Insertion Sort

Adapted from the CLRS book slides

EXAMPLE: INSERTION-SORT

INSERTION-SORT(A, n)

```
for i = 2 to n
    key = A[i]
    // Insert A[i] into the sorted subarray A[1 : i − 1].
    j = i − 1
    while j > 0 and A[j] > key
        A[j + 1] = A[j]
        j = j − 1
    A[j + 1] = key
```

EXAMPLE: INSERTION-SORT

INSERTION-SORT(A, n)

```
for i = 2 to n                                        n − 1 iterations
    key = A[i]
    // Insert A[i] into the sorted subarray A[1 : i − 1].
    j = i − 1
    while j > 0 and A[j] > key                        at most i − 1 iterations
        A[j + 1] = A[j]
        j = j − 1
    A[j + 1] = key
```

Claim: INSERTION-SORT runs in $O(n^2)$ time

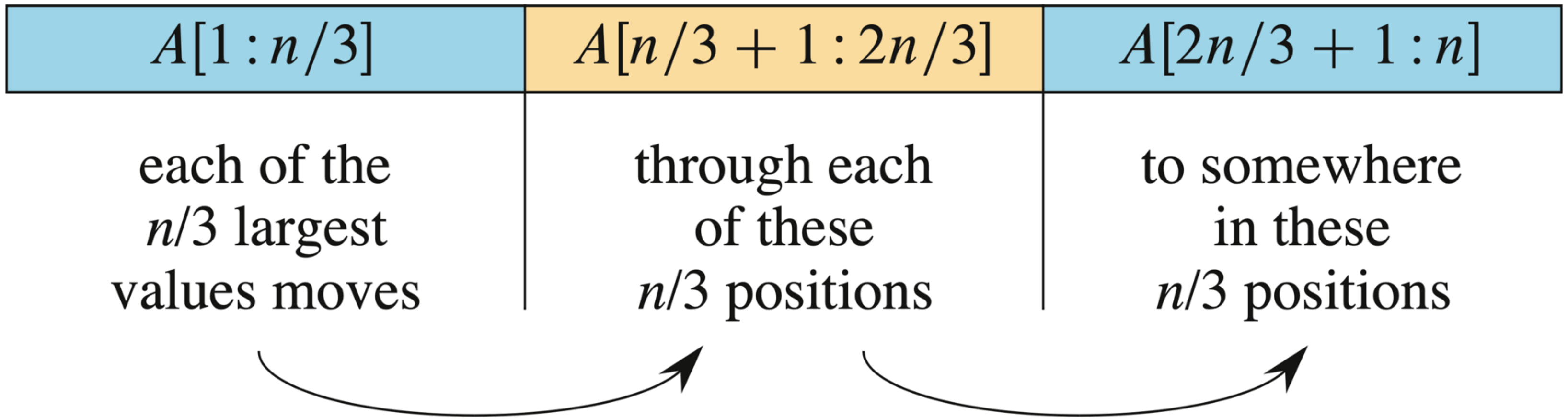
- The total number of iterations of the inner loop is at most $(n-1)(n-1)$, which is less than n^2.
- Each inner loop iteration takes constant time, for a total of at most cn^2 for some constant c, or $O(n^2)$.

EXAMPLE: INSERTION-SORT (continued)

Now show that INSERTION-SORT has a worst-case running time of $\Omega(n^2)$:

Observe that for a value to end up k positions to the right of where it started, the line $A[j+1] = A[j]$ must have been executed k times.

Assume that n is a multiple of 3 so that we can divide the array A into groups of $n/3$ positions.



EXAMPLE: INSERTION-SORT (continued)

Because at least $n/3$ values must pass through at least $n/3$ positions, the line $A[j+1] = A[j]$ executes at least $(n/3)(n/3) = n^2/9$ times, which is $\Omega(n^2)$. For this input, INSERTION-SORT takes time $\Omega(n^2)$.

Since we have shown that INSERTION-SORT runs in $O(n^2)$ time in all cases and that there is an input that makes it take $\Omega(n^2)$ time, we can conclude that the worst-case running time of INSERTION-SORT is $\Theta(n^2)$.

The constant factors for the upper and lower bounds may differ. That doesn't matter.