3_2-selectionSort

1 Pengurutan/Sorting

Selection Sort ***

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

1. Section 1.1
2. Section 1.2

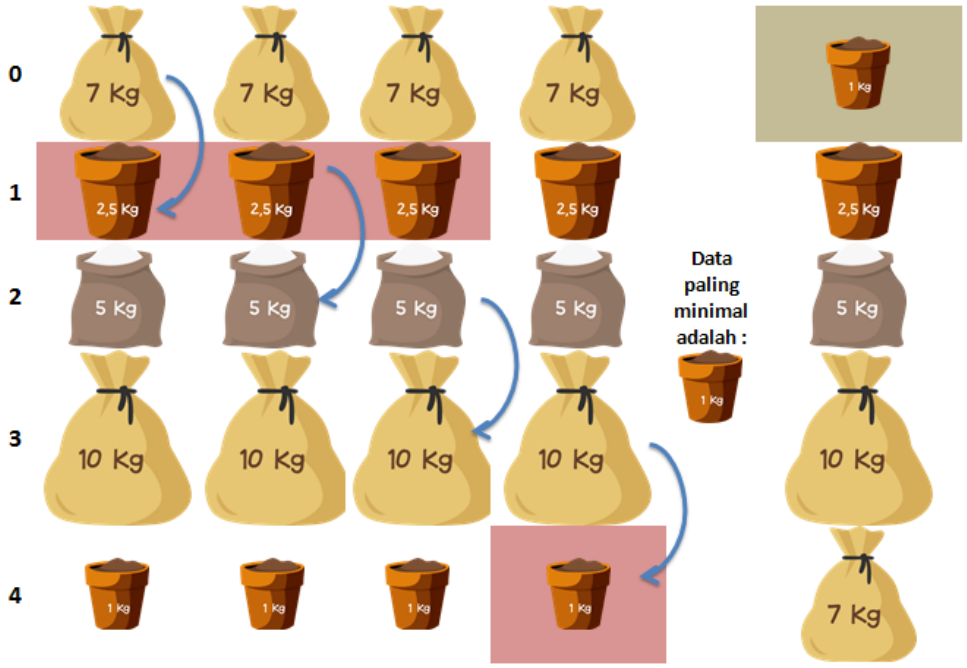
1.1 Algoritma Selection Sort

Algoritma **selection sort** memiliki waktu komputasi yang lebih cepat dibandingkan dengan algoritma **bubble sort**. Hal ini dikarenakan terjadi **reduksi waktu pada saat proses pertukaran data**. Jika algoritma bubble sort, pada setiap iterasi,setiap dua data yang berdekatan akan dibandingkan, dan akan dilakukan pertukaran data jika syarat memenuhi. Maka pada **algoritma selection sort tidak akan melakukan dua proses ini (perbandingan dan pertukaran) pada satu waktu**, akan tetapi, pada algoritma selection sort, akan dicari data yang memenuhi syarat terlebih dahulu (perbandingan data), kemudian data yang terpilih ini ini akan diletakkan pada indeks data yang tepat. Oleh karena itu algoritma ini dinamakan **selection** (pemilihan data).

Berikut tahapan algoritma selection sort (ascending order): 1. Algoritma ini dimulai dari indeks awal sampai dengan indeks akhir data 2. Cari data dengan nilai paling minimal (dari indeks awal sampai dengan indeks akhir) melalui proses perbandingan 3. Letakkan data minimal ini di indeks awal 4. Ulangi lagi proses pencarian data paling minimal (dari indeks awal+1 sampai dengan indeks akhir, karena indeks awal sudah terisi data yang tepat). 5. Letakkan data ini pada indeks awal+1 6. Ulangi lagi proses pencarian data paling minimal (dari indeks awal+2 sampai dengan akhir) dan letakkan data ini pada indeks awal+2, dst.

