Pertemuan 11

SORTING

SORTING

Operasi Pengurutan (Sorting) adalah operasi yang sangat banyak dilakukan dalam ‘Bussiness Data Processing’. Dalam hal ini pengurutan yang dilakukan adalah secara Ascending (menaik dari kecil ke besar)

Macam-macam Sorting (Pengurutan) :

1. SELECTION SORT
2. BUBBLE SORT
3. MERGE SORT
4. QUICK SORT
5. INSERTION SORT

1. SELECTION SORT

Metode pengurutan Selection Sort, Prosedur atau Algoritmanya adalah sbb :

1. Pengecekan dimulai dari data ke –1 sampai dengan data ke – n
2. Tentukan bilangan dengan index terkecil dari data bilangan tersebut
3. Tukar bilangan dengan index terkecil tersebut dengan bilangan pertama (I = 1) dari data bilangan tersebut
4. Lakukan langkah 2 dan 3 untuk bilangan berikut (I = I+1) sampai didapatkan urutan yang optimal.

Contoh	**:**	**22**	**10**	**15**	**3**	**8**	**2**

<u>Iterasi 1</u>

	1	**2**	**3**	**4**	**5**	**6**
Langkah 1 :	**22**	**10**	**15**	**3**	**8**	**2**
Langkah 2 :	**22**	**10**	**15**	**3**	**8**	**2**
Langkah 3 :	**2**	**10**	**15**	**3**	**8**	**22**
Langkah 4 :	**Ulangi langkah 2 dan 3**					

Iterasi 2

Langkah 1: 2 10 15 3 8 22

Langkah 2: 2 10 15 3 8 22

Langkah 3: 2 3 15 10 8 22

Langkah 4: Ulangi langkah 2 dan 3 .

Lakukan Iterasi selanjutnya sampai iterasi ke-6

Prosedur Program Selection Sort (Dengan program C++)

```
void selection_sort(int data[]){
  for(int i=0;i<n-1;i++){
    pos = i;
    for(int j=i+1;j<n;j++){
      if(data[j] < data[pos]) pos = j; //ascending
    }
    if(pos != i) tukar(&data[pos],&data[i]);
  }
}
```

2. BUBBLE SORT

Metode pengurutan Bubble Sort mempunyai Algoritma atau Prosedur sebagai berikut :

1. Pengecekan dimulai dari data ke – 1 sampai dengan data ke – n
2. Bandingkan data ke – n dengan data sebelumnya (n – 1), jika lebih kecil maka tukar bil. Tsb dengan data yang ada didepannya (sebelumnya) satu persatu (n – 1, n - 2, n – 3, ….dst)
3. Lakukan langkah ke- 2 sampai didapatkan urutan yang optimal.

Contoh : 22 10 15 3 8 2

terasi 1

		1	2	3	4	5	6
Langkah 1	**:**	**22**	**10**	**15**	**3**	**8**	**2**
Langkah 2	**:**	**22**	**10**	**15**	**3**	**8**	**2**
Langkah 3	**:**	**22**	**10**	**15**	**3**	**2**	**8**
Langkah 4	**:**	**Ulangi langkah 2 dan 3**					

Hasil iterasi 1 : 2 22 10 15 3 8

Iterasi 2

Langkah 1 : 2 22 10 15 3 8

Langkah 2 : 2 22 10 15 3 8

ket: 8>3, maka 8 tidak pindah, untuk selanjutnya bandingkan data sebelunya yaitu 3.

Langkah 3 : 2 22 10 3 15 8

Langkah 4 : Ulangi langkah 2 dan 3

Hasil Iterasi 2 : 2 3 22 10 15 8

Lakukan Iterasi selanjutnya sampai iterasi ke- 6

Prosedur Program Bubble Sort (Dengan program C++)

```
void bubble_sort(int data[]){
  for(int i=1;i<n;i++){
    for(int j=n-1;j>=i;j--){
      if(data[j]<data[j-1]) tukar(&data[j],&data[j-1]); //ascending
    }
  }
}
```

3. MERGE SORT

Menggunakan Metode Iteratif Merge Sort mempunyai Algoritma atau Prosedur sebagai berikut :

1. Kelompokan deret bilangan kedalam 2 bagian, 4 bagian, 8 bagian ….dst ….
2. Urutkan secara langsung bilangan dalam kelompok tersebut
3. Lakukan langkah di atas untuk kondisi bilangan yang lain sampai didapatkan urutan yang optimal .

Contoh : 22 10 15 3 8 2

Iterasi 1

		1	2	3	4	5	6
Langkah 1	:	22	10	15	3	8	2
Langkah 2	:	10	22	3	15	2	8

Iterasi 2

Langkah 1	:	10	22	3	15	2	8
Langkah 2	:	3	10	15	22	2	8

Iterasi 3

Langkah 1	**:**	**3**	**10**	**15**	**22**	**2**	**8**
Langkah 2	**:**	**2**	**3**	**8**	**10**	**15**	**22**

Prosedur Program Merge Sort (Dengan program C++)

```
int main(void)
{
    int ar[100];

    int i, v, len;

    for (i=0; i<100; i++) {
        cout << "enter a number (-1 to quit): ";
        cin >> v;

        if (v < 0) break;

        ar[i] = v;
    }
```

```
    len = i;

    cout << "main:  before sort:\n";
    for (i=0; i<len; i++) {
        cout << "main:  ar[" << i << "] = " << ar[i] << endl;
    }

    mergesort ms(ar, len);

    cout << "main:  after sort:\n";
    for (i=0; i<len; i++) {
        cout << "main:  ar[" << i << "] = " << ar[i] << endl;
    }
```

4. QUICK SORT

Sangat baik untuk tabel data yang sangat besar.

