3_3-InsertionSort

1 Pengurutan/Sorting

Insertion Sort ***

Algoritma *sorting* yang ketiga adalah *Insertion Sort*. Penjelasan algoritma ini dibagi menjadi dua bagian yaitu :

1. Section 1.1
2. Section 1.2

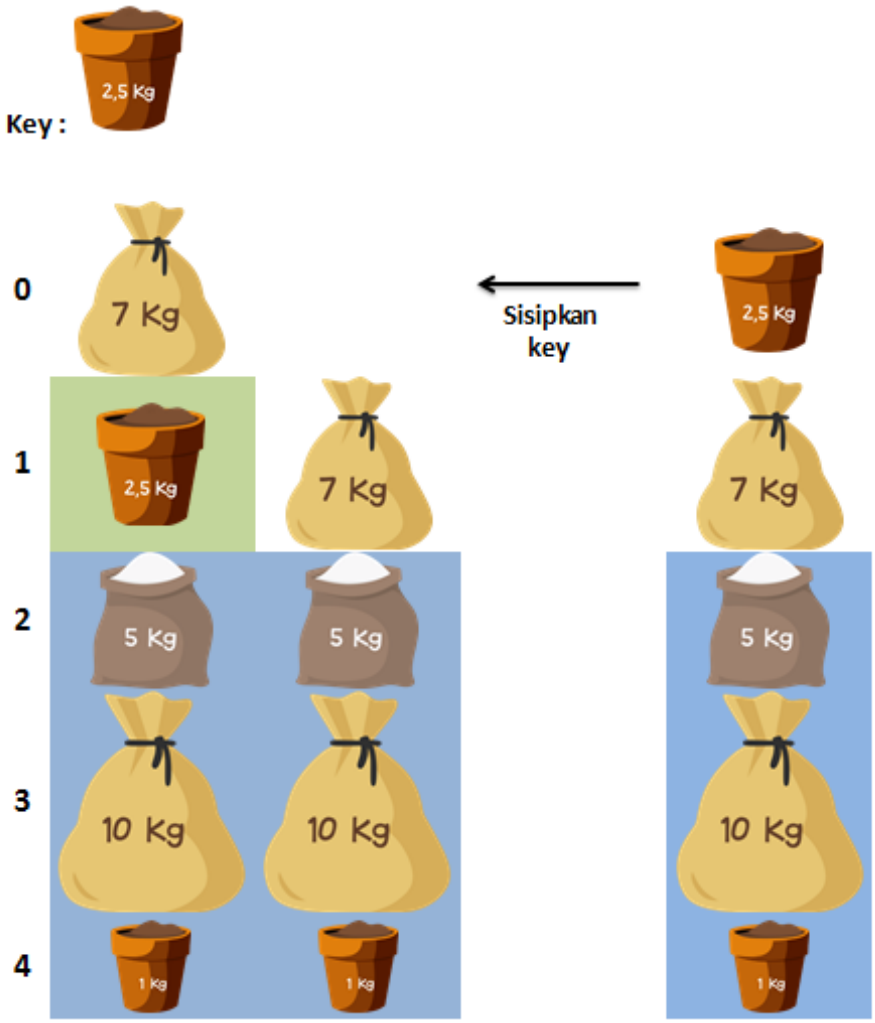
1.1 Algoritma Insertion Sort

Pada algoritma insertion sort, diasumsikan bahwa sebagian data sudah dalam keadaan terurut (pada posisi sebelah kiri dari data list), sehingga data berikutnya akan dibandingkan dengan data yang sudah dalam keadaan terurut ini, dan **disisipkan (insertion)** pada bagian data yang sudah terurut tersebut. Dengan konsep seperti ini, posisi sebelah kiri dari list data (sebelum ada data baru yang akan disisipkan), selalu dalam keadaan terurut.

Berikut algoritma Insertion Sort : 1. Asumsi bahwa data ke-1 sampai dengan $(n/2)$ sudah dalam keadaan terurut 2. Sisipkan data ke $(n/2)$+1 atau *key* ke dalam bagian data yang sudah dalam keadaan terurut, dengan cara: - lakukan perbandingan data antara key dengan data pada indeks sebelumnya, jika *key* bernilai lebih kecil, maka lakukan pergeseran posisi data. - lakukan terus perbandingan data *key* ini dengan data pada indeks sebelumnya, sampai data *key* tidak lebih kecil lagi atau atau sampai dengan data pada posisi pertama.

Berikut ilustrasi insertion sort (ascending). Misalkan terdapat data yang memiliki nilai [7, 2.5, 5, 10, 1].

