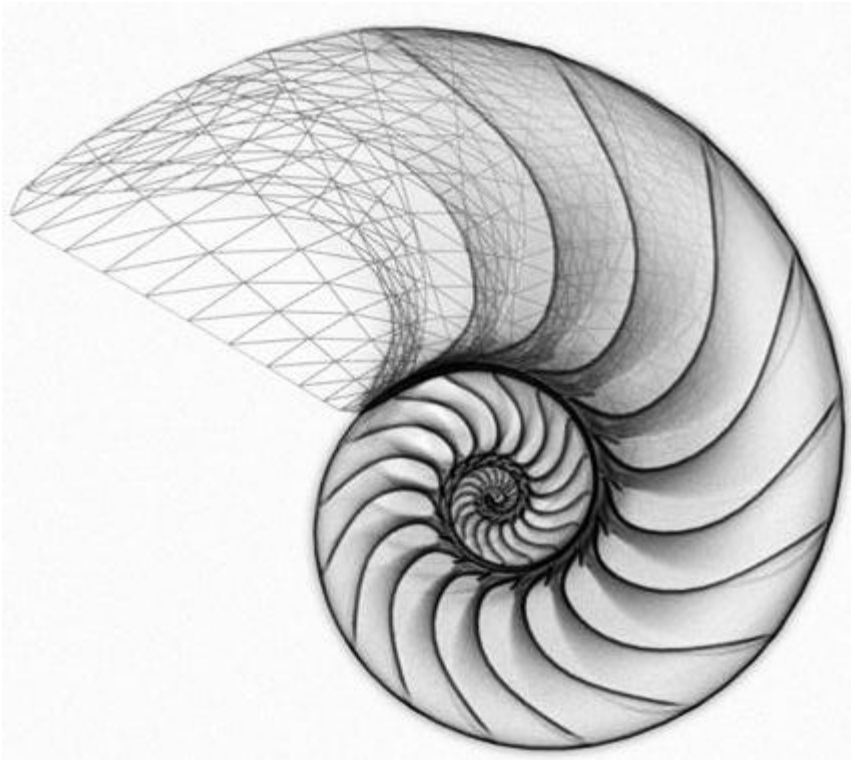


Lecture 09

Insertion Sort

- **Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands.**

- **The array is virtually split into a sorted and an unsorted part.**

- **Values from the unsorted part are picked and placed at the correct position in the sorted part.**

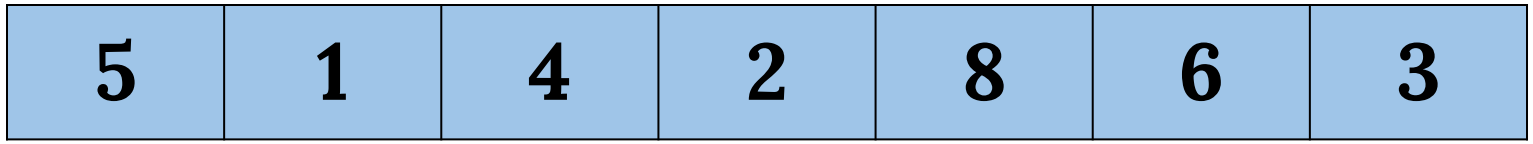
0 1 2 3 4

Insertion Sort

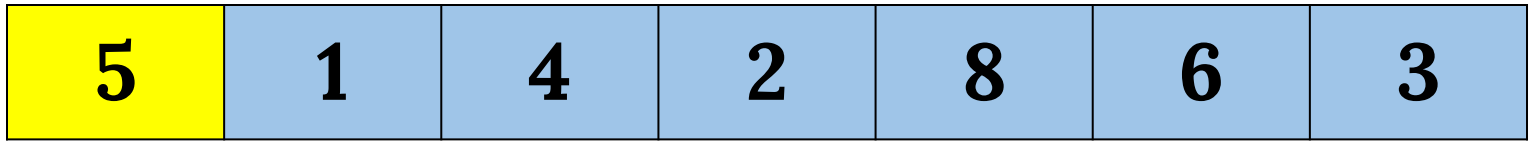
Characteristics of Insertion Sort:

- **This algorithm is one of the simplest algorithm with simple implementation**
- **Basically, Insertion sort is efficient for small data values**
- **Insertion sort is adaptive in nature, i.e. it is appropriate for data sets which are already partially sorted**

