**Analisis Algoritma**

**Perbandingan marge sort dan insertion sort**

****

**Disusun Oleh :**

NAMA : Immanuel K.P Rini

NIM : 1306081007

**LABORATORIUM KOMPUTER**

**JURUSAN ILMU KOMPUTER**

**FAKULTAS SAINS DAN TEKNIK**

**UNIVERSITAS NUSA CENDANA**

**KUPANG 2015**

**I.Pendahuluan**

**1.Latar Belakang**  
*Sorting* atau pengurutan adalah proses menyusun elemen – elemen dari masukan awal acak menjadi keluaran akhir tertata dengan urutan tertentu]. Proses tersebut diimplementasikan dalam bermacam aplikasi.Contoh penerapannya antara lain berupa rincian transaksi sesuai urutan tanggal dan jam pada perbankan, daftar hadir yang diurutkan berdasarkan nomor induk dan daftar pustaka yang diurutkan sesuai abjad pengarang ataupun katalog buku di perpustakaan. Fungsi-fungsi statistik seperti median dan pembuatan kuartil data (*quarter*), desil dan percentil (*percentile*)mensyaratkan data untuk diurutkan terlebih dahulu.Beberapa macam algoritma sorting telah dibuat karena proses tersebut sangat mendasar dan sering digunakan. Oleh karena itu,pemahaman atas algoritma – algoritma yang  
ada sangatlah berguna.

Ada dua macam urutan yang biasa digunakan dalam proses pengurutan yaitu:  
• urut naik (ascending) yaitu dari data yang mempunyai nilai paling kecil sampai paling besar  
• urut turun (descending) yaitu data yang mempunyai nilai paling besar sampai paling kecil.

Keuntungan dari data yang sudah dalam keadaan terurutkan antara lain :  
• data mudah dicari (misalnya dalam buku telepon atau kamus bahasa), mudah untuk dibetulkan, dihapus, disisipi atau digabungkan. Dalam keadaan terurutkan, kita  
mudah melakukan pengeekan apakah ada data yang hilang  
• melakukan komppilasi program komputer jika tabel-tabel simbol harus dibentuk  
• mempercepat proses pencarian data yang harus dilakukan berulang kali.

**II.Materi**

**2.Pengertian**

**Insertion Sort**  
Salah satu algoritma sorting yang paling sederhana adalah insertion sort. Insertion Sort disebut-sebut sebagai metode pertengahan. Artinya, metode ini memiliki kecepatan ratarata antara metode primitif (bubble dan selection) dan modern (merge dan quick).Metode ini, didasarkan pada sebuah kunci  
yang diambil pada elemen ke-2 pada sebuah larik, lalu menyisipkan elemen tersebut jika kondisi percabangan terpenuhi. Metode penyisipan (insertion) bertujuan untuk menjadikan bagian sisi kiri larik terurutkan sampai dengan seluruh larik berhasil diurutkan.Algoritma untuk pemogramannya adalah sebagai berikut :

1. Simpan nilai Ti kedalam variabel sementara, dengan i = 1.
2. Bandingkan nilainya dengan elemen sebelumnya.
3. Jika elemen sebelumnya (Ti-1) lebih besar nilainya daripada Ti, maka tindih nilai Ti dengan nilai Ti-1 tersebut. Decrement i (kurangi nilainya dengan 1).
4. Lakukan terus poin ke-tiga, sampai Ti-1 ≤ Ti.
5. Jika Ti-1 ≤ Ti terpenuhi, tindih nilai di Ti dengan variabel sementara yang disimpan sebelumnya.
6. Ulangi langkah dari poin 1 di atas dengan i di-increment (ditambah satu).

