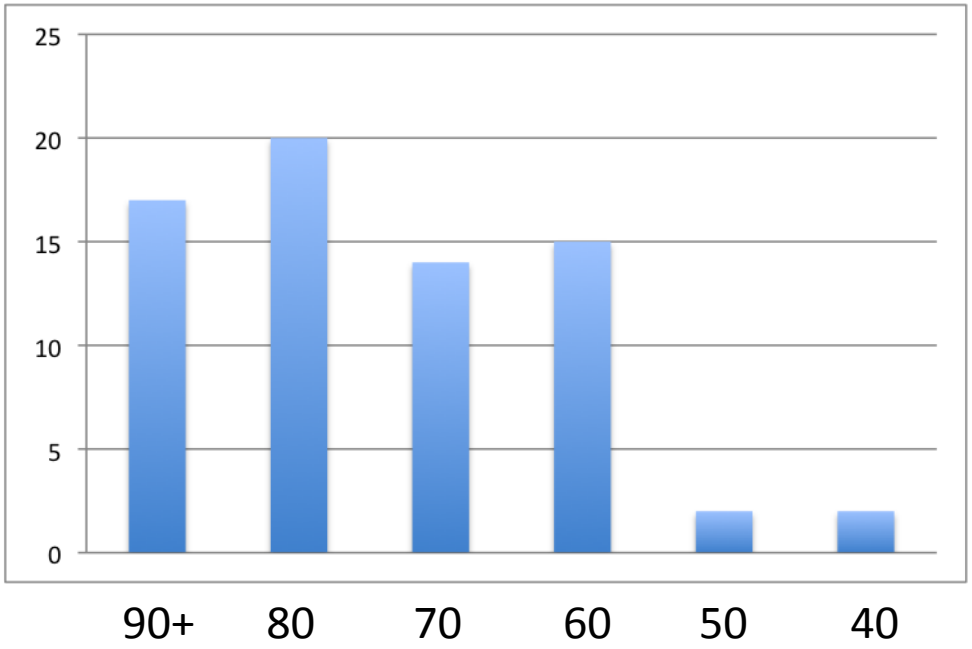
CS 150: Mergesort

Cynthia Taylor
Oberlin College
April 21st 2014

I love CS! What do I take next?

- CS 151: Datastructures
  - First look at more complicated algorithms
  - Learn java/object-orient programming
  - Learn more complicated ways to handle data

Test 2



- Mean of 78
- Median of 82
- Will be handing back in lab this week

Do all these sorts have the same Big O run time?

```
def selectionSort(L):
    for i in range(len(L)):
        min_index = i
        for j in range (i,len(L)):
            if L[j] < L[min_index]:
                min_index = j
    temp = L[i]
    L[i] = L[min_index]
    L[min_index]  = temp
```

```
def insertionSort(L):
    for i in range(len(L)):
        e = L[i]
        while (j > 0 and L[j-1] > e):
            L[j] = L[j-1]
            j = j - 1
        L[j] = e
```

```
def bubbleSort(L):
    swapped = True
    while (swapped):
        swapped = False
        for i in range(len(L)-1):
            if L[i] > L[i+1]:
                temp = L[i]
                L[i] = L[i+1]
                L[i+1] = L[i]
                swap = True
```

A. Yes

B. No

C. I don't know

Will they have the same execution time on every list?

```
def selectionSort(L):
    for i in range(len(L)):
        min_index = i
        for j in range (i,len(L)):
            if L[j] < L[min_index]:
                min_index = j
    temp = L[i]
    L[i] = L[min_index]
    L[min_index]  = temp
```

```
def insertionSort(L):
    for i in range(len(L)):
        e = L[i]
        while (j > 0 and L[j-1] > e):
            L[j] = L[j-1]
            j = j - 1
        L[j] = e
```

A. Yes

B. No

C. I don't know

```
def bubbleSort(L):
    swapped = True
    while (swapped):
        swapped = False
        for i in range(len(L)-1):
            if L[i] > L[i+1]:
                temp = L[i]
                L[i] = L[i+1]
                L[i+1] = L[i]
                swap = True
```

Which will perform best on a sorted list?

```
def selectionSort(L):
    for i in range(len(L)):
        min_index = i
        for j in range (i,len(L)):
            if L[j] < L[min_index]:
                min_index = j
    temp = L[i]
    L[i] = L[min_index]
    L[min_index]  = temp
```

```
def insertionSort(L):
    for i in range(len(L)):
        e = L[i]
        while (j > 0 and L[j-1] > e):
            L[j] = L[j-1]
            j = j - 1
        L[j] = e
```

```
def bubbleSort(L):
    swapped = True
    while (swapped):
        swapped = False
        for i in range(len(L)-1):
            if L[i] > L[i+1]:
                temp = L[i]
                L[i] = L[i+1]
                L[i+1] = L[i]
                swap = True
```

