**CS1110**

Lecture 21: Invariants, Quicksort, Partitioning

**Upcoming schedule**

Friday Apr 18: A4 due

Monday Apr 21: deadline for lab 11 checkoff

Tuesday Apr 22

- Lecture = (optional) review session
- Lab sessions = (optional) drop-in office hours. No new lab.
- Prelim: 7:30-9pm, 200 Baker Lab (= the auditorium)

Wednesday Apr 23: No labs or office hours

Prelim grades announced by email from CMS, hopefully by Thursday morning.

---

**At most one line needs to be fixed**

1. def ap1(s):
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 

(A) Error in line 3
(B) line 6
(C) line 7
(D) line 8
(E) no error

---

**At most one line needs to be fixed**

1. def ap2(s):
2. 
3. 
4. 
5. 
6. 
7. 
8. 

(A) Error in line 3
(B) line 6
(C) line 7
(D) line 8
(E) no error

---

**Sorting: A Key Algorithmic Family**

**Q:** Given a list of items, how can we arrange for them to be sorted in increasing order, in a time- and space-efficient manner?

Applications: making items easier to find.¹

**def sort(b, h, k):**

```python
"""Sort b[h..k] in place. Pre: b: list of ints; k>=h-1"""
# Start with b[h], and organize the rest according to it. No...
# Note: we have h & k explicit to simplify recursive structure.
```

¹Also, computing poker-hand probabilities.

---

**Prelim 2 coverage:** Lectures 12-22 (this Thursday), associated labs and assignments: recursion, using classes effectively, loops and loop invariants, sequence algorithms.

**Preparing for the exam**

Past exams are posted on the Exams section of the website. Profs Lee and Marschner wrote the Spring 2013 one, and the topic coverage is equivalent.

Review all lectures up to and including April 17. Be able to do A3, A4, labs 8-11 from scratch, cold. Be able to do the worked invariants exercises yourself.

As always, come to our many office hours/consulting hours for in-person help; see the Staff section of the webpage.

As always, watch Piazza for announcements, for helpful answers to other people's questions, etc.

---

**Motivation: A Famous Sorting Function**

**def qsort(b, h, k):**

```python
"""Make b[h..k] sorted. Pre: b: list of ints; k>=h-1"""
```

**def partition(b, h, k):**

```python
"""Let x = b[h] be the pivot value. Rearrange b[h..k] so that there is an i where b[h..i-1] <= x, b[i]=x; b[i+1..k] >=x. Return i. Pre: k>=h-1"""
```

**Clicker Q1: recursive case**

# Can you do this without creating extra lists?
Pictorial Notation for Sequence Assertions

\[ \begin{array}{c|c|c}
0 & h & k \\
\hline
\text{some property } p & & \text{some property } q \\
\end{array} \]

Equivalent to:

Property \( p \) holds on all items in \( b[0..h-1] \), and property \( q \) holds on all items in \( b[h..k] \).

(The precise location of the "vertical bars" matters.)

Can also indicate single items.

\[ ((h+1) - h = 1; \text{it's all consistent, hurrah.}) \]

Motivation: A Famous Sorting Function

```python
def qsort(b, h, k):
    """Make \( b[h..k] \) sorted."
    Pre: \( b: \text{list of ints}; k \geq h-1 ""
    if k < h:
        # empty is sorted
        return
    i = partition(b, h, k)

    # invariant: \( b[h..i-1] < x, b[i] = x, b[j..k] \geq x, b[i+1..j-1] \text{ unknown} \)
    while True:
        if b[i+1] >= x:
            # Move to end of block.
            b[i+1], b[j-1] = b[j-1], b[i+1]
            j = j - 1
        else:
            # b[i+1] < x
            CLICKER Q1
            CLICKER Q2
            j = j - 1
    return i
```

Partition Algorithm Implementation

```python
def partition(b, h, k):
    """Partition list \( b[h..k] \) around a pivot \( x = b[h] \); return index of pivot point"
    Pre: \( k \geq h ""
    what goes here?
    # invariant: \( b[h..j-1] < x, b[j] = x, b[j+1..k] \geq x, b[i..j-1] \text{ unknown} \)
    while what goes here?
        if b[i+1] >= x:
            # Move to end of block.
            b[i+1], b[j-1] = b[j-1], b[i+1]
            j = j - 1
        else:
            # b[i+1] < x
            CLICKER Q3
            # post: \( b[h..j-1] < x, b[j] = x, \text{ and } b[i..j-1] \geq x \)
            return i
```

An Invariant to Guide Our Thinking

```
(A) qsort(b,h,i-1)
qsort(b,h,i+1,k)
qsort(b,h,i+1,k)
qsort(b,h,i+1,k)
```

An Invariant to Guide Our Thinking

```
(A) \[ b[i+1], b[j-1] = b[j-1], b[i-1] \]
    \[ j = j - 1 \]
```

```
(B) \[ b[i+1], b[j-1] = b[j-1], b[i-1] \]
```

```
(C) \[ b[i+1], b[i] = b[i], b[i+1] \]
```

```
(D) \[ b[i+1], b[i] = b[i], b[i+1] \]
    \[ i += 1 \]
```