CSCI 204: Data Structures & Algorithms

Revised by Xiannong Meng based on textbook author's notes

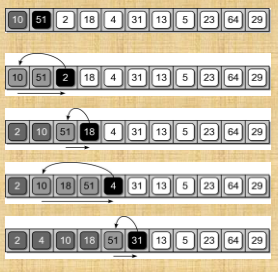
Insertion Sort

- Another commonly studied algorithm.
- Arranges the items by
  - iterating over the sequence one complete time.
  - inserts each unsorted item into its proper place.

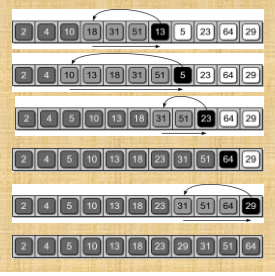
Insertion Sort Code

```
def insertionSort( theSeq ):
  n = len( theSeq )
  for i in range( 1, n ) :
    value = theSeq[i]

    pos = i
    while pos > 0 and value < theSeq[pos - 1] :
      theSeq[pos] = theSeq[pos - 1]
      pos -= 1

    theSeq[pos] = value
```


Insertion Sort Example



Insertion Sort Example

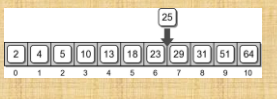


Working With Sorted Lists

- The efficiency of some algorithms can be improved when working with sorted sequences.
  - For non-static collections, it would be inefficient to re-sort a sequence for each add/remove.
  - Better to maintain a sorted sequence.

Maintaining a Sorted List

- To maintain a sorted list, new items must be inserted into their proper position.
  - Can not simply be appended at the end.
  - Must locate the proper position and use `insert()`.



7

Compare Different Sorting Algorithms

- So far, we have studied three different sorting algorithms
  - Bubble sort: in each round, bubble the smallest items to the top (or sink the largest item to the bottom)
  - Selection sort: in each round, select the correct position for the current item
  - Insertion sort: in each round, insert the current item in its correct location so the partial list is sorted

Complexity and Timing

- In the classroom activity, you will come up the big-Oh notation for each of the three algorithms
- You will also measure the timing of the three algorithms with data sets of different sizes