JUSIKOM PRIMA (Junal Sistem Informasi Ilmu Komputer Prima)
Vol. 2 No. 1, Juli 2018 e-ISSN : 2580-2879

OPTIMASI ALGORITMA *SHELL SORT* DALAM PENGURUTAN DATA HURUF DAN ANGKA

Heru Satria Tambunan[1], Sumarno[2], Indra Gunawan[3], Eka Irawan[4]

STIKOM Tunas Bangsa

[1,4]Program Studi Sistm Informasi, [2,3]Program Studi Teknik Informatika

heru@amiktunasbangsa.ac.id[1], sumarno@amiktunasbangsa.ac.id[2], indra@amiktunasbangsa.ac.id[3], eka.irawan@amiktunasbangsa.ac.id[4]

ABSTRAK

Algoritma merupakan peran penting didalam dunia komputer yang tujuan untuk membantu cara kerja. Algoritma yang diterapkan diperangkat lunak sangat begitu penting. Dari segi perkembangan kemajuan teknologi banyak keinginan yang ingin dicapai untuk menyelesaikan permasalahan dari berbagai metode-metode yang diterapkan untuk pemecahan permasalahan, apalagi dalam masalah pengurutan sebuah data, dari data yang diterima mendapatkan data yang berbentuk acakan. setiap perangkat lunak yang dibangun menggunakan algoritma yang digunakan pasti membutuhkan sebuah data. Didalam penelitian yang dikemukakan oleh penulis yaitu optimasi algoritma *shell sort* dan pengurutan data huruf dan angka. Didalam pengurutan huruf dan angka menggunakan aplikasi *DEV C++* menggunakan bahasa pemrograman C.

Keyword : Algoritma *Quick sort*, algoritma *Shell Sort*, *Dev C++*

1. PENDAHULUAN

1.1. Latar Belakang

Disetiap algoritma dalam perancangan perangkat lunak pasti membutuhkan sebuah data, baik data analog maupun data digital, yang mana data tersebut banyak berupa numeric bahkan huruf yang aan dikelolah sehingga menjadi hasil yang diinginkan. Dari data yang digunakan berupa data acak, yang mana data tersebut sulit untuk diurutkan dikarena adanya data berupa huruf dan angka.

Penguruatan data atau yang disebut dengan *sorting* proses menyusun kembali data berupa numerik ataupun karakter yang sebelumnya telah disusun dengan suatu pola tertentu, sehingga tersusun secara teratur menurut aturan tertentu. Baik itu secara ascending maupun dengan descending

Beberapa metode pengurutan data atau sorting yaitu Pengurutan berdasarkan perbandingan (*comparison-based sorting)* yang memiliki jenis pengurutan yakni Bubble sort, exchange sort. Untuk pengurutan berdasarkan prioritas (*priority queue sorting method)* yakni *Selection sort, heap sort* (menggunakan tree), untuk pengurutan berdasarkan penyisipan dan penjagaan terurut (*insert and keep sorted method*) yakni *Insertion sort, tree sort*, sedangkan pengurutan berdasarkan pembagian dan penguasaan (*devide and conquer method*) yakni *Quick sort, merge sort*, dan yang teakhir adalah pengurutan berkurang menurun (*diminishing increment sort method*) yaitu *Shell sort (pengembangan insertion)* yang mana masing-masing pengurutan tersebut memiliki kelebihan.

Algoritma *Shell Sort* merupakan juga dengan metode pertambahan menurun (diminishing increment). Metode ini dikembangkan oleh Donald L. Shell pada tahun 1959, sehingga sering disebut dengan Metode Shell Sort. Metode ini mengurutkan data dengan cara membandingkan suatu data dengan

data lain yang memiliki jarak tertentu, kemudian dilakukan penukaran bila diperlukan.

1.2. Tujuan Penulisan

Dari penjelasan diatas, tujuan yang ingin dicapai dari penelitian ini adalah :

1. Menerapkan algoritma Shell Sort dalam pengurutan huruf maupun angka.
2. Mengetahui bagaimana kecepatan pengurutan huruf dan angka.