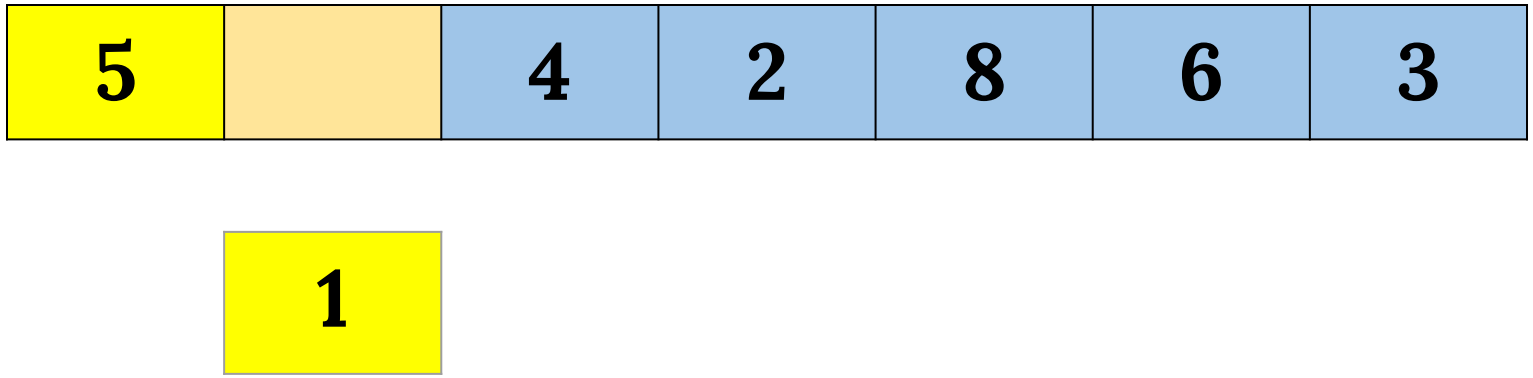
Insertion Sort



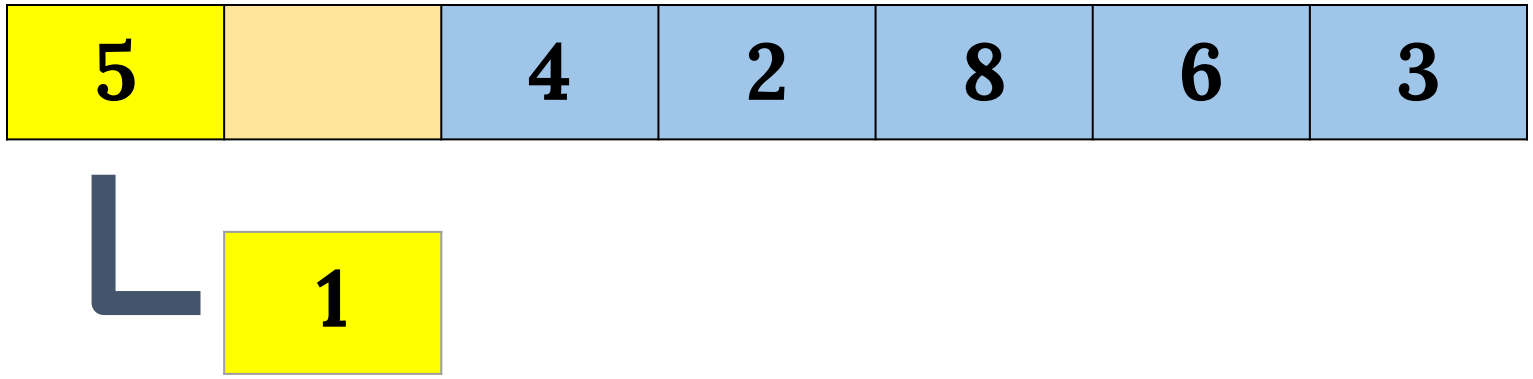
Insertion Sort



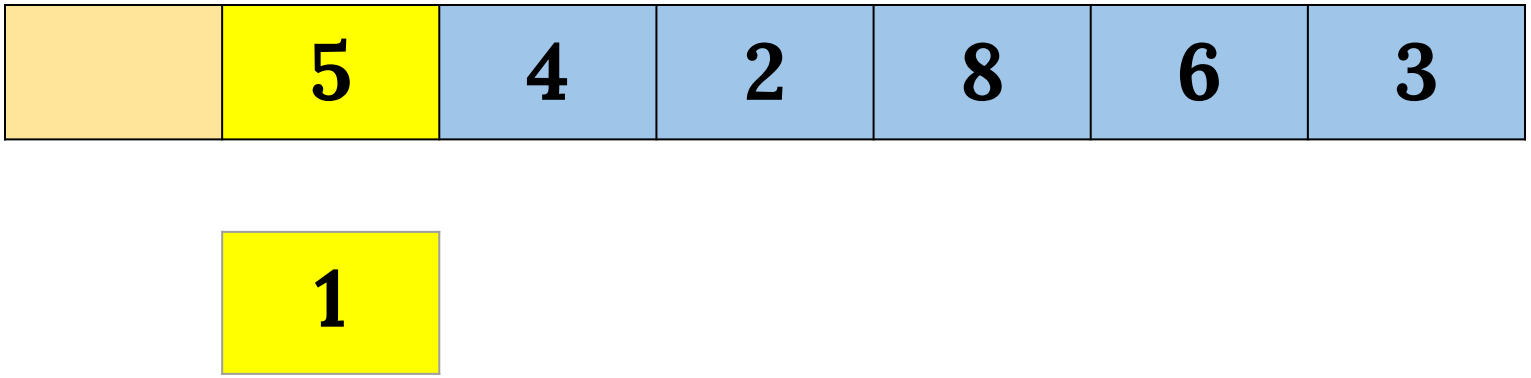
Insertion Sort



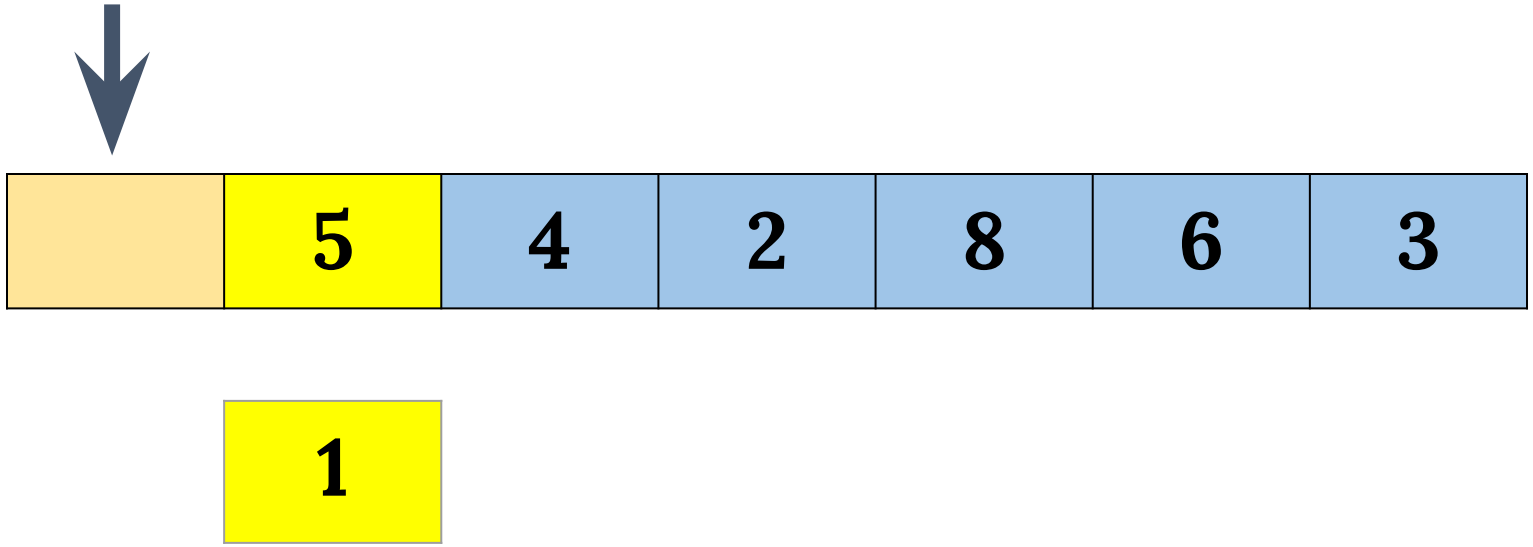
Insertion Sort



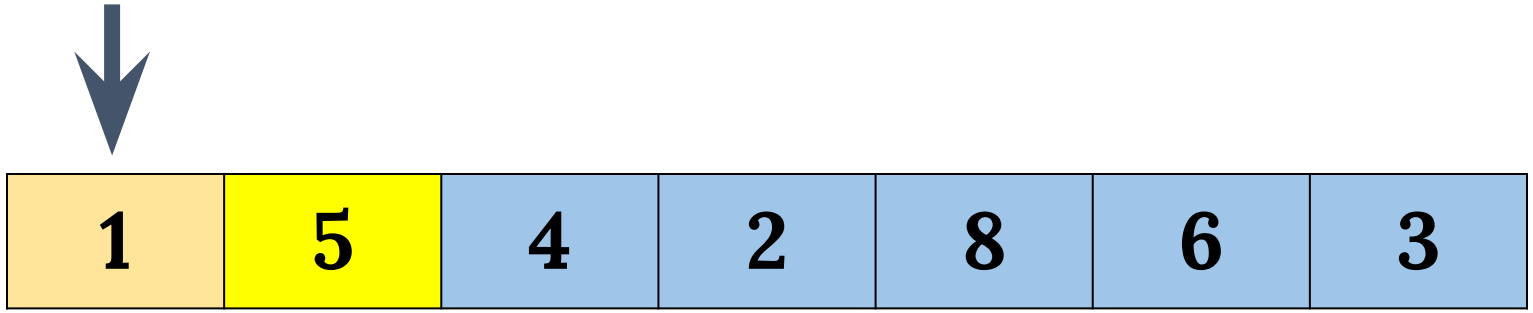
Insertion Sort



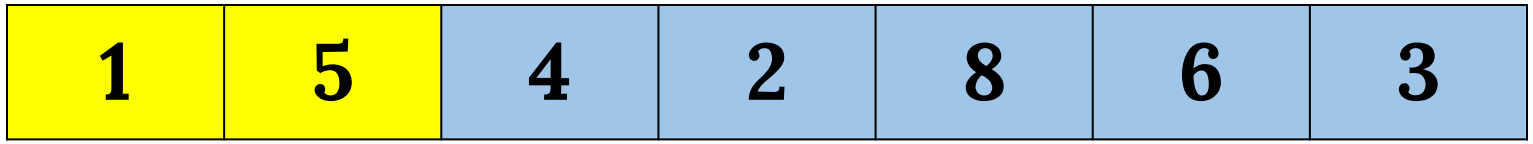
Insertion Sort



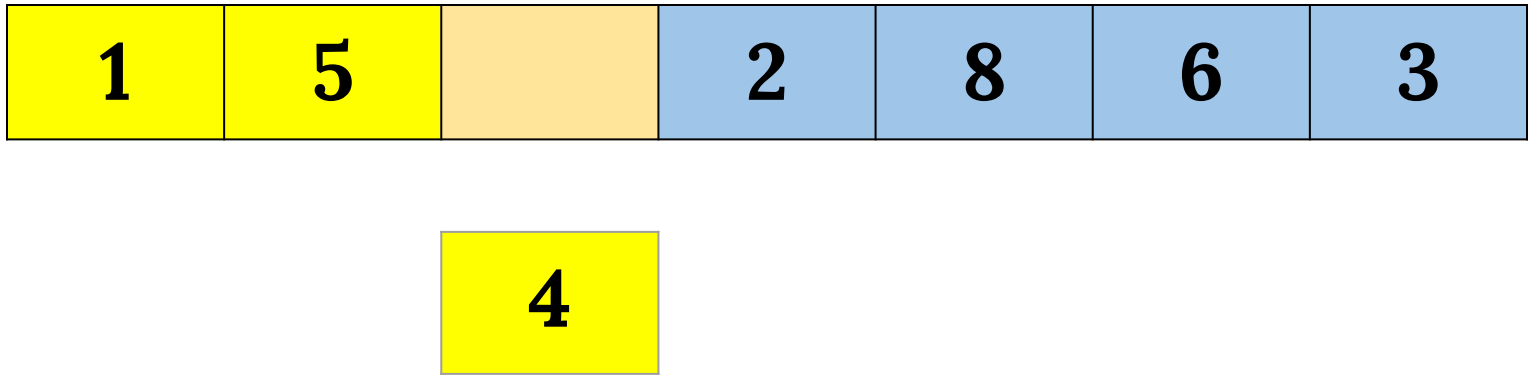
Insertion Sort



