PENGURUTAN

- Yaitu proses pengaturan sekumpulan objek menurut urutan atau susunan tertentu
- Acuan pengurutan dibedakan menjadi :

1. Ascending / menaik

   Syarat : $L[1] \leq L[2] \leq L[3] \leq \ldots \leq L[N]$

2. Descending / menurun

   Syarat : $L[1] \geq L[2] \geq L[3] \geq \ldots \geq L[N]$

- Pengurutan dibedakan menjadi
  - Pengurutan Internal / Pengurutan Array
    - Yaitu pengurutan terhadap sekumpulan data yang disimpan dalam memori utama komputer
    - Umumnya struktur yang digunakan adalah Array
    - Prosesnya lebih cepat & hasil pengurutan bersifat sementara
  - Pengurutan eksternal / Pengurutan File
    - **Yaitu pengurutan terhadap sekumpulan data yang disimpan dalam memori sekunder (biasanya bervolume besar)**
    - **Struktur yang digunakan adalah File**
    - Prosesnya lebih lama tapi hasilnya bersifat tetap

Beberapa teknik pengurutan data yang sering digunakan

- Bubble Sort
- Selection Sort
- Insertion Sort
- Shell sort
- Quick sort
- Heap sort
- Merge sort
- Radix sort
- Tree sort
- Binary sort

Bubble sort

- Diinspirasi oleh gelembung sabun yang ada dipermukaan air, dimana benda yang berat akan terbenam dan yang ringan akan terapung
- Bila pengurutan dengan acuan ascending : elemen yang bernilai besar akan “dibenamkan” melalui proses pembandingan antar elemen yang bersebelahan dan proses pertukaran
  - Proses pertukaran dilakukan sebanyak N-1 langkah, dimana N adalah ukuran array
  - Pada akhir setiap langkah ke I, array L[1..N] akan terdiri atas dua bagian yaitu :
    - Yang sudah terurut
    - Yang belum terurut
  - Setelah langkah terakhir diperoleh array L[1..N] yang terurut ascending

- Contoh : diurutkan secara ascending, N = 6

Lokasi	1	2	3	4	5	6
Data	25	27	10	8	76	21

Langkah 5

Lokasi	1	2	3	4	5	6
Awal	25*	27*	10	8	76	21
1	25	27*	10*	8	76	21
2	25	10	27*	8*	76	21
3	25	10	8	27*	76*	21
4	25	10	8	27	76*	21*
5	25	10	8	27	21	**76**

Langkah 4

Lokasi	1	2	3	4	5	6
Awal	25*	10*	8	27	21	**76**
1	10	25*	8*	27	21	**76**
2	10	8	25*	27*	21	**76**
3	10	8	25	27*	21*	**76**
4	10	8	25	21	**27**	**76**

Langkah 3

Lokasi	1	2	3	4	5	6
Awal	10*	8*	25	21	**27**	**76**
1	8	10*	25*	21	**27**	**76**
2	8	10	25*	21*	**27**	**76**
3	8	10	21	**25**	**27**	**76**

Langkah 2

Lokasi	1	2	3	4	5	6
Awal	8*	10*	21	**25**	**27**	**76**
1	8	10*	21*	**25**	**27**	**76**
2	8	10	**21**	**25**	**27**	**76**

Langkah 1

Lokasi	1	2	3	4	5	6
Awal	8*	10*	21	25	**27**	**76**
1	8	**10**	**21**	**25**	**27**	**76**

```
Algoritma:
Deklarasi
        I       : bilangan bulat   {untuk langkah}
        J       : bilangan bulat   {indek}
        Temp : bilangan bulat  {untuk penampung sementara}
        L               : Array [1 ..N]
        N               : bilangan bulat  {jumlah elemen array}
Deskripsi
        For I ← (N-1) downto  1 do
    For J ← 1 to I do
                                If  L[J] > L [J+1]  then
                                        Temp ← L[J]
                                        L[J] ← L[J+1]
                                        L[J+1] ← temp
                                Endif
                        Endfor
        Endfor
```

Selection sort

- Yaitu memilih nilai yang maksimum/minimum dari suatu array yang akan diurutkan dan menempatkannya pada posisi awal atau akhir array; selanjutnya elemen tersebut diisolasi dan tidak disertakan pada proses berikutnya. Hal ini dilakukan secara terus menerus sampai sebanyak N-1
- Dibedakan menjadi :
  - Algoritma pengurutan maksimum

    Yaitu memilih elemen maksimum sebagai basis pengurutan
  - Algoritma pengurutan minimum

