Modul 5
SORTING (PENGURUTAN)

1. Tujuan Instruksional Umum

a. Mahasiswa mampu melakukan perancangan aplikasi menggunakan Struktur Sorting (pengurutan)
b. Mahasiswa mampu melakukan analisis pada algoritma Sorting yang dibuat
c. Mahasiswa mampu mengimplementasikan algoritma Sorting pada sebuah aplikasi secara tepat dan efisien

2. Tujuan Instruksional Khusus

a. Mahasiswa mampu menjelaskan mengenai algoritma Sorting
b. Mahasiswa mampu membuat dan mendeklarasikan struktur algoritma Sorting
c. Mahasiswa mampu menerapkan dan mengimplementasikan algoritma Sorting

Pendahuluan

Pengurutan data dalam struktur data sangat penting terutama untuk data yang beripe data numerik ataupun karakter. Pengurutan dapat dilakukan secara ascending (urut naik) dan descending (urut turun). Pengurutan (Sorting) adalah proses pengurutan data yang sebelumnya disusun secara acak sehingga tersusun secara teratur menurut aturan tertentu.

Contoh:

Data Acak	: 5 6 8 1 3 25 10
Ascending	: 1 3 5 6 8 10 25
Descending	: 25 10 8 6 5 3 1

Deklarasi Array Sorting

Mendeklarasikan array secara global:

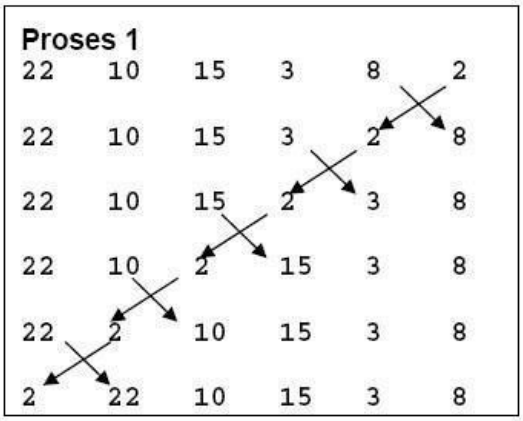
```
int data[100]; int n;
//untuk jumlah data
```

Fungsi Tukar 2 Buah Data:

```
void tukar(int a,int
b){ int tmp; tmp =
data[a]; data[a] =
data[b]; data[b] = tmp;
}
```

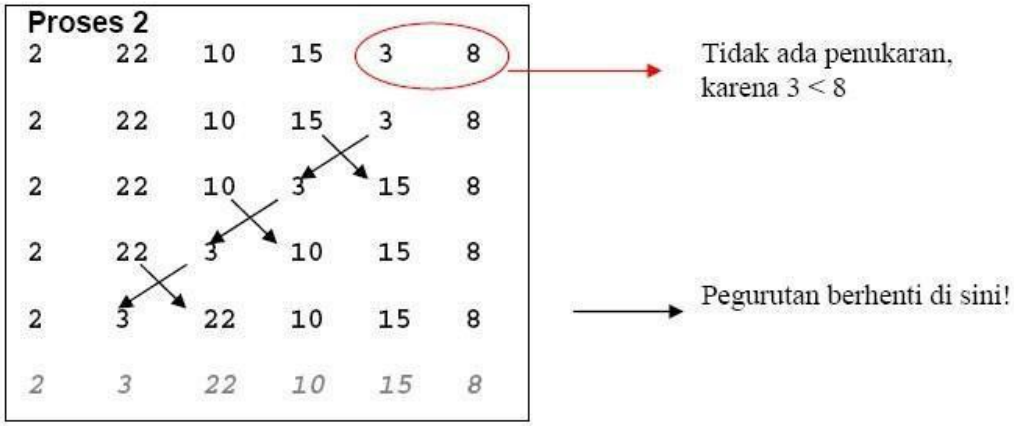
BUBBLE SORT

Merupakan metode sorting termudah, diberi nama “Bubble” karena proses pengurutan secara berangsur- angsur bergerak/berpindah ke posisinya yang tepat, seperti gelembung yang keluar dari sebuah gelas bersoda. Bubble Sort mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya. Jika elemen sekarang lebih besar dari elemen berikutnya maka kedua elemen tersebut ditukar, jika pengurutan ascending. Jika elemen sekarang lebih kecil dari elemen berikutnya, maka kedua elemen tersebut ditukar, jika pengurutan descending. Algoritma ini seolah-olah menggeser satu per satu elemen dari kanan ke kiri atau kiri ke kanan, tergantung jenis pengurutannya. Ketika satu proses telah selesai, maka bubble sort akan mengulangi proses, demikian seterusnya. Kapan berhentinya? Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan, serta tercapai perurutan yang telah diinginkan.