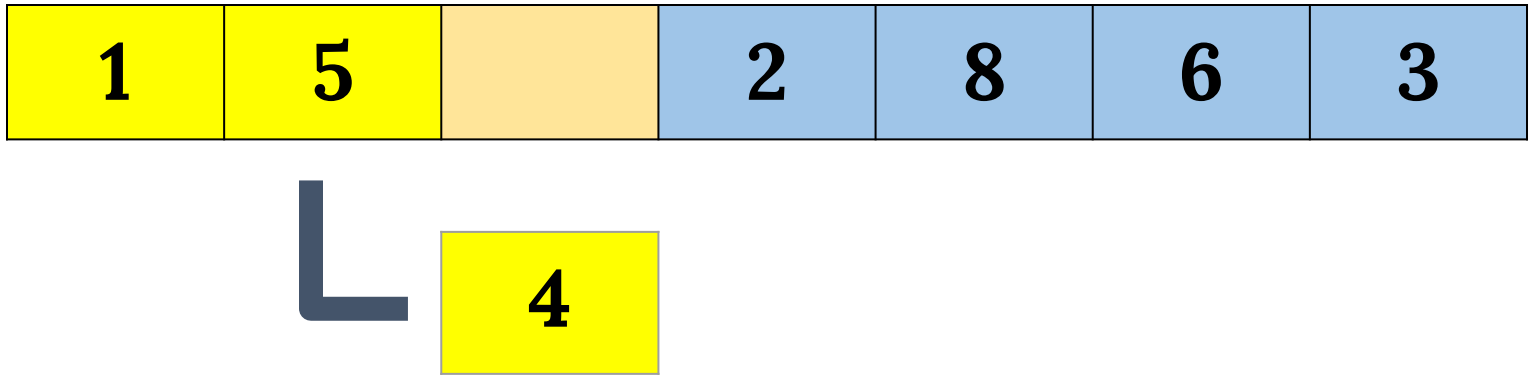
Insertion Sort



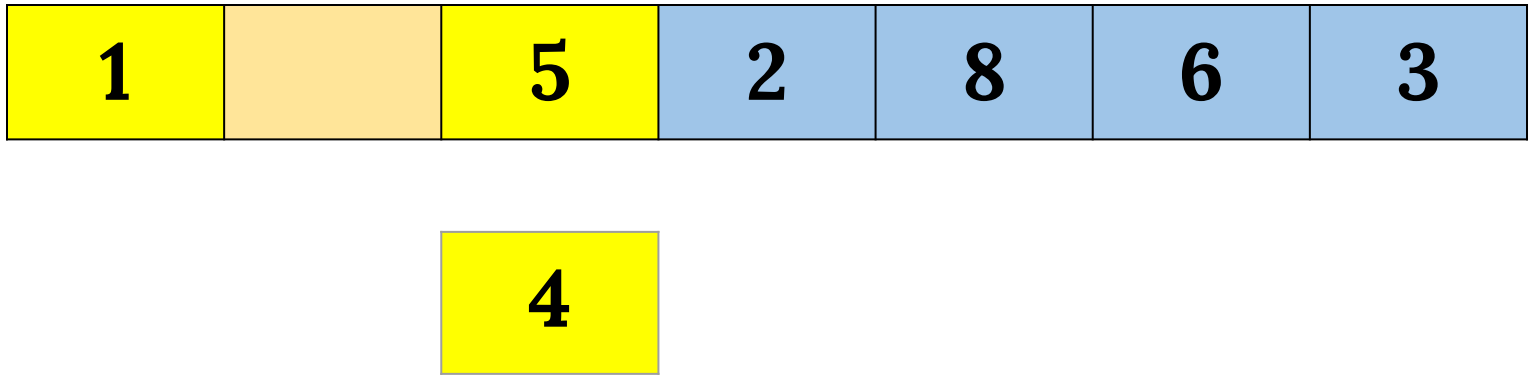
Insertion Sort



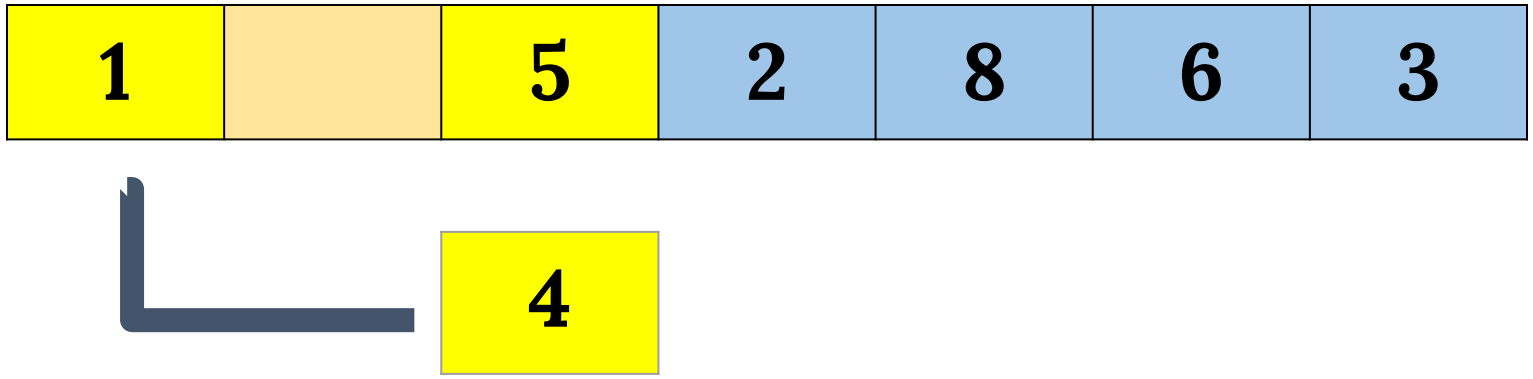
Insertion Sort



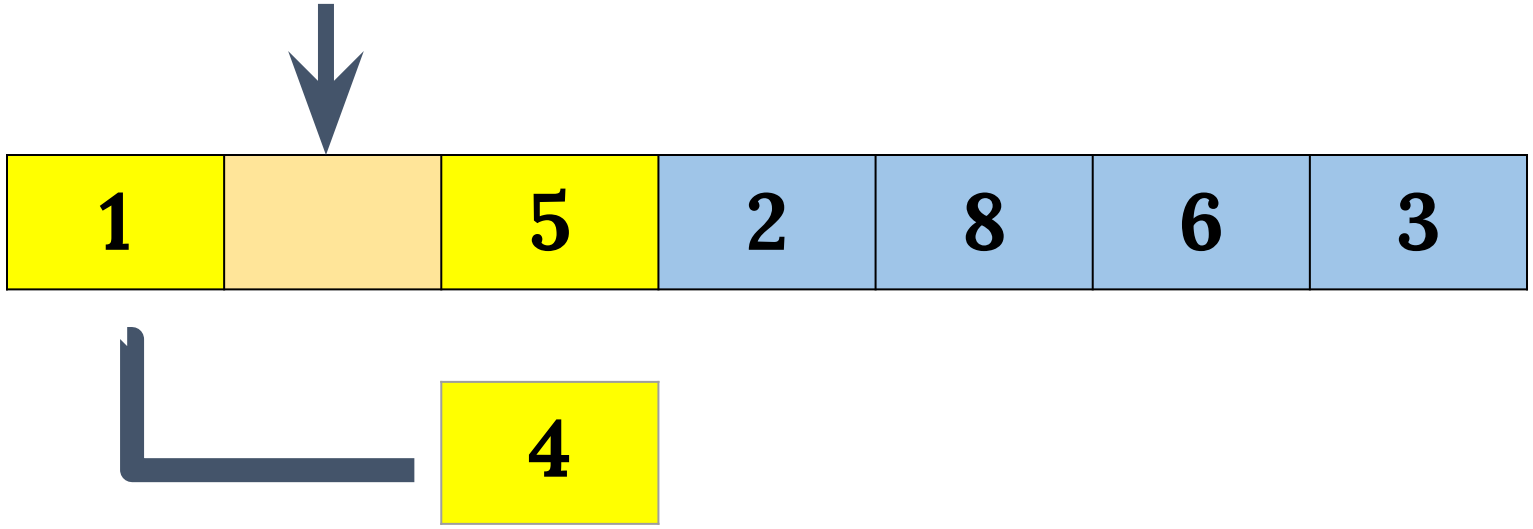
Insertion Sort



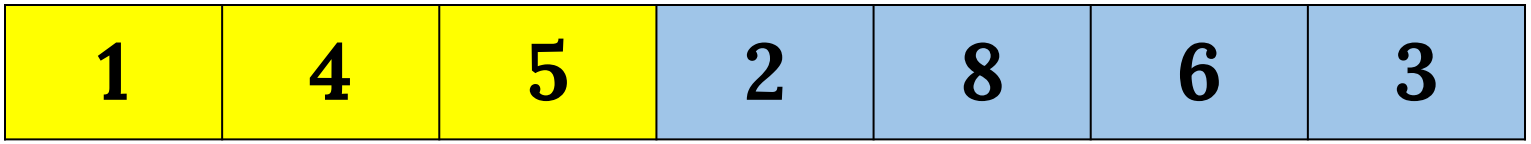
Insertion Sort



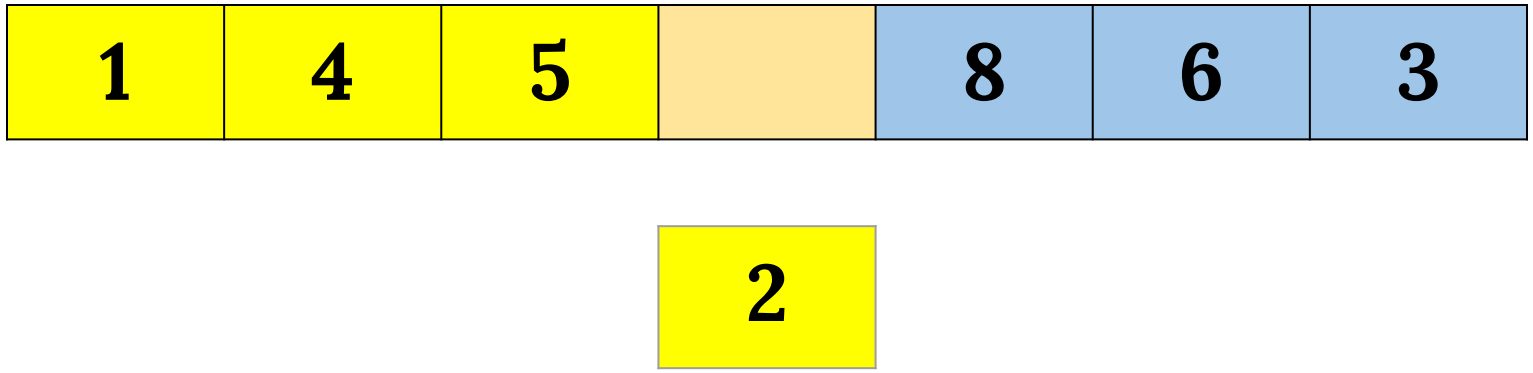
Insertion Sort



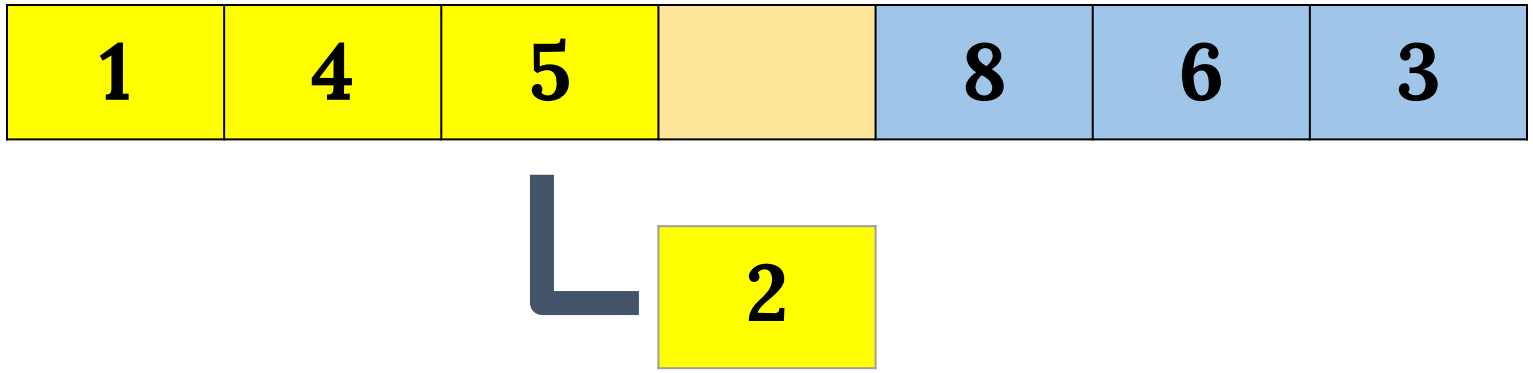
Insertion Sort



Insertion Sort



