MAKALAH SORTING DALAM PEMROGRAMAN

BAHASA C



Oleh :

DIKI CANDRA

NIM 2022903430010

D4 TEKNOLOGI REKAYASA KOMPUTER JARINGAN

POLITEKNIK NEGERI LHOKSEMAWE

2022/2023

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pengurutan atau sorting merupakan jenis operasi penting dalam pengolahan data. Hampir setiap saat dalam kehidupan sehari-hari selalu menjumpai permasalahan yang harus diselesaikan dengan melibatkan operasi pengurutan data. Begitu pentingnya operasi tersebut, sehingga sampai saat ini telah banyak dikembangkan metode-metode pengurutan data dan mungkin akan tetap bermunculan metode-metode baru. Pengurutan data juga merupakan salah satu proses yang sangat dibutuhkan di dalam pemrograman. Sorting atau pengurutan ini adalah proses mengatur sekumpulan objek menurut urutan atau susunan tertentu. Adanya kebutuhan akan pengurutan melahirkan beberapa macam pengurutan. Metodemetode pengurutan antara lain, yaitu bubble sort, selection sort, insertion sort, quick sort, merge sort dan lain sebagainya (Kadir dkk, 2005). Dari kumpulan data yang disimpan dapat mempunyai tipe data yang berbeda-beda dan pengurutan terhadap data tersebut dapat dilakukan untuk satu atribut atau lebih tergantung pada kebutuhan yang diinginkan.

Contoh Penerapannya

antara lain berupa diurutkan berdasarkan nomor induk, menentukan daftar :

* rangking
* hadir
* yang
* tertinggi
* nilai

**Sorting merupakan** suatu proses penyortiran atau pengurutan sebuah data.

Terdapat 2 macam pengurutan data pada sorting yaitu :

* 1. Berdasarkan ascending (kecil ke besar).
* 2. Berdasarkan Descending (besar ke kecil).

1.2 Rumusan Masalah

Berdasarkan judul dari makalah ini, Saya merumuskan masalah yaitu Pointer dalam bahasa pemrograman bahasa C.

1.3 Manfaat

Manfaat yang diperoleh dari makalah tersebut adalah sebagai berikut:

1.Pembaca

* Menjadikan pembaca makalah semata-mata agar pembaca memahami dasar-dasar pemrograman.
* Untuk memotivasi pembaca untuk membuat makalah yang lebih baik dari Ini.

2.Pengarang

* Untuk memberikan semangat pada mata kuliah yang sedang diambil saat ini.
* Sebagai titik awal untuk melakukan pembelajaran agar pembelajaran selanjutnya dapat dilakukan dengan lebih baik lagi.

1.4 Tujuan

* Mengetahui Pointer.
* Dapat menerapkan Bahasa.

BAB II

PEMBAHASAN

2.1 PENGENALAN SORTING

**1. Bubble sort**

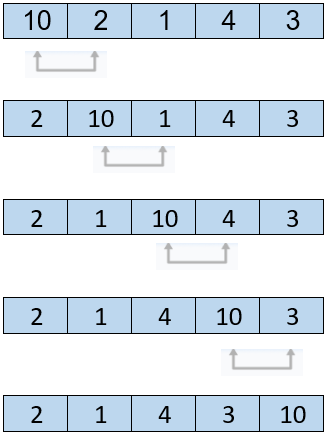
Bubble sort merupakan algoritma pengurutan yang membandingkan dua data yang berdekatan dan menukarnya sampai tidak dalam urutan yang diinginkan.

Cara kerja dari Bubble Sort

Bubble sort menggunakan teknik iterasi. Iterasi merupakan proses melakukan perulangan sebanyak data yang diketahui. Intinya pada iterasi melakukan perbandingan antara dua data.

Misalkan kita mencoba mengurutkan data dalam urutan kecil ke besar (Ascending).

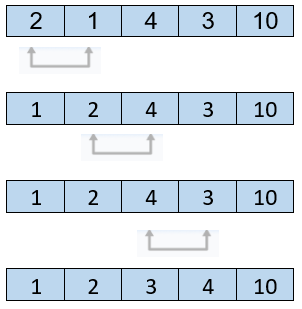
1. Iterasi Pertama (Bandingkan dan Tukar)



Penjelasan :

* Mulai dari indeks pertama, bandingkan data pertama dan kedua.
* Jika data pertama lebih besar dari data kedua, mereka ditukar. Karena contoh tersebut mengurutkan data berdasarkan kecil ke besar (ascending).
* Sekarang, bandingkan data kedua dan ketiga. Tukarkan jika tidak berurutan.
* Proses di atas berlangsung hingga elemen terakhir.

2. Iterasi Kedua (Melanjutkan proses iterasi dari iterasi pertama)

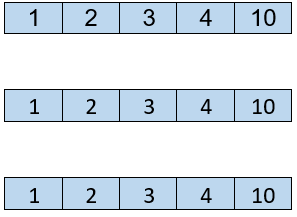


Penjelasan :

* Proses iterasi kedua melanjutkan hasil dari iterasi pertama.
* Proses pertukaran datanya pun sama saja. Jika data yang dibandingkan sudah benar maka tidak perlu terjadi pertukaran.

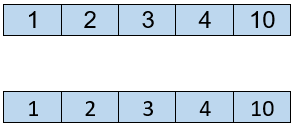
3. Iterasi Ketiga

* Walaupun sudah ke sorting iterasi tetap berlanjut
* Karena iterasi menyesuaikan banyaknya data (contoh soal : 10 2 1 4 3).
* Karena ada 5 data maka harus 5-1 = 4 kali iterasi.
* intinya pada iterasi membandingkan 2 data terus menerus.



4. Iterasi Keempat

* Iterasi keempat penjelasannya sama seperti dengan iterasi ketiga, yaitu banyaknya iterasi bergantung pada banyaknya data.



