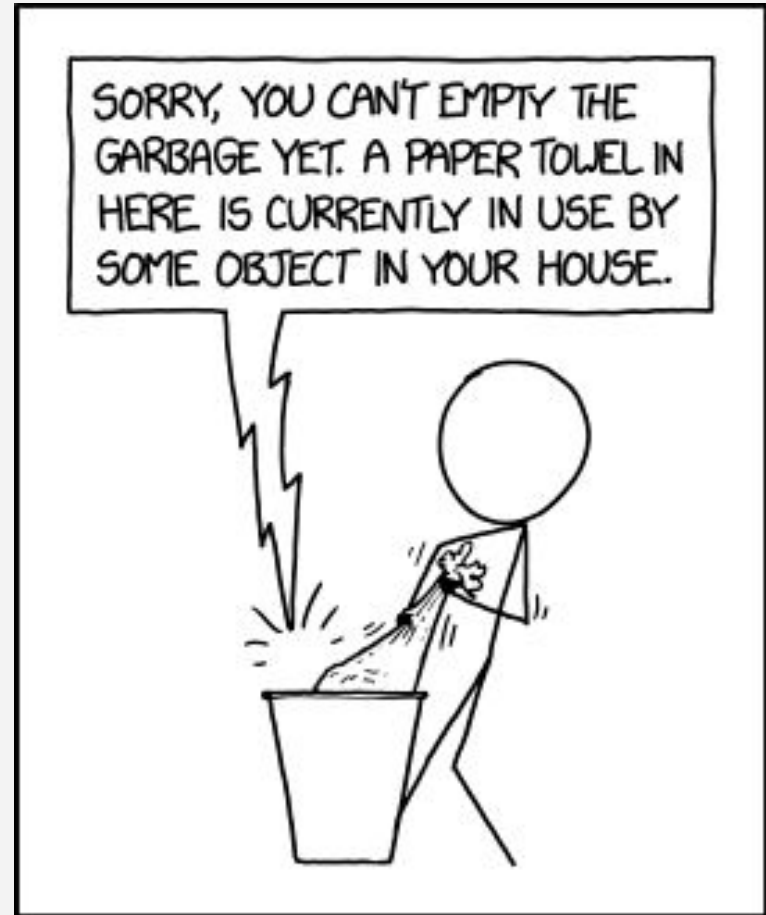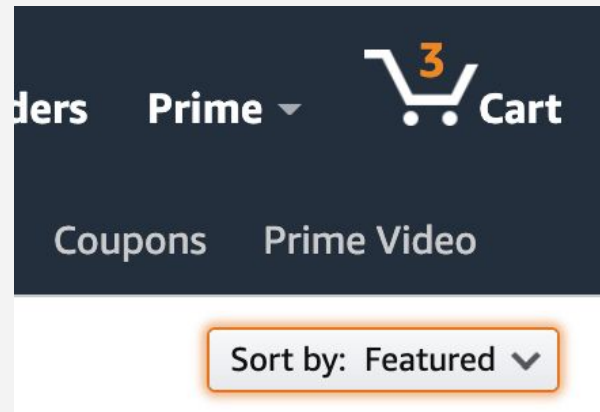# CSc 110

Sort

Benjamin Dicken

Sorting

A number of reasons to want sorted data

- ○ For doing binary search!
- ○ Finding a median value
- ○ Finding the min and max
- ○ Search results
- ○ Others?

Sorting

lists have built-in functionality to rearrange the elements to be in sorted order

`list_var.`**`sort`****`()`**   or   **`sorted`****`(list_var)`**

But, someone at some point had to come up with an **algorithm** for this! How does sorting work, behind the scenes?

What is a sorted list?

A **sorting algorithm** is an algorithm that puts elements of a list in a certain order

- However, there are different types of "ordering"
  - Ascending numeric order   (numbers)
  - Descending numeric order   (numbers)
  - Lexicographic   (strings)
  - Others…
- We will mostly stick with Ascending numeric for the examples in this lecture

What is a sorted list?

`items = [5, 10, 20, 6, 7, 9, 43, 10, 12]`

index	0	1	2	3	4	5	6	7	8
value	5	10	20	6	7	9	43	10	12

Not Sorted

`items = [5, 6, 7, 9, 10, 10, 12, 20, 43]`

index	0	1	2	3	4	5	6	7	8
value	5	6	7	9	10	10	12	20	43

Sorted (ascending)

Come up with a sorting algorithm

- Discuss ideas for how to sort numbers
- Depict your algorithm with drawings/diagrams or with pseudocode
- can't use the `.sort()` function

In-place and Out-of-place

In-place sorting: does not require a secondary data structure

Out-of-place sorting: may require a secondary data structure

Selection Sort

Selection Sort is a very simple sorting algorithm

- Scan the list and find the smallest element
- Swap this element with the beginning element
- Continue these steps for the remaining list, discluding the element just swapped
- Repeat

Selection Sort

Visualizing sorting algorithms with graphics can give one a better understanding

https://visualgo.net/en/sorting

Selection Sort

```python
def selection_sort(items):
    begin = 0
    for i in range(len(items)-1):
        small_i = begin
        for j in range(begin, len(items)):
            if items[small_i] > items[j]:
                small_i = j
        items[begin], items[small_i] = items[small_i], items[begin]
        begin += 1
```

How many total sweeps and swaps to sort this list?

index	0	1	2	3	4
value	3	1	7	2	4

How many total swaps to sort this list?

index	0	1	2	3	4	5	6
value	3	1	7	2	4	8	5

Bubble Sort

Bubble Sort: another sorting algorithm

- ○ Scan through each element in the list, comparing the current element with the next one
- ○ If the next one is smaller, swap the elements
- ○ Continue these iterations until the whole list is sorted

- This causes the large elements to "bubble up" to the top

Bubble Sort

```python
def bubble_sort(items):
    end = len(items)
    for i in range(len(items)-1):
        for j in range(0, end-1):
            if items[j] > items[j+1]:
                items[j], items[j+1] = items[j+1], items[j]
        end -= 1
```

How many sweeps and swaps until it is sorted?

index	0	1	2	3	4
value	3	1	7	2	4

How many sweeps and swaps until it is sorted?

index	0	1	2	3	4	5	6
value	7	1	5	2	4	3	8

Announcements

- Final Exam:
  - Monday Dec 12th 1-3pm
  - Review Session: 5pm on Thursday Dec 8th
    - Room TBA
  - Exam is cumulative
  - Many resources: Study guides, book, slides, preps, etc

Insertion Sort

Insertion Sort: another sorting algorithm

Let's go to the visualization tool

Insertion Sort

```python
def insertion_sort(items):
    for compare_index in range(1, len(items)):
        ci = compare_index
        for j in range(ci-1, -1, -1):
            if ci < 0 or items[ci] >= items[j]:
                break
            else:
                items[ci], items[j] = items[j], items[ci]
                ci -= 1
```

How many TOTAL scans/sweeps to sort?

index	0	1	2	3	4
value	3	1	7	2	4

How many TOTAL swaps to sort?

index	0	1	2	3	4	5	6
value	3	1	7	2	4	8	5

Lots of algorithms

- There are many sorting algorithms
    - **Bogo sort**
    - **Selection sort**
    - **Bubble sort**
    - **Insertion sort**
    - **Merge sort**
    - **Quick sort**
    - …more…

Timing and Vis

- sort_timing.py
- https://www.youtube.com/watch?v=kPRA0W1kECg