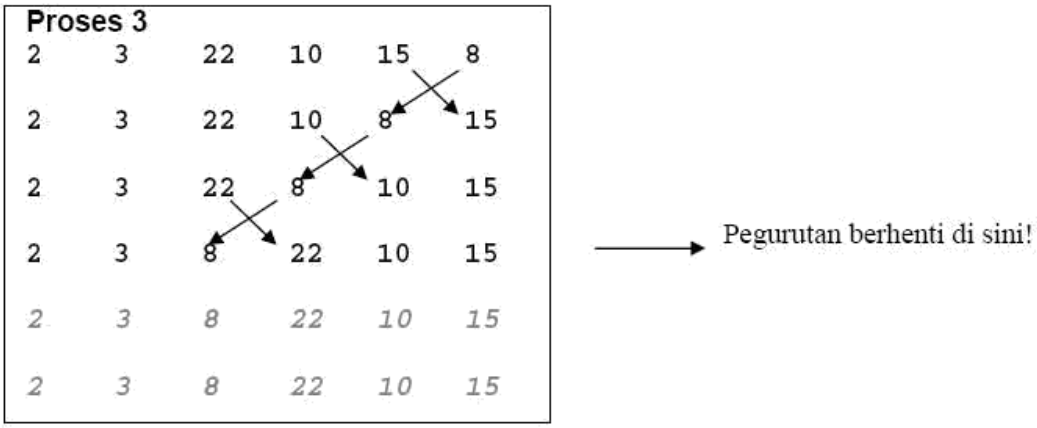
Gambar 1. Proses ke-1 algoritma Bubble Sorting

Pada gambar diatas, pegecekan dimulai dari data yang paling akhir, kemudian dibandingkan dengan data di depannya, jika data di depannya lebih besar maka akan ditukar.