    Yaitu memilih elemen minimum sebagai basis pengurutan

Contoh : Diurutkan secara ascending dengan algoritma pengurutan minimum

Lokasi	1	2	3	4	5	6
Data	25	27	10	8	76	21

Langkah/ Lokasi	1	2	3	4	5	6
1	***8***	27	10	25*	76	21
2	***8***	***10***	27*	25	76	21
3	***8***	***10***	***21***	25	76	27*
4	***8***	***10***	***21***	***25****	76	27
5	***8***	***10***	***21***	***25***	***27***	76*

```
Algoritma :
Deklarasi :
        I       : bilangan bulat   {untuk langkah}
        J       : bilangan bulat   {indek}
        Temp : bilangan bulat  {untuk penampung sementara}
        L               : Array [1 ..N]
        N               : bilangan bulat  {jumlah elemen array}
        K               : Bilangan bulat  {menampung indek nilai terkecil}
X       : Bilangan bulat  {menampung nilai terkecil}


Deskripsi :
        For I ← 1 to (N-1) do
            K ← I
            X ← L[I]
            For J ← ( I+1) to N do
                        If  L[J] < X then
                                    K ← J
                                    X ← L [J]
                        Endif
            Endfor
            Temp ← L[I]
            L[I] ← X
            L[K] ← temp
        Endfor
```

Insertion sort / Sinking Sort / Sifting Sort

- Yaitu metode pengurutan dengan cara menyisipkan elemen array pada posisi yang tepat
- Pada prinsipnya seperti permainan kartu : ambil kartu pertama & pegang, ambil kartu kedua dan letakkan pada posisi yang tepat / berurut, ambil kartu ketiga letakkan pada posisi yang berurut (biasa diawal, ditengah atau diakhir) dst

Contoh :

Lokasi	1	2	3	4	5	6
Data	25	27	10	8	76	21

Langkah/ Lokasi	1	2	3	4	5	6
1	25					
2	25	27				
3	10	25	27			
4	8	10	25	27		
5	8	10	25	27	76	
6	8	10	21	25	27	76

Deklarasi

```
I       : Bilangan bulat   {untuk langkah}
J       : Bilangan bulat   {untuk penelusuran array}
ketemu  : boolean {untuk menyatakan posisi penyisipan ditemukan}
x       : Bilangan bulat {tempat sementara agar L[K] tidak ditimpa
                  selama pergeseran }
```

Deskripsi

```
For I ← 2 to N do
        X ← L[I]
        J ← I – 1
        Ketemu ← False
        While (J ≥ 1) and (not ketemu) do
                If  X < L[J] then
                        L[J+1] ← L[J]
                        J ← J-1
                Else ketemu ← true
                Endif
        Endwhile
        L[J+1] ← X
Endfor
```

Shell sort

- Pada prinsipnya sama dengan bubble sort yaitu membandingkan elemen array dan melakukan proses penukaran; bedanya kalau bubble sort elemen yang dibandingkan adalah elemen yang bersebelahan sedangkan pada shell sort elemen yang dibandingkan mempunyai jarak tertentu
- Langkah pembandingan pertama berjarak *N div 2*, langkah kedua berjarak : *jarak langkah perbandingan pertama div 2* demikian seterusnya sampai jarak perbandingan sama dengan satu. Bila jarak sama dengan satu maka prosesnya sama dengan Bubble sort

Urutkan secara ascending

Lokasi	1	2	3	4	5	6
Data	25	27	10	8	76	21

Loncat = 6 div 2 = 3

Lokasi	1	2	3	4	5	6
Data	25*	27	10	8*	76	21
	8	27*	10	25	76*	21
	8	27	10*	25	76	21*

Loncat = 3 div 2 = 1

Lokasi	1	2	3	4	5	6
	8*	27*	10	25	76	21
1	8	27*	10*	25	76	21
2	8	10	27*	25*	76	21
3	8	10	25	27*	76*	21
4	8	10	25	27	76*	21*
5	8	10	25	27	21	**76**

	8*	10*	25	27	21	**76**
1	8	10*	25*	27	21	**76**
2	8	10	25*	27*	21	**76**
3	8	10	25	27*	21*	**76**
4	8	10	25	21	**27***	**76***

	8*	10*	25	21	**27**	**76**
1	8	10*	25*	21	**27**	**76**
2	8	10	25*	21*	**27**	**76**
3	8	10	21	**25**	**27**	**76**

	8*	10*	21	**25**	**27**	**76**
1	8	10*	21*	**25**	**27**	**76**
2	8	10	**21**	**25**	**27**	**76**

	8*	10*	**21**	**25**	**27**	**76**
1	8	**10**	**21**	**25**	**27**	**76**

Algoritma :
Procedure swap(P,Q : bilangan pecahan)
Deklarasi
Temp : bilangan pecahan
Deskripsi
    Temp ← P
    P ← Q
    Q ← temp
Return

