CSCI 200: Foundational Programming Concepts & Design Lecture 36

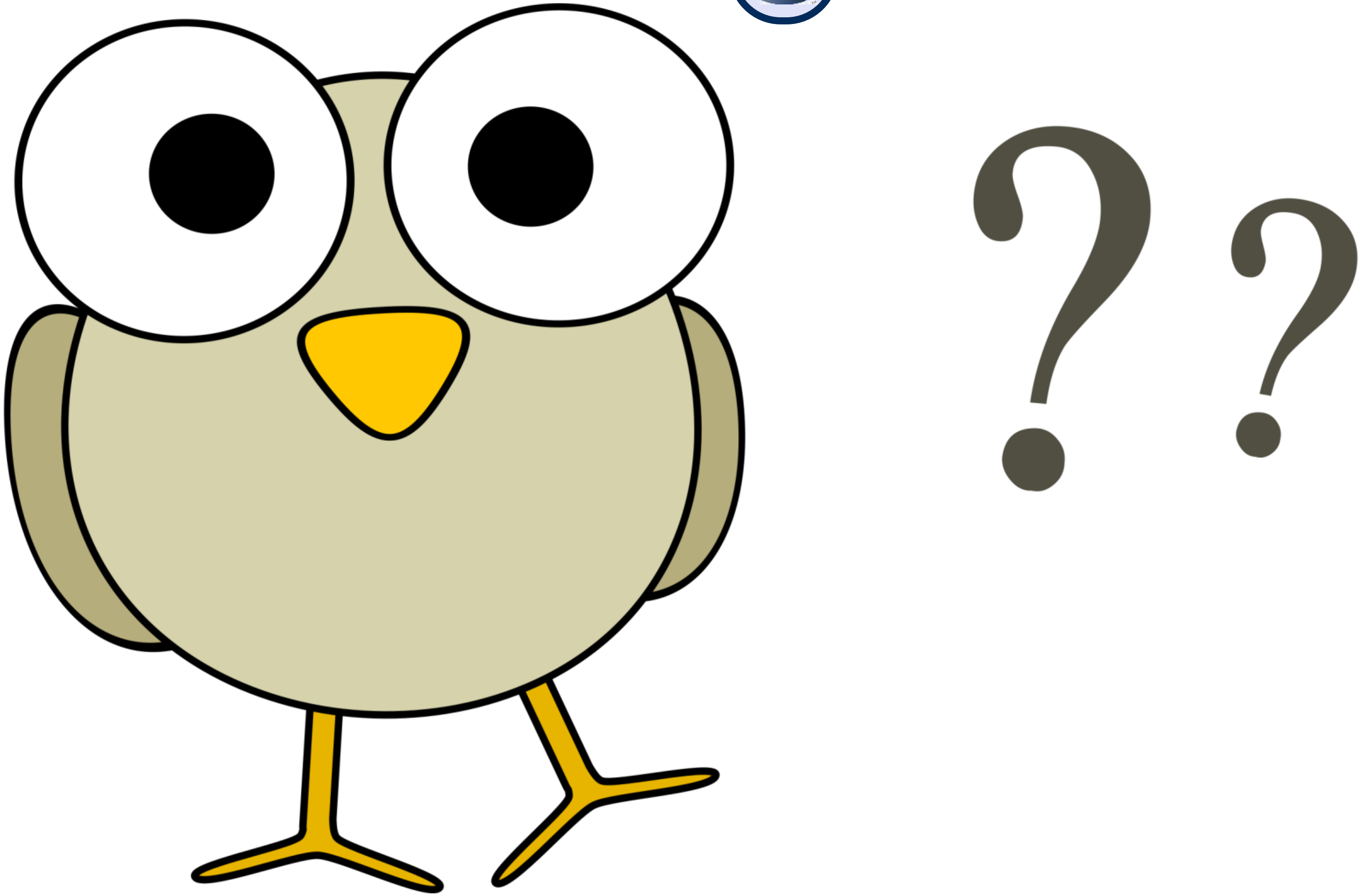
Sorting Algorithms

Previously in CSCI 200

- `try throw catch`
  - `try` to run potentially dangerous code
  - If something dangerous happens, `throw` an exception
  - `catch` the exception and handle the error

```
try {
  // statements that could throw an exception
} catch (ExceptionType1 e) {
} catch (ExceptionType2 e) {
} catch (...) { // generic catch anything that doesn't match above
}
```

Questions?



Learning Outcomes For Today

- Explain how sorting a list affects the performance of searching for a value in a list.
- Generate pseudocode to (1) find the minimum or maximum value in a list (2) sort a list using selection/insertion/bubble sort.
- Discuss the differences of how to swap elements in an array vs a linked list.
- Implement the merge sort algorithm using recursion.

First…Searching

- How does a human search for something?

