



# Pertemuan Ke-14 (SORTING)

---

# Contoh Program Bubble Sort

```
1 Program BubbleSort;
2 uses crt;
3 var L : array [1..100] of integer;
4     i,j,k,n,temp : integer;
5 Begin
6 Clrscr;
7     write('Barapa jumlah elemen array?');readln(n);
8     for i := 1 to n do
9     begin
10        write('L[' ,i, ' ] = ');readln(L[i]);
11    end;
12    writeln('Proses Pengurutan Bubble Sort : ');
13    for i := 1 to n-1 do
14    begin
15        for k := n downto i+1 do
16        begin
17            if L[k] < L[k-1] then
18            begin
19                temp := L[k];
20                L[k] := L[k-1];
21                L[k-1] := temp;
22            end;
23        end;
24
25        {cetak array tiap langkah pengurutan}
26        writeln;
27        writeln;
28        write('Hasil Pengurutan Bubble Sort :');
29        for i := 1 to n do
30            write(L[i], ' ');
31        readln;
32    end.
```

# Contoh Program Gavitasi Sort

```
1 Program Gravitasi_Sort;  
2 Uses crt;  
3 Var  
4     i,j,k,n,temp : integer;  
5     A : Array [1..100] of integer;  
6 begin  
7     Clrscr;  
8     write('Banyaknya Elemen Array : ');readln(n);  
9  
10    {inputData}  
11    for i := 1 to n do  
12    begin  
13        write('Elemen ke-i',i,': ');readln(A[i]);  
14    end;  
15
```

# Contoh Program Gavitasi Sort

```
16 {cetak array sebelum pengurutan}
17 writeln;textcolor(15);
18 write('Sebelum diurutkan : ');
19 for i := 1 to n do
20     write(A[i],' ');
21     writeln; textcolor(15);
22     writeln; writeln('Proses pengurutan Gravitasi: ');
23     for i := 1 to n-1 do
24         begin
25             for j := 1 to n-i do
26                 begin
27                     if A[j] > A[j+1] then
28                         begin
29                             temp := A[j+1];
30                             A[j+1] := A[j];
31                             A[j] := temp;
32                         end;
33                 end;
34
35                 writeln;textcolor(i);
36                 {cetak array tiap langkah pengurutan}
37                 write('Hasil akhir langkah ke-',i,' : ');
38                 for k := 1 to n do
39                     write(A[k],' ');
40             end;
41
```

# Contoh Program Gavitasi Sort

```
41
42     {cetak array setelah pengurutan}
43     writeln;writeln;
44     textcolor(15);write('hasil Pengurutan Gravitasi: ');
45     for i := 1 to n do
46         write(A[i], ' ');
47     writeln;writeln;textcolor(10);
48     readln;
49 end.
```

# Contoh Program Seleksi Maks (Desc)

```
1 Program Selection_Max_Desc;
2 Uses Crt;
3 Var
4     L : array [1..100] of integer;
5     i,j,k,n,imaks,temp : integer;
6
7 Begin
8     Clrscr;
9     Write('Berapa banyaknya elemen array L?');readln(n);
10    for i := 1 to n do
11        Begin
12            Write('L[',i,'] = ');readln(L[i]);
13        End;
14
15    writeln;
16    writeln('Proses Pengurutan Seleksi-Maksimun-
17    Descending:')
18    For i := 1 to n-1 do
19        Begin
20            Imaks := i;
21            For j := i+1 to n do;
22                If L[j] > L[imaks] then
23                    imaks := j;
24            temp := L[imaks];
25            L[imaks] := L[i];
26            L[i] := temp;
27
```

# Contoh Program Seleksi Maks (Desc)

```
34 writeln;
35 writeln;
36     Write('Hasil Pengurutan Insert Sort :');
37     For i := 1 to n do
38         Write(L[i], ' ');
39     Readln;
40 End.
```

# Procedure Insertion Sort

```
10 Procedure Insertion_Sort(n,i,j,x : integer; var data : larikint);
11 begin
12     for i := 2 to n do
13     begin
14         x := data[i];
15         j := i-1;
16         ketemu := false;
17
18         while (j>=1) and (not ketemu) do
19         begin
20             if x < data[j] then
21             begin
22                 data[j+1] := data[j];
23                 j := j-1;
24             end
25             else
26                 ketemu := true;
27             end;{end while}
28
29             data[j+1] := x;
30         end;
31     end;
```