Kurang lebih seperti itu teori dari penggunaan bubble sort. Kamu dapat mengurutkan data dari kecil ke besar (ascending) atau dari besar ke kecil (descending). Prosesnya sama saja yaitu membandingkan dua buah data secara terus menerus.

Contoh program Bubble sort :

#include<stdio.h>

//Melakukan pertukaran jika data tidak sesuai.

void swap(int \*a, int \*b){

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void bubbleSort(int arr[], int N){

for(int i =0; i<N-1; i++){ //N-1 karena index dimulai dr 0.

//Melakukan looping untuk mengakses array pada setiap data.

for(int j=0; j<N-1-i; j++){

//Tanda > mengurutkan berdasarkan ascending

//Tanda < mengurutkan berdasarkan descending

if(arr[j] > arr[j+1]){

//Memanggil function swap yang berfungsi untuk menukar data.

swap(&arr[j], &arr[j+1]);

}

}

}

}

void print(int arr[], int N){

for(int i=0; i<N; i++){

printf("%d ", arr[i]);

}

}

int main(){

int arr[] = {10, 2, 1, 4, 3};

//Mencari panjang array dari data tersebut.

int N = sizeof(arr)/sizeof(arr[0]);

printf("Sebelum Sort: ");

print(arr,N);

puts("");

printf("Sesudah Sort: ");

bubbleSort(arr, N);

print(arr, N);

puts("");

return 0;

}

Penjelasan :

* Pada function swap tedapat 2 variabel dalam parameter, yaitu variabel int \*a dan int \*b yang berfungsi sebagai dua buah data. Terdapat juga variabel penampung yaitu int temp. Function swap berisi pertukaran data a dengan b.
* Pada function bubblesort berfungsi melakukan looping dan eksekusi program bubblesort.
* int N berfungsi sebagai ukuran array atau banyaknya data.
* Pada function print hanya berfungsi untuk mencetak data.
* Pada function main terdapat data yang ingin di sorting. Lalu terdapat pencarian panjang array dan terdapat pemanggilan function sehingga dapat mengeluarkan output.

Output :



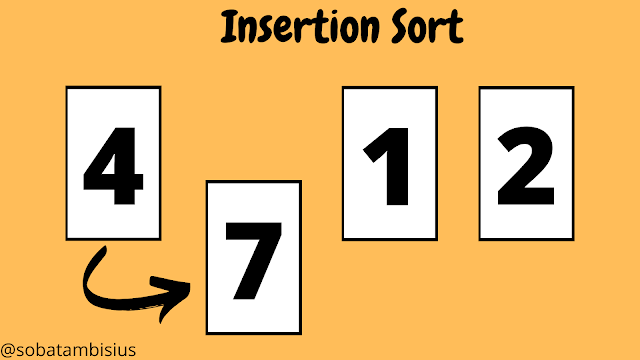
**2. Insertion sort**

Insertion sort merupakan teknik sorting dengan cara menyisipkan atau memasukan setiap elemen secara berulang berulang.

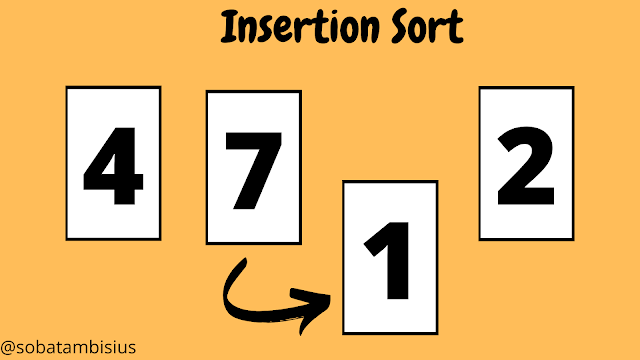
Konsep insertion sort bisa diibaratkan sebuah kartu. Untuk lebih jelasnya perhatikan gambar berikut.

Contohnya kita melakukan sort berdasarkan kecil ke besar (ascending) 4 buah data yaitu {4,7,1,2}.

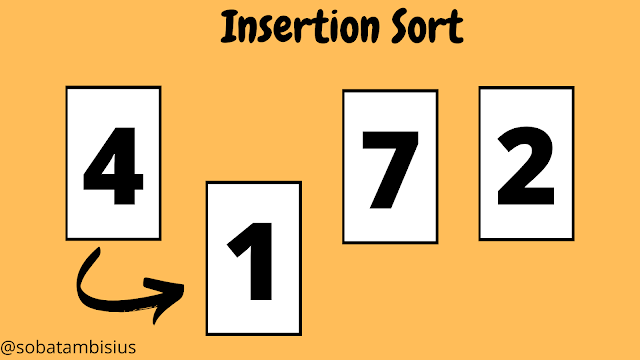
* Langkah pertama bandingkan nomor 4 dengan 7, karena 4 < 7 maka tidak perlu diubah.



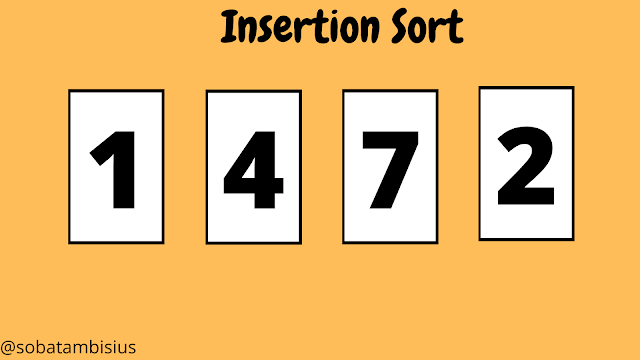
* Langkah kedua bandingkan nomor 7 dengan 1, karena 7 > 1 maka ubah posisi.



