LAPORAN

“ALGORITMA SHELL SORT”

Disusun Oleh :

2209116045

ABDUL RAHMAN

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS MULAWARMAN

2022

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam dunia teknologi informasi, pengolahan data merupakan hal yang sangat penting dan seringkali menjadi fokus utama dalam pengembangan program. Salah satu bagian dari pengolahan data adalah pengurutan (sorting) data. Sorting data diperlukan ketika kita ingin menemukan nilai tertentu dalam data atau ingin menampilkan data dalam urutan tertentu. Namun, mengurutkan data dapat menjadi tugas yang sangat sulit, terutama ketika jumlah datanya sangat besar.

Untuk mengatasi hal ini, berbagai algoritma sorting telah dikembangkan. Salah satu algoritma sorting yang efisien adalah algoritma Shell sort. Algoritma ini pertama kali diperkenalkan oleh Donald Shell pada tahun 1959. Shell sort adalah algoritma sorting yang cepat dan efisien, dan dapat mengurutkan data dengan cepat bahkan untuk ukuran data yang besar.

Dalam laporan ini, penulis akan membahas tentang algoritma Shell sort. Tujuan dari laporan ini adalah untuk memberikan pemahaman yang lebih baik tentang algoritma Shell sort dan bagaimana cara kerjanya.

1.2. Tujuan

Tujuan laporan ini adalah untuk memberikan pemahaman yang lebih baik tentang algoritma Shell sort dan bagaimana cara kerjanya. Dengan demikian, laporan ini akan membahas hal-hal berikut:

1) Mengenal secara apa itu singkat shell sort
2) Keuntungan dan manfaat penggunaan algoritma Shell sort.
3) Implementasi algoritma Shell sort.

1.3. Manfaat

Dari banyak algoritma *sorting,* penulis memilih shell sort sebagai bahan utama dari laporan ini, karena beberapa kelebihannya, diantaranya adalah sebagai berikut:

1) Efisiensi: Shell sort dapat mengurutkan array dengan cepat dan efisien bahkan untuk ukuran data yang besar. Hal tersebut karena algoritma ini membagi data menjadi beberapa kelompok dan mengurutkannya secara terpisah sebelum menggabungkannya kembali menjadi satu kesatuan.

2) Mudah diimplementasikan: Algoritma Shell sort relatif mudah diimplementasikan dan mudah dimodifikasi. Ini membuatnya dapat digunakan di berbagai bahasa pemrograman dan lingkungan pengembangan.

3) Stabilitas: Shell sort stabil dalam hal urutan data. Artinya, urutan elemen dengan nilai yang sama tidak akan berubah setelah diurutkan.

4) Penggunaan memori yang rendah: Shell sort memerlukan sedikit memori untuk melakukan sorting. Ini membuatnya sangat cocok untuk digunakan dalam perangkat lunak yang memerlukan penggunaan memori yang efisien.

5) Cocok untuk array yang hampir terurut: Algoritma Shell sort sangat efektif pada array yang hampir terurut. Hal ini karena algoritma ini dapat memanfaatkan struktur hampir terurut untuk meningkatkan kecepatan sorting.

BAB II
PEMBAHASAN

2.1. Cara Kerja Shell Sort

Shell sort adalah sebuah algoritma sorting yang efisien dan cepat. Algoritma ini merupakan pengembangan dari algoritma insertion sort, yang pada dasarnya membandingkan dan memindahkan elemen satu per satu dalam sebuah array. Shell sort memperbaiki kelemahan insertion sort dengan cara membandingkan dan memindahkan elemen secara terpisah dalam array, menggunakan sebuah increment yang disebut gap.

Gap pada Shell sort mengatur jarak antara elemen yang dibandingkan dan dipindahkan dalam array. Awalnya, gap diatur dengan nilai yang lebih besar dari ukuran array, dan kemudian gap ini dikurangi secara berulang-ulang sampai nilainya menjadi 1.