**Merge Sort**  
Merge sort adalah metode pengurutan yang menggunakan pola divide and conquer[10].  
Strateginya adalah dengan membagi sekelompok data yang akan diurutkan menjadi  
beberapa kelompok kecil terdiri dari maksimal dua nilai untuk dibandingkan dan digabungkan  
lagi secara keseluruhan.  
Langkah kerja dari Merge sort[9]:  
1. Divide  
Memilah elemen – elemen dari rangkaian data menjadi dua bagian dan mengulangi pemilahan hingga satu elemen terdiri maksimal dua nilai.  
2. Conquer  
Mengurutkan masing-masing elemen.  
3. Kombinasi  
Mengkombinasikan dua bagian tersebut secara rekursif untuk mendapatkan rangkaian  
data berurutan. Proses rekursi berhenti jika mencapai elemen dasar. Hal ini terjadi bilamana bagian yang akan diurutkan menyisakan tepat satu elemen. Sisa pengurutan satu elemen tersebut menandakan bahwa bagian tersebut telah terurut sesuai rangkaian. Algoritma Merge sort sebagai berikut :

1. Bagi list yang tak terurut menjadi dua sama panjang atau salah satunya lebih panjang satu elemen.
2. Bagi masing-masing dari 2 sub-list secara rekursif sampai didapatkan list dengan ukuran 1.
3. Gabung 2 sublist kembali menjadi satu list terurut.

III.Implimentasi

*Pseudocode* untuk merge sort adalah sebagai berikut :

*#include <stdio.h>*

*#include <stdlib.h>*

*#define MAX 13*

*int Data[MAX];*

*int temp[MAX];*

*void merge(int Data[], int temp[], int kiri, int tengah, int kanan)*

*{*

*int i, left\_end, num\_elements, tmp\_pos;*

*left\_end = tengah - 1;*

*tmp\_pos = kiri;*

*num\_elements = kanan - kiri + 1;*

*while ((kiri <= left\_end) && (tengah <= kanan))*

*{*

*if (Data[kiri] <= Data[tengah])*

*{*

*temp[tmp\_pos] = Data[kiri];*

*tmp\_pos = tmp\_pos + 1;*

*kiri = kiri +1;*

*}*

*else*

*{*

*temp[tmp\_pos] = Data[tengah];*

*tmp\_pos = tmp\_pos + 1;*

*tengah = tengah + 1;*

*}*

*}*

*while (kiri <= left\_end)*

*{*

*temp[tmp\_pos] = Data[kiri];*

*kiri = kiri + 1;*

*tmp\_pos = tmp\_pos + 1;*

*}*

*while (tengah <= kanan)*

*{*

*temp[tmp\_pos] = Data[tengah];*

*tengah = tengah + 1;*

*tmp\_pos = tmp\_pos + 1;*

*}*

*for (i=0; i <= num\_elements; i++)*

*{*

*Data[kanan] = temp[kanan];*

*kanan = kanan - 1;*

*}*

*}*

*void m\_sort(int Data[], int temp[], int kiri, int kanan)*

*{*

*int tengah;*

*if (kanan > kiri)*

*{*

*tengah=(kanan + kiri) / 2;*

*m\_sort(Data, temp, kiri, tengah);*

*m\_sort(Data, temp, tengah+1, kanan);*

*merge(Data, temp, kiri, tengah+1, kanan);*

*}*

*}*

*void mergeSort(int Data[], int temp[], int array\_size)*

*{*

*m\_sort(Data, temp, 0, array\_size - 1);*

*}*

*int main()*

*{*

*int i,Data[13]={8,-1,7,22,8,61,40,-56,22,91,5,613};*

*printf("\nDATA SEBELUM TERURUT : \n");*

*for (i = 0; i <=*

*MAX; i++)*

*{*

*printf (" Data ke %d:\t %d\n",i+1, Data[i]);*

*}*

*mergeSort(Data, temp, MAX);*

*printf("\nDATA SETELAH TERURUT : ");*

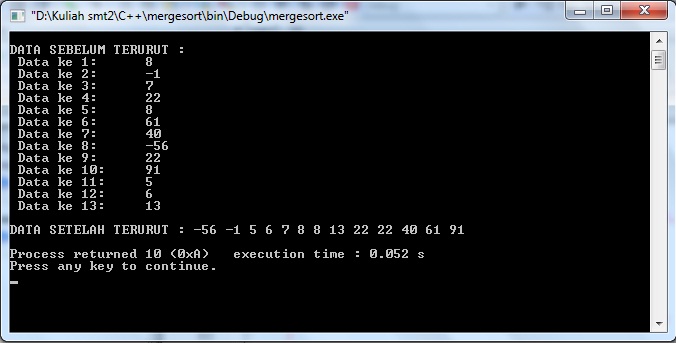
*for (i = 0; i <= MAX; i++)*

*printf("%d ", Data[i]);*

*printf("\n");*

*}*

Hasil pemogramannya :