1.3. Batasan Masalah

Adapun batasan permasalahan dalam penelitian ini adalah sebagai berikut:

1. Algoritma *shell sort* yang digunakan untuk *sorting* huruf dan angka
2. Menggunakan Aplikasi dev C++ pada Pemograman C++.
3. Pengujian dalam pengurutan ini hanya menggunakan algoritma *shell Sort*

2. LANDASAN TEORI

2.1. Algoritma

Algoritma sorting adalah kumpulan langkah-langkah penyelesaian dalam suatu masalah dengan metode tertentu, sedangkan sorting didefinisikan sebagai pengurutan sejumlah data berdasarkan nilai kunci tertentu untuk mengurutkan nilai dari yang terkecil *(ascending)* atau sebaliknya *(descending)* [1].

2.2. Kompleksitas Algoritma

Kompleksitas suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk mendapatkan hasil yang diinginkan.

Hal-hal yang mempengaruhi kompleksitas waktu:

1. Jumlah masukan data untuk suatu algoritma (n).
2. Waktu yang dibutuhkan unutk menjalankan algoritma tersebut.
3. Ruang memori yang dibutuhkan untuk menjalankan algoritma yang berkaitan dengan struktur data dari program.

Kompleksitas mempengaruhi performa atau kinerja dari suatu algoritma. Kompleksitas dibagi menjadi 3 jenis yaitu, worst case, best case, dan average case. Masing-masing jenis kompleksitas ini menunjukkan kecepatan atau waktu yang dibutuhkan algoritma untuk mengeksekusi sejumlah kode.[2]

2.3. Algoritma *Shell Sort*

Algoritma *Shell Sort* merupakan juga dengan metode pertambahan menurun (diminishing increment). Metode ini dikembangkan oleh Donald L. Shell pada tahun 1959, sehingga sering disebut dengan Metode Shell Sort. Metode ini mengurutkan data dengan cara membandingkan suatu data dengan data lain yang memiliki jarak tertentu, kemudian dilakukan penukaran bila diperlukan.

Didalam proses pengurutan dengan metode Shell dapat dijelaskan sebagai berikut :

1. Menentukan jarak mula-mula dari data yang akan dibandingkan, yaitu N / 2.
2. Data pertama dibandingkan dengan data dengan jarak N / 2. Apabila data pertama lebih besar dari data ke N / 2 tersebut maka kedua data tersebut ditukar.
3. Kemudian data kedua dibandingkan dengan jarak yang sama yaitu N / 2.
4. Demikian seterusnya sampai seluruh data dibandingkan sehingga semua data ke-j selalu lebih kecil daripada data ke-(j + N / 2).

Untuk proses selanjutnya *Shell Sort* menggunakan jarak (N / 2) / 2 atau N / 4.

1. Data pertama dibandingkan dengan data dengan jarak N / 4. Apabila data pertama lebih besar dari data ke N / 4 tersebut maka kedua data tersebut ditukar.
2. Kemudian data kedua dibandingkan dengan jarak yang sama yaitu N / 4.

3. Demikianlah seterusnya hingga seluruh data dibandingkan sehingga semua data ke-j lebih kecil daripada data ke-(j + N / 4).
4. Pada proses berikutnya, digunakan jarak (N / 4) / 2 atau N / 8.
5. Demikian seterusnya sampai jarak yang digunakan adalah 1.

Atau jarak yang digunakan untuk *shell sort*adalah :

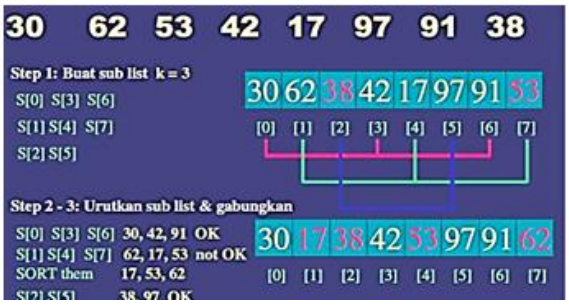
Jarak yang digunakan disebut increment value, atau sequence number k Misal sekuens: 5,3,1.

