Algoritma dan Pemrograman Lanjut

Pertemuan Ke-9
Pengurutan *(Sorting)* 2

Disusun Oleh :
Wilis Kaswidjanti, S.Si.,M.Kom.

Jurusan Teknik Informatika
Fakultas Teknologi Industri
Universitas Pembangunan Nasional "Veteran"
Yogyakarta

Algoritma dan Pemrograman Lanjut

Judul Materi : Pengurutan *(Sorting)* 2

Deskripsi Materi : Materi ini membahas metode sorting tak langsung Metode Shell Sort, Metode Quick Sort dan Metode Merge Sort dengan menggunakan tipe data array dan algoritma rekursif

Tujuan Instruksional Khusus :

1. Memahami dan membandingkan metode sorting menggunakan tipe data array dan algoritma rekursif
2. Mengimplemetasikan penggunaan tipe data array
3. Mendeskripsikan berbagai metode sorting
4. Mengilustrasikan berbagai metode sorting menggunakan tipe data array

Referensi :

- Buku Teks

  Munir, Rinaldi (2005), *Algoritma dan Pemrograman dalam Bahasa Pascal dan C*, Buku 2, Edisi Ketiga, Penerbit Informatika Bandung, , Bab 2, hal 35-76.

  Charibaldi, N. (2004), *Modul Kuliah Algoritma Pemrograman II*, Edisi Kedua, Yogyakarta

- Buku Acuan/Referensi

  Brassard, Gilles (1999), *Fundamentals of algorithma*, PrinteceHall.

  Jarne, Stroustrup B. (1997), *C++ Programming language*, AT &T.

  Kristanto, Andri (2003), *Algoritma pemrograman C++*, Graha Ilmu.

  Schildt,Herbert (2000), *The Complete Reference C++*, McGraw-Hill.

  Sedgewick, R. (2000), *Algoritma Third edition In C part 5*, Addison Wesley.

PENGURUTAN *(SORTING)* 2

PENDAHULUAN

Untuk melakukan proses pengurutan tidak langsung dapat menggunakan beberapa metode :

1. Shell Sort
2. Quick Sort
3. Merge Sort

B. Metode pengurutan tidak langsung :

1. Shell Sort

Disebut juga dengan istilah metode "pertambahan menurun (dimishing increment)" dari Donald L.Shell. Metoda ini memanfaatkan penukaran sepasang elemen untuk mencapai keadaan urut. Dua buah elemen ditukarkan dengan jarak tertentu.

Rumus : ■ Jarak pertama = N div 2

■ Jarak berikutnya = jarak sebelumnya div 2

Contoh :

24	46	11	26	57	38	27	20	17
A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]

Sehingga : ■ Jarak pertama = 9 div 2 = 4

■ Jarak kedua = 4 div 2 = 2

■ Jarak ketiga = 2 div 2 = 1

Ilustrasi

Jarak	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
Awal	**24**	46	11	26	**57**	38	27	20	17
Jarak = 4	24	**46**	11	26	57	**38**	27	20	17
	24	38	**11**	26	57	46	**27**	20	17
	24	38	11	**26**	57	46	27	**20**	17
	24	38	11	20	**57**	46	27	26	**17**
Jarak = 2	**24**	38	**11**	20	17	46	27	26	57
	11	**38**	24	**20**	17	46	27	26	57
	11	20	**24**	38	**17**	46	27	26	57
	11	20	17	**38**	24	**46**	27	26	57
	11	20	17	38	**24**	46	**27**	26	57
	11	20	17	38	24	**46**	27	**26**	57
	11	20	17	38	24	26	**27**	46	**57**
Jarak = 1	**11**	**20**	17	38	24	26	27	46	57
	11	**20**	**17**	38	24	26	27	46	57
	11	17	**20**	**38**	24	26	27	46	57
	11	17	20	**38**	**24**	26	27	46	57
	11	17	20	24	**38**	**26**	27	46	57
	11	17	20	24	26	**38**	**27**	46	57
	11	17	20	24	26	27	**38**	**46**	57
	11	17	20	24	26	27	38	**46**	**57**
	11	17	20	24	26	27	38	46	57