# Contoh Program Insertion Sort

```
1  Program InsertSort;
2  uses crt;
3  const max = 1000;
4    type larikint = array [1..max] of integer;
5  var
6    data : larikint;
7    n,i,j,x :integer;
8    ketemu : boolean;
9
10 Procedure Insertion_Sort(n,i,j,x : integer; var data : larikint);
11 begin
12     for i := 2 to n do
13     begin
14         x := data[i];
15         j := i-1;
16         ketemu := false;
17
18         while (j>=1) and (not ketemu) do
19         begin
20             if x < data[j] then
21             begin
22                 data[j+1] := data[j];
23                 j := j-1;
24             end
25             else
26                 ketemu := true;
27             end;{end while}
28
29             data[j+1] := x;
30         end;
31     end;
```

# Contoh Program Insertion Sort

```
33 Begin
34 Clrscr;
35     write('Masukkan Jumlah Data : ');readln(n);
36     writeln;
37     for i := 1 to n do
38     begin
39         write (' Data Ke ',i,' : ');readln(data[i]);
40     end;
41     WriteLn;
42     write ('Sebelum Di Urutkan : ');
43     for i := 1 to n do
44         Write (data[i],' ');
45
46     Insertion_Sort(n,i,j,x,data);
47     WriteLn;
48     writeln;
49     writeln;
50     write ('Hasil Pengurutan penyisipan langsung : ');
51     for i := 1 to n do
52         write(data[i],' ');
53     ReadLn;
54 end.
```

# Output Program Insertion Sort

 Free Pascal

```
Masukkan Jumlah Data : 8
```

```
Data ke 1 : 34
```

```
Data ke 2 : 29
```

```
Data ke 3 : 57
```

```
Data ke 4 : 14
```

```
Data ke 5 : 38
```

```
Data ke 6 : 66
```

```
Data ke 7 : 45
```

```
Data ke 8 : 52
```

```
Sebelum Diurutkan :34 29 57 14 38 66 45 52
```

```
Hasil Pengurutan Insertion Sort : 14 29 34 38 45 52 57 66
```