Kayak word search.

```
A A Y K A K A Y K A Y Y K K A A K A Y A A A Y K Y K Y A Y K
K Y Y A K Y A A A Y K A A K K Y A K K K A A A K A A Y K A A
K K K Y A K A K A Y K K Y K K Y K K A K Y A A A A K A A K K
Y K A A K Y A A K K K K K A Y A Y Y Y Y Y A A K A A K K K K
K K A Y Y A A K K Y Y A Y A A K A A K K A A Y K A K K A A K
K K A A A K A A A K K A K K K Y A A A Y K Y A Y A K Y K A A
K K Y K A A A A A A A A A A K K A A A K Y Y A K A K A K K Y
K K K K K K A K Y Y K Y Y K K A A Y A K A A A K Y A K K A A
K A K K A K A Y K A A Y A K Y Y K K A Y K K K A A A A A K A
K K A K A K A K A A A K A Y A Y K A K A Y Y A Y K K A K Y Y
A A A Y A K K K A K K A K Y K A Y K K A K K Y K A A K K Y Y
Y K K K Y K K Y A A A K K K K A K A Y K K K K K A A K K K Y
K K A K A Y K A A K A K K Y Y A K A Y A A K Y A A A A A A A
A A A A K K Y A A K K A A Y K K A A A Y A Y A K Y A K A Y K
A K K A K K A Y K A A Y K Y K A K Y A A K K Y K K K K K K A
K Y K A A K K Y A Y K A K K K Y K A K A Y A Y K Y A A A K K
K A Y Y K K A K Y K A Y A A K A Y A A Y Y Y A K K Y K Y K K
K K K K A K A Y A A A K K A Y A Y K K Y A K A A A A Y K A Y
A Y K A K K K A A K A K A Y A A A K K Y K K A Y K A Y A Y Y
K Y A Y K A Y A K A K Y Y K Y K A K A A Y K Y K K A Y K A K
A K A K K A K K A K K K K A A A K Y Y Y Y K A K A A A A A K
K A K Y Y K A K K Y K Y A K Y A A A A A K A K A Y A A K K A
K K A A A K K A K A K K A A K A Y A A Y A A A K K A A Y Y A
K A K K Y Y K K A Y A Y K K A K K Y Y K Y K Y K K A A A K A
```

Words to find:
KAYAK

First…Searching

- How does a human search for something?

Searching & Sorting

- Easier to search for something when the collection is sorted!

On Tap For Today

- MinMax
- Sorting
  - Selection Sort
  - Insertion Sort
  - Bubble Sort
  - Merge Sort
- Practice

On Tap For Today

- MinMax
- Sorting
  - Selection Sort
  - Insertion Sort
  - Bubble Sort
  - Merge Sort
- Practice

Finding the Min/Max Value

- Pseudocode
  - Store the first value of the list as our current min/max
  - For every element in the list
    - Min – if an element is smaller than our current min, then that element is our new min
    - Max - If an element is larger than our current max, then that element is our new max

On Tap For Today

- MinMax
- Sorting
  - Selection Sort
  - Insertion Sort
  - Bubble Sort
  - Merge Sort
- Practice

Sorting Algorithms

- Many different ways to sort a list

- http://www.sorting-algorithms.com/

On Tap For Today

- MinMax
- Sorting
  - Selection Sort
  - Insertion Sort
  - Bubble Sort
  - Merge Sort
- Practice

Selection Sort

- Pseudocode
  - Find the minimum value, swap with the 1st position
  - Find the next minimum value, swap in the 2nd position
  - Continue until you have found the second to largest value and swapped in the second to last position

	8
	5
	2
	6
	9
	3
	1
	4
	0
	7

Selection Sort

- Consider a list of `n` elements
  - After we find the smallest element and put it in the first position
  - `1` element is sorted, `n-1` elements are unsorted

Selection Sort

- Consider a list of `n-1` elements
  - After we find the smallest element and put it in the first position
  - `1` element is sorted, `n-2` elements are unsorted

Selection Sort

- Consider a list of `n` elements
  - After we find the two smallest element and put them in the first two positions
  - `2` smallest elements are sorted, `n-2` elements are unsorted

Selection Sort

- Consider a list of `n` elements
  - After we find the `k` smallest element and put them in the first `k` positions
  - `k` smallest elements are sorted, `n-k` elements are unsorted

Selection Sort Pseudocode

- Can be implemented with two nested for loops

```
for i=0 to end
  k = i
  for j=i+1 to end
    if list[j] < list[k]
      k = j
  swap list[i] & list[k]
```

Sorting Complexities

Algorithm	Worst Case	Best Case	Average Case
Selection Sort			

- When does the worst case scenario occur?
- When does the best case scenario occur?

Sorting Complexities

Algorithm	Worst Case	Best Case	Average Case
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$

On Tap For Today

- MinMax
- Sorting
  - Selection Sort
  - Insertion Sort
  - Bubble Sort
  - Merge Sort
- Practice

Insertion Sort

- Pseudocode
  - For current spot
    - Move towards the front and slide larger elements over
    - Insert current value after a smaller value

6 5 3 1 8 7 2 4

Insertion Sort

- Consider a list of `n` elements
  - After we processing `k` elements and put them in the first `k` positions
  - `k` elements are sorted, `n-k` elements are unsorted

Insertion Sort Pseudocode

- Can be implemented with two nested loops

```
for i=1 to n
  x = list[i]
  j = i-1
  while j >= 0 and list[j] > x
    list[j+1] = list[j]
    j--
  list[j+1] = x
```

- Complexity?