Program berikut, memakai shell sort untuk mengurutkan array yang berisi 50 nilai acak:

Contoh Program *Shell Sort :*

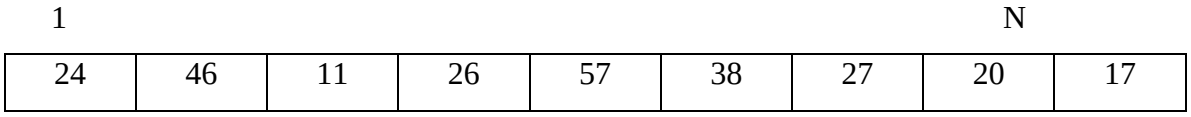
```
/* Shell Sort */
#include <iostream.h>
#include <stdlib.h>
#include <conio.h>
void shell_sort(int array[], int size)
{     int temp, gap, i, exchange_occurred;

      gap= size/2;
      do {
       do {
              exchange_occurred= 0;
                  for (i=0; i<size-gap; i++)
                        if(array[i] > array[i+gap])
                              {
                               temp= array[i];
                               array[i]= array[i+gap];
                               array[i+gap]= temp;
                               exchange_occurred= 1;
                              }
       } while (exchange_occurred);
      } while (gap= gap/2);
}

main()
{
 int values[50], i;

 clrscr();
 //data yang belum diurutkan diambil dari hasil random
 cout << "data yang belum urut : "<< endl;
 for (i=0; i<50; i++)
 {    values[i]= rand()%100;
      cout << values[i] << " ";
 }
 cout << endl;

 shell_sort(values, 50);

 //data yang sudah diurutkan
 cout << "data yang sudah diurutkan : "<< endl;
 for (i=0; i<50; i++)
      cout << values[i] << " ";
 getche();
}
```

2. Quick Sort

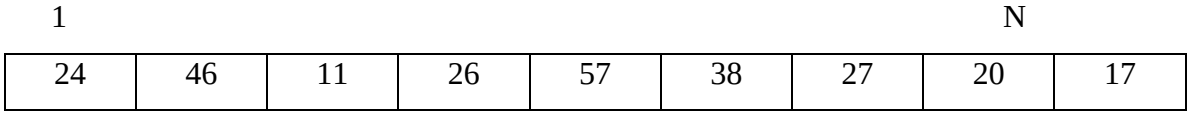
- Sering disebut dengan “partion exchange” sort.
- Langkah-langkah :