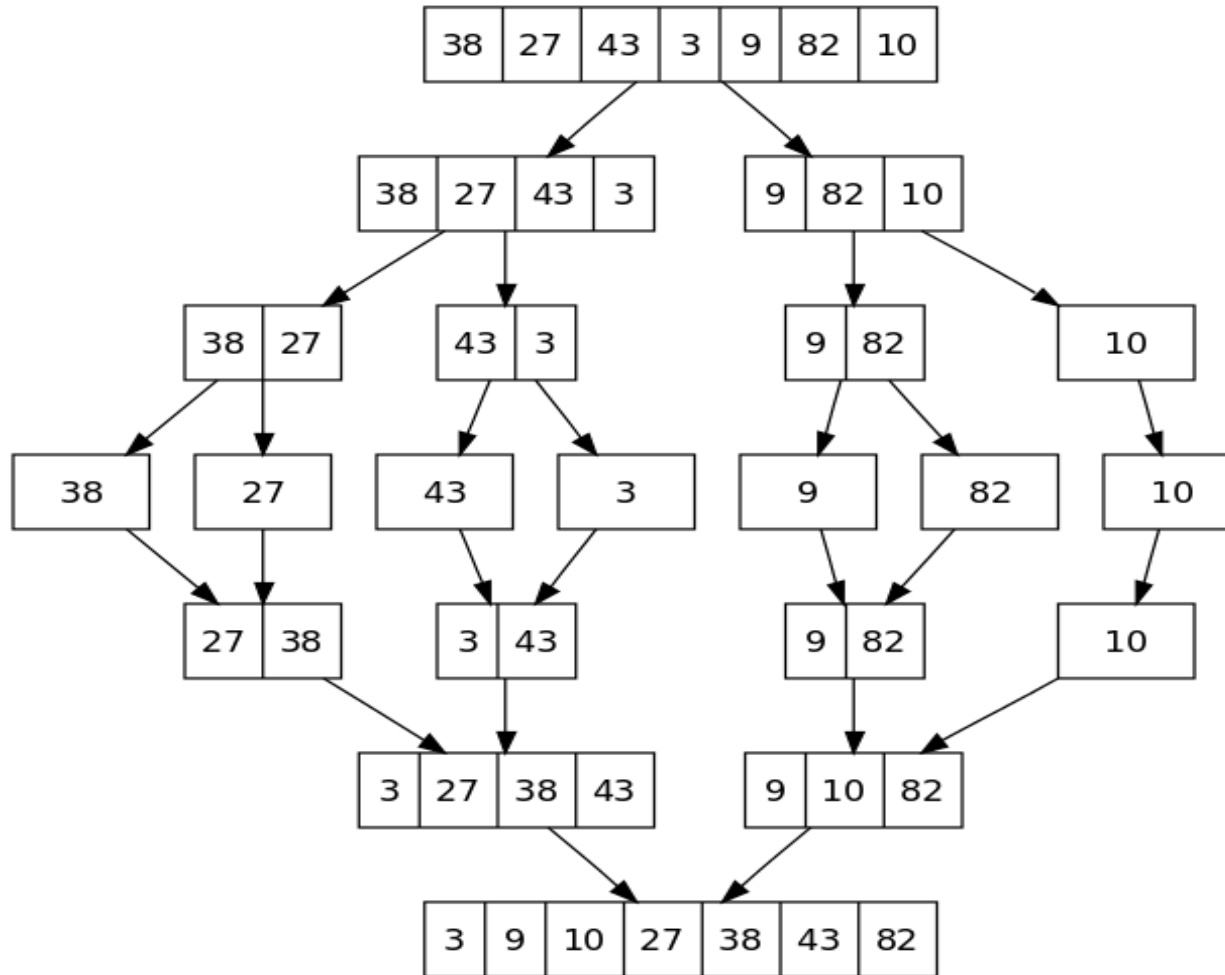
# Merge Sort

- MergeSort adalah algoritma yang berdasarkan strategi divide-and-conquer.
- Algoritma ini terdiri dari dua bagian utama, yaitu bagian pembagian list menjadi sublist-sublist yang lebih kecil dan bagian sort (pengurutan) dan merge (penggabungan) pada sublist-sublist tersebut.

# Merge Sort

- *Divide*: membagi masalah menjadi beberapa submasalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama),
- *Conquer*: memecahkan (menyelesaikan) masing-masing submasalah (secara rekursif), dan
- *Combine*: menggabungkan solusi masing-masing submasalah sehingga membentuk solusi masalah semula.

# Contoh Merge Sort



# Cara kerja *Merge sort*

- ***Divide:***

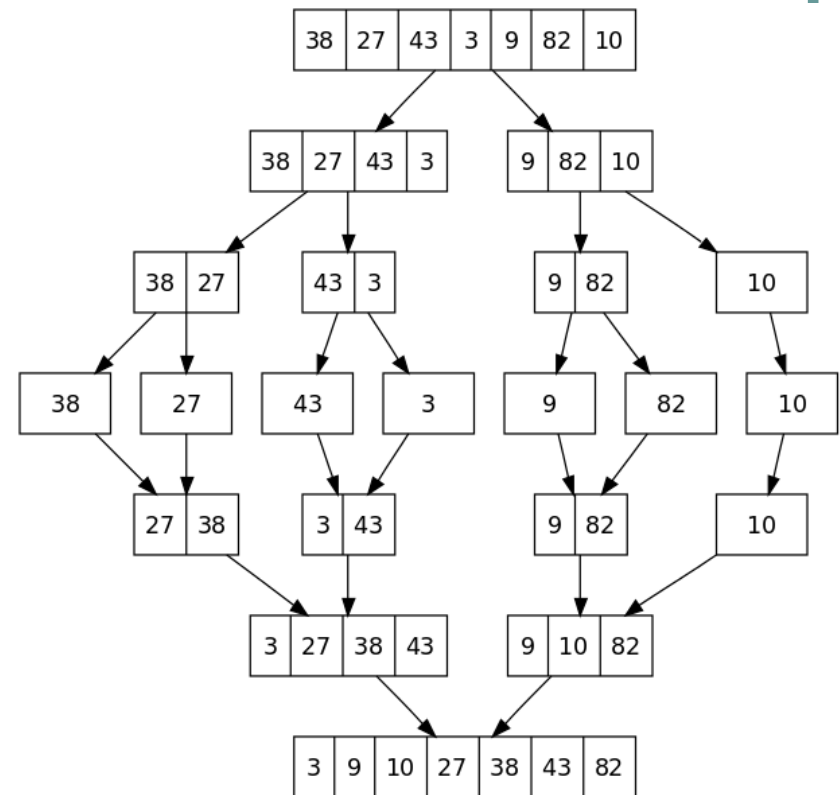
membagi sekuens dg n elemen menjadi 2 subsekuens masing-masing dgn  $n/2$  elemen.

