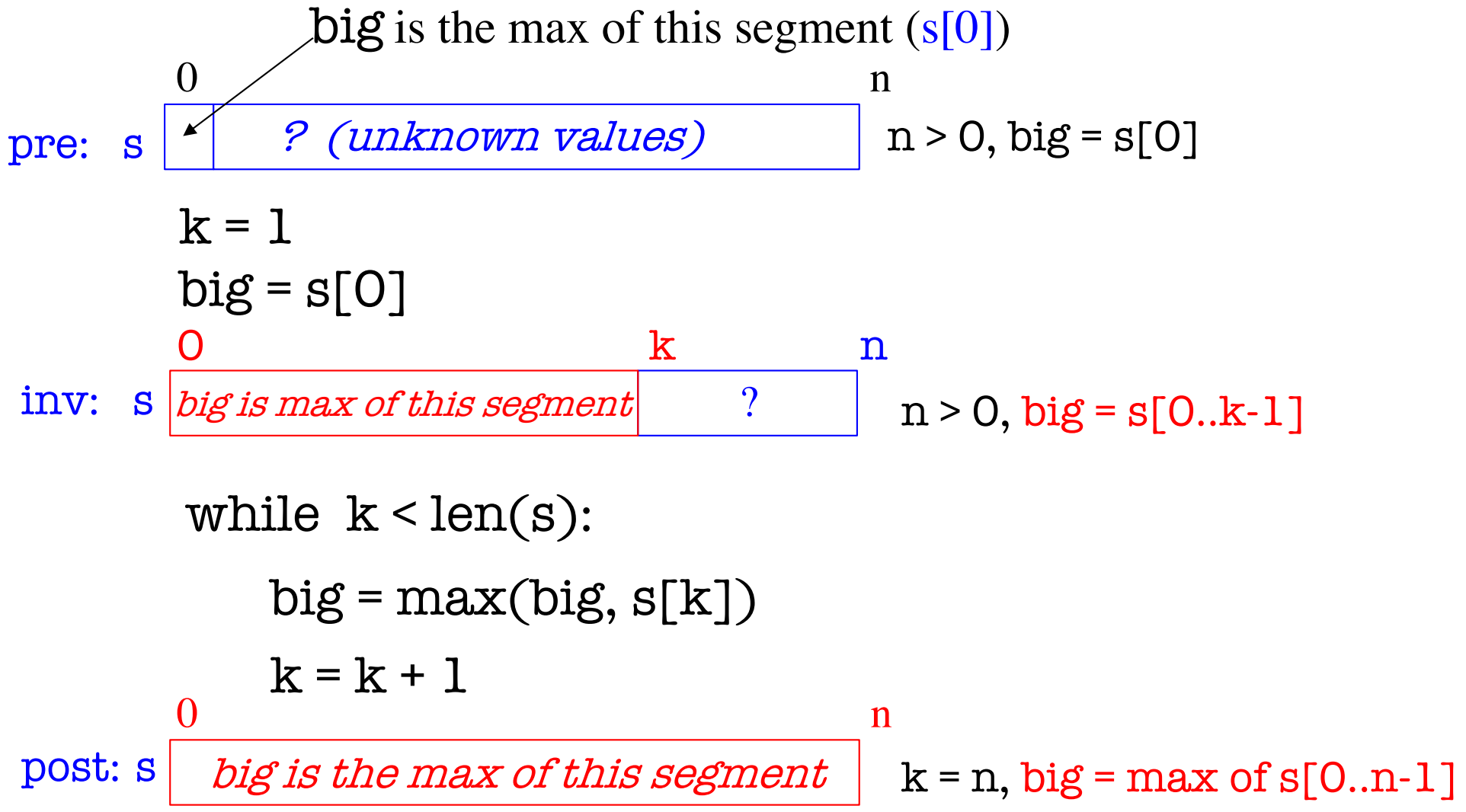
find the max of a seq (1)

Task: find the maximum of a sequence s

```
k = 1
big = s[0]

while  k < len(s):
    big = max(big, s[k])
    k = k + 1
```

find the max of a seq (2)

Task: find the maximum of a sequence s

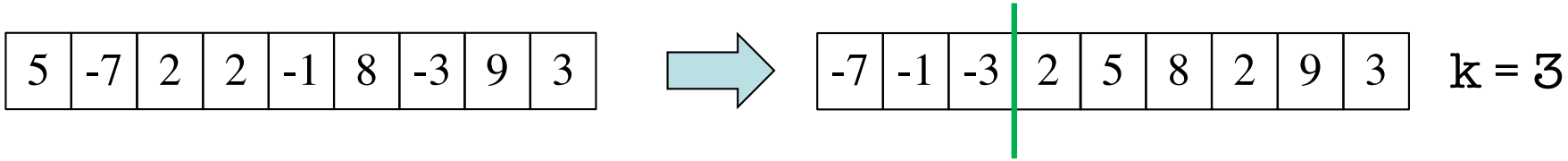


Developing Algorithms on Sequences

- Specify the algorithm by giving its precondition and postcondition as pictures.
- Draw the invariant by drawing another picture that "moves from" the precondition to the postcondition
  - The invariant is true at the beginning and at the end
- The four loop design questions
  1. How does loop start (how to make the invariant true)?
  2. How does it stop (is the postcondition true)?
  3. How does the body make progress toward termination?
  4. How does the body keep the invariant true?