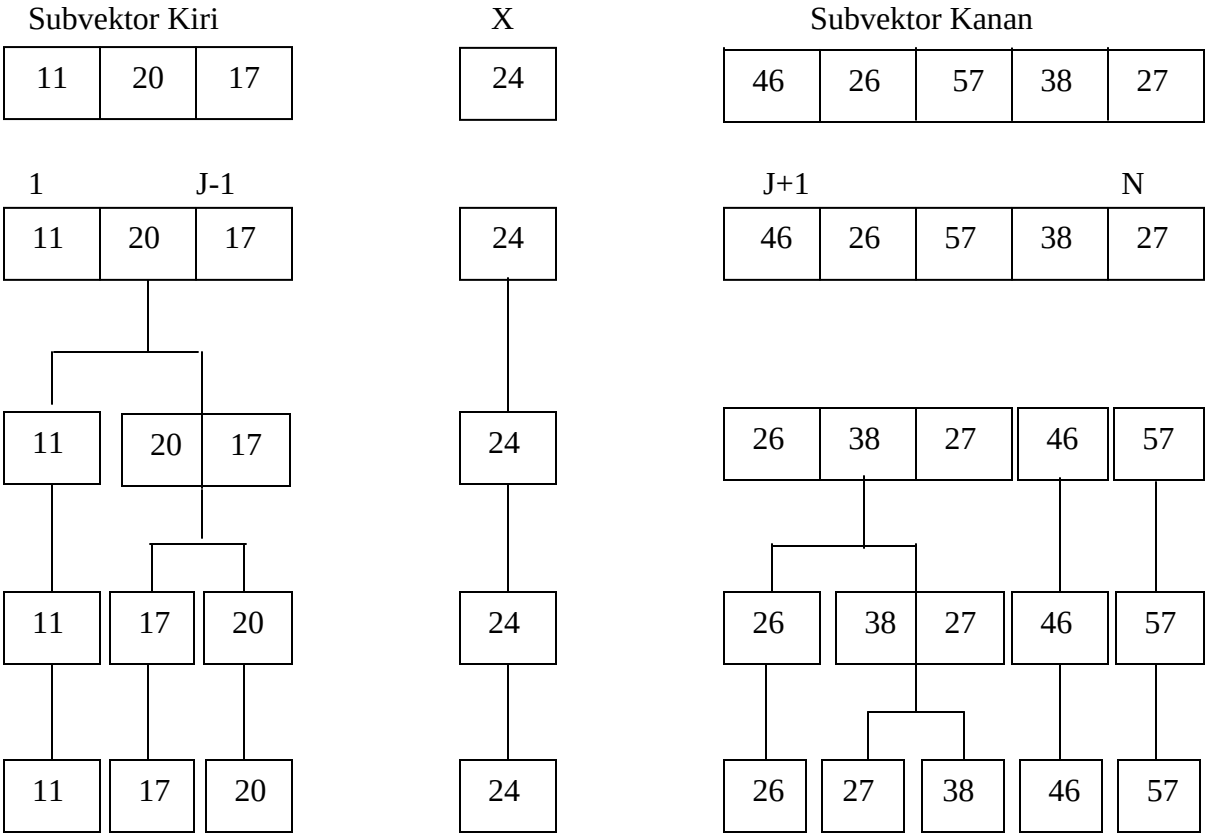
a. Misal vektor A yang memiliki N elemen.
b. Dipilih sembarang elemen, biasanya elemen pertama, misal sebut saja X.
c. Kemudian, semua elemen tersebut dengan menempatkan X pada posisi J sedemikian rupa sehingga :
    i. elemen 1 s/d J-1 memiliki nilai lebih kecil dari X dan
    ii. elemen ke J+1 s/d N memiliki nilai lebih besar dari X.
d. Dengan demikian, terdapat dua buah subvektor.

Contoh :



Ilustrasi





Contoh Program *Quick Sort :*

```
/* Quick Sort*/

#include <iostream.h>
#include <stdlib.h>
#include <conio.h>

void quick_sort(int array[], int first, int last)
{
 int temp, low, high, list_separator;

 low= first;
 high= last;
 list_separator= array[(first+last)/2];
 do {
           while (array[low]<list_separator) low++;
           while (array[high]>list_separator) high--;
           if (low<=high)
                 {
                  temp= array[low];
                  array[low++]= array[high];
                  array[high--]= temp;
                 }
 } while (low<=high);

 if (first<high) quick_sort(array, first, high);
 if (low<last) quick_sort(array, low, last);
}

main()
{
 int values[100], i;
 clrscr();
 //data yang belum diurutkan diambil dari hasil random
 cout << "data yang belum urut : "<< endl;
 for (i=0; i<100; i++)
 {    values[i]= rand()%100;
      cout << values[i] << " ";
 }
 cout << endl;

 quick_sort(values, 0, 99 );

 //data yang sudah diurutkan
 cout << "data yang sudah diurutkan : "<< endl;
 for (i=0; i<100; i++)
      cout << values[i] << " ";
 getche();
}
```