Insertion Sort



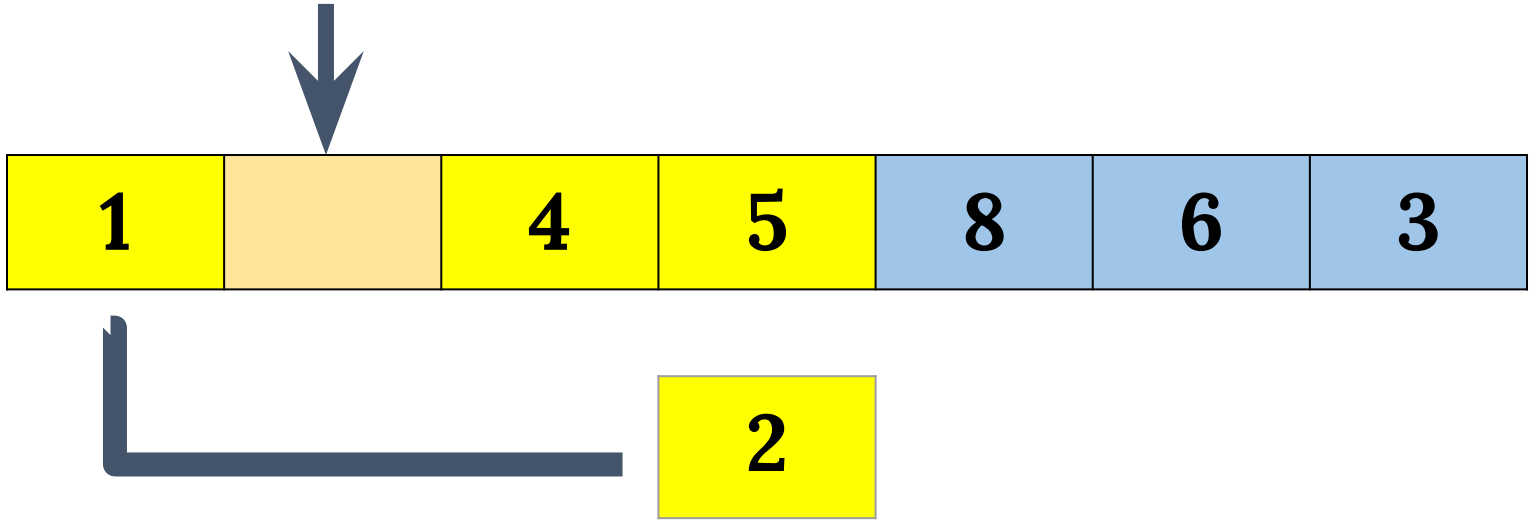
Insertion Sort

Insertion Sort

Insertion Sort

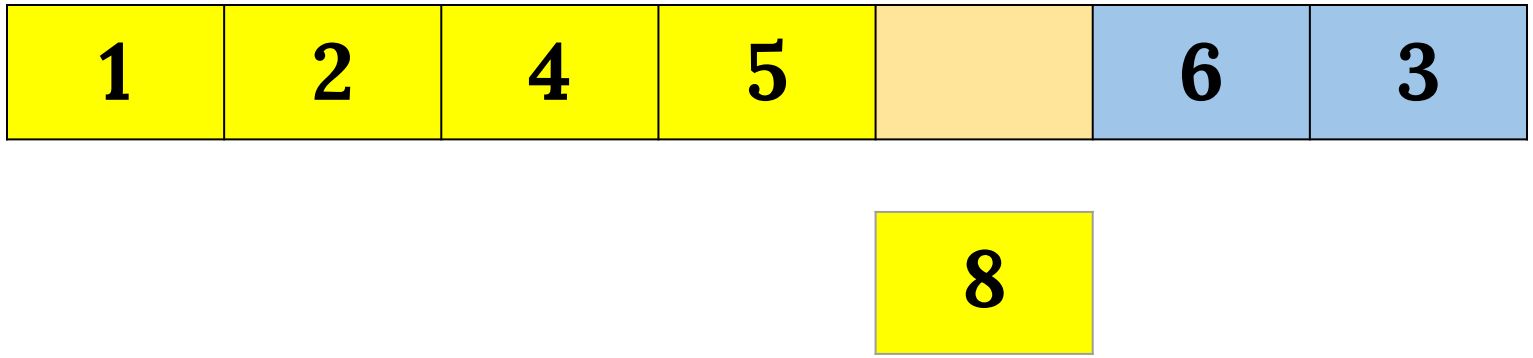
Insertion Sort

Insertion Sort

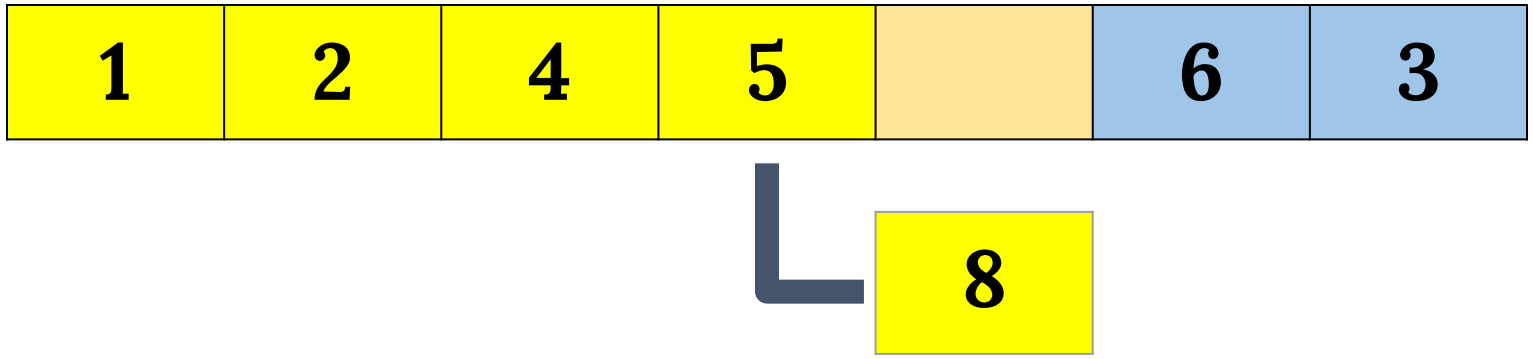


Insertion Sort

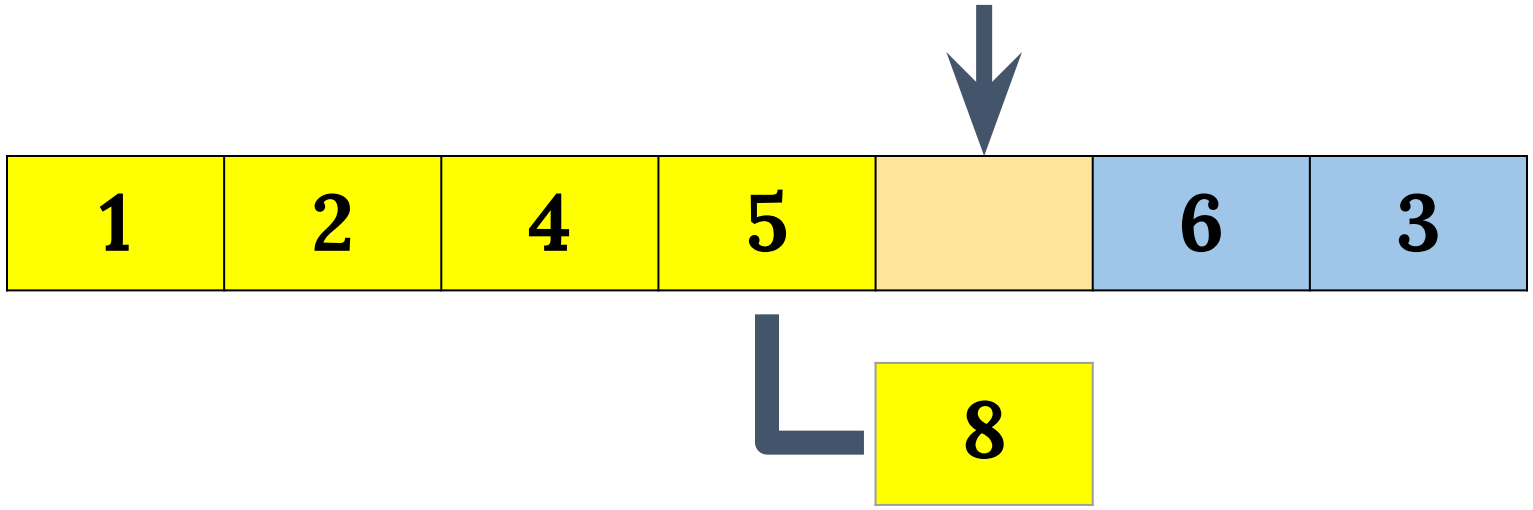
Insertion Sort



Insertion Sort

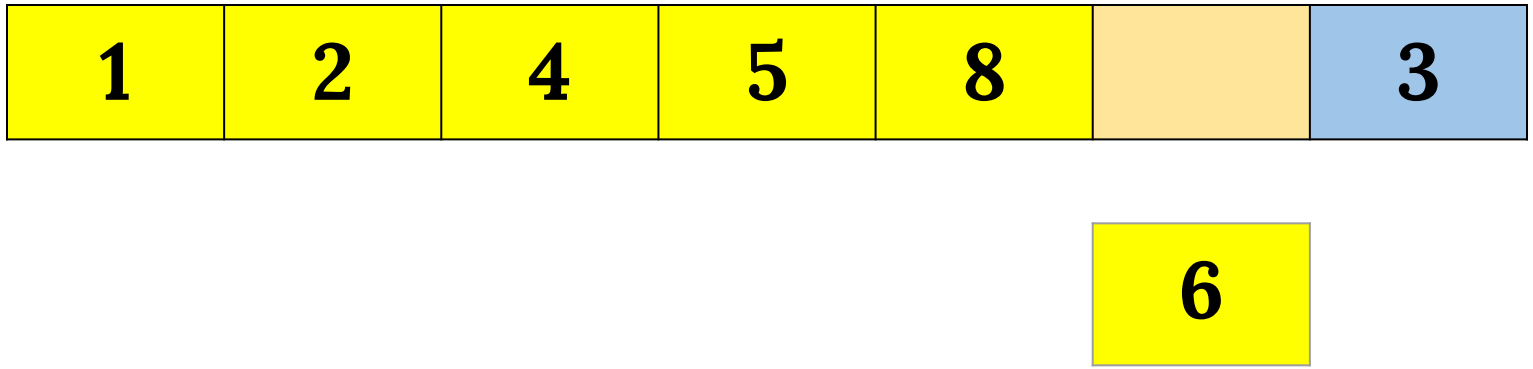


Insertion Sort

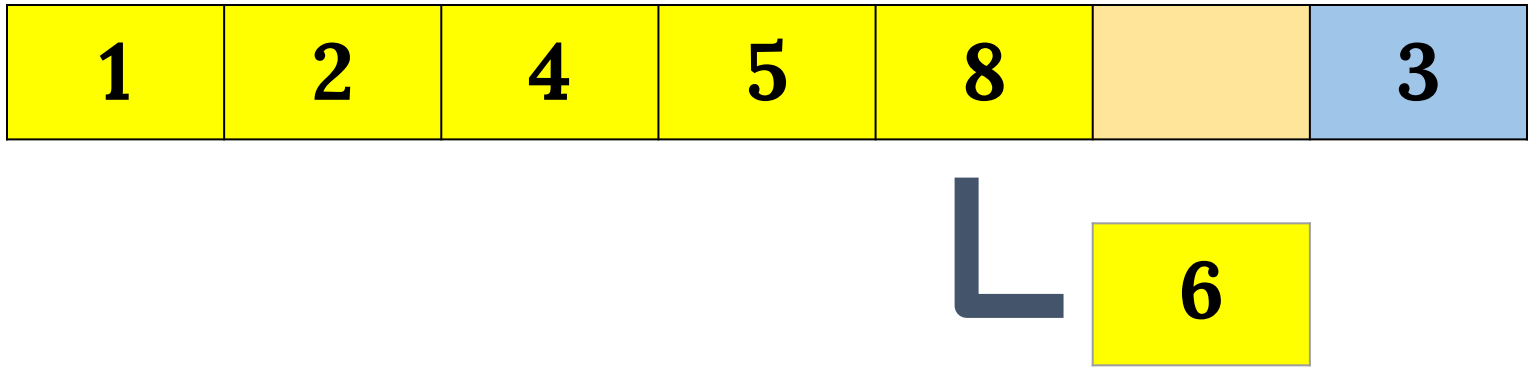


Insertion Sort

