Insertion Sort

ALGORITHM

Unplugged Activity:

Sorting a hand of playing cards

1. Start with an empty left hand and 10 cards are facing down on the table.

2. Remove one card at a time from the table, and insert it into the correct position in the left hand.

   - compare it with each of the cards already in the hand, from right to left

3. The cards held in the left hand are sorted

Unplugged Activity:

First card : 10

10

10

Second card : 6

Second card : 6

Swap to place them in order

Third card : 36

Since 36 is larger than 10, no swapping is required.

Forth card : 24

Swap to place them in order

Check whether further swapping is required

Fifth card : 12

Make multiple swap until they are placed in order

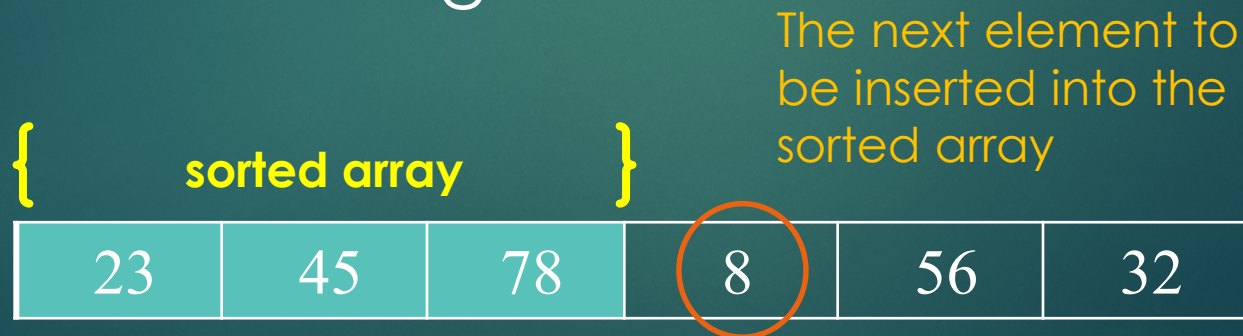Fifth card : 12

Make multiple swap until they are placed in order

We have a sorted set of cards when there is no more swapping.

6 10 12 24 36

Description :

Insertion Sort is a simple sorting algorithm that works the way we sort playing cards in our hands. We sort the first two cards and then place the third card in the appropriate position within the first two, and then the fourth is positioned within the first three, and so on until the whole hand is sorted.

During an iteration, an element of the list is inserted into the sorted portion of the array to its left. So, basically, for each iteration, we have **an array of sorted elements** to the left, and an array of other elements still to be sorted to the right.

The next element to be inserted into the sorted array

{ **sorted array** }

| 23 | 45 | 78 | 8 | 56 | 32 |

Basic skill #1:

For a list containing only 2 elements, swap the elements, if necessary, such that the smaller element will be shifted to the left.

Basic skill #2:

For a list with all the elements sorted in ascending order except the last element, comparing two elements at a time, and swapping when necessary, arrange the elements in an ascending order.

Insertion Sort:

For an unsorted list, perform Basic Skill #2 on a first sub-list of 2 elements, then on a sub-list of 3 elements … until the whole list is sorted in an ascending order.

In-Place Insertion Sort

This algorithm is efficient for smaller lists, but very inefficient for larger lists. Also, it does not require any additional storage space, as it operates in-place.

Order of Growth

Insertion Sort : **O(n^2)**

For an array of **n** unsorted items, to shift one element to the 'correct' position, the Insertion Sort algorithm needs to scan through the whole array of **(n-1)** terms, hence the Order of Growth is **O(n^2)**.