

# LAPORAN INDIVIDU : SHELL SORT (Kelompok 7)

Revina Aurigha Firdaus (21091397003)

## 1. Laporan kodingan

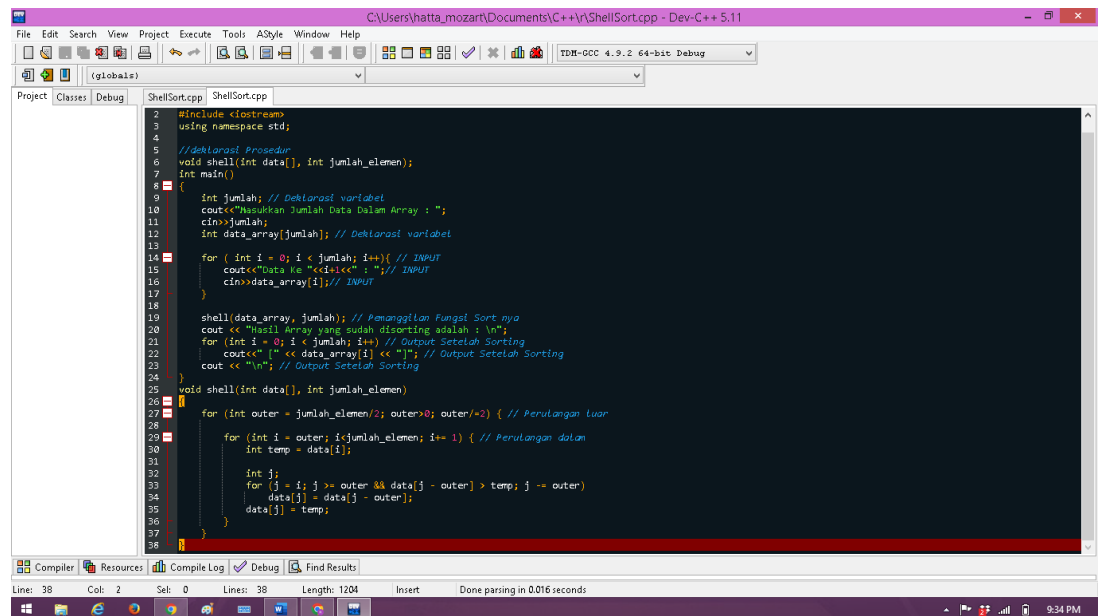
Menurut Atrinawati (2007) menjelaskan Shell Sort adalah algoritma dengan kompleksitas algoritma  $O(n^2)$  dan yang paling efisien dibandingkan algoritma-algoritma lain dengan kompleksitas algoritma yang sama. Algoritma shell sort lima kali lebih cepat dibandingkan algoritma pengurutan gelembung dan dua kali lebih cepat dibandingkan algoritma pengurutan dengan penyisipan.

Shell Sort adalah versi umum dari insertion sort. Ini adalah semacam perbandingan di tempat. Pengurutan Shell juga dikenal sebagai pengurutan peningkatan yang semakin berkurang, ini adalah salah satu algoritma pengurutan tertua yang ditemukan oleh Donald L. Shell (1959.)

Algoritma ini menggunakan insertion sort pada interval elemen yang besar untuk disortir. Kemudian interval pengurutan terus menurun secara berurutan hingga interval mencapai 1. Interval ini dikenal sebagai urutan celah.

Algoritma ini bekerja cukup efisien untuk array ukuran kecil dan menengah karena kompleksitas waktu rata-ratanya mendekati  $O(n)$ .

### • Kode tipe Shell Sort



```
1 #include <iostream>
2 using namespace std;
3
4 //deklarasl Prosedur
5 void shell(int data[], int jumlah_elemen);
6
7 int main()
8 {
9     int jumlah; // Deklarasi variabel
10    cout<<"Masukkan Jumlah Data Dalam Array : ";
11    cin>>jumlah;
12    int data_array[jumlah]; // Deklarasi variabel
13
14    for (int i = 0; i < jumlah; i++){ // INPUT
15        cout<<"Data ke "<<i<<" : "; // INPUT
16        cin>>data_array[i]; // INPUT
17    }
18
19    shell(data_array, jumlah); // Pemanggilan Fungsi Sortirnya
20    cout<<"Masih Array yang sudah disorting adalah : \n";
21    for (int i = 0; i < jumlah; i++) // Output Setelah Sorting
22        cout<<"[" << data_array[i] << " ]"; // Output Setelah Sorting
23    cout<<"\n"; // Output Setelah Sorting
24 }
25
26 void shell(int data[], int jumlah_elemen)
27 {
28     for (int outer = jumlah_elemen/2; outer>0; outer/=2) { // Perulangan luar
29         for (int i = outer; i<jumlah_elemen; i+= 1) { // Perulangan dalam
30             int temp = data[i];
31
32             int j;
33             for (j = i; j >= outer && data[j - outer] > temp; j -= outer)
34                 data[j] = data[j - outer];
35             data[j] = temp;
36         }
37     }
38 }
```

- **Hasil run Shell Sort**

```

C:\Users\hatta_mozart\Documents\C++\ShellSort.exe
Masukkan Jumlah Data Dalam Array : 5
Data Ke 1 : 4
Data Ke 2 : 20
Data Ke 3 : 3
Data Ke 4 : 9
Data Ke 5 : 13
Hasil Array yang sudah disorting adalah :
[3] [4] [9] [13] [20]
Process exited after 11.29 seconds with return value 0
Press any key to continue . . .
  
```

## 2. BIG O

- Shell Sort adalah penyortiran berdasarkan perbandingan.
- Kompleksitas waktu dari Shell Sortir tergantung pada urutan celah. Kompleksitas waktu kasus terbaiknya adalah  $O(n \cdot \log n)$  dan kasus terburuknya adalah  $O(n \cdot \log^2 n)$ . Kompleksitas waktu dari jenis Shell umumnya diasumsikan mendekati  $O(n)$  dan kurang dari  $O(n^2)$  karena menentukan kompleksitas waktunya masih merupakan masalah terbuka.
- Shell Sort adalah sortir yang tidak stabil karena urutan relatif elemen dengan nilai yang sama dapat berubah.
- Diamati bahwa shell sort 5 kali lebih cepat dari bubble sort dan dua kali lebih cepat dari insertion sort pesaing terdekatnya.
- Terdapat berbagai macam increment sequence atau gap sequence pada shell sort yang menghasilkan berbagai kompleksitas antara  $O(n)$  dan  $O(n^2)$ .

## 3. Kelebihan dan kekurangan Shell sort

- **Kelebihan Shell sort :**
  1. Algoritma ini sangat rapat dan mudah untuk diimplementasikan.
  2. Operasi pertukarannya hanya dilakukan sekali saja.
  3. Waktu pengurutan dapat lebih ditekan.
  4. Mudah menggabungkannya kembali.
  5. Kompleksitas selection sort relatif lebih kecil.
- **Kekurangan Shell Sort :**
  1. Membutuhkan method tambahan.
  2. Sulit untuk membagi masalah.