A. Bubble sort
B. Insertion sort
C. Selection sort
D. A and B
E. I don't know

We have two sorted lists, B and C. What is the MOST EFFICIENT way to combine them into one sorted list?

n^2

A. Add C to the end of B, Bubblesort the new list
B. Insertion sort, treating B as the sorted part and C as the unsorted part

$n * C \geq n^2$

C. Create a new list, insert the smallest element from either list in the new list until you have inserted all the elements

n

D. Something else
E. I don't know

Write Code for Merge

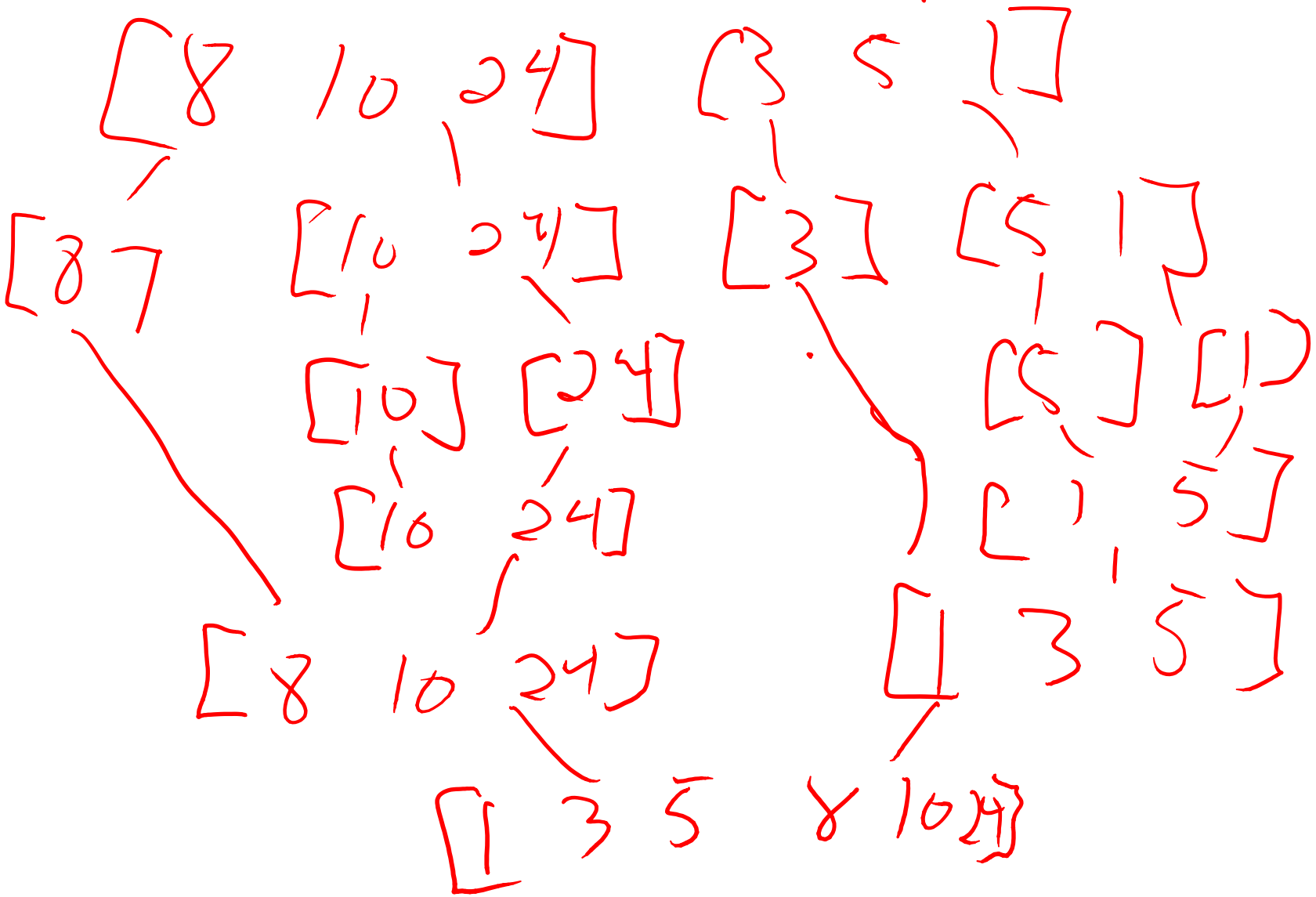
sorted lists

Mergesort

- Split the list in half recursively until you have a bunch of 1 element lists (by definition sorted)

- Merge these sorted lists back together

Mergesort: [8 10 24 3 5 1]



Write Code for Mergesort

Next Time

- Sets and Dictionaries
  - Read Section 11.6

- Lab 9 – Tuesday at 10 pm