Pengurutan/Sorting

Indah Agustien Siradjuddin

Bubble Sort

Data tidak hanya sekedar diinput, tetapi data juga butuh diolah sehingga dapat diambil suatu analisa. Proses pengurutan merupakan salah satu proses yang sangat penting dalam pengolahan data. Misalkan terdapat data mahasiswa, terkadang dibutuhkan data yang urut berdasarkan nilai, penghasilan orang tua, sehingga dari data yang urut tersebut dapat diambil keputusan mengenai mahasiswa yang berhak menerima beasiswa.

Secara umum, proses pengurutan data ini dilakukan dengan membandingkan antara data yang satu dengan data yang lain.

Algoritma sorting yang pertama, adalah Bubble Sort, yang terdiri dari:

1. Algoritma Bubble Sort
2. Code

Bubble Sort

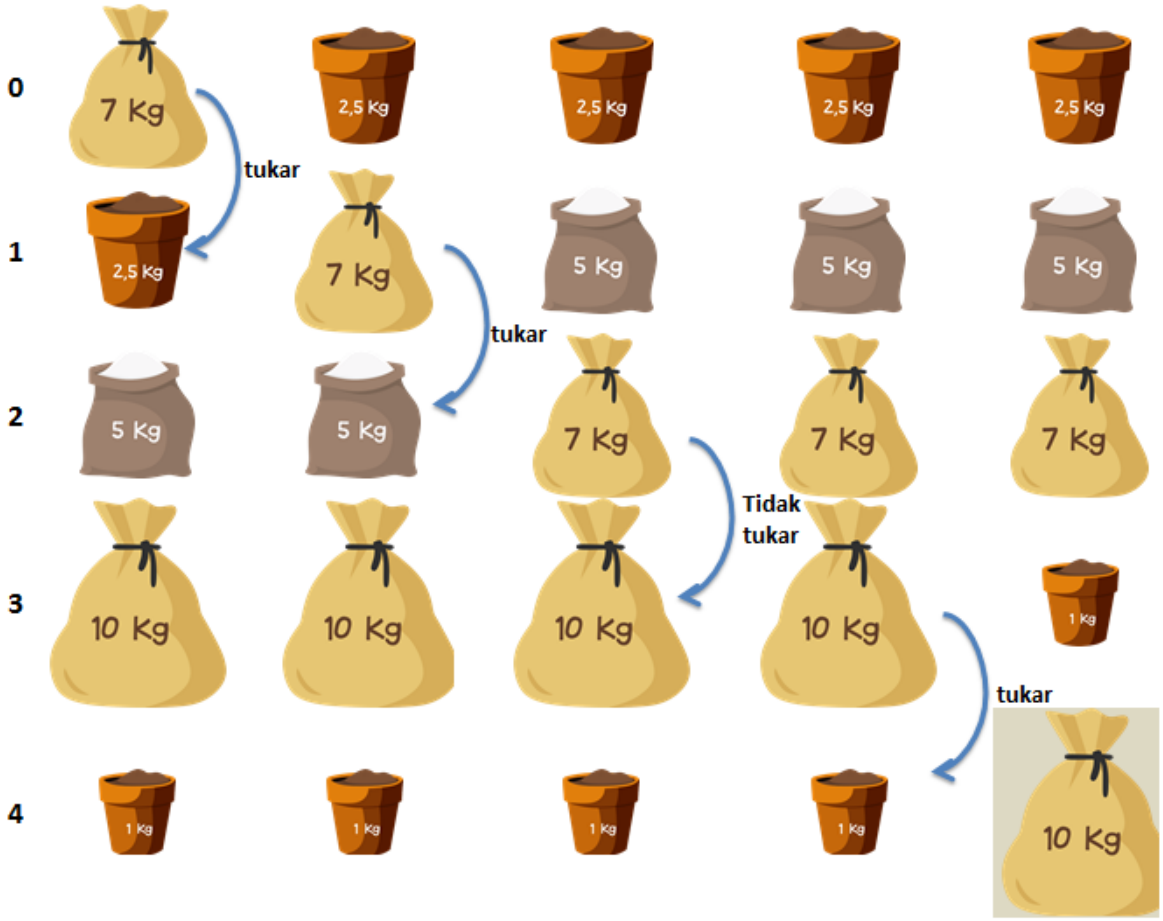
Algoritma pengurutan yang pertama adalah *Bubble Sort*. Algoritma ini dikenal sebagai algoritma pengurutan yang membutuhkan waktu paling lama, hanya saja algoritma ini adalah algoritma pengurutan yang paling sederhana.