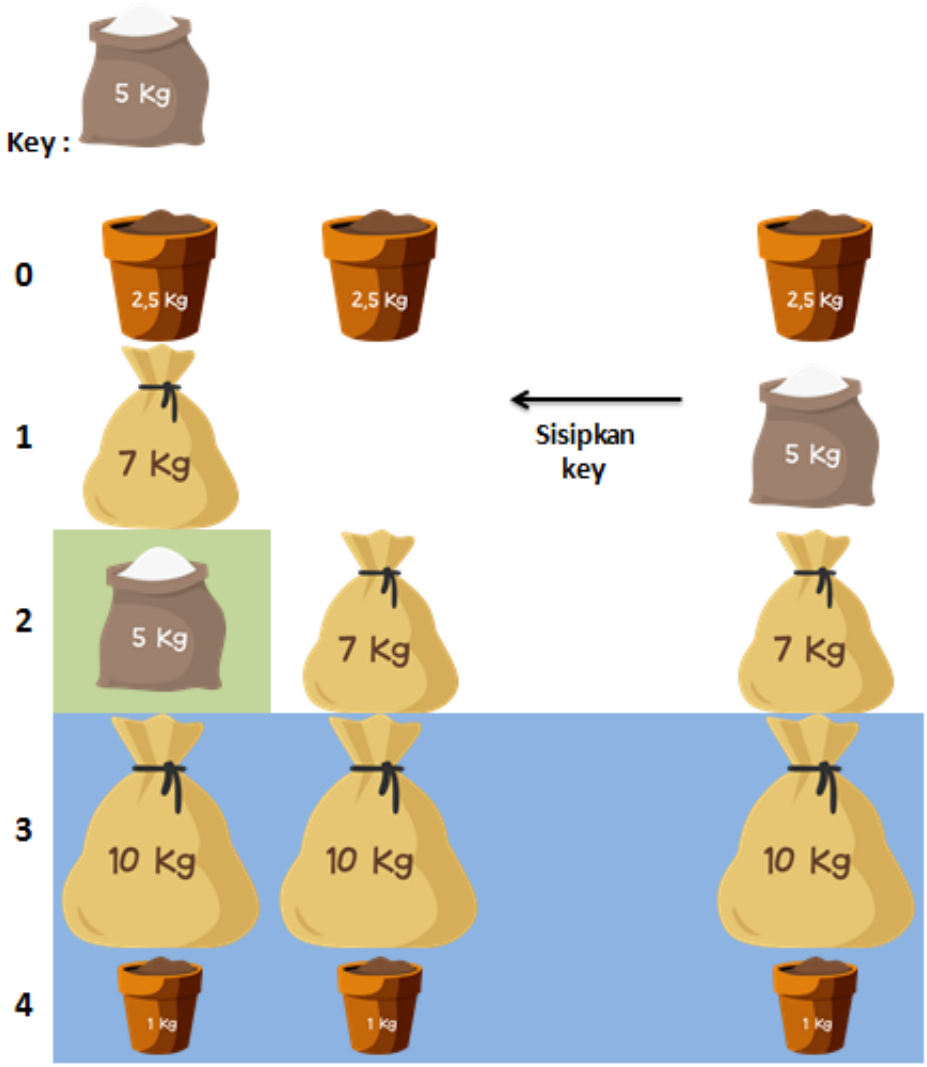
Iterasi Pertama. Pada iterasi pertama, diasumsikan indeks ke-0, adalah data yang sudah dalam keadaan terurut. Data yang digunakan sebagai **key**, adalah data indeks ke-1. Langkah berikutnya adalah menyisipkan key pada kumpulan data terurut sebelumnya, sehingga data saat ini menjadi [2.5, 7, 5, 10, 1]. Akhir dari iterasi pertama adalah, data pada indeks ke-0 dan indeks ke-1, sudah dalam keadaan terurut. Ilustrasi ini dapat dilihat pada Gambar 1.