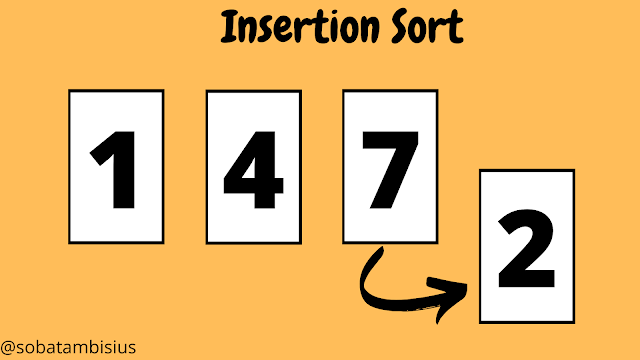
* Langkah ketiga bandingkan nomor 4 dengan 1, karena 4 > 1 maka ubah posisi.



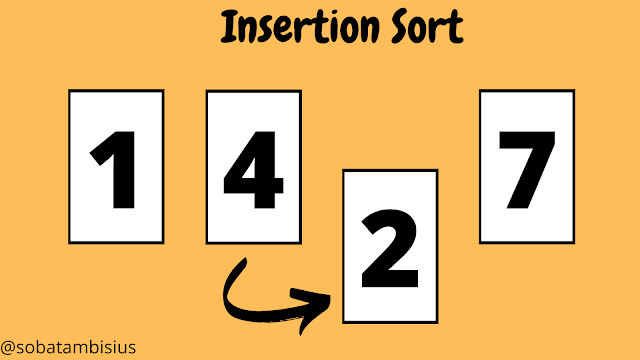
* Langkah keempat cari angka berikutnya yang belum terurut.



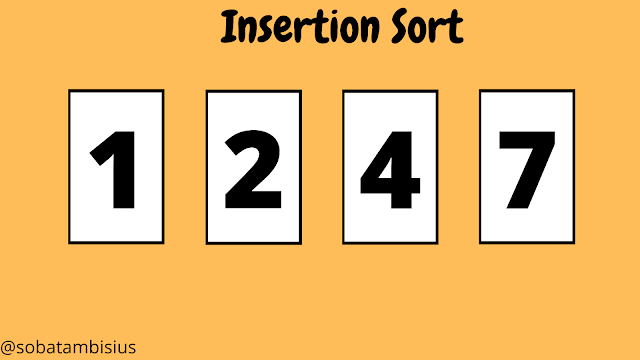
* Langkah kelima bandingkan nomor 7 dengan 2, karena 7 > 2 maka ubah posisi.



* Langkah keenam bandingkan nomor 4 dengan 2, karena 4 > 2 maka ubah posisi.



* Jika data sudah berurut maka tidak ada yang perlu dibandingkan dan sorting berhasil.



Contoh program Insertion sort :

#include<stdio.h>

void insertionSort(int \*Data, int n){

int temp, j;

for (int i = 1; i <= n-1; i++){ //n-1 karena index dimulai dr 0.

temp = Data[i];

j = i-1;

//Data[j] > temp berfungsi untuk mengurutkan data secara ascending.

//Data[j] < temp berfungsi untuk mengurutkan data secara descending.

while(j>=0 && Data[j] > temp){

Data[j+1] = Data[j];

j--;

}

Data[j+1] = temp;

}

}

void print(int\* Data, int n){

for (int i = 0; i < n; i++){

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

}

printf("\n");

}

int main(){

int Data[] = {4, 7, 1, 2};

//Mencari panjang array dan menyesuaikan ukuran array dari data tersebut.

int n = sizeof(Data)/sizeof(Data[0]);

printf("Sebelum Sort: ");

print(Data, n);

printf("Sesudah Sort: ");

insertionSort(Data, n);

print(Data, n);

return 0;

}

Penjelasan :

* Pada function Insertionsort berfungsi melakukan looping dan eksekusi program Insertionsort.
* int n berfungsi sebagai ukuran array atau banyaknya data.
* Pada function print hanya berfungsi untuk mencetak data.
* Pada function main terdapat data yang ingin di sorting. Lalu terdapat pencarian panjang array dan terdapat pemanggilan function sehingga dapat mengeluarkan output.

Output :



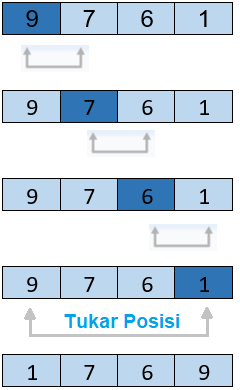
**3. Selection Sort**

Selection sort menggunakan perbandingan data, selection sort akan membandingkan data 1 dengan lainnya dan jika menemukan data yang terkecil maka data tersebut menjadi acuannya.

Contohnya kali ini kita akan mengerjakan teori dari selection sort. Contoh soal diketahui terdapat 4 buah data, dengan angka 9,7,6,1. Urutkanlah data tersebut berdasarkan ascending.

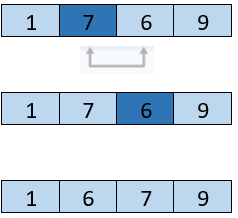
Langkah 1 :

* Gunakanlah data pertama yaitu 9 sebagai acuannya, lalu bandingkan dengan data kedua yaitu 7. Karena 7 < 9 maka sekarang 7 yang menjadi acuannya.
* Bandingkan angka 7 dengan data selanjutnya yaitu 6. Karena 6 < 7 maka angka 6 menjadi acuannya.
* Kemudian bandingkan angka 6 dengan data selanjutya yaitu 1. Karena 1 < 6 maka angka 1 menjadi acuannya.
* Lalu tukar angka 1 dengan angka 9 yang menjadi angka acuan awal.



Langkah 2 :

* Bandingkan data kedua yaitu 7 dengan data selanjutnya yaitu 6. Karena 6 < 7 maka angka 6 menjadi acuannya.
* Kemudian tukar angka 6 dan 7.



Contoh program Selection sort :

#include <stdio.h>

//Melakukan pertukaran jika data tidak sesuai.

void swap(int \*a, int \*b){

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void SelectionSort (int Data[ ], int n) {

// variabel sementara untuk menyimpan posisi data minimum

int min;

// mengurangi ukuran efektif array sebanyak satu di setiap iterasi.

for(int i = 0; i < n-1 ; i++) {

// anggap saja data pertama menjadi minimum dari array yang tidak disortir .

min = i ;

// memberikan ukuran efektif dari array yang tidak disortir.

for(int j = i+1; j < n ; j++ ) {

if(Data[ j ] < Data[ min ]) { //mencari data minimum

min = j ;

}

}

// meletakkan data minimum pada posisi yang tepat

swap (&Data[ min ], &Data[ i ]) ;

}

}

void print(int\* Data, int n){

for (int i = 0; i < n; i++){

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

}

printf("\n");

}

int main(){

int Data[] = {9, 7, 6, 1};

//Mencari panjang array dan menyesuaikan ukuran array dari data tersebut.

int n = sizeof(Data)/sizeof(Data[0]);

printf("Sebelum Sort: ");

print(Data, n);

printf("Sesudah Sort: ");

SelectionSort(Data, n);

print(Data, n);

return 0;

}

Penjelasan :

* Pada function Selectionsort berfungsi melakukan looping dan eksekusi program Selectionsort.
* int n berfungsi sebagai ukuran array atau banyaknya data.
* Pada function print hanya berfungsi untuk mencetak data.
* Pada function main terdapat data yang ingin di sorting. Lalu terdapat pencarian panjang array dan terdapat pemanggilan function sehingga dapat mengeluarkan output.

Output :



**4. Counting Sort**

Counting sort merupakan algoritma soring (pengurutan) yang dapat dilakukan apabila data dapat dinyatakan sebagai bilangan yang menyatakan besarannya. Rentang nilai bilangan itu tidak boleh terlalu besar, karena akan dibuat array yang besarnya sama dengan rentangnya. Ide Counting sort seperti kita melakukan perhitungan pemilu yaitu dengan mencatat frekuensi atau banyaknya kemunculan data. Namun metode ini hanya cocok digunakan bila data yang digunakan bertipe integer dan dibatasi pada range tertentu. Biasanya counting sort hanya digunakan untuk mengurutkan angka.

Berikut keunggulan dan kelemahan Algoritma Counting Sort berdasarkan hasil pengamatan kami:

Keungggulan Algoritma Counting Sort adalah dapat mengurutkan dengan waktu yang relatif lebih singkat, karena tidak membandingkan dengan elemen lain.

Kelemahan Algoritma Counting Sort adalah menggunakan array yang terlalu banyak.

Contoh Program Counting Sort dalam Bahasa C

#include <stdio.h>

int main()

{

int L[20],temp,i,j,n=6,idx;

printf("pengurutan berdasarkan Counting sort \nmasukkan %d elements: \n",n);

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

scanf("%d",&L[i]);}

printf("\nsebelum sorting: ");

for(i=0;i<n;i++){printf("%d ",L[i]);}

for (i=0;i<(n-1);i++){

/\*4 6 1 2 ===> 4 6 1 2, 4 6 1 2, 4 6 1 2, 1 6 4 2 // 1 6 4 2, 1 6 4 2, 1 2 4 6 // 1 2 4 6, 1 2 4 6\*/

idx=i;

for (j=i+1;j<n;j++){

if(L[j]<L[idx]){

idx=j;

}

}

temp=L[i];

L[i]=L[idx];

L[idx]=temp;

}

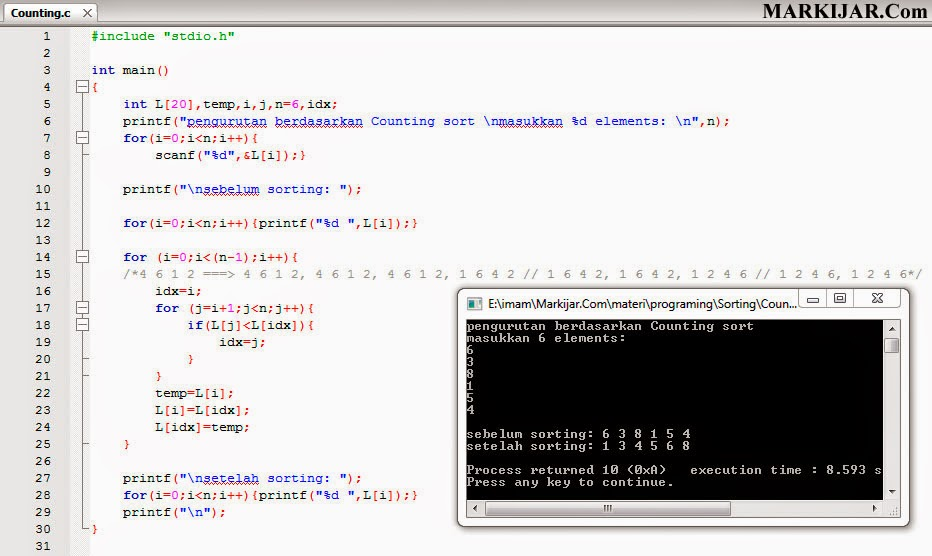
printf("\nsetelah sorting: ");

for(i=0;i<n;i++){printf("%d ",L[i]);}

printf("\n");

}

Output :



5. Merge Sort

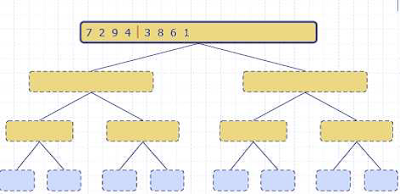
Metode pengurutan merge sort adalah metode pengurutan lanjut, sama dengan metode Quick Sort. Metode ini juga menggunakan konsep devideand conquer yang membagi data S dalam dua kelompok yaitu S1 dan S2 yang tidak beririsan (disjoint). Proses pembagian data dilakukan secara rekursif sampai data tidak dapat dibagi lagi atau dengan kata lain datadalam sub bagian menjadi tunggal.

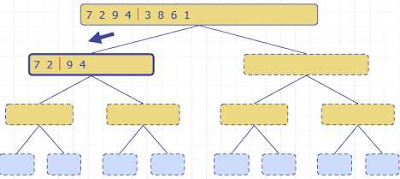
Setelah data tidak dapat dibagi lagi, proses penggabungan (merging) dilakukan antara sub-sub bagian dengan memperhatikan urutan data yang diinginkan (ascending/kecil ke besar atau descending/besar ke kecil). Proses penggabungan ini dilakukan sampai semua data tergabung dan terurut sesuai urutan yang diiginkan. Kompleksitas algoritma merge sort adalah O(n log n).

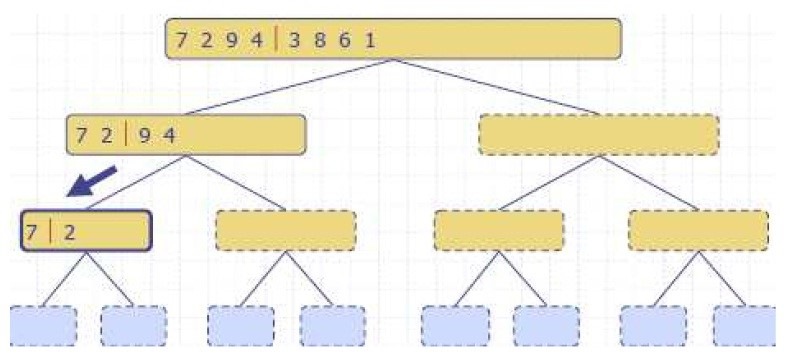
Cara kerja Merge Sort

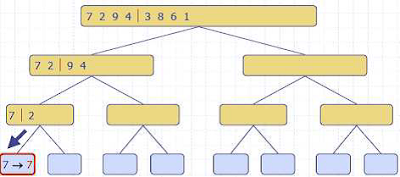
Merge Sort adalah membagi larik data yang diberikan menjadi dua bagian yang lebih kecil. Kedua larik yang baru tersebut kemudian akan diurutkan secara terpisah. Setelah kedua buah list tersusun, maka akan dibentuk larik baru sebagai hasil penggabungan dari dua buah larik sebelumnya.

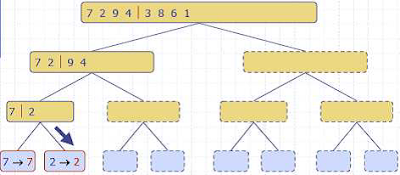
Ilustrasi Algoritma Merge Sort

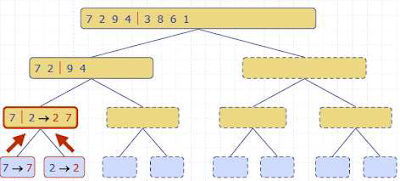


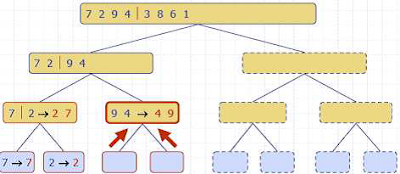
****

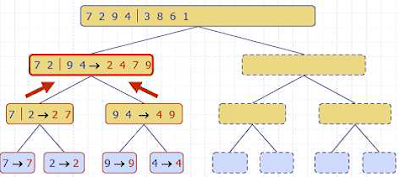
****

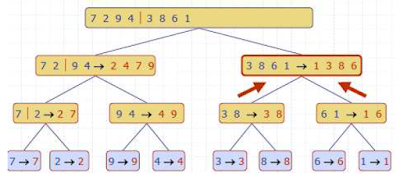


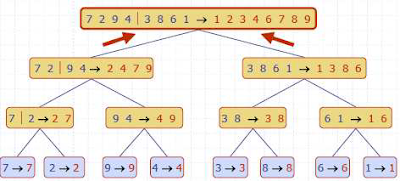












Contoh Program :

#include <stdio.h>

#define MAX 5

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;

printf("Masukkan DATA SEBELUM TERURUT : \n");

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

{

printf ("Data ke %i : ", i+1);

scanf ("%d", &Data[i]);

}

mergeSort(Data, temp, MAX);

printf("\nDATA SETELAH TERURUT : ");