Algoritma utama:
Deklarasi
    Loncat : bilangan bulat
    N : Bilangan Bulat
    Kondisi : Boolean
    J : bilangan bulat
    I : bilangan bulat
    L : array [1 .. N]

```
Deskripsi
        Loncat ← N
        While loncat > 1 do
                Loncat ← loncat div 2
                Repeat
                        Kondisi ← true
                        For J ← 1 to (N- loncat) do
                                I ← J + Loncat
                                If L [J] > L [I] then
                                        Swap (L[J], L[I])
                                        Kondisi ← false
                                Endif
                        Endfor
                Until kondisi
        Endwhile
```

Quick sort

- Langkah :
  a. Elemen 1 pindahkan pada penampung yang berfungsi sebagai patokan
  b. Dari kanan ke kiri secara berurutan cari nilai yang lebih kecil dari patokan dan pindahkan nilainya pada posisi patokan
  c. Dari kiri ke kanan secara berurutan cari nilai yang lebih besar dari patokan dan pindahkan pada posisi data terakhir yang dipindah
  d. Lakukan langkah b dan c sampai posisi patokan ditemukan, yaitu
     - Bila posisi yang ditinggalkan dan yang ditempati saling bersebelahan
     - Bila dari kanan ke kiri tidak ada lagi data yang lebih kecil dari patokan
     - Bila dari kiri ke kanan tidak ada lagi data yang lebih besar dari patokan

Urutkan secara ascending

Lokasi	1	2	3	4	5	6
Data	25	27	10	8	76	21

Patokan = 25

Lokasi	1	2	3	4	5	6
kn	*	27	10	8	76	**21**
kr	21	**27**	10	8	76	*
kn	21	*	10	**8**	76	27
kr	21	8	10	*	76	27
	21	8	10	**25**	76	27

Patokan = 21

Lokasi	1	2	3	4	5	6
	*	8	**10**	**25**	76	27
kn	10	8	*			
kr	10	8	**21**			

Patokan = 10

Lokasi	1	2	3	4	5	6
	*	**8**	**21**	**25**	76	27
kn	8	**10**				

Patokan = 76

Lokasi	1	2	3	4	5	6
	8	**10**	**21**	**25**	*	27
kn					27	**76**

Hasil

Lokasi	1	2	3	4	5	6
hasil	8	**10**	**21**	**25**	27	**76**

Algoritma :

Procedure swap(P,Q : bilangan pecahan)

Deklarasi

Temp : bilangan pecahan

Deskripsi

Temp ← P

P ← Q

Q ← temp

Return

```
Procedure bagi ( I, J , M : bilangan bulat)
Deklarasi
        Saklar : bilangan bulat
Deskripsi
        Saklar ← -1
        While I < J do
                If  A[I] > A[J] then
                        Swap( A[I} , A[J])
                        Saklar ← - saklar
                Endif
                If saklar ← -saklar then J ← J-1
                Else I ← I +1
                Endif
                M ← I
        Endwhile
Return
```

```
Procedure quicksort (L,R : bilangan bulat)
deklarasi
        K, mid : bilangan bulat;
Deskripsi
        Write (‘L = ‘, L:2, ‘, R = ‘, R:2, ‘ : ‘);
        For K ← L to R do
                Write (A[K] : 4)
                If  L >= R then
                                If A[L] > A[R] then
swap (A[L], A[R])
                                endif
                else
                                bagi(L,R,Mid)
                                call Quicksort(L,Mid-1)
                                Call quicksort(Mid+1, R)
                Endif
        Endfor
Return
```

```
Algoritma utama
Deklarasi
        A : Array [1..N]
        I, N            : bilangan bulat
Deskripsi
        Write('Jumlah komponen = ')
        Readln(N)
        For I ← 1 to N do
Read(A[I])
        endfor
        Call Quicksort(1,N)
        For I ← 1 to N do
write(A[I]:4)
        endfor
```

Radix Sort

a. Susun bilangan secara vertikal
b. Lakukan penambahan 0 didepan bilangan yang digitnya kurang (samakan digitnya)
c. lakukan pengurutan secara vertikal mulai dari digit pertama dari kiri dan lakukan pengelompokan
d. Dalam tiap-tiap kelompok, lakukaan pengurutan secara vertikal untuk digit ke dua dari kiri, Bila dalam pengelompokan hanya terdiri dari satu data maka posisinya tetap (tidak perlu diurutkan)
e. Lakukan langkah d sampai dengan digit terakhir. Hasil pengurutan adalah hasil pengelompokan digit terakhir