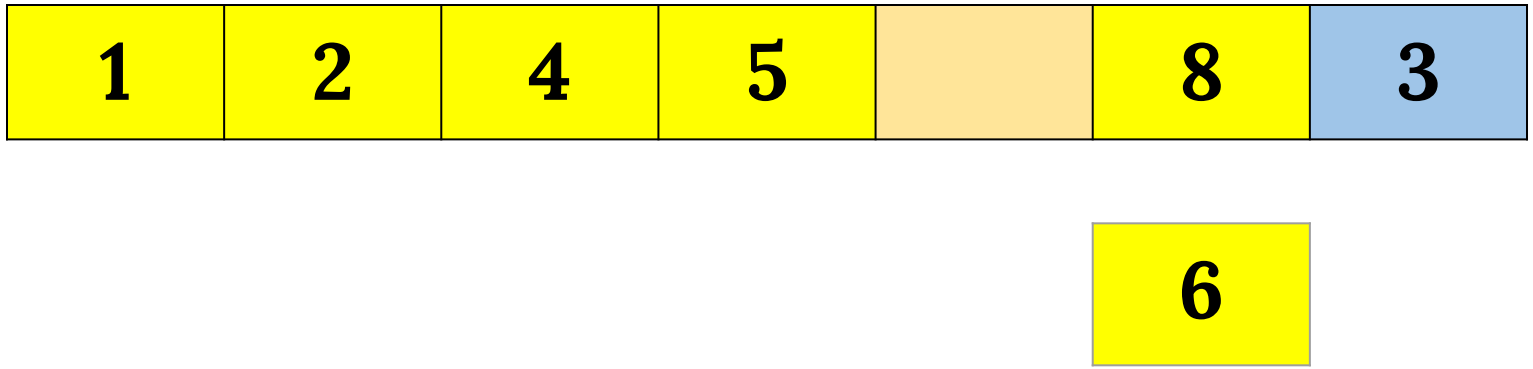
Insertion Sort



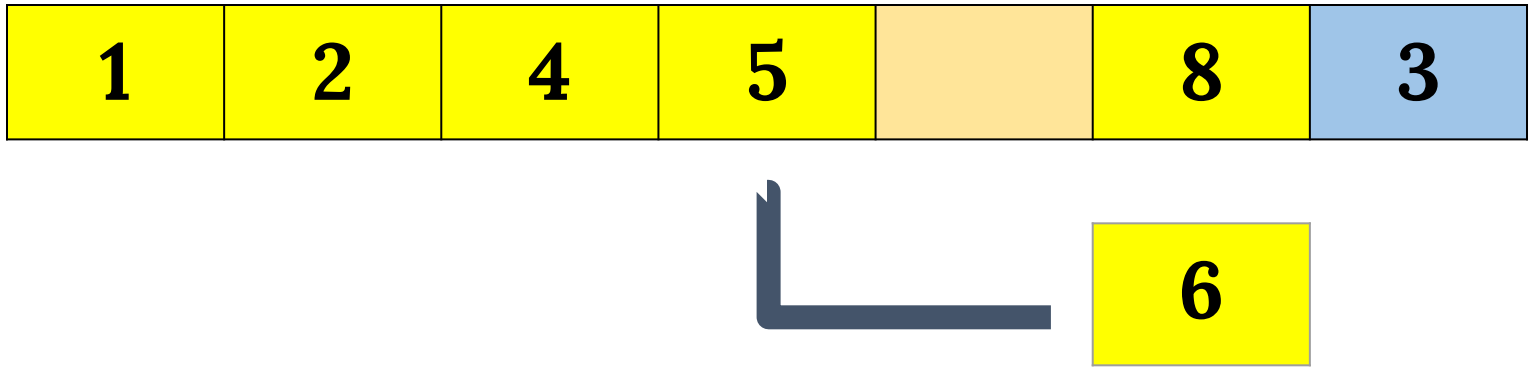
Insertion Sort



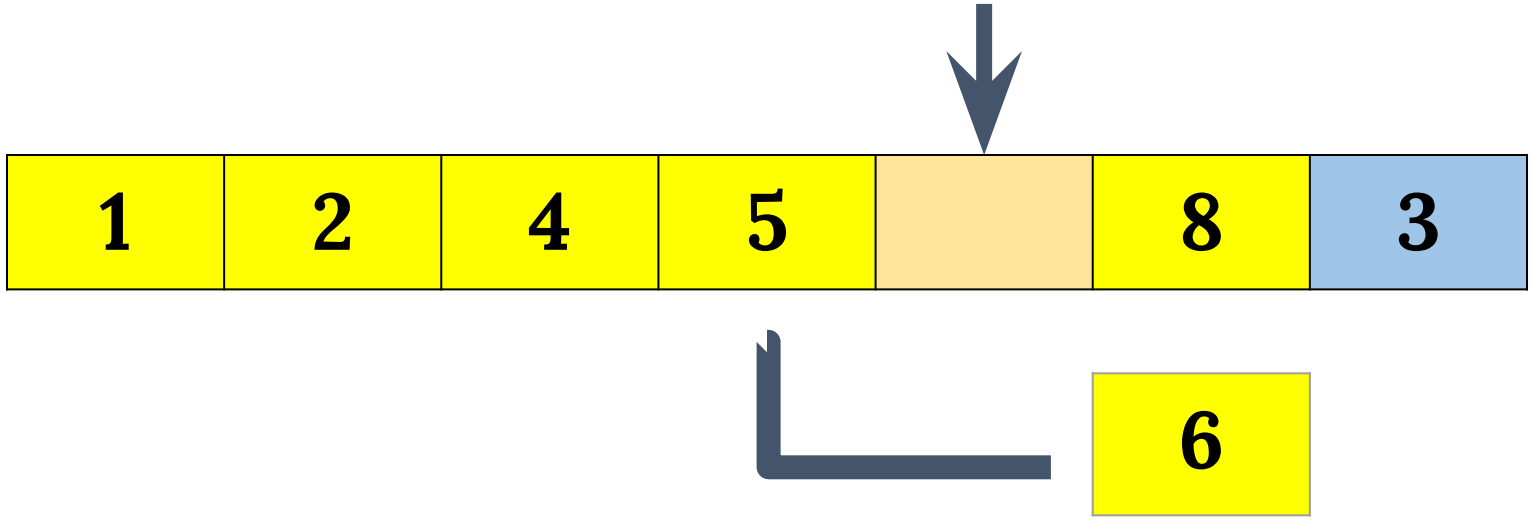
Insertion Sort



Insertion Sort

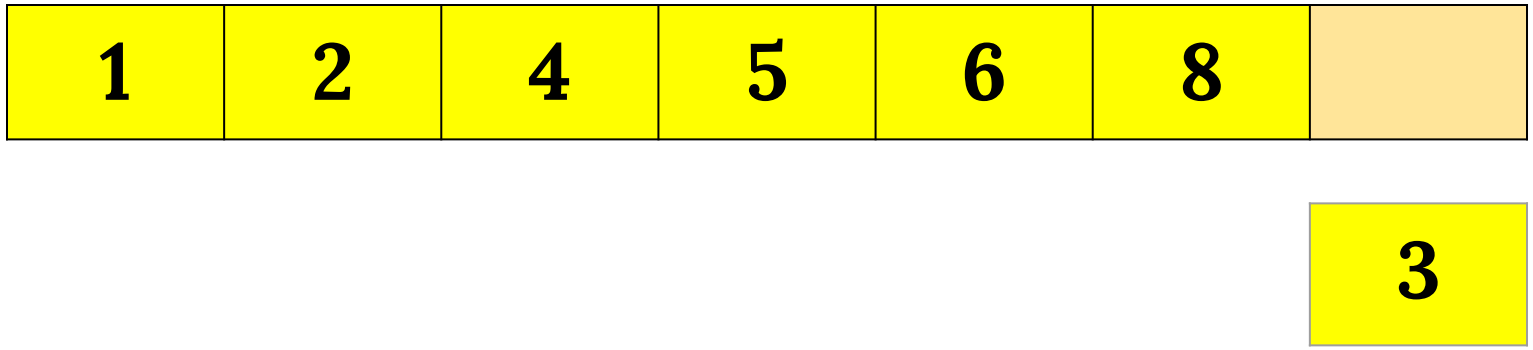


Insertion Sort

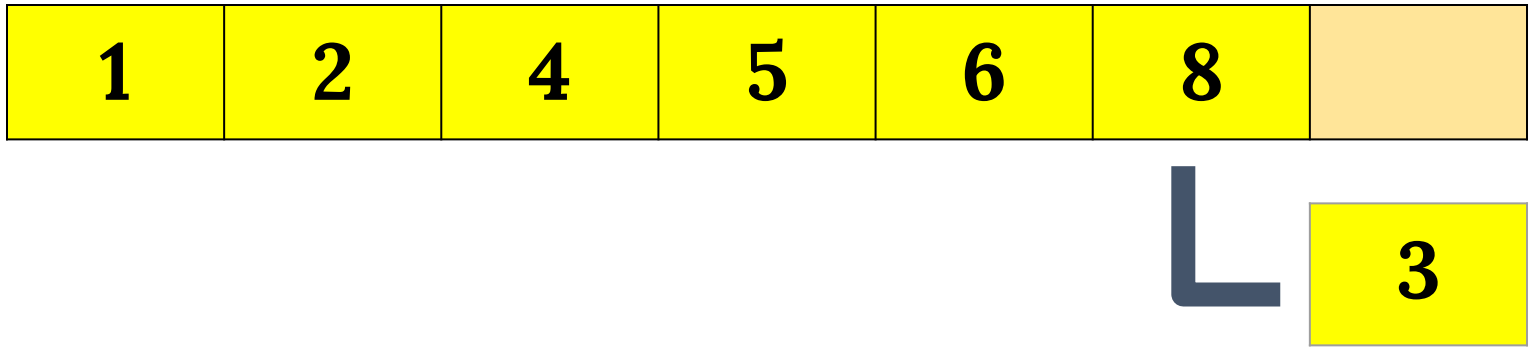


Insertion Sort

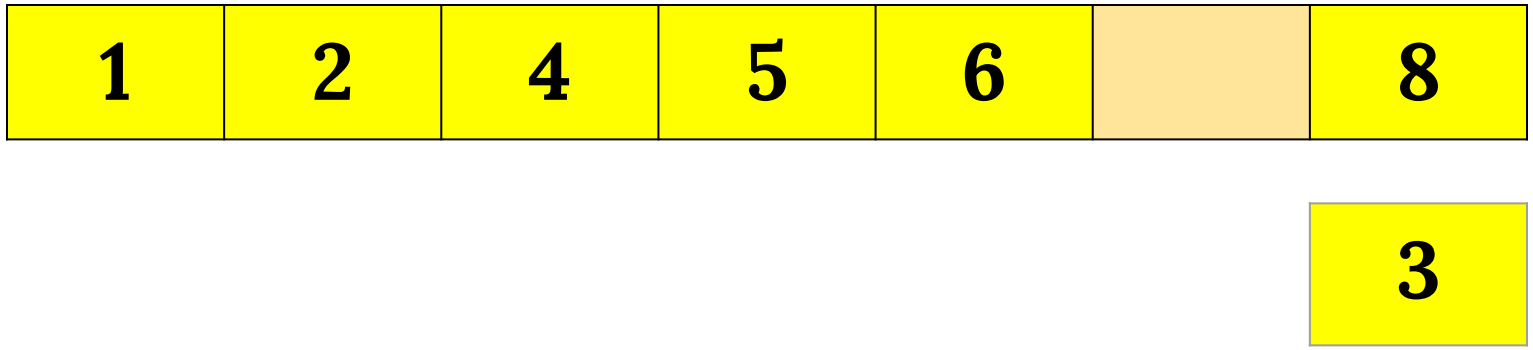
Insertion Sort



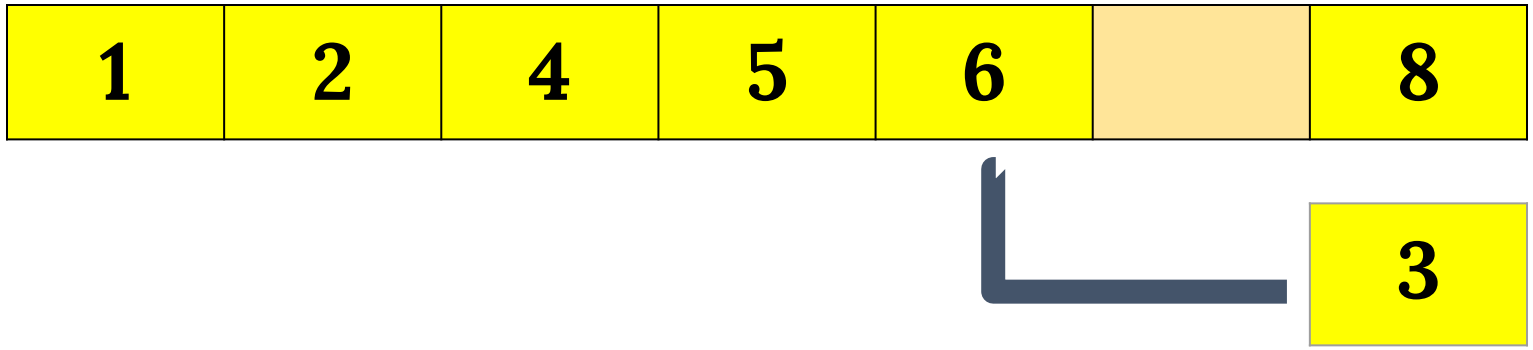
Insertion Sort