- ***Conquer:***

mengurutkan 2 subsekuens tersebut secara rekursif menggunakan *merge sort*.

- ***Combine:***

menggabungkan 2 subsekuens yang telah diurutkan untuk menghasilkan output yang dikehendaki.



# Algoritma Merge Sort

procedure mergesort(input/output a : tabelint, input i, j : integer)

*{ mengurutkan tabel L[i..j] dengan algoritma merge sort masukan: tabel L dengan n elemen*

*keluaran: tabel a yang terurut }*

**deklarasi:**

k : integer

**algoritma:**

if i < j then { ukuran(L) > 1 }

k ← (i+j) div 2

mergesort(L, i, k)

mergesort(L, k+1, j)

merge(L, i, k, j)

endif

endprocedure



# Contoh Program Merge Sort

```
1 Program ContohMergeSort;  
2 uses crt;  
3 type arr = array [1..100] of integer;  
4  
5 var  
6 ArrMain, ArrUrut : arr;  
7 n, m : integer;  
8
```


```
9 Procedure Merge(Left:arr; pjgL:integer; Right : arr; pjgR:integer; var ArrMerge : arr);
10 var
11     i,j,k,m,panjang: integer;
12     hasil : arr;
13 begin
14     i:=1;
15     j:=1;
16     k:=1;
17     panjang:=pjgL+pjgR;
18     while ((pjgL>0) and (pjgR>0)) do
19     begin
20         if(Left[i]<= Right[j]) then
21         begin
22             hasil[k]:=Left[i];
23             i:=i+1;
24             k:=k+1;
25             pjgL:=pjgL-1;
26         end
27         else
28         begin
29             hasil[k]:=Right[j];
30             j:=j+1;
31             k:=k+1;
32             pjgR:=pjgR-1;
33         end;
34     end;
35
```

```
36 while (pjpgL>0) do
37 begin
38     hasil[k]:=Left[i];
39     i:=i+1;
40     k:=k+1;
41     pjpgL:=pjpgL-1;
42 end;
43
44 while (pjpgR>0) do
45 begin
46     hasil[k]:=Right[j];
47     j:=j+1;
48     k:=k+1;
49     pjpgR:=pjpgR-1;
50 end;
51
52 {Mengembalikan nilai ke variabel output ArrMerge}
53 ArrMerge:=hasil;
54 for m:= 1 to panjang do
55     writeln('Array Merge ke-',m,' : ',hasil[m]);
56 end;
```


```
58 Procedure Mergesort (pjtg:integer;A : arr;var ArrHasil : arr);
59 var
60     middle,i,pjtgLeft,pjtgRight : integer;
61     ArrLeft,ArrRight : arr;
62 begin
63     if pjtg <= 1 then
64         ArrHasil := A
65     else
66     begin
67         middle := pjtg div 2;
68         for i:=1 to middle do
69             ArrLeft[i]:=A[i];
70
71         for i:=(middle+1) to pjtg do
72             ArrRight[i-middle]:=A[i];
73             pjtgLeft := pjtg div 2;
74             pjtgRight := (pjtg+1) div 2;
75
76         for m:= 1 to pjtgLeft do
77             writeln('Array Left ke-',m,' : ',ArrLeft[m]);
78
79         for m:= 1 to pjtgRight do
80             writeln('Array Right ke-',m,' : ',ArrRight[m]);
81             mergesort (pjtgLeft,ArrLeft,ArrLeft);
82             mergesort (pjtgRight,ArrRight,ArrRight);
83             merge (ArrLeft,pjtgLeft,ArrRight,pjtgRight,ArrHasil);
84     end;
85 end;
```

```
87 begin
88 clrscr;
89 write('Jumlah array : ');readln(n);
90     for m := 1 to n do
91         begin
92             write('Array ke-',m,' : ');
93             readln(ArrMain[m]);
94         end;
95 writeln;
96 writeln('LANGKAH MERGE SORT');
97 writeln('-----');
98 mergesort(n,ArrMain,ArrUrut);
99 writeln;
100 writeln('HASIL ARRAY TERURUT');
101 writeln('-----');
102 for m:= 1 to n do
103     writeln('Array Urut ke-',m,' : ',ArrUrut[m]);
104 readln;
105 end.
```