Pengambilan sekuens bebas, asal menurun

a. Jika k=5, sublistnya:
   1) Data[0], Data[5], Data[10], dst
   2) Data[1], Data[6], Data[11], dst
   3) Data[2], Data[7], Data[12], dst
b. Jika k=3, sublistnya:
   1) Data[0], Data[3], Data[6], dst
   2) Data[1], Data[4], Data[7], dst
   3) Data[2], Data[5], Data[8], dst

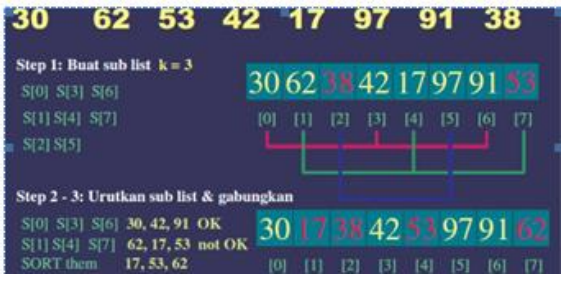
Contoh langkah perhitungan *shell sort*

Proses pertama, jarak=(N/2)+1=(8/2)+1=5



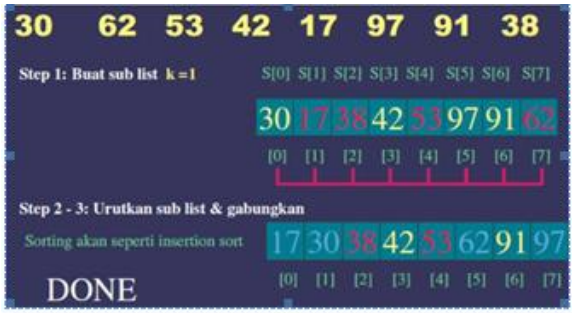
Gambar 1. Proses Perhitungan Jarak Pertama

Proses kedua, jarak=(N/4)+1=(8/4)+1=3



Gambar 2. Proses Perhitungan Jarak Kedua

Proses Ketiga jarak=(N/8)=(8/4)=1



Gambar 3. Proses Perhitungan Jarak Ketiga

Adapun kelebihan dan kekurangan dari algoritma *Shell Sort* yaitu:

Kelebihan dari algoritma *shellsort* yaitu:

1. Algoritma ini sangat rapat dan mudah diimplementasikan.
2. Operasi pertukarannya hanya dilakukan sekali saja
3. Waktu pengurutannya dapat lebih ditekan.
4. Mudah menggabungkannya kembali
5. Kompleksitas selection sort relatif lebih kecil

Kekurangan dari algoritma *shellsort* yaitu:

1. Membutuhkan method tambahan
2. Sulit untuk membagi masalah.

3. HASIL DAN PEMBAHASAN

Hasil dan pembahasan ini menggunakan perangkat lunak *Dev C++* dengan bahasa pemrograman C untuk mengurutkan dan mendapatkan waktu pengurutannya.

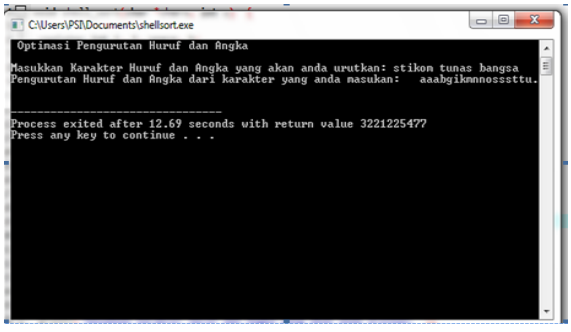
3.1. Pseudocode *Shell Sort*

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
void shell_sort(char *chars, int c)  {

 register int i, j, space, k;
 char x, a[5];
```

```
 a[0]=9; a[1]=5; a[2]=3; a[3]=2; a[4]=1;

 for(k=0; k < 5; k++) {
  space = a[k];
  for(i=space; i < c; ++i) {
   x = chars[i];
   for(j=i-space; (x < chars[j]) && (j >= 0); j=j-space)
    chars[j+space] = chars[j];
   chars[j+space] = x;
  }
 }
}

int main() {
 char string[3];
 printf (" Optimasi Pengurutan Huruf dan Angka \n\n");