Lokasi	1	2	3	4	5	6
Data	25	17	10	8	36	21

Data awal
25
17
10
08
36
21

Digit 1 dari kiri
08
17
10
25
21
36

Digit 2 dari kiri
08
10
17
21
25
36

Binary Sort

a. Ubah bilangan desimal ke dalam biner
b. Susun secara vertikal
c. Lakukan penambahan 0 didepan bilangan yang digitnya kurang (samakan digitnya)
d. lakukan pengurutan secara vertikal mulai dari digit pertama dari kiri dan lakukan pengelompokan
e. Dalam tiap-tiap kelompok, lakukaan pengurutan secara vertikal untuk digit ke dua dari kiri, Bila dalam pengelompokan hanya terdiri dari satu data maka posisinya tetap (tidak perlu diurutkan)
f. Lakukan langkah e sampai dengan digit terakhir. Hasil pengurutan adalah hasil pengelompokan digit terakhir

Contoh : Urutkan secara ascending

data : 8,5,4,2,10,7

Ubah ke dalam bilangan binary dan lakukan pengurutan dari digit 1 dari kiri

8	=	0	1	0	1
5	=	0	1	0	0
4	=	0	0	1	0
2	=	0	1	1	1
10	=	1	0	0	0
7	=	1	0	1	0

Tanpa mengubah urutan digit 1 dari kiri, lakukan pengurutan digit 2 dari kiri

0	0	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	0	0	0
1	0	1	0

Tanpa mengubah urutan digit 1 dan 2 dari kiri, lakukan pengurutan digit 3 dari kiri

0	0	1	0
0	1	0	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0

Tanpa mengubah urutan digit 1,2 dan 3 dari kiri, lakukan pengurutan digit 4 dari kiri

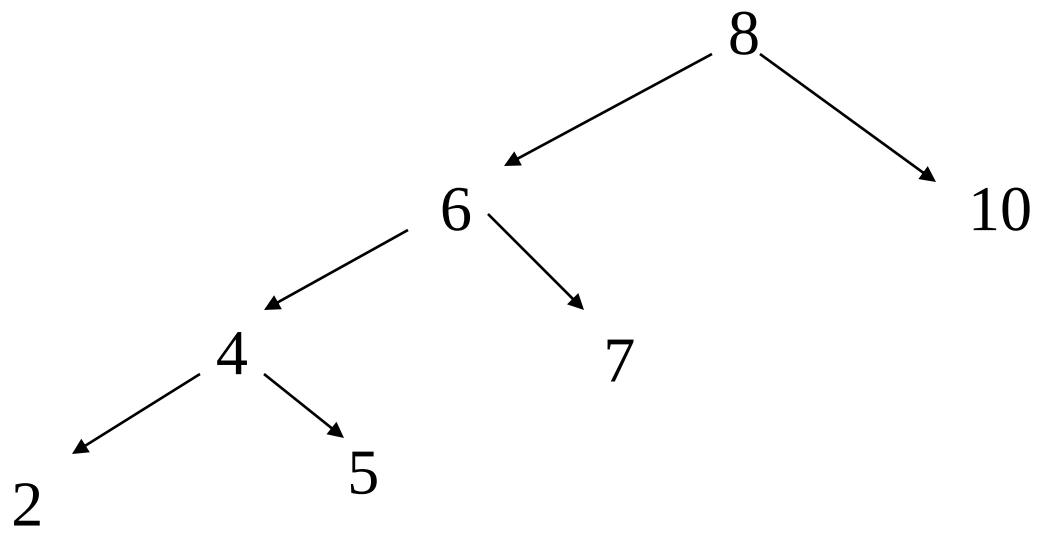
0	0	1	0	=	2
0	1	0	0	=	4
0	1	0	1	=	5
0	1	1	1	=	7
1	0	0	0	=	8
1	0	1	0	=	10

Hasil Pengurutan secara ascending = 2, 4, 5, 7, 8, 10

Three sort

a. Ambil data pertama dan tempatkan sebagai root
b. Ambil data ke dua dan bandingkan dengan root, bila nilai data kedua lebih kecil dari root maka tempatkan sebagai anak kiri root, bila lebih besar tempatkan sebagai anak kanan root
c. Ambil data berikutnya, bandingkan dengan root bila lebih kecil dari root bandingkan dengan anak kiri , bila lebih kecil maka akan menjadi anak kiri dari anak kiri root. begitu juga untuk anak kanannya
d. Lakukan langkah c sampai data terakhir
e. Urutan pembacaannya (bila ascending) adalah anak kiri yang paling kiri, root, anak kanan paling kiri, anak kanan

Contoh : Urutkan secara ascending data : 8,6,4,5, 2,10,7



Hasil : 2, 4, 5, 7, 8, 10

Anak kiri yang paling kiri – root – anak kanan yang paling kiri – root – anak kanan – root - dst