# Output Program Merge Sort

 Free Pascal

```
Jumlah Array : 7  
Array ke-1 : 38  
Array ke-2 : 27  
Array ke-3 : 43  
Array ke-4 : 3  
Array ke-5 : 9  
Array ke-6 : 82  
Array ke-7 : 10
```

 Free Pascal

```
Array Merge ke-2 : 68  
Array Left ke-1 : 3  
Array Right ke-1 : 9  
Array Merge ke-1 : 3  
Array Merge ke-2 : 9  
Array Left ke-1 : 82  
Array Right ke-1 : 10  
Array Merge ke-1 : 10  
Array Merge ke-2 : 82  
Array Merge ke-1 : 3  
Array Merge ke-2 : 9  
Array Merge ke-3 : 10  
Array Merge ke-4 : 82  
Array Merge ke-1 : 3  
Array Merge ke-2 : 9  
Array Merge ke-3 : 10  
Array Merge ke-4 : 27  
Array Merge ke-5 : 38  
Array Merge ke-6 : 43  
Array Merge ke-7 : 82
```

Hasil Array Terurut

```
-----  
Array urut ke-1 : 3  
Array urut ke-2 : 9  
Array urut ke-3 : 10  
Array urut ke-4 : 27  
Array urut ke-5 : 38  
Array urut ke-6 : 43  
Array urut ke-7 : 82
```

# Quick Sort

- Metode Quick sering disebut juga metode partisi (partition exchange sort).
- Metode ini mempunyai efektifitas yang tinggi dengan teknik menukarkan dua elemen dengan jarak yang cukup besar.

# Quick Ascending

```
Procedure Asc_quick(L,R : integer); { Prosedur ascending }
```

```
Var
```

```
  i, j : integer;
```

```
  begin
```

```
    if L < R then
```

```
      begin
```

```
        i := L ; j := R+1;
```

```
        repeat
```

```
          repeat inc(i) until data[ i ] >= data [ l ];
```

```
          repeat dec(j) until data[ j ] <= data [ l ];
```

```
          if i < j then TukarData (data[i], data[j]);
```

```
        until i > j;
```

```
        TukarData(data[ l ], data [ j ]);
```

```
        Asc_quick(L, j-1);
```

```
          Asc_quick(j+1, R);
```

```
      End;
```

```
End;
```



# Contoh Program Quick-Sort

```
1 Program QuickSort;
2 uses crt;
3 type Larik = array [1..100] of integer;
4
5 Procedure INPUT(var L : Larik; var n : integer);
6 var i : integer;
7 begin
8     write('Jumlah data ; '); readln(n);
9     for i := 1 to n do
10        begin
11            write('L[' , i, ' ] = '); Readln(L[i]);
12        end;
13        writeln;
14    end;
15
16 Procedure TUKAR(var a,b : integer);
17 var t : integer;
18 begin
19     t := a;
20     a := b;
21     b := t;
22 end;
23
```

# Contoh Program Quick-Sort

```
24 Procedure CETAK(L : Larik; c: integer);
25 var i : integer;
26 begin
27     for i := 1 to c do
28         write(L[i]:5);
29         writeln;
30 end;
31
32 Procedure quicksort(var L:Larik; a,b:integer);
33 var
34 a1,b1,pivot: integer;
35 begin
36     a1:=a; b1:=b;
37     pivot:=L[(a+b) div 2];
38     repeat
39         while(L[a1] < pivot) do inc(a1);
40         while(L[b1]>pivot) do dec(b1);
41         if (a1<=b1) then
42             begin
43                 tukar(L[a1],L[b1]);
44                 inc(a1);
45                 dec(b1);
46             end;
47     until (a1>b1);
48     CETAK(L,b);
49     if (a < b1) then quicksort(L, a, b1);
50     if (a1 < b ) then quicksort(L,a1, b);
51 end;
52
```

# Contoh Program Quick-Sort

```
53 {Program Utama}
54 var
55 Data :Larik;
56 n:integer;
57
58 begin
59 clrscr;
60     INPUT (Data,n) ;
61     writeln('Proses Sorting Data');
62     CETAK (data,n) ;
63     quicksort (data,1,n) ;
64     writeln;
65     writeln('Hasil Sorting Data ');
66     cetak (data,n) ;
67     readkey;
68 end.
```

# Tugas

Buatlah program pengurutan dengan menggunakan metode Selection Maximum (Descending)

Thank you!