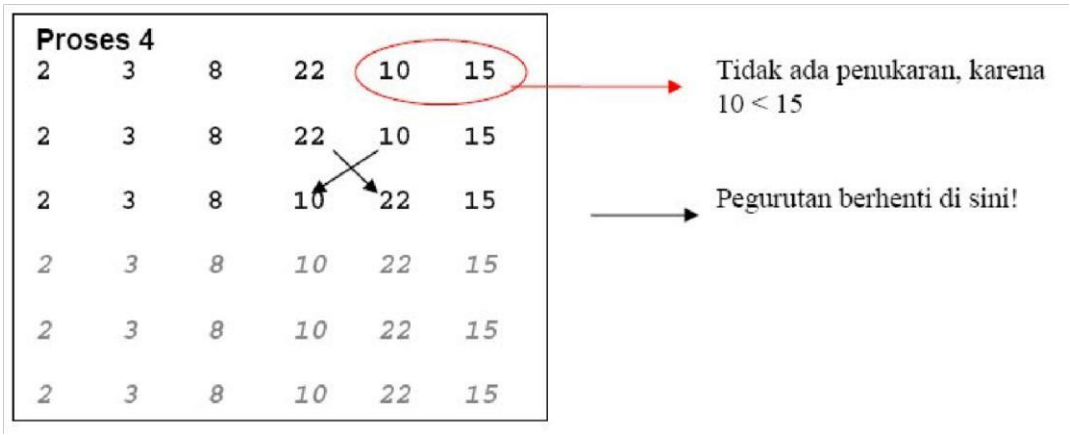


Gambar 2. Proses ke-2 algoritma Bubble Sorting

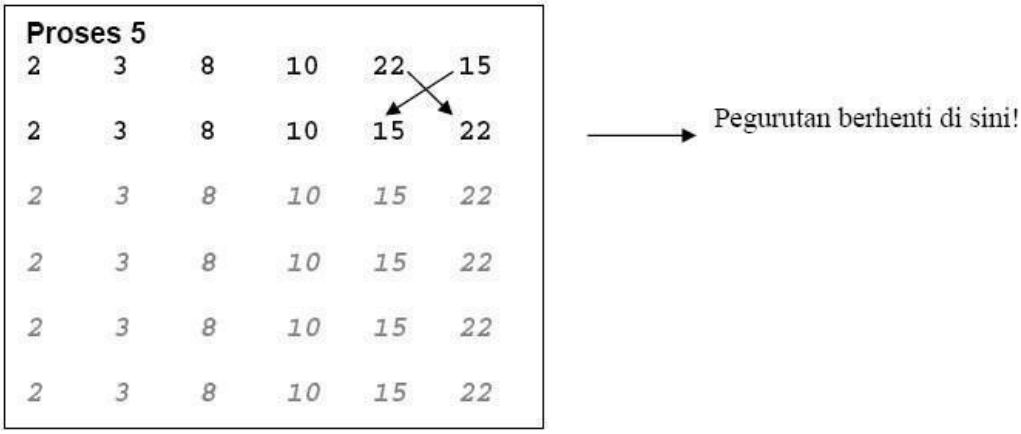
Pada proses kedua, pengecekan dilakukan sampai dengan data ke-2 karena data pertama pasti sudah paling kecil.



Gambar 3. Proses ke-3 algoritma Bubble Sorting



Gambar 4. Proses ke-4 algoritma Bubble Sorting

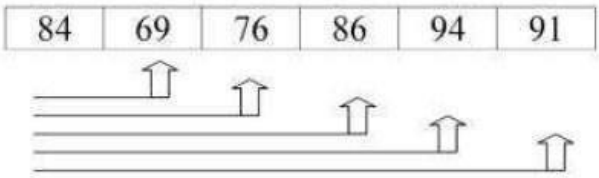


Gambar 5. Proses ke-5 algoritma Bubble Sort

Algoritma Bubble Sorting mudah dalam sintaks, tetapi lebih lambat dibandingkan dengan algoritma sorting yang lain

EXCHANGE SORT

Sangat mirip dengan Bubble Sort, dan banyak yang mengatakan Bubble Sort sama dengan Exchange Sort. Pebedaan ada dalam hal bagaimana membandingkan antar elemen-elemennya. Exchange sort membandingkan suatu elemen dengan elemen-elemen lainnya dalam array tersebut, dan melakukan pertukaran elemen jika perlu. Jadi ada elemen yang selalu menjadi elemen pusat (pivot). Sedangkan Bubble sort akan membandingkan elemen pertama/terakhir dengan elemen sebelumnya/sesudahnya, kemudian elemen sebelum/sesudahnya itu akan menjadi pusat (pivot) untuk dibandingkan dengan elemen sebelumnya/sesudahnya lagi, begitu seterusnya.



Proses 1

Pivot (Pusat)

84	69	76	86	94	91
84	69	76	86	94	91
84	69	76	**86**	94	91
86	69	76	**84**	**94**	91
94	69	76	84	**86**	91
94	69	76	84	86	91

Proses 2

Pivot (Pusat)

94	69	76	84	86	91
94	76	69	84	86	91
94	84	69	76	86	91
94	86	69	76	84	91
94	91	69	76	84	86

Proses 3

Pivot (Pusat)

94	91	69	76	84	86
94	91	76	69	84	86
94	91	84	69	76	86
94	91	86	69	76	84

Proses 4

Pivot (Pusat)

94	91	86	69	76	84
94	91	86	76	69	84
94	91	86	84	69	76

Proses 5

Pivot (Pusat)

94	91	86	84	69	76
94	91	86	84	76	69

Gambar 6. Proses algoritma Exchange Sorting

SELECTION SORT

Merupakan kombinasi antara sorting dan searching. Untuk setiap proses, akan dicari elemen-elemen yang belum diurutkan yang memiliki nilai terkecil atau terbesar akan dipertukarkan ke posisi yang tepat di dalam array. Misalnya untuk putaran pertama, akan dicari data dengan nilai terkecil dan data ini akan ditempatkan di indeks terkecil (data[0]), pada putaran kedua akan dicari data kedua terkecil, dan akan ditempatkan di indeks kedua (data[1]). Selama proses, pembandingan dan pengubahan hanya dilakukan pada indeks pembanding saja, pertukaran data secara fisik terjadi pada akhir proses.

