UMBC

AN HONORS UNIVERSITY IN MARYLAND

CMSC201 Computer Science I for Majors

Lecture 24 – Sorting

Prof. Katherine Gibson

Prof. Jeremy Dixon

Last Class We Covered

- Searching
  - Linear search
  - Binary search
- Asymptotic Performance
  - How fast an algorithm “runs”
  - Why certain algorithms are “better” than others

Any Questions from Last Time?

Today’s Objectives

- To learn about sorting algorithms
  - Selection Sort
  - Bubble Sort
  - Quick Sort
  - Radix Sort
- To examine which of these algorithms is best for different sorting situations
  - How quickly do they scale?

Sorting

Sorting Algorithms

- Sorting algorithms put the elements of a list in a specific order

- A sorted list is necessary to be able to use certain other algorithms
- Like binary search!
  – If sorted once, we can search many, many times

Sorting Algorithms

- There are many different ways to sort a list
- What method would you use?
- Now imagine you can only look at ***at most*** two elements at a time

- Computer science has a number of commonly used sorting algorithms

Selection Sort

Selection Sort Algorithm

- Here is a simple way of sorting a list:

1. Find the smallest number in a list
2. Move that to the end of a new list
3. Repeat until the original list is empty

Selection Sort Run Time

- What is the Big Oh of finding the lowest number in a list?

- For a list of size **`N`**, what is the <u>worst case</u> number of elements you'd have to look through to find the min?
- **`N`**

Selection Sort Run Time

- For a list of size **`N`**, how many times would we have to find the min to sort the list?
- **`N`**

- What is the Big Oh of this sorting algorithm?
- **`O(N`^{2}`)`**

Bubble Sort

Bubble Sort Algorithm

- Let’s take a look at another sorting method!

1. We look at the first pair of items in the list, and if the first one is bigger than the second one, we swap them
2. Then we look at the second and third one and put them in order, and so on
3. Once we hit the end of the list, we start over at the beginning
4. Repeat until the list is sorted!

Bubble Sort Example

```
[ 4, 8, 1, 10, 13, 14, 6]
```

<u>First pass:</u>
4 and 8 are in order
8 and 1 should be swapped:

```
[ 4, 1, 8, 10, 13, 14, 6]
```

8 and 10 are in order
10 and 13 are in order
13 and 14 are in order
6 and 14 should be swapped:

```
[ 4, 1, 8, 10, 13, 6, 14]
```

Bubble Sort Example (Cont)

```
[ 4, 1, 8, 10, 13, 6, 14]
```

<u>Second pass:</u>

4 and 1 should be swapped:

```
[ 1, 4, 8, 10, 13, 6, 14]
```

4 and 8 are in order
8 and 10 are in order
10 and 13 are in order
13 and 6 should be swapped:

```
[ 1, 4, 8, 10, 6, 13, 14]
```

13 and 14 are in order

Bubble Sort Example (Cont)

```
[ 1, 4, 8, 10, 6, 13, 14]
```

Third pass:

10 and 6 should be swapped:

```
[ 1, 4, 8, 6, 10, 13, 14]
```

Fourth pass:

8 and 6 should be swapped:

```
[ 1, 4, 6, 8, 10, 13, 14]
```

Bubble Sort Run Time

- For a list of size **N**, how much work do we do for a single pass?
  - **N**
- How many passes will we have to do?
  - **N**
- What is the Big Oh of Bubble Sort?
  - O(**N**^{2})

Quicksort

Quicksort Algorithm

- Here's another method:

1. Start with the number on the far right
2. Put everything less than that number on the left of it and everything greater than it on the right of it
3. Quicksort the left side and the right side

- Does this method remind you of anything?

Quicksort Run Time

- For a list of size **`N`**, how many steps does it take to move everything less than the last number to the left and everything greater than the last number to the right?

- **`N`**

Quicksort Run Time

- How many times will the algorithm divide the list in half?
- **`lg(N)`**

- What is the Big Oh of Quick Sort?
- **`O(N lg(N))`**

Radix Sort

Improving Run Time

- Most of the time, **`O(Nlg(N))`** is the best we can do for sorting
- However if we make the problem slightly easier, we can do even better!

- Imagine we know for a fact that the list we are sorting is <u>only</u> integers between 0 and 9

Radix Sort Algorithm

- We can make a list of size 10 filled with zeroes
- The first element of this list represents the number of zeroes we’ve seen so far in the list we’re sorting
- The second number is the number of ones we’ve seen, and so on

Radix Sort Algorithm

- So say we have the list:
  - `[0, 3, 2, 1, 6, 8, 4, 4, 7, 7, 7]`
- We make our counting list:
  - `[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`
- And iterate over the list we want to sort

Radix Sort Algorithm

- The first number is a zero, so we add one to the zeroth element of our counting list:
  - `[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]`

- The next number is a 3, so we add one to the third element of our counting list:
  - `[1, 0, 0, 1, 0, 0, 0, 0, 0, 0]`

Radix Sort Algorithm

- Then 2:
  - `[1, 0, 1, 1, 0, 0, 0, 0, 0, 0]`
- Then 1:
  - `[1, 1, 0, 1, 0, 0, 0, 0, 0, 0]`

- When we're done, the list looks like this:
  - `[1, 1, 1, 1, 2, 0, 1, 3, 1, 0]`

Radix Sort Algorithm

- For an index **`i`**, we know that if **`countList[i] == 1`**, there was one **`i`** in the original list

- One pass over the counting list to figure out which numbers were there and we've sorted it!

Radix Sort Run Time

- How many operations do we need to do to fill out our counting list with zeros?
  - `10`
- How many operations do we need to do to fill out our counting list with the right values?
  - **`N`**

Radix Sort Run Time

- How many operations do we need to do to reconstruct our sorted list?
  - **`N`**
- This gives us a total run time of **`2N + 10`** operations
  - So our final run time is simply
  - **`O(N)`**

Any Other Questions?

General Announcements

- Survey #2 is out – due Monday, May 9th

- Project 2 is out
  - Due by Monday, May 9th at 8:59:59 PM
  - Do NOT procrastinate!

- Next Class: Review for the Final

Announcements: Final Exam

- Final Exam will held be on Friday, **May 13th from 6 to 8 PM**
- Being held in three separate rooms
    - ENGR 027 – Sections 2, 5, 11, 12, 15, 22, and 23
    - ITE 102 – Sections 4, 14, and 21
    - ITE 104 – Sections 3, 9, 10, 16, 18, and 20
- Make sure you go to the correct room!

Final Exam Room Assignments

- Room assignments by TA…

- ENGR 027
  - Alex, Grace, Joe, Rishi, Ted, Tyler
- ITE 102
  - Chris, Jeremy, Michael
- ITE 104
  - Hailee, Jacob, Katie, Liz, Patrick