CSCI 204: Data Structures & Algorithms

Revised by Xiannong Meng based on textbook author's notes

Sorting

- The process of arranging a collection of items such that each item and its successor satisfy a preferred order.
  - **sequence sort** – sorting within a sequence such as a list or an array.
  - **sort key** – values on which items are ordered such as priority in a priority queue.
  - items arranged in ascending or descending order.
  - sorted in place – within the same structure.

Compared to the Priority Queues We Learned

- We've learned how to maintain a priority queue, inserting items into a queue based on their priorities.
- Difference and similarity between a priority queue and sorting a list in order
  - Once an item is inserted into a priority queue, the queue as a whole is "sorted."
    - A priority queue may be in an un-sorted order, like the one in the textbook, removing the top-priority item involves a search
  - The action of sorting re-arranges a list originally un-ordered into an ordered sequence.

There are many different sorting and searching algorithms. The famous sequence of books by Knuth "The Art of Programming" dedicates an entire book of **800 pages** to just sorting and searching!

https://en.wikipedia.org/wiki/The_Art_of_Computer_Programming

We will examine a few here in CSCI 204.

Bubble Sort

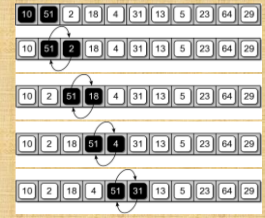
- A simple solution to the sorting problem.
- Arranges the items by
  - iterating over the sequence multiple times.
  - smaller values bubble to the top (or large values "sink" to the bottom).

Bubble Sort Code ("sink")

```
def bubble_sort( the_seq ):
  n = len( the_seq )
  for i in range( n - 1, 0, -1 ) :
    for j in range( i ) :
      if the_seq[j] > the_seq[j + 1] :
      # swap the two items
        tmp = the_seq[j]
        the_seq[j] = the_seq[j + 1]
        the_seq[j + 1] = tmp
```

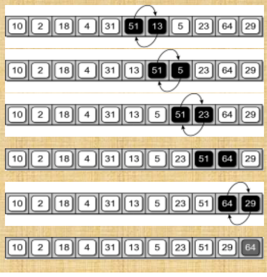
Bubble Sort Example

- First complete iteration of the inner loop.



7

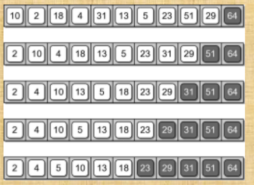
Bubble Sort Example



8

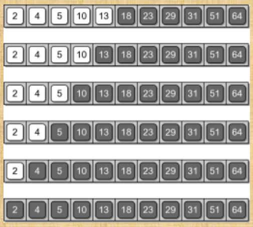
Bubble Sort Example

- Results after each iteration of the outer loop.



9

Bubble Sort Example



10

Two Issues to Consider

- The above program works correctly. But we can think about the following two issues.
  - The algorithm “sinks” the largest value to the end of the list. How can we re-write it so that the smallest value “bubbles” to the beginning?
  - Do we really need to run the outer loop to its full count? How can we stop earlier? E.g., when the sequence becomes 2,4,5,10,13,18,**23,29,31,51,64** in the 5th round, we could’ve stopped!

Selection Sort

- Improves on the bubble sort.
- Works in a fashion similar to what a human may use to sort a sequence.
- Instead of swapping many items,
  - repeatedly selects the next largest (or the smallest) item from among the unsorted items, puts it in the right place.
  - requires a search to select the smallest item in each round.

12

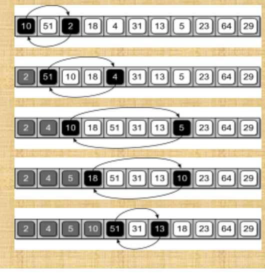
Selection Sort Code

```
def selection_sort( the_seq ):
  n = len( the_seq )
  for i in range( n - 1 ):
    small_index = i
    for j in range( i + 1, n ):
      if the_seq[j] < the_seq[small_index] :
        small_index = j

    if small_index != i :  # found a new small
       tmp = the_seq[i]
       the_seq[i] = the_seq[small_index]
       the_seq[small_index] = tmp
```

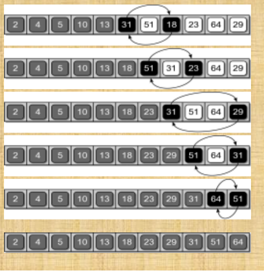
13

Selection Sort Example



14

Selection Sort Example



15