3. Merge Sort

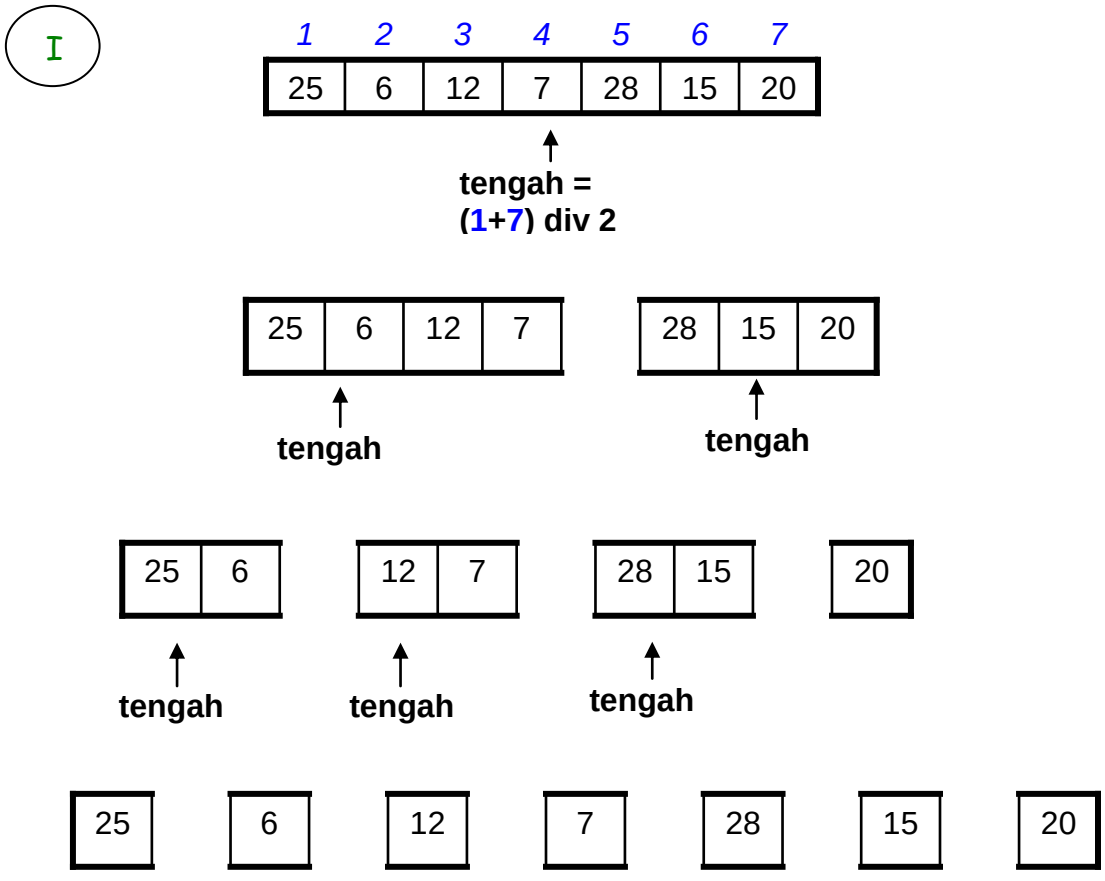
Ide metode Merge Sort :