Bubble sort mengurutkan data dengan cara sebagai berikut :

1. data dibaca mulai dari paling kiri atau mulai dari indeks ke 0 (phyton) dari suatu list data
2. Bandingkan antara dua buah data yang saling berdekatan (antara data ke *i* dan *i+1*), tukar posisi jika data ke-*i* lebih besar dibandingkan data ke-*i+1* (pengurutan secara *ascending*)
3. pindah satu posisi
4. lakukan perbandingan seperti langkah ke-2 dan pindah satu posisi kembali seperti langkah ke-3. Lakukan terus menerus sampai indeks data terakhir.
5. Setelah langkah ke-4 dilakukan, maka data yang berada di posisi paling kanan (indeks ke-*N*) adalah data terbesar .
6. Ulangi lagi langkah 1-4 dimulai dari indeks data ke 0 sampai dengan indeks ke *N-1*, *N-2*, ..., 0

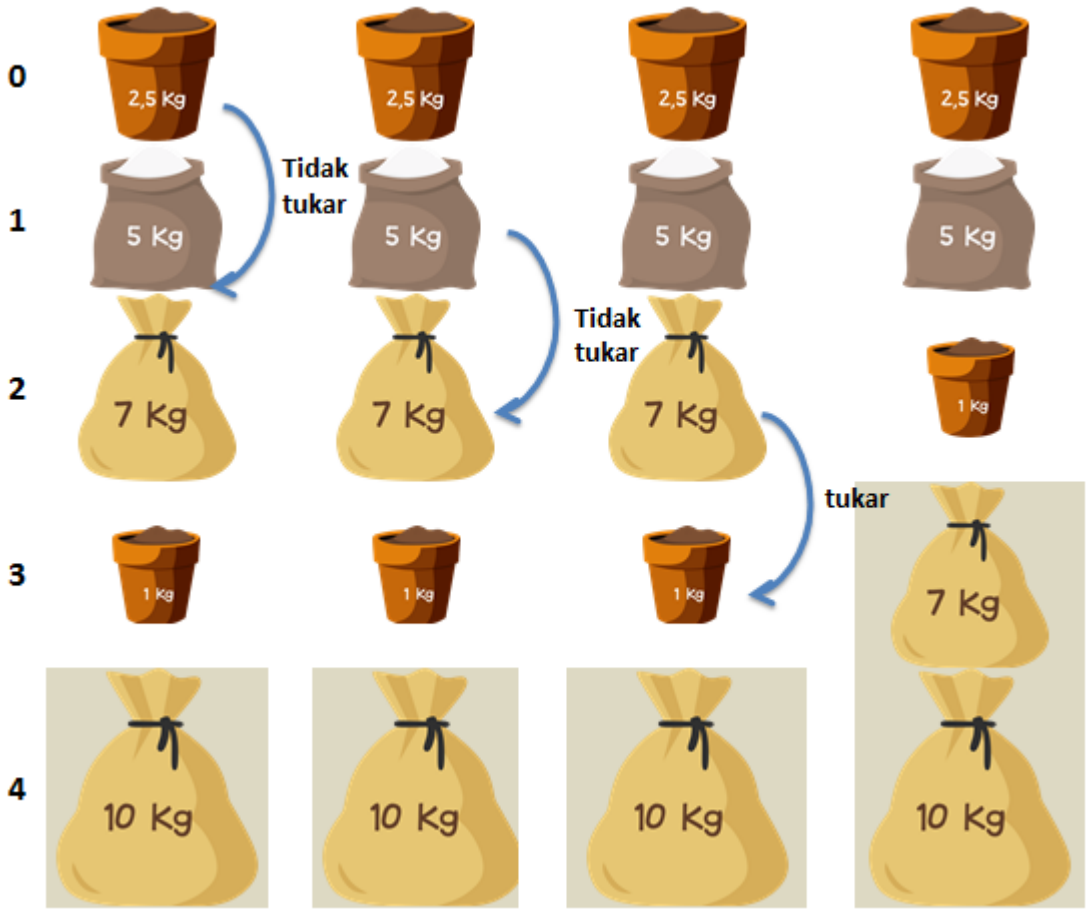
Setiap iterasi berakhir, data yang terletak di posisi paling kanan adalah data yang paling besar, oleh karena itu algoritma ini disebut dengan 'bubble' karena data terbesar disetiap iterasi berakhir bergerak keatas seperti halnya gelembung. Berikut ilustrasi algoritma bubble sort pada tiap iterasi. Misalkan terdapat data [7, 2.5, 5. 10, 1].