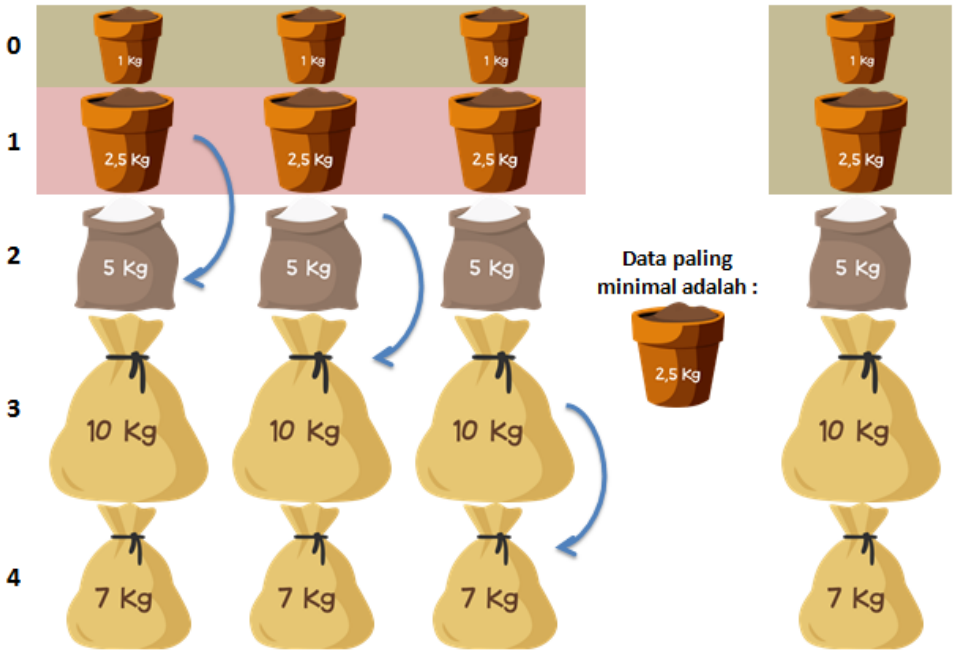
Berikut ilustrasi algoritma **selection sort** ini. Misalkan terdapat suatu data, yaitu [7, 2.5, 5, 10, 1]. Maka proses pengurutan secara ascending (dari nilai kecil sampai dengna nilai besar) dengan menggunakan algoritma Selection Sort adalah sebagai berikut :

Iterasi Pertama. Pada iterasi pertama ini, data yang memiliki nilai paling kecil, akan menempati posisi dengan indeks paling awal. Untuk mencari data dengan nilai paling kecil ini, dilakukan perbandingan sebagai berikut: 1. antara indeks ke-0 dengan ke-1, sehingga nilai terkecil saat ini adalah 2.5 2. antara indeks ke-1 dengan ke-2, nilai terkecil tetaplah 2.5 3. antara indeks ke-2 dengan ke-3, nilai terkecil tetaplah 2.5 2. antara indeks ke-3 dengan ke-4, didapatkan nilai terkecil adalah 1. Pada saat pencarian nilai terkecil ini tidak dilakukan proses pertukaran posisi (berbeda dengan a;goritma bubble sort). Setelah didapatkan nilai terkecil, maka nilai 1 ini akan ditukar dengan data pada posisi dengan indeks ke-0, sehingga setelah iterasi pertama berakhir, data 1, akan menempati indeks ke-0. Ilustrasi iterasi pertama ini dapat dilihat pada Gambar 1.



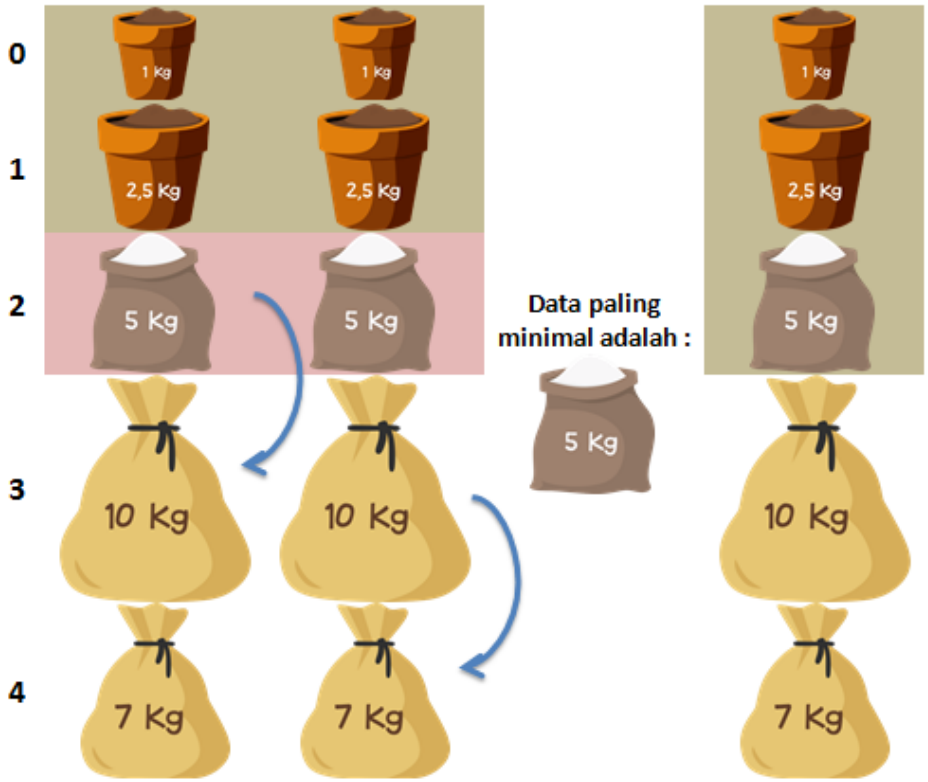
Gambar 1. Iterasi Pertama Algoritma Selection Sort