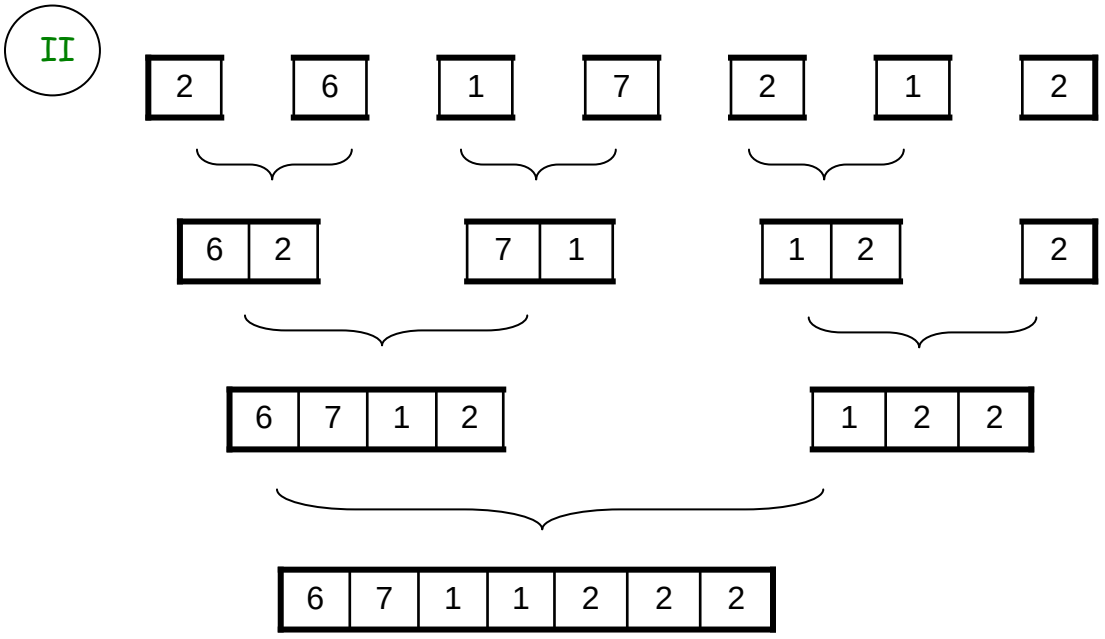
1. Pembagian array data menjadi dua bagian (bagian kiri dan bagian kanan)

   Ulangi langkah 1 secara rekursi terhadap kedu abagian tersebut, sampai tiap subarray hanya terdiri dari satu elemen.
2. Gabungkan masing-masing bagian itu sekaligus mengurutkannya, sesuai pohon yang terbentuk saat membagi array, sampai membentuk array pertama saat sebelum dibagi.

   Ulangi langkah 2 secara rekursi pula.

Ilustrasi Merge Sort





Algoritma MergeSortRekursi

Deklarasi

Type TipeData : Array[1..100] of integer

Data : TipeData

n,i : integer

Procedure MergeSort(input/output Data : TipaData;

input awal,akhir : integer)

Deskripsi

output(‘Banyaknya elemen array :’)

input(n)

i traversal[1..n]

$Data_i \leftarrow$ random(n)

output(‘Data yang belum terurut : ‘)

i traversal[1..n]

output($Data_i$)