Adapun secara singkat cara kerja Shell sort adalah sebagai berikut:

1) Tentukan increment (gap / jarak) awal. Biasanya, nilai gap diatur dengan ukuran array dibagi 2.
2) Lakukan pengurutan insertion sort pada elemen-elemen dengan jarak gap yang telah ditentukan.
3) Kurangi gap menjadi setengah dari gap sebelumnya, dan lakukan kembali pengurutan insertion sort pada elemen dengan jarak gap yang telah ditentukan.
4) Ulangi langkah 3 sampai gap menjadi 1.
5) Setelah fase shell selesai, elemen-elemen pada array telah diurutkan. Kompleksitas waktu Shell sort tergantung pada increment yang digunakan.

2.2. Coding / Pseuducode Shell Sort

```
import random
# Import untuk membuat library array random

def shell_sort(array):
    n = len(array)
    # n adalah panjang dari data yaitu array yang telah dibuat dengan modul
random
    # Dalam hal ini n = 12
```

```
    interval = n // 2
    # Interval adalah loncatan / jarak antar array yang akan dibandingkan
    # Misalnya [80, 19, 10, 62, 31, 70, 51, 29, 5, 45, 93, 12] dengan interval 6
    # Maka interval 6 = [80, 51] [19,29] dst...
    # Jika interval 3 = [80, 62] [19, 31] dst...
    # Patut diingat bahwa interval selalu dibagi 2
```

```
    while interval > 0:
        # Jika interval tidak sama dengan 0 maka akan looping
        # Jika interval 6, maka akan looping hingga interval adalah 0 dimana
seluruh array telah disusun
        for i in range(interval, n):
            # i adalah angka / array didalam data yang akan diurutkan
            temp = array[i]
            # temp atau temporary sebagai penyimpanan sementara dari array yang
telah diurutkan
            j = i
            while j >= interval and array[j - interval] > temp:
                # Jika array
                array[j] = array[j - interval]
                j -= interval
                array[j] = temp
        interval //= 2
    return array
```

```
def data_acak():
    # Random akan menghasilkan angkat dari 1 hingga 100 acak, dimana jumlah yang
dihasilkan hanya 12
    array = [random.randint(1, 100) for _ in range(12)]
    return array
```

```
# Panggil fungsi data_acak untuk menginisialisasi data.
data_awal = data_acak()
```

```
print('Data (Array) sebelum shellsort:')
```

```
print(data_awal)
print()
```

```
# Panggil fungsi shell_sort untuk mengurutkan data.
data_urut = shell_sort(data_awal)
```

```
print('Data (Array) setelah shellsort:')
print(data_urut)
```

```
print()
print('Array sebelum shellsort: [80, 19, 10, 62, 31, 70, 51, 29, 5, 45, 93, 12]')
print('Array menjadi sub-set: [80, 19, 10, 62, 31, 70] dan [51, 29, 5, 45, 93,
12]')
print('Interval k = 6 adalah [51, 19, 5, 45, 31, 12, 80, 29, 10, 62, 93, 70]')
print('Interval k = 3 adalah [45, 19, 5, 51, 29, 10, 62, 31, 12, 80, 93, 70]')
print('Interval k = 1 adalah [5, 10, 12, 19, 29, 31, 45, 51, 62, 70, 80, 93]')
print('Array setelah shellsort: [5, 10, 12, 19, 29, 31, 45, 51, 62, 70, 80, 93]')
```

2.3. Cara Kerja Algoritma Shell Sort

Dibawah ini merupakan contoh dari algoritma shell sort, dalam contoh ini saya menggunakan array dengan jumlah n = 12

```
[80, 19, 10, 62, 31, 70, 51, 29, 5, 45, 93, 12]
```