*Pseudocode* untuk insertion sort adalah sebagai berikut :

*#include <stdio.h>*

*#include <stdlib.h>*

*void insertion\_sort (int \*a, int n)*

*{*

*int i, j, t;*

*for (i = 1; i < n; i++)*

*{*

*t = a[i];*

*for (j = i; j > 0 && t < a[j - 1]; j--) {*

*a[j] = a[j - 1];*

*}*

*a[j] = t;*

*}*

*}*

*int main ()*

*{*

*int a[13] = {8, -1, 7, 22, 8, 61, 40, -56, 22, 91, 5, 6, 13};*

*int n = sizeof a / sizeof a[0];*

*int i;*

*printf("\nDATA SEBELUM TERURUT : \n");*

*for (i = 0; i < n; i++)*

*{*

*printf("\n %d%s", a[i], i == n - 1 ? "\n" : " ");*

*}*

*insertion\_sort(a, n);*

*printf("\nDATA SETELAH TERURUT : \n");*

*for (i = 0; i < n; i++)*

*{*

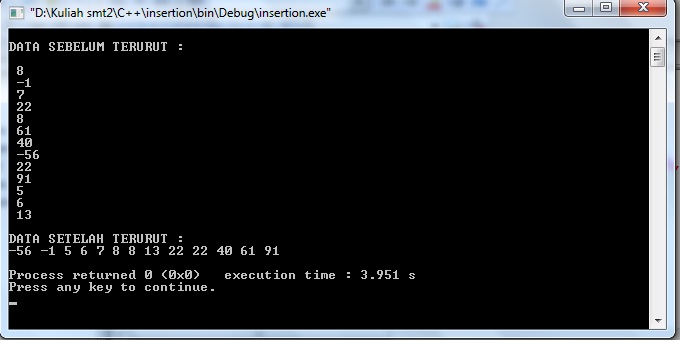
*printf("%d%s", a[i], i == n - 1 ? "\n" : " ");*

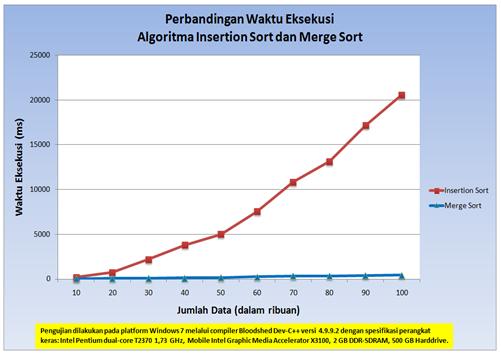
*}*

*return 0;*

*}*

Hasil pemograman :



Hasil perbandingan waktu eksekusi antara marge soert dan insertion sort:

IV.Kesimpulan .

Dari hasil implementasi maupun pengujian yang telah dilakukan dapat ditarik beberapa kesimpulan sebagaimana berikut ini:

1.Dari hasil pengujian diketahui bahwa algoritma merge sort lebih cepat dibandingkan insertion sort untuk data yang lebih banyak, khususnya untuk jumlah data > 10000.  
2. Kelemahan utama merge sort adalah algoritma ini membutuhkan setidaknya ruang atau memori dua  
kali lebih besar karena dilakukan secara rekursif dan memakai dua elemen terpisah.  
3. Pada kasus *best case*, algoritma insertion sort lebih unggul daripada merge sort, sehubungan dengan  
kompleksitas yang lebih rendah yaitu nilai O(n) dibandingkan dengan O(n log n).  
4. Pada kasus *average* maupun *worst* *case*, algoritma merge sort lebih unggul terhadap insertion sort,  
sehubungan dengan kompleksitas yang lebih rendah yaitu nilai O (n log n), dibandingkan dengan O (n2).  
5. Algoritma insertion sort secara teknis lebih mudah diterapkan dibandingkan dengan merge sort, berkaitan dengan panjangnya instruksi yangdiperlukan.