Praktikum 11

Algoritma Pengurutan (Bubble Sort dan Shell Sort)

A. TUJUAN PEMBELAJARAN

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

1. Memahami mengenai algoritma pengurutan *bubble sort* dan *shell sort*.
2. Mampu mengimplementasikan algoritma pengurutan *bubble sort* dan *shell sort* secara *ascending* dan *descending*.

B. DASAR TEORI

B.1 Algoritma *Bubble Sort*

Bubble sort (metode gelembung) adalah metode/algoritma pengurutan dengan dengan cara melakukan penukaran data dengan tepat disebelahnya secara terus menerus sampai bisa dipastikan dalam satu iterasi tertentu tidak ada lagi perubahan. Jika tidak ada perubahan berarti data sudah terurut. Disebut pengurutan gelembung karena masing-masing kunci akan dengan lambat menggelembung ke posisinya yang tepat.

Metode pengurutan gelembung (*Bubble Sort*) diinspirasikan oleh gelembung sabun yang berada dipermukaan air. Karena berat jenis gelembung sabun lebih ringan daripada berat jenis air, maka gelembung sabun selalu terapung ke atas permukaan. Prinsip di atas dipakai pada pengurutan gelembung.

Algoritma *bubble sort* adalah salah satu algoritma pengurutan yang paling simple, baik dalam hal pengertian maupun penerapannya. Ide dari algoritma ini adalah mengulang proses pembandingan antara tiap-tiap elemen array dan menukarnya apabila urutannya salah. Pembandingan elemen-elemen ini akan terus

diulang hingga tidak perlu dilakukan penukaran lagi. Algoritma ini termasuk dalam golongan algoritma *comparison sort*, karena menggunakan perbandingan dalam operasi antar elemennya. Berikut ini adalah gambaran dari algoritma *bubble sort*. Misalkan kita mempunyai sebuah array dengan. Elemen-elemen "4 2 5 3 9". Proses yang akan terjadi apabila digunakan algoritma *bubble sort* adalah sebagai berikut.

Pass pertama

(4 2 5 3 9) menjadi (2 4 5 3 9)

(2 4 5 3 9) menjadi (2 4 5 3 9)

(2 4 5 3 9) menjadi (2 4 3 5 9)

(2 4 3 5 9) menjadi (2 4 3 5 9)

Pass kedua

(2 4 3 5 9) menjadi (2 4 3 5 9)

(2 4 3 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

Pass ketiga

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

Dapat dilihat pada proses di atas, sebenarnya pada pass kedua, langkah kedua, array telah terurut. Namun algoritma tetap dilanjutkan hingga pass kedua berakhir. Pass ketiga dilakukan karena definisi terurut dalam algoritma *bubble sort* adalah tidak ada satupun penukaran pada suatu pass, sehingga pass ketiga dibutuhkan untuk memverifikasi keurutan array tersebut.

B.2 Algoritma *Shell Sort*

Metode *shell sort* dikembangkan oleh Donald L. Shell pada tahun 1959. Dalam metode ini jarak antara dua elemen yang dibandingkan dan ditukarkan tertentu. Secara singkat metode ini dijelaskan sebagai berikut. Pada langkah pertama, kita ambil elemen pertama dan kita bandingkan dan kita bandingkan dengan elemen pada jarak tertentu dari elemen pertama tersebut. Kemudain elemen kedua kita

bandingkan dengan elemen lain dengan jarak yang sama seperti jarak yang sama seperti diatas. Demikian seterusnya sampai seluruh elemen dibandingkan. Pada langkah kedua proses diulang dengan langkah yang lebih kecil, pada langkah ketiga jarak tersebut diperkecil lagi seluruh proses dihentikan jika jarak sudah sama dengan satu.

Contoh dari proses pengurutan dengan menggunakan metode *Shell Sort* :

Jarak	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
Awal	22	10	15	3	2	8
Jarak = 3	22	10	15	3	2	8
	3	10	15	22	2	8
	3	2	15	22	10	8
	3	2	8	22	10	15
Jarak = 1	3	2	8	22	10	15
	2	3	8	22	10	15
	2	3	8	22	10	15
	2	3	8	22	10	15
	2	3	8	10	22	15
	2	3	8	10	15	22
Akhir	2	3	8	10	15	22

C. TUGAS PENDAHULUAN

Jawablah pertanyaan berikut ini :

1. Tuliskan algoritma pengurutan *bubble sort* secara *ascending.*
2. Tuliskan algoritma pengurutan *shell sort* secara *ascending*.

D. PERCOBAAN

Percobaan 1 : *Bubble sort* secara *ascending*.