 printf("Masukkan Karakter Huruf dan Angka yang akan anda urutkan: ");
 gets(string);
 shell_sort(string, strlen(string));
 printf("Pengurutan Huruf dan Angka dari karakter yang anda masukan: %s.\n", string);
 return 0;
}
```

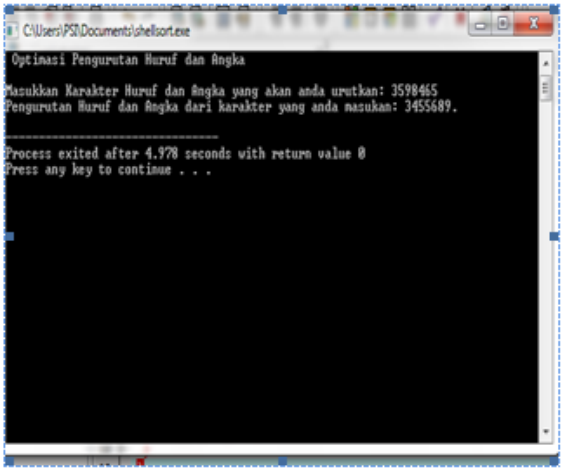
3.2. Pengujian

Dari *pseudocode* sebelumnya maka hasil pengujian dari data yang di *input* berbeda-beda sebagai berikut :



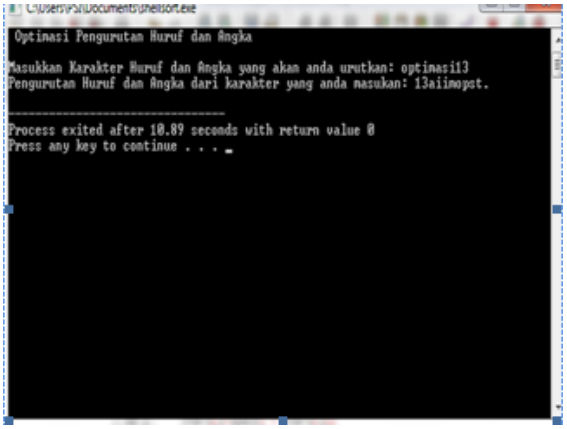
Gambar 4. Hasil pengurutan huruf

Dari pengurutan huruf dari gambar diatas membutuhkan waktu 12.69 second untuk mengurutkan stikom tunas bangsa dengan jumlah karakter 19.



Gambar 5. Pengurutan Angka

Sedangkan dalam pengurutan angka, dari jumlah angka sebanyak 7 karakter membutuhkan waktu 4.978 second untuk mengurutkan data awal 3598465



Gambar 6. Pengurutan Huruf dan Angka

Sedangkan pengurutan huruf dan angka dari data optimasi13 pada gambar diatas membutuhkan waktu 10.89 *second* untuk mengurutkan dengan jumlah karakter 10.

3.3. Perangkat yang digunakan

Pengujian dilaksanakan dengan menggunakan compiler Bloodshed *Dev-C++* versi 5.11 pada

platform Windows 7 dan komputer PC dengan spesifikasi:

a. Intel(R) Core(TM) i3-2100 CPU @3.10GHz
b. Memory 2GB
c. System operasi 32 Bit
d. HDD 500GB

4. KESIMPULAN

Dari hasil implementasi maupun pengujian yang telah dilakukan dapat ditarik beberapa kesimpulan sebagai:

1. Pengujian dilakukan dari beberapa data yang berbeda disetiap pengujian.
2. Pengujian yang dilakukan hanya menggunakan algoritma *shell sort* untuk huruf dan angka.
3. Pengujian menggunakan bahasa pemrograman bahasa C dan program *dev C++*

DAFTAR PUSTAKA

[1] Wisudawan, Wahyu Fahmy, 2007. Kompleksitas Algoritma Sorting yang Populer Dipakai. Bandung: Teknik Informatika, Institut Teknologi Bandung

[2] Anisyah S, Febrian N, "analisis perbandingan algoritma *bubble sort, merge sort,* dan *quick sort* dalam proses pengurutan kombinasi angka dan huruf". Jurnal Pseudocode, Vol II No. 2, September 2015