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

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

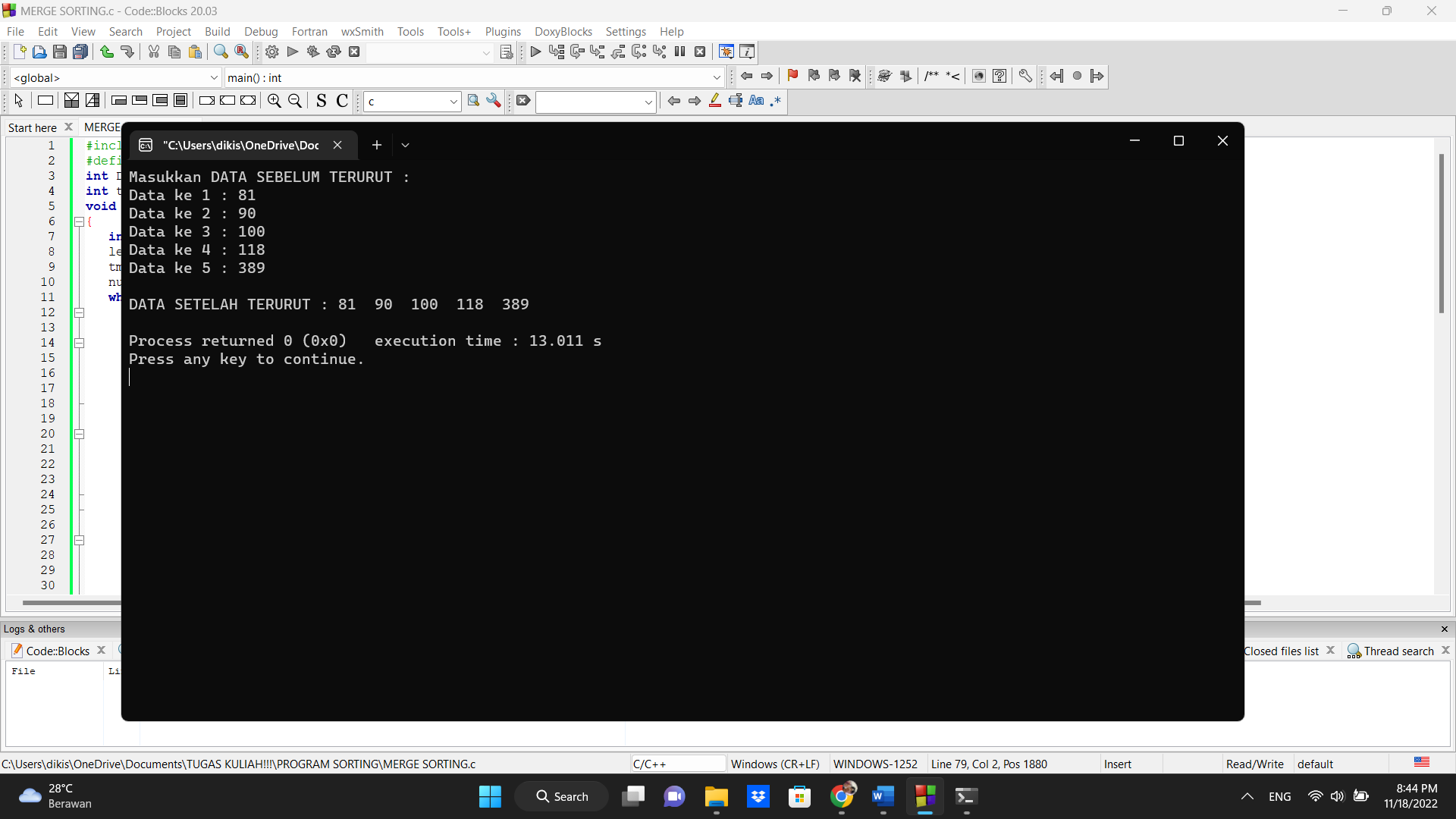
printf("\n");

//scanf("%d");

return(0);

}

Output :



**6. Quick Sort**

Pada pembahasan sebelumnya Anda telah mengetahui bahwa setidaknya ada dua metoda pengurutan yang populer, yaitu Bubble Sort dan Quick Sort. Sengaja metoda Bubble Sort dipilih untuk dibahas terlebih dahulu karena logikanya mudah untuk diikuti.

Nah, sekarang kita menginjak pada metoda pengurutan yang kedua yaitu Quick Sort. Metoda ini mungkin memang tidak semudah Bubble Sort, akan tetapi unggul dalam segi kecepatan, terutama apabila ada banyak data yang harus diurutkan.

Sebelum kita menginjak pada pembahasan tentang program pengurutan dengan metoda Quick Sort, terlebih dahulu akan diberikan ilustrasi terlebih dahulu.

Katakanlah Anda memiliki 7 buah angka yang harus diurutkan, yaitu 5, 1, 3, 10, 8, 6, dan 7. Lihat ilustrasi pada gambar 1. Dengan menggunakan metoda Quick Sort, langkah pertama yang harus diambil adalah menentukan elemen pivot. Elemen pivot adalah suatu data yang dipilih untuk menjadi dasar acuan dalam pengurutan data. Elemen ini boleh diambil sembarang, salah satu dari ketujuh data tersebut. Dalam contoh ini akan diambil angka 7 sebagai elemen pivot. Lihat ilustrasi pada gambar 2.

Langkah berikutnya adalah melakukan pembandingan seluruh elemen yang ada dengan elemen pivot tersebut. Sebagai hasil dari pembandingan tersebut, bilangan yang lebih kecil daripada elemen pivot akan diletakkan di sebelah kiri elemen pivot dan bilangan yang lebih besar daripada elemen pivot diletakkan di sebelah kanan elemen pivot. (Kasus ini berlaku untuk pengurutan dari bilangan terkecil ke bilangan terbesar atau ascending).

Proses pembandingan ini dapat dilakukan dari dua arah, yaitu dari arah kiri dan dari arah kanan. Nantinya dalam program, gerakan dari arah kiri akan dinotasikan dengan huruf i dan gerakan dari arah kanan akan dinotasikan dengan huruf j. Lihat ilustrasi pada gambar 3. Apabila elemen i memiliki nilai lebih besar daripada elemen pivot dan elemen j memiliki nilai lebih kecil daripada elemen pivot, maka kedua elemen ini ditukar letaknya. Proses ini akan dilakukan terus hingga kedua arah gerakan bertemu di suatu titik tertentu. Lihat ilustrasi pada gambar 4. (Perhatikan bahwa letak urutan angka telah berubah). Selanjutnya letak elemen pivot ditukar dengan letak titik temu tersebut. Lihat ilustrasi pada gambar 5. Hasil akhir proses ini terlihat pada gambar 6.

Perhatikan baik-baik gambar 6 tersebut. Pada keadaan tersebut, seluruh bilangan yang terletak di sebelah kiri elemen pivot bernilai lebih kecil daripada elemen pivot dan seluruh bilangan yang terletak di sebelah kanan elemen pivot bernilai lebih besar daripada elemen pivot. Jumlah elemen yang terletak di sebelah kiri dan kanan tidak harus sama.

Sekalipun sekarang seluruh bilangan yang terletak di sebelah kiri elemen pivot bernilai lebih kecil daripada elemen pivot dan seluruh bilangan yang terletak di sebelah kanan elemen pivot bernilai lebih besar daripada elemen pivot, namun bilangan-bilangan tersebut masih belum berurutan. Jadi proses Quick Sort harus dilakukan lagi untuk bilangan yang terletak di sebelah kiri elemen pivot dan di sebelah kanan elemen pivot.

Proses ini akan terus berlangsung hingga seluruh bilangan akan terurut.

Nah, berdasarkan ilustrasi di atas, sekarang akan kita bangun program yang akan melakukan proses pengurutan dengan metoda Quick Sort. Program tersebut diberikan pada listing 1.

Contoh Program:

#include <stdio.h>

void quickSort( int[], int, int);

int partition( int[], int, int);

int total;

void main(){

int total;

int a[1000];

int i;

printf("masukkan jumlah data total = ");scanf("%d",&total);

for(i=0;i<total;i++){

printf("masukkan data index ke %d = ",i+1);scanf("%d",&a[i]);

}

printf("\n\nsebelum Di- sorting: ");

for(i = 0; i < total; ++i){

printf(" %d ", a[i]);

}

quickSort( a, 0, total-1);

printf("\n\nsesudah Di- sorting: ");

for(i = 0; i < total; ++i){

printf(" %d ", a[i]);

}

printf("\n");

}

void quickSort( int a[], int l, int r){

int j;

if( l < r ){

// divide and conquer

j = partition( a, l, r);

quickSort( a, l, j-1);

quickSort( a, j+1, r);

}

}

int partition( int a[], int l, int r) {

int pivot, i, j, t;

pivot = a[l];

i = l; j = r+1;

while( 1){

do ++i; while( a[i] <= pivot && i <= r );

do --j; while( a[j] > pivot );

if( i >= j ) break;

t = a[i]; a[i] = a[j]; a[j] = t;

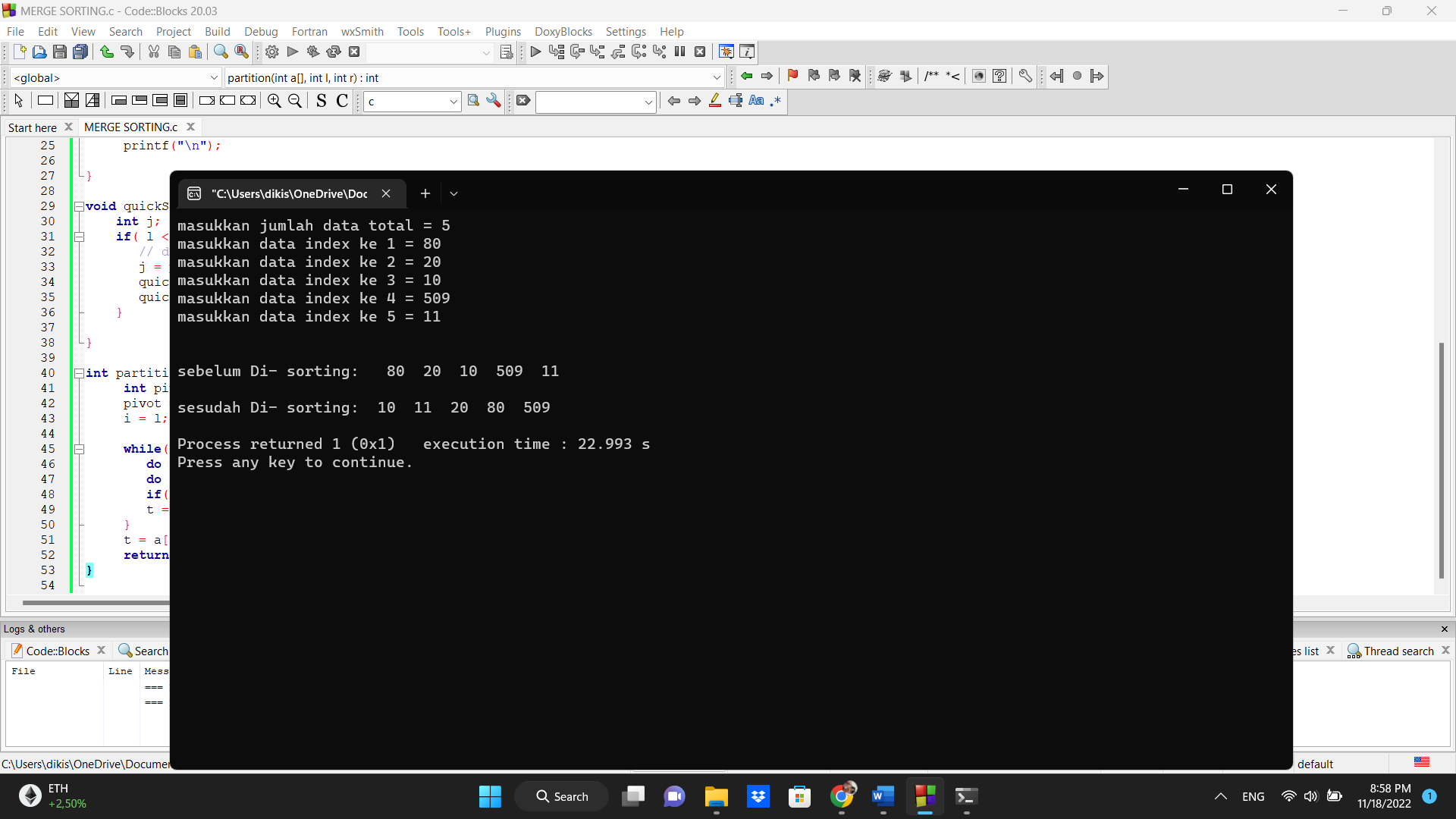
}

t = a[l]; a[l] = a[j]; a[j] = t;

return j;

}

Output :



**7. Counting sort**

Adalah sebuah algoritma sorting linear yang digunakan untuk mengurutkan ‘item’ ketika urutannya telah ditentukan dan memiliki panjang yang terbatas. Bilangan interval yang telah tetap, katakana k1 ke k2 adalah contoh dari ‘item’ tersebut. Counting sort sebenarnya merupakan metode pengurutan yang memanfaatkan index variabel array. Hanya effektif pada data yang nilainya kecil.