Iterasi Kedua. Pada iterasi kedua ini, data yang memiliki nilai paling kecil kedua, akan menempati posisi dengan indeks setelah data paling kecil, yaitu indeks ke-1. Untuk mencari data dengan nilai paling kecil kedua ini, dilakukan perbandingan, hanya saja data pada indeks ke-0 tidak perlu dibandingkan lagi, karena data pada indeks ke-0 sudah berisi data yang tepat. Perbandingan yang dilakukan untuk mencari nilai minimal kedua adalah sebagai berikut: 1. antara indeks ke-1 dengan ke-2, nilai terkecil tetaplah 2.5 2. antara indeks ke-2 dengan ke-3, nilai terkecil tetaplah 2.5 3. antara indeks ke-3 dengan ke-4, didapatkan nilai terkecil adalah 2.5 Setelah didapatkan nilai minimal, yaitu 2.5, maka nilai ini akan ditukar dengan data pada posisi dengan indeks ke-1, sehingga setelah iterasi kedua berakhir, data 2.5, akan menempati indeks ke-1. Ilustrasi iterasi kedua ini dapat dilihat pada Gambar 2.



Gambar 2. Iterasi Kedua Algoritma Selection Sort

Iterasi Ketiga. Iterasi ketiga bertujuan untuk menempatkan data minimal yang ketiga pada posisi yang tepat, yaitu indeks ke-2. Untuk mencari data minimal ketiga ini, dilakukan perbandingan, hanya saja data pada indeks ke-0 dan ke-1 tidak perlu dibandingkan lagi, karena data pada indeks ke-0 dan ke-1, sudah berisi data yang tepat. Perbandingan yang dilakukan untuk mencari nilai minimal kedua adalah sebagai berikut: 1. antara indeks ke-2 dengan ke-3, nilai terkecil adalah 5 2. antara indeks ke-3 dengan ke-4, didapatkan nilai terkecil adalah 5 Setelah didapatkan nilai minimal, yaitu 5, maka nilai ini akan ditukar dengan data pada posisi dengan indeks ke-2, sehingga setelah iterasi kedua berakhir, data 5, akan menempati indeks ke-2. Ilustrasi iterasi kedua ini dapat dilihat pada Gambar 3.



Gambar 3. Iterasi Ketiga Algoritma Selection Sort

Proses ini dilakukan secara terus menerus sampai proses pengecekan dilakukan pada data terakhir, sehingga akhir dari algoritma selection sort ini, data sudah dalam keadaan terurut, seperti

yang terlihat pada Gambar 4. **Gambar 3. Iterasi Terakhir Algoritma Selection Sort**

Section 1

1.2 Code

Berikut adalah code untuk algoritma selection sort, yang terdiri dari dua buah iterasi. Iterasi pertama digunakan untuk menempatkan data atau menukar data, sehingga data dengan nilai minimal, akan menempati indeks-indeks awal. Sedangkan iterasi kedua (iterasi dalam), digunakan untuk mencari nilai minimal.

```
In [1]: def selectionSort(listData):
            print('Algoritma Selection Sort konvensional')
            print('Data Awal=',listData)
            for outIter in range(len(listData)-1):
                minIndex=outIter
                for i in range(outIter+1,len(listData)):
                    if listData[i]<listData[minIndex]:
                        minIndex=i

                temp=listData[outIter]
                listData[outIter]=listData[minIndex]
                listData[minIndex]=temp
                print('Iterasi ke-',outIter+1,':',listData)
```

```
            print('Data Urut=',listData)
```

```
In [2]: a=[10,2,5,8,1,20,7,12,4]
        selectionSort(a)
```

```
Algoritma Selection Sort konvensional
Data Awal= [10, 2, 5, 8, 1, 20, 7, 12, 4]
Iterasi ke- 1 : [1, 2, 5, 8, 10, 20, 7, 12, 4]
Iterasi ke- 2 : [1, 2, 5, 8, 10, 20, 7, 12, 4]
Iterasi ke- 3 : [1, 2, 4, 8, 10, 20, 7, 12, 5]
Iterasi ke- 4 : [1, 2, 4, 5, 10, 20, 7, 12, 8]
Iterasi ke- 5 : [1, 2, 4, 5, 7, 20, 10, 12, 8]
Iterasi ke- 6 : [1, 2, 4, 5, 7, 8, 10, 12, 20]
Iterasi ke- 7 : [1, 2, 4, 5, 7, 8, 10, 12, 20]
Iterasi ke- 8 : [1, 2, 4, 5, 7, 8, 10, 12, 20]
Data Urut= [1, 2, 4, 5, 7, 8, 10, 12, 20]
```

1.3 Latihan - 2

1. Buat data secara random dengan jumlah 500 data, lakukan sorting dengan algoritma bubble sort dan selection sort, bandingkan waktu komputasi antara dua buah algoritma tersebut
2. Modifikasi code selection sort tersebut, agar iterasi dalam melakukan dua buah pencarian, yaitu nilai minimal dan maksimal, jika sudah ditemukan, maka letakkan nilai minimal pada indeks-indeks awal, dan nilai maksimal pada indeks-indeks akhir

Section 1