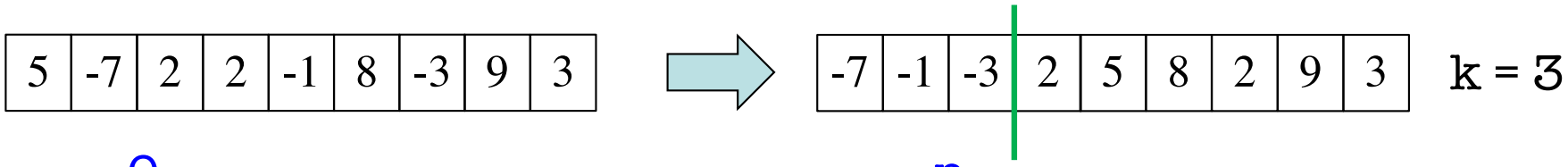
Task: separate + from – in a list

Task: Put negative values before nonnegative ones and return the split index

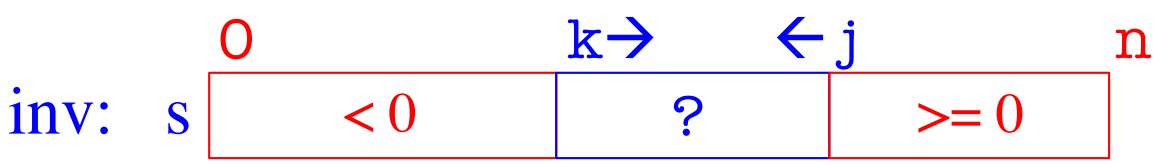


Invariants: separate + from – in a list

Task: Put negative values before nonnegative ones and return the split index

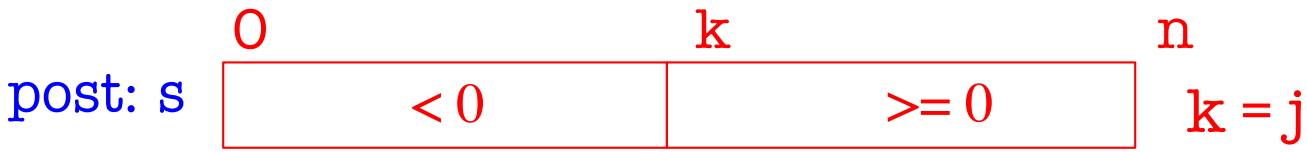



```
k = 0
j = n
```



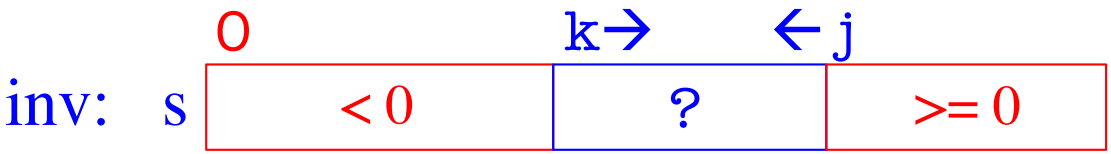
s[0..k-1] negative
s[j..n-1] zero or +
s[k..j-1] unknown

```
while k < j:

    <body goes here>
```



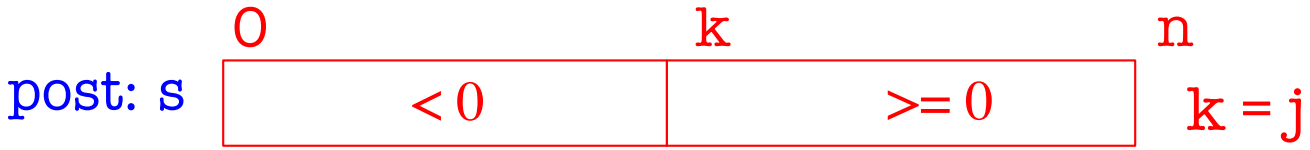
Body: separate + from – in a list

```
k = 0
j = n
```



s[0..k-1] negative
s[j..n-1] zero or +
s[k..j-1] unknown

```
while k < j:
    if s[k] < 0: # kth elem stays where it is
        k = k + 1
    elif s[j-1] >= 0: # (j-1)th elem stays where it is
        j = j - 1
    else: # both elements in the wrong place
    swap(s, k, j-1)
        k = k + 1
        j = j - 1
```



k = j