Proses 1

0	1	2	3	4	5
32	75	69	58	21	40

Pembanding	**Posisi**
32 < 75	0
32 < 69	0
32 < 58	0
32 > 21 (**tukar idx**)	4
21 < 40	4

Tukar data ke-0 (**32**) dengan data ke-4 (**21**)

0	1	2	3	4	5
21	75	69	58	32	40

Proses 2

0	1	2	3	4	5
21	**75**	69	58	32	40

Pembanding	**Posisi**
75 > 69 (**tukar idx**)	2
69 > 58 (**tukar idx**)	3
58 > 32 (**tukar idx**)	4
32 < 40	4

Tukar data ke-1 (**75**) dengan data ke-4 (**32**)

0	1	2	3	4	5
21	32	69	58	75	40

Proses 3

0	1	2	3	4	5
21	32	69	58	75	40

Pembanding	**Posisi**
69 > 58 (**tukar idx**)	3
58 < 75	3
58 > 40	5

Tukar data ke-2 (**69**) dengan data ke-5 (**40**)

0	1	2	3	4	5
21	32	40	58	75	69

Proses 4

0	1	2	3	4	5
21	32	40	58	75	69

Pembanding	**Posisi**
58 < 75	3
58 < 69	3

Tukar data ke-3 (**58**) dengan data ke-3 (**58**)

0	1	2	3	4	5
21	32	40	58	75	69

Proses 5

0	1	2	3	4	5
21	32	40	58	75	69

Pembanding	**Posisi**
75 > 69	5

Tukar data ke-4 (**75**) dengan data ke-5 (**69**)

0	1	2	3	4	5
21	32	40	58	69	75

Gambar 7. Proses algoritma Selection Sorting

INSERTION SORT

Mirip dengan cara orang mengurutkan kartu, selembar demi selembar kartu diambil dan disisipkan (insert) ke tempat yang seharusnya. Pengurutan dimulai dari data ke-2 sampai dengan data terakhir, jika ditemukan data yang lebih kecil, maka akan ditempatkan (diinsert) diposisi yang seharusnya. Pada penyisipan elemen, maka elemen-elemen lain akan bergeser ke belakang.

Proses 1

0	1	2	3	4	5
22	**10**	15	3	8	2

Temp	Cek	Geser
10	Temp<22?	Data ke-0 ke posisi 1

Temp menempati posisi ke -0

0	1	2	3	4	5
10	22	15	3	8	2

Proses 2

0	1	2	3	4	5
10	22	**15**	3	8	2

Temp	Cek	Geser
15	Temp<22	Data ke-1 ke posisi 2
15	Temp>10	-

Temp menempati posisi ke-1

0	1	2	3	4	5
10	**15**	22	3	8	2

Proses 3

0	1	2	3	4	5
10	15	22	**3**	8	2

Temp	Cek	Geser
3	Temp<22	Data ke-2 ke posisi 3
3	Temp<15	Data ke-1 ke posisi 2
3	Temp<10	Data ke-0 ke posisi 1

Temp menempati posisi ke-0

0	1	2	3	4	5
3	10	15	22	8	2

Proses 4

0	1	2	3	4	5
3	10	15	22	**8**	2

Temp	Cek	Geser
8	Temp<22	Data ke-3 ke posisi 4
8	Temp<15	Data ke-2 ke posisi 3
8	Temp<10	Data ke-1 ke posisi 2
8	Temp>3	-

Temp menempati posisi ke-1

0	1	2	3	4	5
3	**8**	10	15	22	2

Proses 5

0	1	2	3	4	5
3	8	10	15	22	**2**

Temp	Cek	Geser
2	Temp<22	Data ke-4 ke posisi 5
2	Temp<15	Data ke-3 ke posisi 4
2	Temp<10	Data ke-2 ke posisi 3
2	Temp<8	Data ke-1 ke posisi 2
2	Temp<3	Data ke-0 ke posisi 1

Temp menempati posisi ke-0

0	1	2	3	4	5
2	3	8	10	15	22

Gambar 9. Proses algoritma Insertion Sorting