

The screenshot shows a Jupyter Notebook titled 'Sort1\_G211220038.ipynb' in Google Colab. The code defines two sorting algorithms: Bubble Sort and Selection Sort. The Bubble Sort function iterates through the array, comparing adjacent elements and swapping them if they are in the wrong order. The Selection Sort function finds the minimum element in the unsorted portion of the array and swaps it with the first element. Driver code is provided for both algorithms to test them on specific arrays.

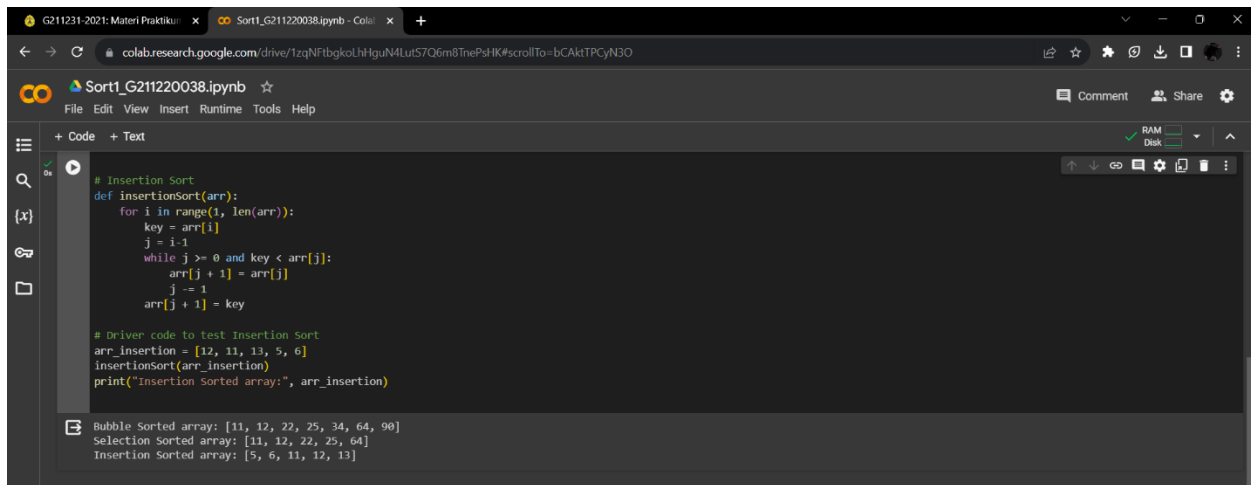
```
# Bubble Sort
def bubbleSort(arr):
    n = len(arr)
    for i in range(n-1):
        for j in range(0, n-i-1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

# Driver code to test Bubble Sort
arr_bubble = [64, 34, 25, 12, 22, 11, 90]
bubbleSort(arr_bubble)
print("Bubble Sorted array:", arr_bubble)

# Selection Sort
def selectionSort(arr):
    for i in range(len(arr)):
        min_idx = i
        for j in range(i+1, len(arr)):
            if arr[min_idx] > arr[j]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]

# Driver code to test Selection Sort
arr_selection = [64, 25, 12, 22, 11]
selectionSort(arr_selection)
print("Selection Sorted array:", arr_selection)

# Insertion Sort
```



The screenshot shows the same Jupyter Notebook with the Insertion Sort algorithm implemented. The Insertion Sort function shifts elements of the sorted portion of the array to the right to make space for the new element being inserted. Driver code is provided to test it on a specific array. The output of the previous algorithms is also displayed in the notebook's output area.

```
# Insertion Sort
def insertionSort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

# Driver code to test Insertion Sort
arr_insertion = [12, 11, 13, 5, 6]
insertionSort(arr_insertion)
print("Insertion Sorted array:", arr_insertion)
```

Bubble Sorted array: [11, 12, 22, 25, 34, 64, 90]  
Selection Sorted array: [11, 12, 22, 25, 64]  
Insertion Sorted array: [5, 6, 11, 12, 13]

## 1. Bubble Sort:

### Algoritma Bubble Sort:

- Iterasi dilakukan sebanyak  $n-1$  kali, di mana  $n$  adalah jumlah elemen dalam array.

- Pada setiap iterasi, dibandingkan pasangan elemen berturut-turut dalam array.
- Jika elemen pertama lebih besar dari elemen kedua, tukar posisi keduanya.
- Iterasi terus berlanjut hingga tidak ada pertukaran yang dilakukan pada suatu iterasi, menandakan bahwa array sudah terurut.

Contoh: arr = [64, 34, 25, 12, 22, 11, 90]

Hasil setelah Bubble Sort: [11, 12, 22, 25, 34, 64, 90]

## 2. Selection Sort:

Algoritma Selection Sort:

- Iterasi dilakukan sebanyak n kali, di mana n adalah jumlah elemen dalam array.
- Pada setiap iterasi, mencari elemen terkecil dari bagian array yang belum diurutkan.
- Tukar elemen terkecil tersebut dengan elemen pertama dari bagian yang belum diurutkan.
- Iterasi terus berlanjut hingga seluruh array terurut.

Contoh: arr = [64, 25, 12, 22, 11]

Hasil setelah Selection Sort: [11, 12, 22, 25, 64]

## 3. Insertion Sort:

Algoritma Insertion Sort:

- Iterasi dimulai dari elemen kedua hingga elemen terakhir dalam array.
- Pada setiap iterasi, elemen saat ini dibandingkan dengan elemen-elemen sebelumnya yang sudah diurutkan.
- Jika elemen saat ini lebih kecil, geser elemen-elemen yang lebih besar ke kanan untuk membuat ruang bagi elemen saat ini.
- Masukkan elemen saat ini ke posisi yang sesuai dalam array yang sudah diurutkan

Contoh: arr = [12, 11, 13, 5, 6]

Hasil setelah Insertion Sort: [5, 6, 11, 12, 13]