Sorting Complexities

Algorithm	Worst Case	Best Case	Average Case
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort			

Sorting Complexities

Algorithm	Worst Case	Best Case	Average Case
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n^2)$	$O(n)$	$O(n^2)$

- Why is best case better for insertion sort?

Insertion Sort Pseudocode

- Can be implemented with two nested loops

```
for i=1 to n
  x = list[i]
  j = i-1
  while j >= 0 and list[j] > x
    list[j+1] = list[j]
    j--
  list[j+1] = x
```

- Complexity?

On Tap For Today

- MinMax
- Sorting
  - Selection Sort
  - Insertion Sort
  - Bubble Sort
  - Merge Sort
- Practice

Bubble Sort

- Pseudocode
  - Find largest element by
    - Repeatedly compare neighboring elements
    - If out of order, swap
  - Repeat for all i^{th} largest elements

- Values "bubble up" to the top

6 5 3 1 8 7 2 4

Bubble Sort

- Consider a list of `n` elements
  - After we processing `k` elements and put them in the last `k` positions
  - `k` largest elements are sorted, `n-k` elements are unsorted

Bubble Sort Pseudocoe

- Can be implemented with two nested loops

```
for i=0 to n
  for j=1 to n-i
    if list[j-1] > list[j]
      swap list[j-1] & list[j]
```

- Complexity?

Sorting Complexities

Algorithm	Worst Case	Best Case	Average Case
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n^2)$	$O(n)$	$O(n^2)$
Bubble Sort			

Sorting Complexities

Algorithm	Worst Case	Best Case	Average Case
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n^2)$	$O(n)$	$O(n^2)$
Bubble Sort	$O(n^2)$	$O(n)$	$O(n^2)$

Sorting Complexities

Algorithm	Worst Case	Best Case	Average Case
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n^2)$	$O(n)$	$O(n^2)$
Bubble Sort	$O(n^2)$	$O(n)$	$O(n^2)$

```
for i=0 to n
  numSwaps = 0
  for j=1 to n-i
    if list[j-1] > list[j]
      swap list[j-1] & list[j]
      numSwaps++
  if numSwaps == 0
    break
```

Sorting Complexities

Algorithm	Worst Case	Best Case	Average Case
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n^2)$	$O(n)$	$O(n^2)$
Bubble Sort	$O(n^2)$	$O(n)$	$O(n^2)$

- Not ideal

On Tap For Today

- MinMax
- Sorting
  - Selection Sort
  - Insertion Sort
  - Bubble Sort
  - Merge Sort
- Practice

Merge Sort Idea

1. Split list in half
2. Sort each half
3. Merge the two halves

Merge Sort Idea

1. Split list in half
2. Sort each half
   - for each half

3. Merge the two halves

Merge Sort Idea

1. Split list in half
2. Sort each half
   - for each half
     1. Split half in half (into quarters)
     2. Sort each quarter
     3. Merge the two quarters
3. Merge the two halves

Merge Sort Idea

1. Split list in half
2. Sort each half
   - for each half
     1. Split half in half (into quarters)
     2. Sort each quarter
     3. Merge the two quarters
3. Merge the two halves

Merge Sort Idea

1. Split list in half
2. Sort each half
   - for each half
     1. Split half in half (into quarters)
     2. Sort each quarter
     3. Merge the two quarters
3. Merge the two halves

Merge Sort Idea

1. Split list in half
2. Sort each half
   - for each half
     1. Split half in half (into quarters)
     2. Sort each quarter
     3. Merge the two quarters
3. Merge the two halves

Merge Sort Idea

1. Split list in half
2. Sort each half
   - for each half
     1. Split half in half (into quarters)
     2. Sort each quarter
     3. Merge the two quarters
3. Merge the two halves

- Defined in terms of itself →Recursion!

Sorting Complexities

Algorithm	Worst Case	Best Case	Average Case
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n^2)$	$O(n)$	$O(n^2)$
Bubble Sort	$O(n^2)$	$O(n)$	$O(n^2)$
Merge Sort	???	???	???

On Tap For Today

- MinMax
- Sorting
  - Selection Sort
  - Insertion Sort
  - Bubble Sort
  - Merge Sort
- Practice

To Do For Next Time

- Rest of semester
  - M 11/24: Recursion + Merge Sort, L6A due
  - T 11/25, A5 due

  - M 12/01: Linear & Binary Search
  - W 12/03: 2D Lists + BFS/DFS
  - F 12/05: Stack & Queue

  - M 12/08: Trees & Graphs, L6B due, Quiz 6
  - W 12/10: Exam Review, L6C due, Exam XC due
  - R 12/11: A6, AXC, Final Project due

  - M 12/15 8am - 10am: Final Exam