Iterasi Kertama, bertujuan untuk memindahkan data terbesar, yaitu 10, ke posisi terakhir atau paling kanan dari list (karena proses pengurutan secara ascending (dari kecil-ke besar). Proses iterasi pertama ini dapat dilihat pada Gambar 1.



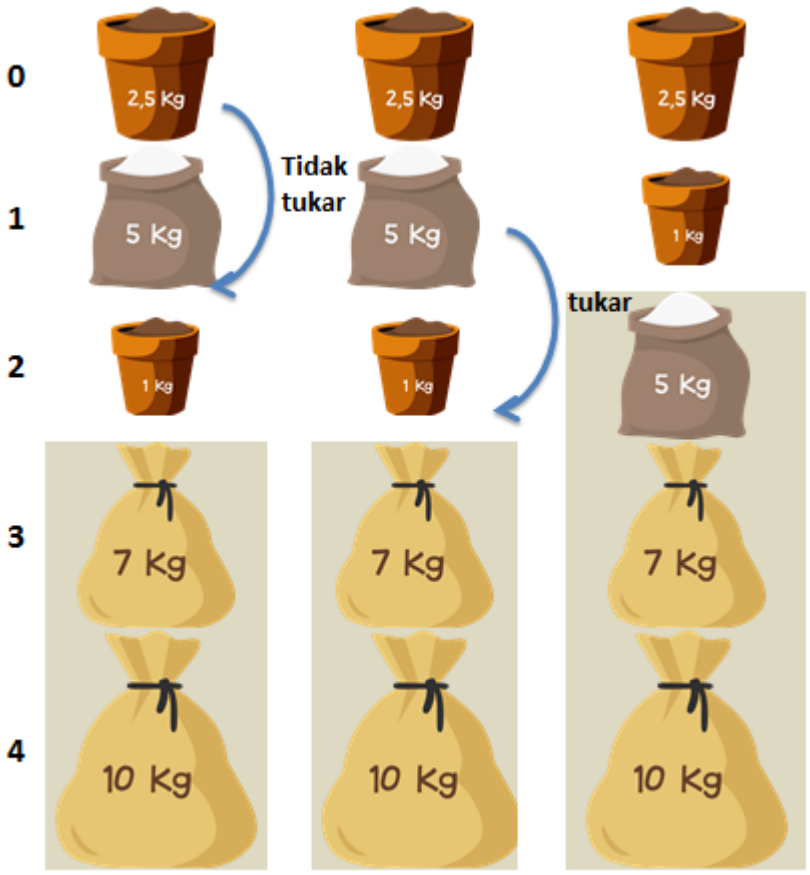
Gambar 1. Iterasi Pertama Bubble Sort

Iterasi Kedua, bertujuan untuk memindahkan data terbesar kedua, yaitu 7, agar menempati posisi tepat sebelum data 10, seperti yang terlihat pada Gambar 2.



Gambar 2. Iterasi Kedua Bubble Sort

Iterasi Ketiga, bertujuan untuk memindahkan data terbesar ketiga, yaitu 5, untuk menempati posisi tepat sebelum data 7, seperti yang ditunjukkan pada Gambar 3.



Gambar 3. Iterasi Ketiga Bubble Sort

Proses ini dilakukan terus menerus sampai tidak ada data lagi yang akan dibandingkan. Data dalam keadaan terurut, dapat dilihat pada Gambar 4.

Gambar 4. Iterasi Terakhir Bubble Sort

[Kembali ke Menu Awal]

Code

Berikut implementasi algoritma *bubble sort*.

In [7]:
```
def bubbleSort(listData):
    print('Data yang akan diurutkan : ', listData)
    count=0
    for outIter in range(len(listData)-1,-1,-1):
        print('outIter=',outIter)
        count=count+1
        print ('Iterasi ke-', count,':')
        for i in range(outIter):
            if listData[i]>listData[i+1]:
                temp=listData[i]
                listData[i]=listData[i+1]
                listData[i+1]=temp
                #listData[i],listData[i+1]=listData[i+1],listData[i]
            print(outIter,'=',listData)
    print('Data urut-',listData)
```