Gambar 1. Iterasi

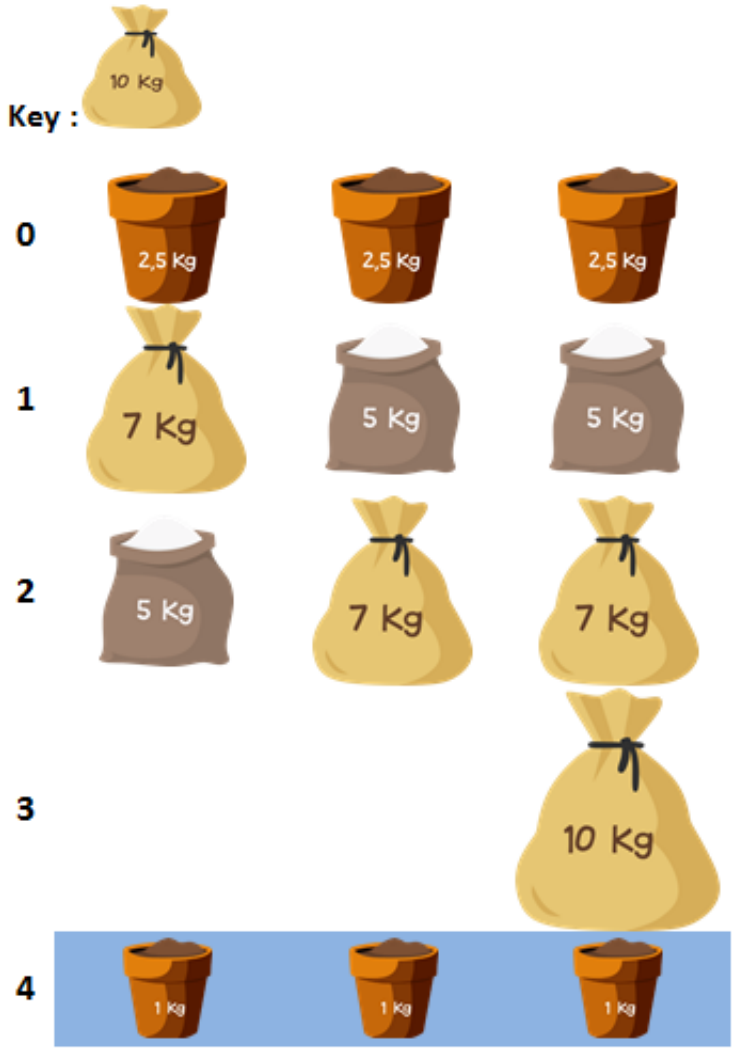
Pertama Algoritma Insertion Sort

Iterasi Kedua. Pada iterasi kedua menjadikan data pada indeks ke-2, yaitu 5, menjadi **key**. Sehingga pada iterasi kedua ini, berusaha menyispikan key pada kumpulan data terurut sebelumnya, yaitu [2.5, 7], dengan cara membandingkan satu persatu. Karena nilai key lebih kecil dari 7, dan lebih besar dari 2.5, maka key disisipkan diantara kedua nilai tersebut, sehingga data setelah iterasi kedua, menjadi [2.5, 5, 7, 10, 1]. Dapat dilihat bahwa data indeks ke-0, sampai dengan indeks ke-2, sudah dalam keadaan terurut. Ilustrasi ini dapat dilihat pada Gambar 2.



Gambar 2. Iterasi Kedua Algoritma Insertion Sort

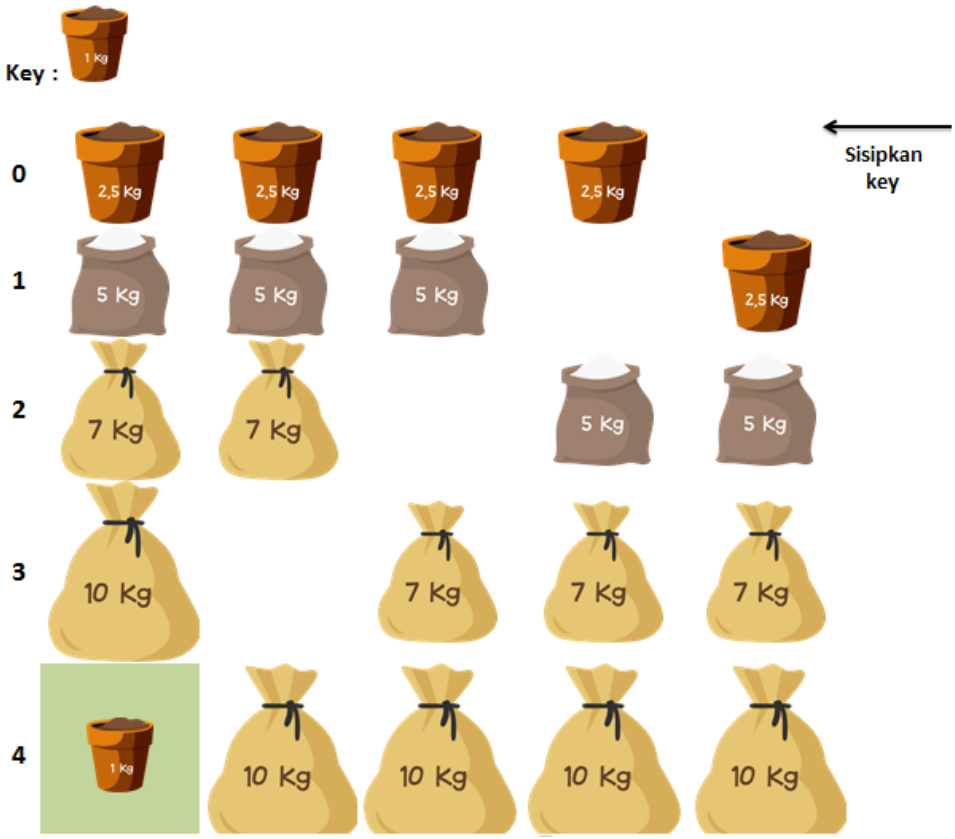
Iterasi Ketiga. Key pada iterasi ketiga adalah data pada indeks ke-3, yaitu 10. Tujuan akhir dari iterasi ketiga adalah mencari lokasi yang tepat untuk data 10 diantara data yang sudah urut sebelumnya, yaitu [2.5, 5, 7]. Karena perbandingan pertama yang dilakukan adalah dengan data 7, dan 10 > 7, maka data 10 berada pada tempat yang sama. Ilustrasi iterasi ketiga ini, dapat dilihat



