COMPUTER SCIENCE

UNIT E WEEK 7, TUESDAY FEB 27TH

WWW.CITA.UTORONTO.CA/~WOODFORD

WOODFORD@CITA.UTORONTO.CA

THIS WEEK IN CS AND STEM

- Filtering water efficiently with graphene
  - https://futurism.com/film-graphene-filtering-dirty-water-easier-ever/
- A blood test for… concussions?
  - https://futurism.com/doctors-blood-detect-concussions-faster/
- Bacteria Bots
  - https://futurism.com/nanobots-kill-tumors-blood-supply/
- "Unless they figure out how to open doors" – Part II
  - https://futurism.com/boston-dynamics-spotmini-open-doors/

ASSIGNMENT 13

- …will be assigned next week (March 6th)
- Will focus on searching and sorting material from Unit E
- Due date: March 18th

FINAL PROJECT: TOPIC LAUNCH WILL BE MID-MARCH

SORTING

- We already know of some of the built-in sorting functions – how do they work?
  - How do they sort numbers? (ints and floats)
  - How do they sort strings?
  - How do they sort Booleans?
  - How do they sort 2D lists and arrays of ints, floats, bools?
  - Do lists and arrays behave differently?
  - How do they sort mixed lists? (ie. More than one variable type)

SORTING METHODS

- We actually heard about some of the situations and methods in the documentary “Algorithms” that we watched a couple of weeks ago.
- There are two main types of sorting algorithms:
  - Comparison sorts: compare each element to see if it belongs to the “left” or “right” of another element
  - Integer (or counting) sorts: for each element, count how many elements are “beneath” it
- What could be some benefits and drawbacks of each?

POPULAR SORTING METHODS

- Bubble sort: Start at the beginning of the data set. If the current element is larger than the one to the right of it, then swap them. Keep moving through.
- Insertion sort: Insert elements one at a time into a new array so that they're sorted. Ie. Move elements one at a time into the correct position.
- Selection sort: Find the smallest element, put in the first position. Find next smallest, put in second position, and so on.
- Merge sort: Split into smaller groups – halve until each group has 1 or no elements. Sort split groups, and then merge neighbours together

EFFICIENCY OF SORTING METHODS

- Remember our discussion of the Order of a function, or how many operations it requires to complete?
  - Ex. Linear search was O(n), Binary search was O(logn)
- What do you think the orders are for bubble, insertion, selection, and merge sort? Why?

EFFICIENCY OF SORTING METHODS

- Remember our discussion of the Order of a function, or how many operations it requires to complete?
  - Ex. Linear search was O(n), Binary search was O(logn)
- What do you think the orders are for bubble, insertion, selection, and merge sort? Why?
  - Comparison-based sorting algorithms have a time complexity of O(nlogn). For example, if an algorithm had a worst-case running time of O(nlogn), then it is guaranteed that the algorithm will never be slower than O(nlogn), and if an algorithm has an average-case running time of $O(n^2)$, then on average, it will not be slower than $O(n^2)$. Counting-based sorting algorithms have a time complexity of $O(n^2)$.

WHAT ELSE?

- We've discussed types of sorting algorithms, done some examples, and looked at time complexities. What else might we need to consider when choosing a sorting algorithm to use?

WHAT ELSE?

- We've discussed types of sorting algorithms, done some examples, and looked at time complexities. What else might we need to consider when choosing a sorting algorithm to use?

- Stability: A sorting algorithm is stable if it preserves the original order of elements with equal key values (where the key is the value the algorithm sorts by). Ie. You keep the original order of equal/identical elements.
  - Why is this desirable? Is it always important to consider?

MORE COMPLEX, BUT MORE EFFICIENT

- Quick Sort: Choose a pivot element, sort into 2 arrays where one is all the elements less than the pivot and the other is all elements greater than the pivot. Repeat by choosing new pivots.
- Heap Sort: Using a graph structure – divides input into sorted and unsorted regions, shrink sorted region by extracting the largest element and moving it to the sorted region.
- Counting Sort: for each element, determine the number of elements less than it. Then place the current element into that number slot. Repeat.

REFERENCES

- https://brilliant.org/wiki/sorting-algorithms/
- http://python3.codes/popular-sorting-algorithms/