In [8]:

```
a=[12,0,5,1,11,20,4,2]
bubbleSort(a)
b=[12,11,5,1,0]
print('--')
bubbleSort(b)
```

```
Data yang akan diurutkan :  [12, 0, 5, 1, 11, 20, 4, 2]
outIter= 7
Iterasi ke- 1 :
7 = [0, 12, 5, 1, 11, 20, 4, 2]
7 = [0, 5, 12, 1, 11, 20, 4, 2]
7 = [0, 5, 1, 12, 11, 20, 4, 2]
7 = [0, 5, 1, 11, 12, 20, 4, 2]
7 = [0, 5, 1, 11, 12, 20, 4, 2]
7 = [0, 5, 1, 11, 12, 4, 20, 2]
7 = [0, 5, 1, 11, 12, 4, 2, 20]
outIter= 6
Iterasi ke- 2 :
6 = [0, 5, 1, 11, 12, 4, 2, 20]
6 = [0, 1, 5, 11, 12, 4, 2, 20]
6 = [0, 1, 5, 11, 12, 4, 2, 20]
6 = [0, 1, 5, 11, 12, 4, 2, 20]
6 = [0, 1, 5, 11, 4, 12, 2, 20]
6 = [0, 1, 5, 11, 4, 2, 12, 20]
outIter= 5
Iterasi ke- 3 :
5 = [0, 1, 5, 11, 4, 2, 12, 20]
5 = [0, 1, 5, 11, 4, 2, 12, 20]
5 = [0, 1, 5, 11, 4, 2, 12, 20]
5 = [0, 1, 5, 4, 11, 2, 12, 20]
5 = [0, 1, 5, 4, 2, 11, 12, 20]
outIter= 4
Iterasi ke- 4 :
4 = [0, 1, 5, 4, 2, 11, 12, 20]
4 = [0, 1, 5, 4, 2, 11, 12, 20]
4 = [0, 1, 4, 5, 2, 11, 12, 20]
4 = [0, 1, 4, 2, 5, 11, 12, 20]
outIter= 3
Iterasi ke- 5 :
3 = [0, 1, 4, 2, 5, 11, 12, 20]
3 = [0, 1, 4, 2, 5, 11, 12, 20]
3 = [0, 1, 2, 4, 5, 11, 12, 20]
outIter= 2
Iterasi ke- 6 :
2 = [0, 1, 2, 4, 5, 11, 12, 20]
2 = [0, 1, 2, 4, 5, 11, 12, 20]
outIter= 1
Iterasi ke- 7 :
1 = [0, 1, 2, 4, 5, 11, 12, 20]
outIter= 0
Iterasi ke- 8 :
Data urut- [0, 1, 2, 4, 5, 11, 12, 20]
--
Data yang akan diurutkan :  [12, 11, 5, 1, 0]
outIter= 4
```

```
Iterasi ke- 1 :
4 = [11, 12, 5, 1, 0]
4 = [11, 5, 12, 1, 0]
4 = [11, 5, 1, 12, 0]
4 = [11, 5, 1, 0, 12]
outIter= 3
Iterasi ke- 2 :
3 = [5, 11, 1, 0, 12]
3 = [5, 1, 11, 0, 12]
3 = [5, 1, 0, 11, 12]
outIter= 2
Iterasi ke- 3 :
2 = [1, 5, 0, 11, 12]
2 = [1, 0, 5, 11, 12]
outIter= 1
Iterasi ke- 4 :
1 = [0, 1, 5, 11, 12]
outIter= 0
Iterasi ke- 5 :
Data urut- [0, 1, 5, 11, 12]
```

In [0]:
```
b=[11,12,1,5,0]
bubbleSort(b)
```

In [0]:
```
10,2,5,8,1,20,2,2,4]
```

In [0]:
```
b=[10,2,5,8,1,20,2,2,4]
bubbleSort(b)
```

Latihan - 1

1. Pada contoh pengurutan diatas (data b), urutan data hasil dari empat iterasi tidak mengalami perubahan apapun (karena data sudah dalam keaadaan terurut pada setelah iterasi ke-4. Modifikasi algoritma bubble sort tersebut agar iterasi terhenti ketika tidak ada lagi data yang akan ditukar posisinya (semua data sudah berada di tempat yang tepat).
2. Buat modifikasi algoritma bubble sort. Jika pada algoritma bubble sort, pointer bergerak dari kiri ke kanan (menuju index terakhir), sedemikian hingga data terbesar berada pada index terakhir. Modifikasi yang dilakukan adalah, setelah data terbesar terletak pada index terakhir. Pointer bergerak lagi dengan arah sebaliknya, yaitu dari kanan ke kiri untuk membawa data terkecil ke index paling awal (bidirectional bubble sort)

In [0]: