

CSCI 241

Lecture 5

Stability, Comparison Sorts, Radix Sort

Happenings next week

Monday, 1/21 – Martin Luther King Jr. Day, No Class!

Tuesday, 1/22 – CS SMATE Faculty Candidate **Gerald Raj**: Research Talk – 4 pm in CF 316

Tuesday, 1/22 – [ACM Research Talk: Computer Vision with Dr. Scott Wehrwein](#) – 5 pm in CF 316

Wednesday, 1/23 – CS SMATE Faculty Candidate **Gerald Raj**: Teaching Talk – 4 pm in CF 316

Wednesday, 1/23 – [Peer Lecture Series: CS Success Workshop](#) – 5 pm in CF 420

Thursday, 1/24 – CS SMATE Faculty Candidate **Cecily Heiner**: Research Talk – 4 pm in CF 316

Friday, 1/25 – CS SMATE Faculty Candidate **Cecily Heiner**: Teaching Talk – 4 pm in **CF 115**




THE MENTORS PROGRAM PRESENTS

MENTORS+

A Peer Mentorship Program

RESEARCH
INTERNSHIPS
WORKLOAD
RESOURCES
NETWORKING



Come talk to Mentors about Computer Science... and life!
This time is for YOU to ask us questions about anything and
everything about our experiences as Computer Science
students at WWU.

CF 167

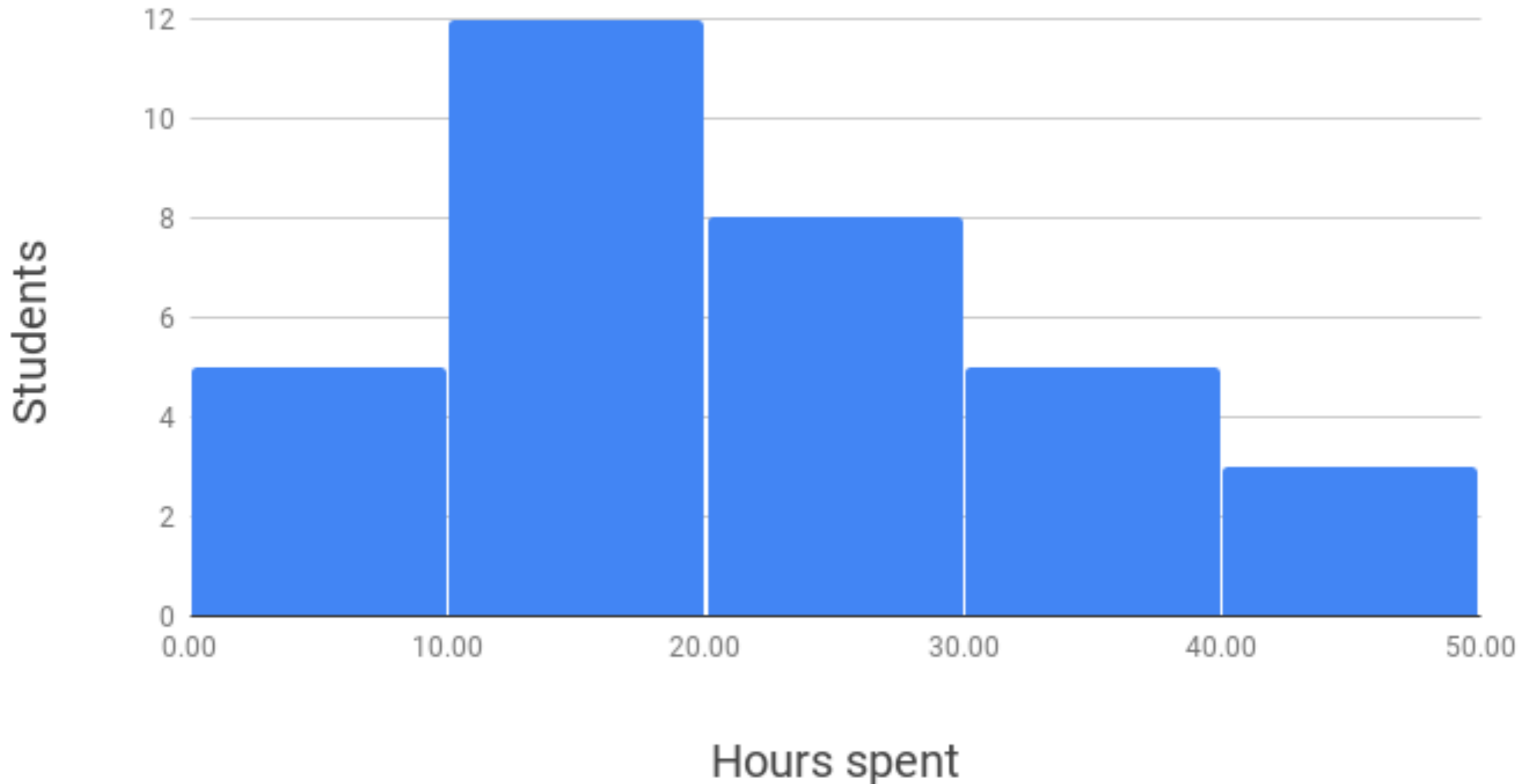
3:30-4:00

Tuesday, Wednesday & Thursday



Announcements

Hours spent on A1 last quarter



Goals:

- Know what it means for a sorting algorithm to be **stable**
- Understand the distinction between comparison and non-comparison sorts.
- Be prepared to implement radix sort.

Stability

Objects can be sorted on **keys** - **different** objects may have the same value.

- e.g., sorting a collection of Students by first name only.

A **stable** sort maintains the order of distinct elements with the same key.

Stability

A **stable** sort maintains the order of elements with the same value.

[6^* 2^* 6^+ 2^+ 3 4]

Stably sorted: [2^* 2^+ 3 4 6^* 6^+]

Unstably sorted: [2^+ 2^* 3 4 6^* 6^+]

Stability

A **stable** sort maintains the order of elements with the same value.

In groups: Sort this list using insertion and selection sort. Is either algorithm stable?

[6* 2* 6+ 2+ 3 4]

```
insertionSort(A):  
  i = 0;  
  while i < A.length:  
    // push A[i] to  
    // its sorted position  
    // in A[0..i]  
    // increment i
```

```
selectionSort(A):  
  i = 0;  
  while i < A.length:  
    // find min of A[i..A.length]  
    // swap it with A[i]  
    // increment i
```


Comparison sorts operate by comparing pairs of elements.

How do you sort without comparing elements?

Suppose I gave you 10 sticky notes with the digits 0 through 9.

What algorithm would you use to sort them?

How many times did you need to look at each sticky note?

What if there are duplicates?

Refresher:

Stacks and Queues

(LIFO)

(FIFO)

```
Stack s;  
Queue q;  
  
for i in 1..5:  
    s.add(i) // push  
    q.add(i) // enqueue  
for i in 1..5:  
    print s.remove() // pop  
    print q.remove() // dequeue
```

ABCD: What is printed?

A. 1 1 2 2 3 3 4 4

B. 1 4 2 3 3 2 4 1

C. 4 1 3 2 2 3 1 4

D. 4 4 3 3 2 2 1 1

Stability

Objects can be sorted on **keys** - **different** objects may have the same value.

A **stable** sort maintains the order of distinct elements with the same key.

- Example: sorting on 10's place only

[**6^{*}** **2^{*}** **6⁺** **2⁺** **3** **4**]

[**61** **21** **63** **23** **35** **48**]