INDEX	0	1	2	3	4	5	6	7	8	9	10	11
ARRAY	80	19	10	62	31	70	51	29	5	45	93	12

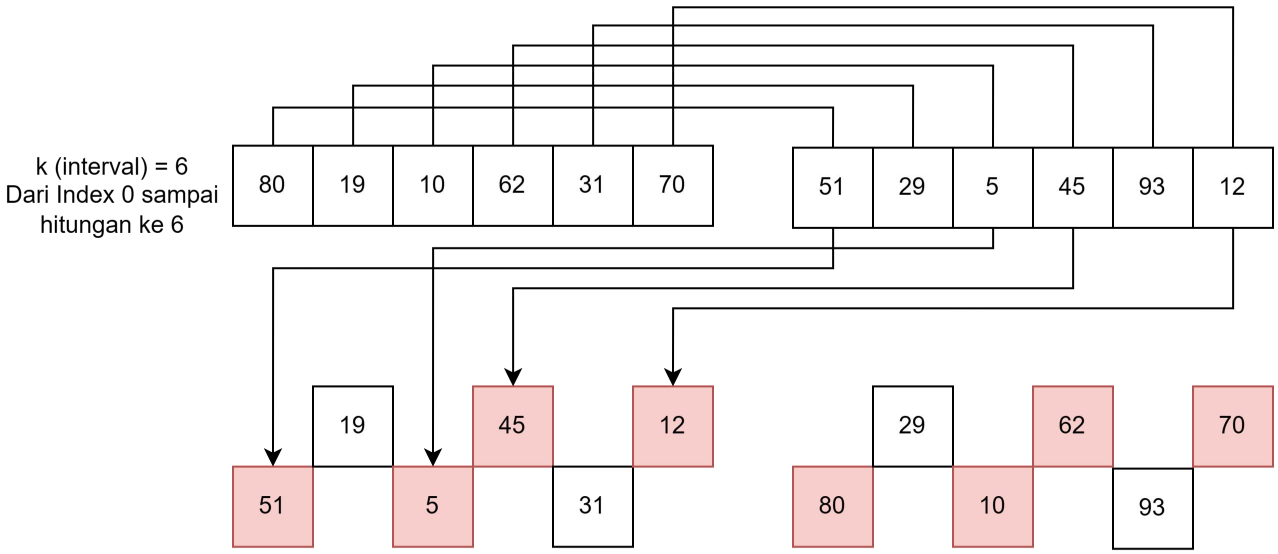
Pada contoh array diatas, terdapat index yaitu 0 hingga 11, meski n = 12. Index 0 digunakan untuk urutan dalam code program array

Selanjutnya adalah membagi kumpulan array. Dalam hal ini adalah n / 2 = 6
Sehingga array menjadi 2 set, seperti dibawah ini

n / 2 = 6	80	19	10	62	31	70

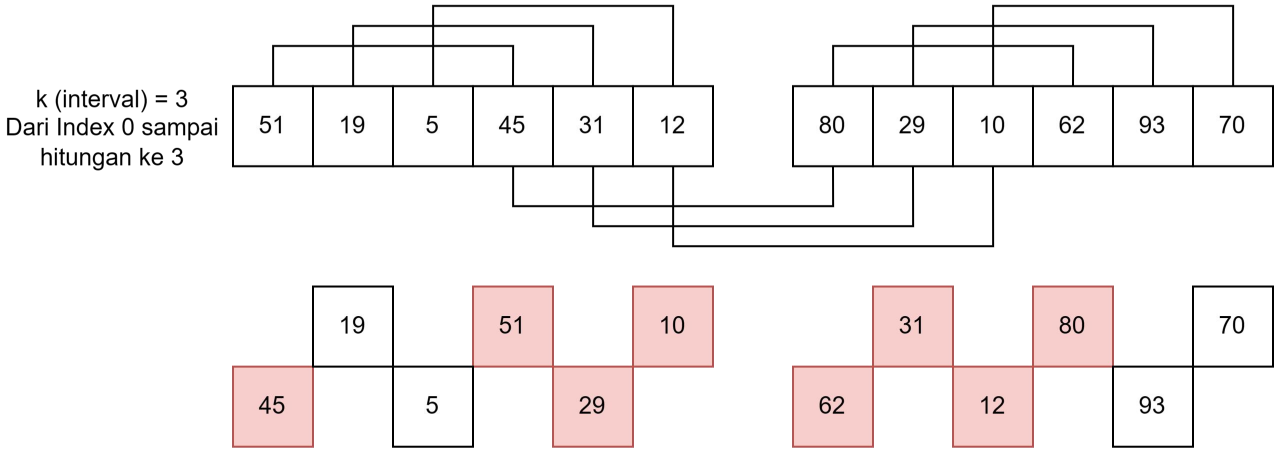
51	29	5	45	93	12

Setelah membagi array menjadi 2 set, maka kita lakukan perbandingan dengan interval / k = 6
perhitungan dimulai dari index 1 hingga posisi 6, lalu index 2 hingga 7 dan seterusnya...
seperti contoh dibawah ini



