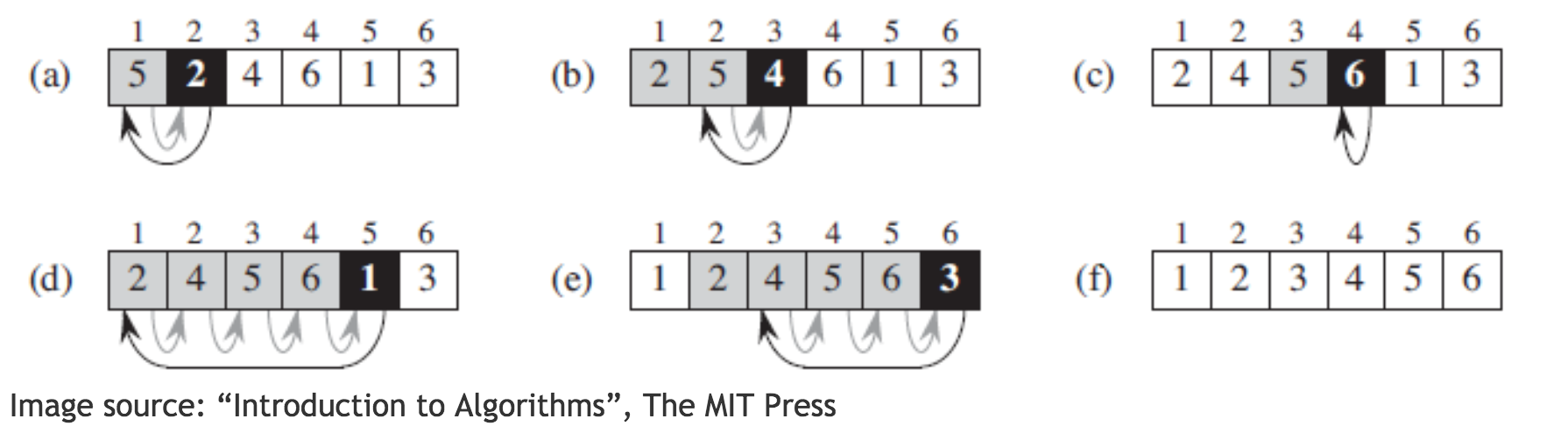
|  |
| --- |
|  |
| **Program: Implement insertion sort in java.**  Insertion sort is a simple sorting algorithm, it builds the final sorted array one item at a time. It is much less efficient on large lists than other sort algorithms.  Advantages of Insertion Sort:   1) It is very simple. 2) It is very efficient for small data sets. 3) It is stable; i.e., it does not change the relative order of elements with equal keys. 4) In-place; i.e., only requires a constant amount O(1) of additional memory space.  Insertion sort iterates through the list by consuming one input element at each repetition, and growing a sorted output list. On a repetition, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain. |



The best case input is an array that is already sorted. In this case insertion sort has a linear running time (i.e., Θ(n)). During each iteration, the first remaining element of the input is only compared with the right-most element of the sorted subsection of the array. The simplest worst case input is an array sorted in reverse order. The set of all worst case inputs consists of all arrays where each element is the smallest or second-smallest of the elements before it. In these cases every iteration of the inner loop will scan and shift the entire sorted subsection of the array before inserting the next element. This gives insertion sort a quadratic running time (i.e., O(n2)). The average case is also quadratic, which makes insertion sort impractical for sorting large arrays. However, insertion sort is one of the fastest algorithms for sorting very small arrays, even faster than quicksort; indeed, good quicksort implementations use insertion sort for arrays smaller than a certain threshold, also when arising as subproblems; the exact threshold must be determined experimentally and depends on the machine, but is commonly around ten.