Algorithma atau Prosedur Quick Sort adalah sbb :

1. Tentukan bilangan yang dinyatakan sebagai batas bawah (Lower Bound (I = 1)) dan bilangan yang dinyatakan sebagai batas atas (Upper Bound(I = N))
2. Syarat pemindahan adalah LB > UB, dengan melihat perbandingan antara UB (awal bilangan)dan LB (akhir bilangan)
3. Jika LB > UB lakukan pertukaran antara kedua bilangan tersebut, jika tidak lakukan perpindahan LB (I = I +1, I = I + 2,.....) ke bilangan selanjutnya dan bandingkan kembali dengan UB (I = N, I = N – 1, I = N – 2,.....)
4. Lakukan langkah 2 dan 3 untuk bilangan selanjutnya sampai didapatkan urutan yang optimal.

Contoh	**:**	**22**	**10**	**15**	**3**	**8**	**2**

<u>Iterasi 1</u>

		1	**2**	**3**	**4**	**5**	**6**
Langkah	**1**	**: 22**	**10**	**15**	**3**	**8**	**2**
		LB					**UB**

Langkah	**2**	**:2**	**10**	**15**	**3**	**8**	**22**

Iterasi 2

Langkah	**1**	**: 2**	**10**	**15**	**3**	**8**	**22**
		LB/UB					
Langkah	**2**	**:2**	**10**	**15**	**3**	**8**	**22**
			LB			**UB**	

Iterasi 3

Langkah	**1**	**:2**	**10**	**15**	**3**	**8**	**22**
			LB			**UB**	
Langkah	**2**	**:2**	**8**	**15**	**3**	**10**	**22**

Iterasi 4

Langkah	**1**	**:2**	**8**	**15**	**3**	**10**	**22**
			LB		**UB**		
Langkah	**2**	**:2**	**3**	**15**	**8**	**10**	**22**

Lakukan Iterasi selanjutnya sampai iterasi ke- 6

Prosedur Program Quick Sort (Dengan program C++)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#include <codecogs/array/sort/quick_sort.h>

int main()
{
  double vals[25];
  int n=25;

  srand((unsigned int) time(NULL));
  for (int i=0; i<n; i++) vals[i]=((double) n*rand())/RAND_MAX;
```

```
    printf("\nArray to be sorted:\n");
    for (int i=0; i<n; i++) printf("%3.0f ", vals[i]);
    printf("\n");

    Array::Sort::quickSort<double>(vals, n);

    printf("Sorted array:\n");
    for (int i=0; i<n; i++) printf("%3.0f ", vals[i]);
    printf("\n");
    return 0;
}
```

5. INSERTION SORT

kukan insert suatu record dalam record-record yang telah diurutkan.

Metode pengurutan Bubble Sort mempunyai Algoritma atau Prosedur sebagai berikut :

1. Pengecekan dimulai dari data ke – 1 sampai dengan data ke – n
2. Pengurutan dilakukan dengan cara membandingkan data ke – I (dimana I dimulai dari data ke-2 sampai dengan data yang terakhir.
3. Bandingkan data ke – I tersebut dengan data sebelumnya (I –1), jika lebih kecil maka data tersebut dapat disisipkan ke data awal (depan) sesuai dengan posisi yang seharusnya.
4. Lakukan langkah ke- 2 dan 3 untuk bilangan selanjutnya (I = I+1) sampai didapatkan urutan yang optimal

Contoh : 22 10 15 3 8 2

Iterasi 1

	1	2	3	4	5	6
Langkah 1:	22	10	15	3	8	2
Langkah 2:	22	10	15	3	8	2
Langkah 3:	10	22	15	3	8	2
Langkah 4:	Ulangi langkah 2 dan 3					

Iterasi 2

Langkah 1:	**10**	**22**	**15**	**3**	**8**	**2**
Langkah 2:	**10**	**22**	**15**	**3**	**8**	**2**
Langkah 3:	**10**	**15**	**22**	**3**	**8**	**2**
Langkah 4:	**Ulangi langkah 2 dan 3**					

Lakukan Iterasi selanjutnya sampai iterasi ke- 6

Catatan : Setiap ada pemindahan, maka elemen. Yang sudah ada akan di insert sehingga akan bergeser kebelakang.

Prosedur Program Insertion Sort (Dengan program C++)

```
void insertion_sort(int data[]){
      int temp;
      for(int i=1;i<n;i++){
        temp = data[i];
        j = i -1;
        while(data[j]>temp && j>=0){
          data[j+1] = data[j];
          j--;
        }
        data[j+1] = temp;
      }
}
```