Algorithms in CLRS

Ken Harvey
CS Student at UNLV
github.com/kennethken73/DSA_CLRS
kennethken73@gmail.com

Ongoing Study of CLRS

Literate Programming via orgmode's live-code snippets

---

1 Insertion Sort

CLRS pp.18 describes Insertion Sort, with the following pseudo-code:

```
1   for j = 2 to A.length
2     key = A[j]
3     // Insert A[j] into the
4     // sorted sequence A[1..j-1],
5     i = j-1
6     while i > 0 and A[i] > key
7       A[i+1] = A[i]
8       i = i-1
9     A[i+1] = key
```

NOTES ON THE PRECEDING PSEUDO-CODE :

1. `1  for j = 2 to A.length`

   - Initially, the first item in the array comprises the sorted set, which at this point
     is only of length = 1, and therefore is sorted, by force.

   - The other items (the last item, the second item, and all items in between) are considered unsorted.

   - It is useful to state again, clearly, that this pseudo-code is 1-indexed (not 0-indexed).

2. ```
   2     key = A[j]
   3     // Insert A[j] into the
   4     // sorted sequence A[1..j-1],
   ```

   - On the first iteration, the second item is being placed into the sorted "hand"
     (like a "hand" of a deck of cards).

   - This item-being-sorted (namely: `j`) is
     - initially: 2, and
     - finally: `A.length`

     - So, if (in this 1-indexed array) the array has 5 elements, then
       the last element is at position 5.
       * Note, that in 0-indexed arrays, this last element will be at `A.length - 1`!
     - The sorted "Left-hand" set is comprised of elements A[1..j-1], which is:
       * initially: A[1..(2-1)] = A[1..1] = A[1], and
       * finally: A[1..(A.length-1)]
         · Note, that the final state of the algorithm takes the last item (namely: A[j]),
           and places it into the "left-hand" sorted-set, comprised now of elements A[1] through A[j-1].

3. ```
   5     i = j-1
   6     while i > 0 and A[i] > key
   7       A[i+1] = A[i]
   8       i = i-1
   9     A[i+1] = key
   ```

   - We set the index into the sorted "left-hand" sorted-set at the right-most element, and
     then iterate backwards.

   - On each backwards iteration (i–), elements greater than the key-value (A[j]) are moved rightward
     in the sorted-set (This is what line 7 accomplishes).

   - Line 9 then places the key item-being-placed-into-its-correct-position at this correct position (namely: A[i+1]).

   - Note that line 5 guarantees that there *is* some such i+1, as i is, at maximum (j-1) = (A.length - 1).