Algoritma ini diproses dengan mendefinisikan sebuah hubungan urutan antara ‘item’ yang akan disorting. Katakana ‘item’ yang akan disorting adalah variable A. Maka, terdapat sebuah array tambahan dengan ukuran yang serupa dengan array A. katakana array tersebut adalah array B. untuk setiap element di A, sebut e, algoritma ini menyimpan jumlah ‘item’ di A lebih kecil dari atau sama dengan e di B(e). jika hasil sorting yang terakhir disimpan di array C, maka untuk masing-masing e di A, dibuat dalam arah yang sebaliknya, yaitu C[B(e)]=e. setelah step di atas, niali dari B(e) berkurang dengan 1.

Algoritma ini membuat 2 passover A dan passover B. Jika ukuran dari range k lebih kecil dari ukuran input n, maka time complexity = O(n). perhatikan juga bahwa algoritma ini stabil yang berarti bahwa sambungan diselesaikan dengan langsung mengabarkan element-element yang muncul pertama kali.

Adapun syarat algoritma ini berjalan dengan baik ialah:

* Data harus bilangan bulat yang bernilai lebih besar atau sama dengan nol
* Range data diketahui

Ada 3 macam array yang terlibat:

* Array untuk mengisi bilangan yang belum diurutkan.
* Array untuk mengisi frekuensi bilangan itu, sekaligus sebagai penghitung kejadian.
* Array untuk mengisi bilangan yang sudah diurutkan.

Algoritmanya :

countingsort(A[], B[], min, max, n)

for i = min to max do

C[i] = 0

for j = 1 to n do

C[A[j]] = C[A[j]] + 1

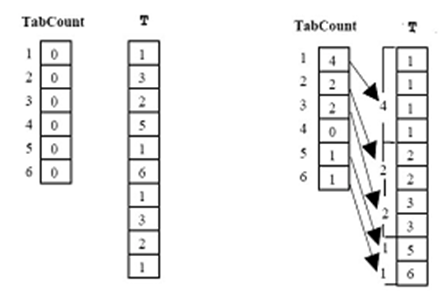
for i = min + 1 to max do

C[i] = C[i] + C[i-1]

for j = n downto 1 do

B[C[A[j]]] = A[j]

C[A[j]] = C[A[j]] – 1



BAB III

PENUTUP

**Kesimpulan**

Berdasarkan uraian bahasan “jenis-jenis algoritma searching dan sorting” dapat disimpulkan bahwa :

1. Terdapat banyak algoritma searching dan sorting yang bisa programmer gunakan sehingga dapat digunakan dalam situasi dan kondisi masalah tertentu

2. Pembuat algoritma baru dirasakan sangat penting karena setiap masalah akan memerlukan teknik sorting dan searching yang baik dan sesuai

**Saran**

Adapun saran-saran yang dapat penulis sampaikan dalam hal ini adalah sebagai berikut :

* Semoga pembaca dapat memahami dan mengerti tentang algoritma searching dan sorting yang dijelaskan
* Dengan adanya makalah ini, diharapkan pembaca mendapat informasi terbaru tentang algoritma-algoritma sorting dan searching.
* Semoga dengan membaca makalah ini, pembaca mendapat inspirasi-inspirasi terbaru yang berguna dalam pembuatan algoritma-algoritma lainnya
* Dalam kesempatan-kesempatan berikutnya sebaiknya ditambahkan kemungkinan algoritma-algoritma baru sehingga ke depannya akan didapt lebih banyak algoritma yang lebih efisien dan efektif

Demikianlah Makalah ini saya buat dengan sebaik-baiknya. Terima kasih buat Orang tua yang selalu mendukung saya baik materil maupun moril,dan terima kasih untuk abang dan kakak assisten laboratorium paket applikasi, juga terima kasih buat kawan-kawan yang telah membantu dalam mengerjakan makalah ini.semoga makalah ini dapat berguna buat kita semua dan buat angkatan ke depannya kelak.

lebih dan kurangnya saya mohon maaf, kepada abang dan kakak saya ucapkan terima kasih

Wassalammu’alaikum Wr.Wb.

**DAFTAR PUSTAKA**

1. <https://ikkholis27.wordpress.com/2013/02/01/sorting-dan-searching-array/>
2. <https://polidamar.wordpress.com/bahasa-c/metoda-pengurutan-quick-sort-dengan-bahasa-c/>
3. <https://fakultas-ti.blogspot.com/2016/11/metode-sorting-menggunakan-merge-sort.html>
4. <https://www.sobatambisius.com/2021/09/belajar-bahasa-c-12-sorting.html>
5. <https://www.belajarstatistik.com/blog/2022/03/22/contoh-program-pengurutan-dalam-bahasa-c/>
6. <https://www.mahirkoding.com/selection-sort-dalam-c/>
7. <https://www.materipelajaran.web.id/2017/05/macam-macam-sorting-pada-bahasa-c.html>
8. <https://www.kopicoding.com/bubble-sort-bahasa-c/>