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

#define MAX 20

void BubbleSort(int arr[])
{
     int i, j, temp;
     for(i=0; i<MAX-1; i++){
         for(j=0; j<MAX-i-1; j++){
             if(arr[j] > arr[j+1]){
                 temp = arr[j+1];
                 arr[j+1] = arr[j];
                 arr[j]= temp;
             }
         }
```

```
      }
}


void main()
{
       int data_awal[MAX], data_urut[MAX];
       int i;
       long k1, k2;

       printf("Sebelum pengurutan : \n");
       for(i=0; i<MAX; i++){
             srand(time(NULL) * (i+1));
             data_awal[i] = rand() % 100 + 1;
             printf("%d ", data_awal[i]);
       }
       printf("\nSetelah pengurutan : \n");
       for(i=0; i<MAX; i++)
             data_urut[i] = data_awal[i];

       time(&k1);
       BubbleSort(data_urut);
       time(&k2);
       for(i=0; i<MAX; i++)
             printf("%d ", data_urut[i]);
       printf("\nWaktu = %ld\n", k2-k1);
}
```

Percobaan 2 : *Bubble sort* secara *ascending* dengan *flag*.

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

#define MAX 20

void BubbleSortFlag(int arr[])
{
     int i=0, j, temp;
     bool did_swap=true;

     while(i < MAX-1 && did_swap){
         for(j=0; j<MAX-i-1; j++){
             did_swap=false;
             if(arr[j] > arr[j+1]){
                 temp = arr[j+1];
                 arr[j+1] = arr[j];
                 arr[j]= temp;
                 did_swap=true;
             }
         }
         i++;
     }
}


void main()
{
       int data_awal[MAX], data_urut[MAX];
       int i;
```

```
      long k1, k2;

      printf("Sebelum pengurutan : \n");
      for(i=0; i<MAX; i++){
            srand(time(NULL) * (i+1));
            data_awal[i] = rand() % 100 + 1;
            printf("%d ", data_awal[i]);
      }
      printf("\nSetelah pengurutan : \n");
      for(i=0; i<MAX; i++)
            data_urut[i] = data_awal[i];

      time(&k1);
      BubbleSortFlag(data_urut);
      time(&k2);
      for(i=0; i<MAX; i++)
            printf("%d ", data_urut[i]);
      printf("\nWaktu = %ld\n", k2-k1);
}
```

Percobaan 3 : *Shell sort* secara *ascending*.

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

#define MAX 20

void ShellSort(int arr[])
{
     int i, jarak, temp;
     bool did_swap=true;

       jarak = MAX;
     while(jarak > 1){
         jarak = jarak / 2;
         did_swap = true;
         while( did_swap ){
             did_swap = false;
             i = 0;
             while(i < (MAX-jarak)){
                 if(arr[i] > arr[i+jarak]){
                     temp = arr[i];
                     arr[i] = arr[i+jarak];
                     arr[i+jarak] = temp;
                     did_swap=true;
                 }
                 i++;
             }
         }
     }
}

void main()
{
      int data_awal[MAX], data_urut[MAX];
      int i;
      long k1, k2;

      printf("Sebelum pengurutan : \n");
```

```
      for(i=0; i<MAX; i++){
            srand(time(NULL) * (i+1));
            data_awal[i] = rand() % 100 + 1;
            printf("%d ", data_awal[i]);
      }
      printf("\nSetelah pengurutan : \n");
      for(i=0; i<MAX; i++)
            data_urut[i] = data_awal[i];

      time(&k1);
      ShellSort(data_urut);
      time(&k2);
      for(i=0; i<MAX; i++)
            printf("%d ", data_urut[i]);
      printf("\nWaktu = %ld\n", k2-k1);
}
```

E. LATIHAN

1. Dari percobaan 1 tambahkan fungsi untuk melakukan pengurutan *bubble sort* secara *descending*.
2. Dari percobaan 2 tambahkan fungsi untuk melakukan pengurutan *bubble sort* dengan *flag* secara *descending*.
3. Dari percobaan 3 tambahkan fungsi untuk melakukan pengurutan *shell sort* secara *descending*.
4. Buatlah struktur `Mahasiswa` dengan variable `nrp` dan `nama` yang memiliki tipe `String` dan `kelas` yang bertipe `int`. Buatlah fungsi pengurutan dengan *bubble sort, bubble sort* dengan *flag* dan *shell sort* berdasarkan `nrp`.

```
struct Mahasiswa {
    char nrp[10];
    char nama[20];
    int kelas;
};
```

F. LAPORAN RESMI

1. Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.
2. Tuliskan kesimpulan dari percobaan dan latihan yang telah anda lakukan.