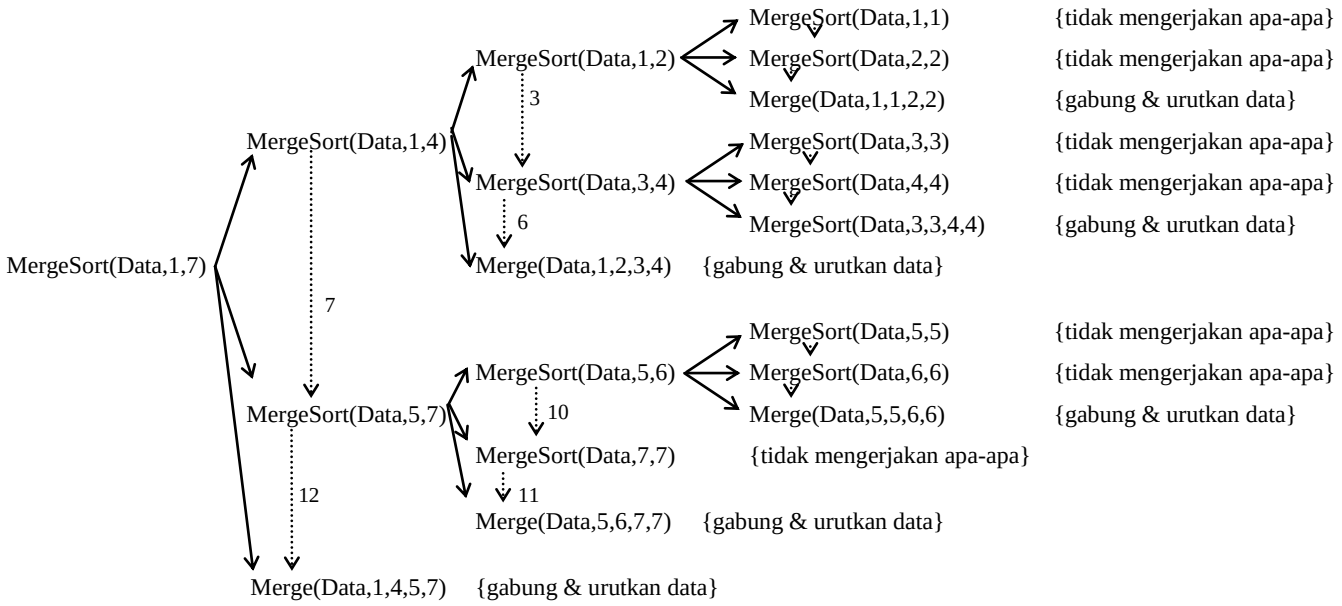
```
MergeSort(Data,1,n)


output('Data yang sudah terurut :')
i traversal[1..n]
   output(Data_i)
```

```
Procedure MergeSort(input/output Data : TipaData;
                                input awal,akhir : integer)
Deklarasi Lokal
 tengah : integer
 Procedure Merge(input/output Data : TipaData;
                          input awalkiri,akhirkiri,
                          awalkiri,akhirkiri : integer)
Deskripsi
  if (awal<akhir) then
        tengah ← (awal+akhir) div 2
        MergeSort(Data,awal,tengah)
        MergeSort(Data,tengah+1,akhir)
        Merge(Data,awal,tengah,tengah+1,akhir)
  endif
```

```
Procedure Merge(input/output Data : TipaData;
                          input awalkiri,akhirkiri,
                          awalkiri,akhirkiri : integer)
Deklarasi Lokal
 temp : TipeData
 i,kiri,kanan : integer
Deskripsi
  kiri ← awalkiri
  kanan ← awalkanan
  i ← awalkiri
```

```
while (kiri<=akhirkiri) or (kanan<=akhirkanan) do
 if (Data_kiri<=Data_kanan) or (kanan>akhirkanan) then
        tempi ← Data_kiri
        kiri ← kiri+1
 endif
 if (Data_kiri>Data_kanan) or (kiri>akhirkiri) then
        tempi ← Data_kanan
        kanan ← kanan+1
 endif
 i ← i + 1
endwhile
i traversal[awalkiri..akhirkanan]
  Data_i ← temp_i
```

MergeSort bila ditest dengan data dari ilustrasi



Keterangan : ⟶ = memanggil

⋯⋯→ = alur logik

PENUTUP

Proses pengurutan tidak langsung dapat menggunakan beberapa metode : Shell Sort, Quick Sort dan Merge Sort. Metode pengurutan tak langsung, implementasinya dapat menggunakan fungsi rekursi.

SOAL-SOAL

1. Tambahkan ketiga metode pengurutan tak langsung di atas untuk menampilan data dalam pengolahan data nilai suatu mata kuliah pada soal latihan kuliah pertemuan ke-8.
2. Buat program menghitung nilai tengah (median) dari data-data integer yang diinput lewat keyboard, dan cari nilai Z dengan $Z = (\text{median})^2$.
   (Keterangan : cari dengan bantuan procedure pengurutan atau sorting. Perhatikan untuk data yang jumlahnya ganjil atau genap)
   Misal :
   Input : Data ke-1 = 3
   Data ke-2 = 2
   Data ke-3 = 6
   Data ke-4 = 5
   Output : Median = 4
   Z = 16