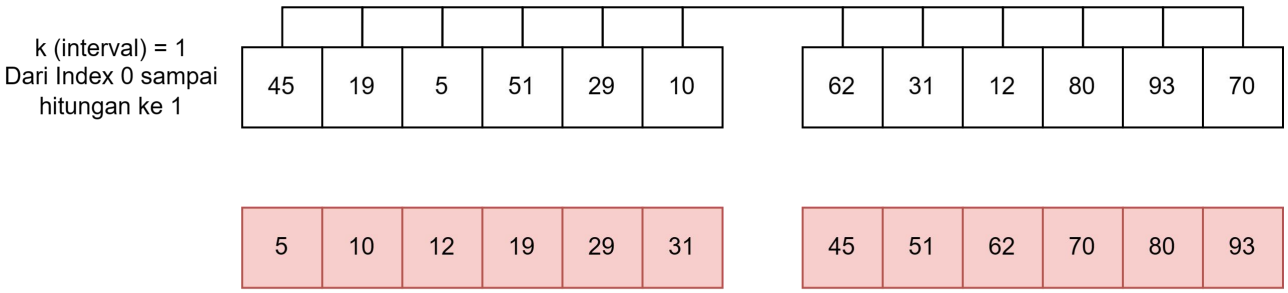
Setelah menemukan perbandingannya, maka kita akan membandingkannya, seperti contoh diatas: 80 > 51 maka 51 berada di kiri, 19 < 29 maka 19 berada di kiri, dan seterusnya, angka terkecil dapat diletakkan di bagian kiri jika ascending dan kanan jika descending

Setelah terurut seperti diatas, maka akan dilakukan perbandingan lagi dengan interval 3 yang berasal dari 6 / 2 = 3. Disini array akan disusun lalu dibandingkan dengan interval 3
yaitu [51, 45] [19, 31] [5, 12] dan seterusnya.



Setelah menemukan perbandingannya, maka akan dibandingkan, yaitu terkecil ke terbesar, dalam hal ini 51 > 45, sehiingga 45 berada di ujung kiri (dibawah 51), hal tersebut diulang hingga mencapai array terakhir yaitu 70.

Jika telah mengurutkannya namun belum terurut dengan benar, maka akan diurutkan lagi ke interval terkecil yaitu interval 1.



Diatas, kita menggunakan k = 1, lalu hasil akhirnya akan disusun menjadi angka ascending (terkecil ke terbesar)

KESIMPULAN

Shell sort adalah salah satu dari banyaknya algoritma sorting, shell sort dapat dikatakan efisien dan cepat. Cara kerja shell sort yaitu dengan membagi array menjadi dua set lalu membandingkannya dengan jarak yang disebut interval, yaitu jarak antar array yang dibandingkan. Algoritma shell sort menggunakan sebuah increment yang disebut gap.

Gap pada Shell sort mengatur jarak antara elemen yang dibandingkan dan dipindahkan dalam array. Awalnya, gap diatur dengan nilai yang lebih besar dari ukuran array, dan kemudian gap ini dikurangi secara berulang-ulang sampai nilainya menjadi 1. Proses ini disebut sebagai fase shell.

Dengan memahami konsep dan prinsip dasar dari algoritma Shell sort serta bagaimana mengimplementasikannya secara efektif, penulis berharap pembaca dapat meningkatkan efisiensi pengolahan data pada program yang digunakan dan meningkatkan kecepatan program. Meski begitu penulis dengan sadar bahwa laporan ini belum sempurna, oleh karena itulah penulis berharap untuk mendapat kritik serta saran yang membangun demi kesempurnaan laporan yang sama dimasa mendatang.