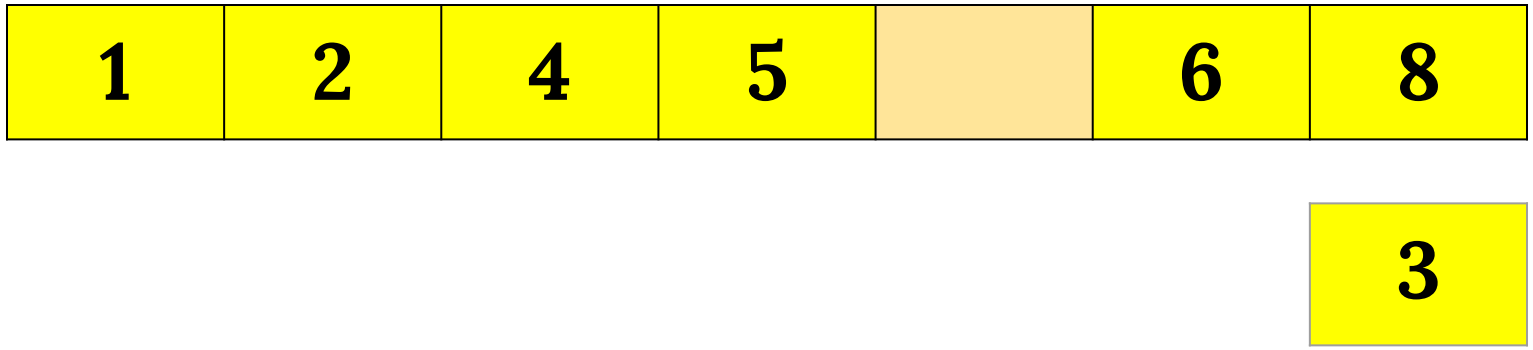
Insertion Sort



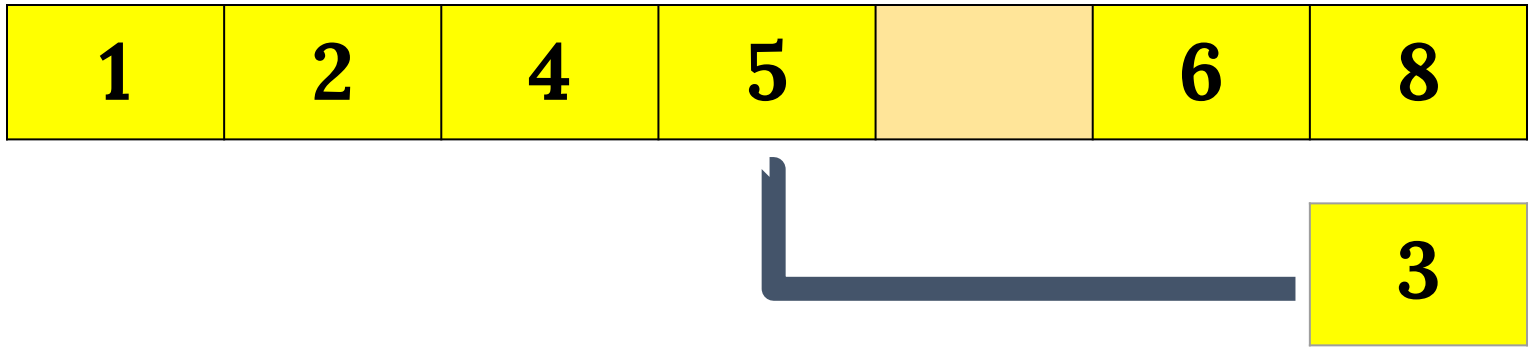
Insertion Sort



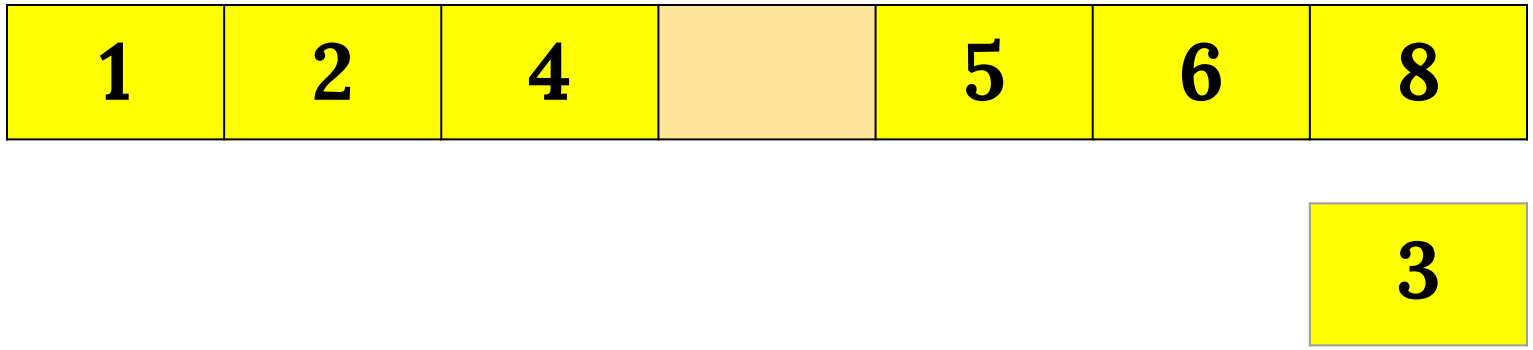
Insertion Sort



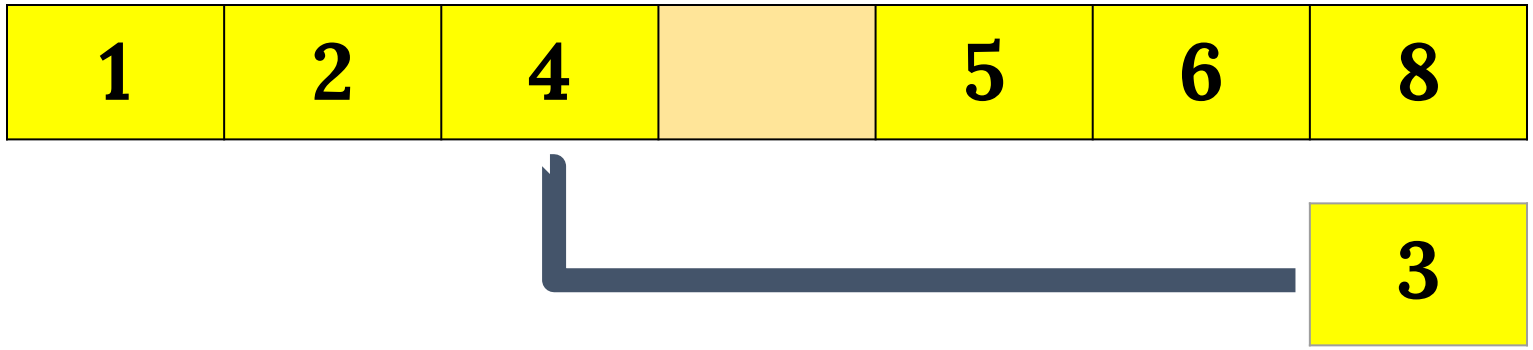
Insertion Sort



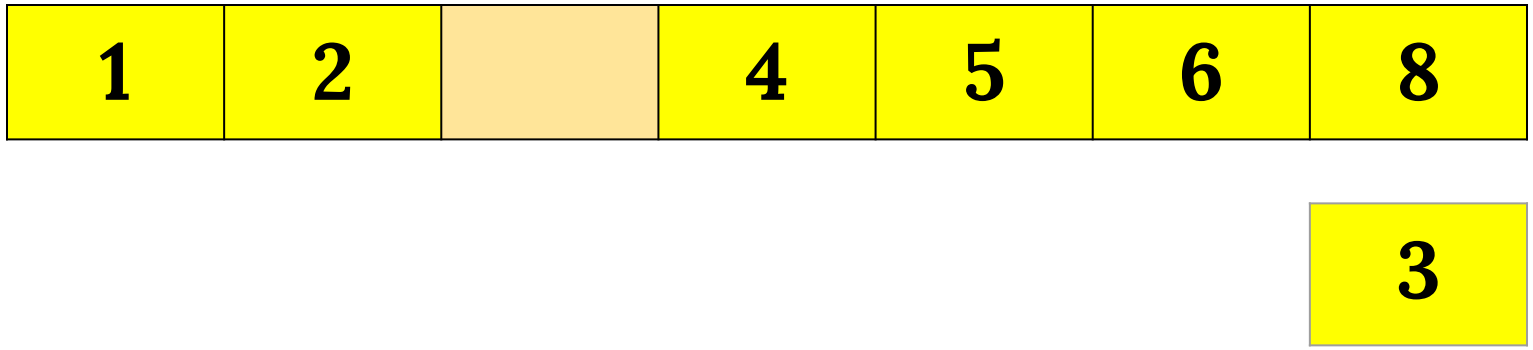
Insertion Sort



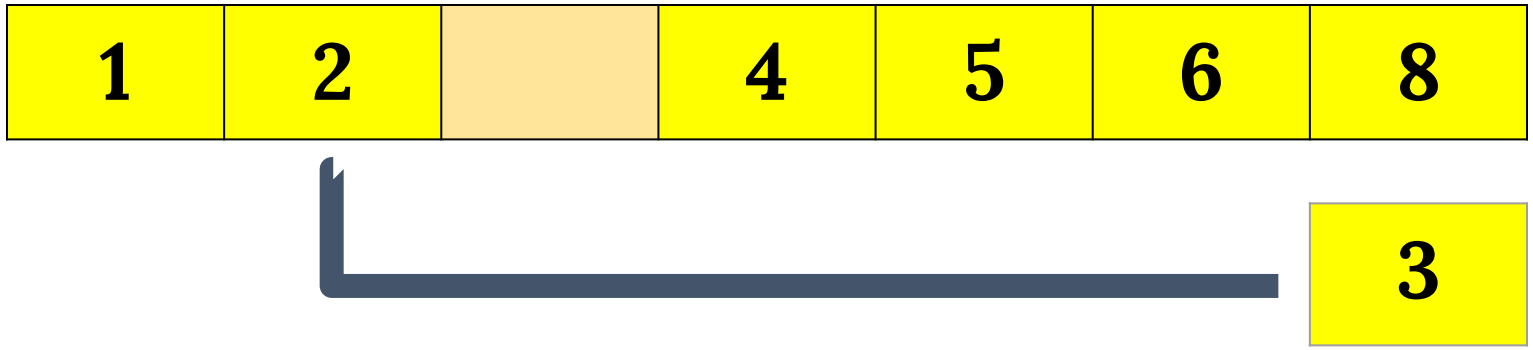
Insertion Sort



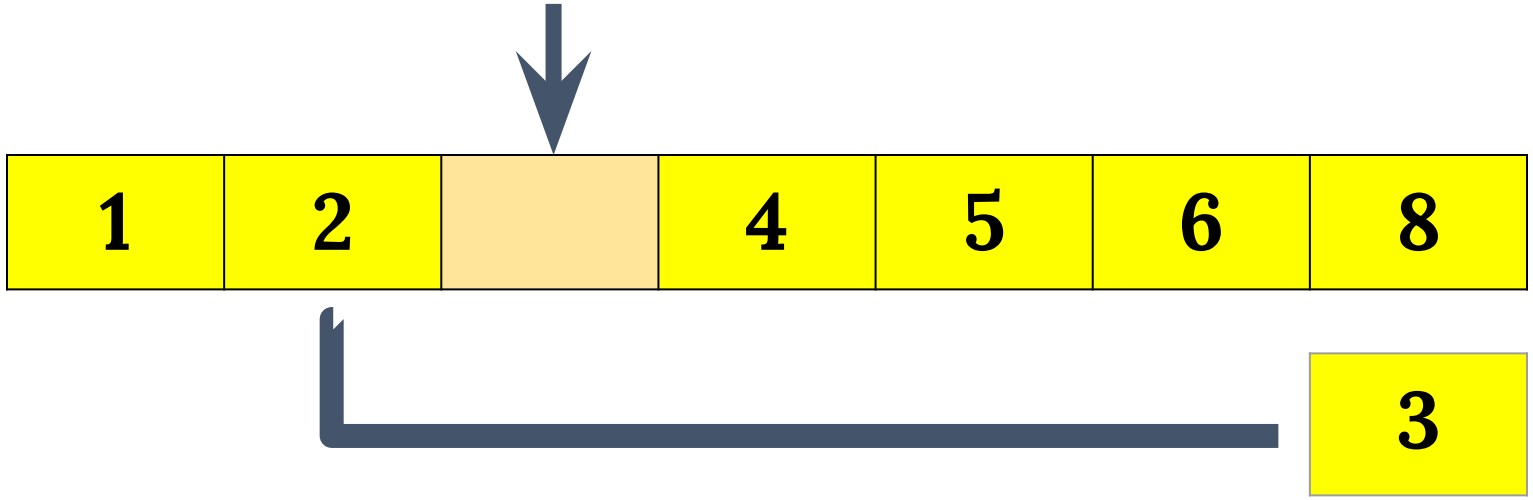
Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort

Insertion Sort

```
void insertionSort(int array[], int size) {

      for (int step = 1; step < size; step++) {
            int key = array[step];
            int j = step - 1;

            while (key < array[j] && j >= 0) {
                  array[j + 1] = array[j];
                  --j;
            }

            array[j + 1] = key;
      }
 }
```

Insertion Sort

```
void insertionSort(int array[], int size) {
     for (int step = 1; step < size; step++) {
          int key = array[step];
          int j = step - 1;
          while (key < array[j] && j >= 0) {
               array[j + 1] = array[j];
               --j;
          }
          array[j + 1] = key;
     }
}


int main() {
     int data[] = {9, 5, 1, 4, 3};
     int size = sizeof(data) / sizeof(data[0]);
     insertionSort(data, size);
}
```