pada Gambar 3. **Gambar 3.**

Iterasi Ketiga Algoritma Insertion Sort

Iterasi Keempat. Key untuk iterasi keempat adalah data pada indeks ke-4, yaitu 1. Pada iterasi keempat ini bertujuan untuk menyisipkan data 1 pada data yang sudah dalam keaadan urut sebelumnya, yaitu [2.5, 5, 7, 10]. Perbandingan dilakukan sebanyak empat kali untuk mencari posisi yang tepat untuk data 1 ini. Sehingga data 1 harus disisipkan, sebelum data 2.5. Ilustrasi ini dapat dilihat pada Gambar 4.



Gambar 4. Iterasi

Keempat Algoritma Insertion Sort

Akhir dari iterasi keempat ini adalah data sudah dalam keaadan terurut, seperti yang ditun-

jukkan pada Gambar 5.

Gambar 5. Hasil Pengurutan

Section 1

1.2 Code

Berikut code untuk pengurutan dengan menggunakan algoritma Insertion Sort. Iterasi dalam, menggunakan iterasi while, karena iterasi akan berhenti atau tidak dikerjakan, ketika nilai pada key lebih dari data terakhir yang akan dibandingkan.

```
In [5]: def insertionSort(listData):

            for outIter in range(1,len(listData)):
                print(listData)
                key=listData[outIter]
                ind=outIter
                while (ind>0 and listData[ind-1]>key):
                    listData[ind]=listData[ind-1]
                    ind=ind-1
                    print('inner=',listData)
                listData[ind]=key

            print('sortedData=',listData)
```

```
In [6]: b=[10,2,5,8,1,20,2,2,4]
        insertionSort(b)
```

```
[10, 2, 5, 8, 1, 20, 2, 2, 4]
inner= [10, 10, 5, 8, 1, 20, 2, 2, 4]
[2, 10, 5, 8, 1, 20, 2, 2, 4]
inner= [2, 10, 10, 8, 1, 20, 2, 2, 4]
[2, 5, 10, 8, 1, 20, 2, 2, 4]
inner= [2, 5, 10, 10, 1, 20, 2, 2, 4]
[2, 5, 8, 10, 1, 20, 2, 2, 4]
inner= [2, 5, 8, 10, 10, 20, 2, 2, 4]
inner= [2, 5, 8, 8, 10, 20, 2, 2, 4]
inner= [2, 5, 5, 8, 10, 20, 2, 2, 4]
inner= [2, 2, 5, 8, 10, 20, 2, 2, 4]
[1, 2, 5, 8, 10, 20, 2, 2, 4]
[1, 2, 5, 8, 10, 20, 2, 2, 4]
inner= [1, 2, 5, 8, 10, 20, 20, 2, 4]
inner= [1, 2, 5, 8, 10, 10, 20, 2, 4]
inner= [1, 2, 5, 8, 8, 10, 20, 2, 4]
inner= [1, 2, 5, 5, 8, 10, 20, 2, 4]
[1, 2, 2, 5, 8, 10, 20, 2, 4]
inner= [1, 2, 2, 5, 8, 10, 20, 20, 4]
inner= [1, 2, 2, 5, 8, 10, 10, 20, 4]
inner= [1, 2, 2, 5, 8, 8, 10, 20, 4]
inner= [1, 2, 2, 5, 5, 8, 10, 20, 4]
[1, 2, 2, 2, 5, 8, 10, 20, 4]
inner= [1, 2, 2, 2, 5, 8, 10, 20, 20]
inner= [1, 2, 2, 2, 5, 8, 10, 10, 20]
inner= [1, 2, 2, 2, 5, 8, 8, 10, 20]
inner= [1, 2, 2, 2, 5, 5, 8, 10, 20]
sortedData= [1, 2, 2, 2, 4, 5, 8, 10, 20]
```

Section 1