Nama : Dzikri Algiffari  
NPM : 140810180053

Tugas 4

## **Studi Kasus 1: MERGE SORT**

Setelah Anda mengetahui Algoritma Merge-Sort mengadopsi paradigma divide & conquer, lakukan Hal berikut:

1. Buat program Merge-Sort dengan bahasa C++
2. Kompleksitas waktu algoritma merge sort adalah O(n lg n). Cari tahu kecepatan komputer Anda dalam memproses program. Hitung berapa running time yang dibutuhkan apabila input untuk merge sort-nya adalah 20?

**Merge Sort**

/\*

Nama Program : MergeSort

\*/

#include <iostream>

#include <chrono>

using namespace std;

void satu(int\* in, int p, int q,int r){

int n1 = q-p+1;

int n2 = r-q;

int L[n1+1];

int R[n2+1];

for (int i=1; i<=n1; i++){

L[i-1] = in[(p-1)+i-1];

}

for (int j=1; j<=n2; j++){

R[j-1] = in[(q-1)+j];

}

int i=0;

int j=0;

L[n1]=2147483647;

R[n2]=2147483647;

for (int k=(p-1); k<r; k++){

if(L[i]<=R[j]){

in[k]=L[i];

i = i+1;

}

else{

in[k]=R[j];

j = j+1;

}

}

}

void msort(int\* in, int p, int r){

int q;

if(p<r){

q = (p+r)/2;

msort(in, p, q);

msort(in, q+1, r);

satu(in, p, q, r);

}

}

void input(int\* a, int& n){

cout << "Input banyak data: "; cin >> n;

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

cout << "Input angka: "; cin >> a[i];

}

}

int main(){

int in[100];

int n;

input(in,n);

auto start = chrono::steady\_clock::now();

msort(in,1,n);

auto end = chrono::steady\_clock::now();

cout << "Hasil: ";

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

cout << in[i] << " ";

}

cout<<endl;

cout << "Elapsed time in nanoseconds : "

<< chrono::duration\_cast<chrono::nanoseconds>(end - start).count()

<< " ns" << endl;

return 0;

}

## **Studi Kasus 2: SELECTION SORT**

Selection sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma selection sort, lakukan langkah-langkah berikut:

* Pelajari cara kerja algoritma selection sort
* Tentukan T(n) dari rekurensi (pengulangan) selection sort berdasarkan penentuan rekurensi divide & conquer:

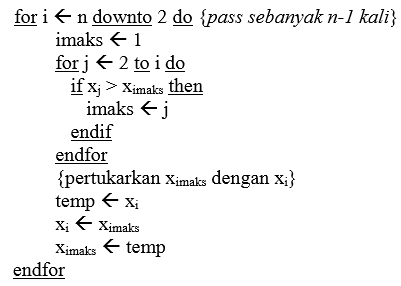
## (1) if ≤ + ( ) + ( ) otherwise

### ( ) =

* Selesaikan persamaan rekurensi T(n) dengan **metode recursion-tree** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ

Lakukan implementasi koding program untuk algoritma selection sort dengan menggunakan bahasa C+

**SelectionSort**



Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses pembagian = n

Waktu proses penggabungan = n



T(n) = cn + cn-c +cn-2c + ..... + 2c +cn

= c((n-1)(n-2)/2) + cn

= c((n2-3n+2)/2) + cn

= c(n2/2)-(3n/2)+1 + cn

=O(n2)

T(n) = cn + cn-c +cn-2c + ..... + 2c +cn

= c((n-1)(n-2)/2) + cn

= c((n2-3n+2)/2) + cn

= c(n2/2)-(3n/2)+1 + cn

= Ω (n2)

T(n) = cn2

= Θ(n2)

Source Code :

/\*

Program : Selection Sort

\*/

#include <iostream>

#include<conio.h>

using namespace std;

int data[50],data2[50];

int n;

void switch(int a, int b)

{

int t;

t = data[b];

data[b] = data[a];

data[a] = t;

}

void selectionSort()

{

int pos,i,j;

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

{

pos = i;

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

{

if(data[j] < data[pos]) pos = j;

}

if(pos != i) switch(pos,i);

}

}

int main()

{

cout<<"\nMasukkan Jumlah Data : ";cin>>n;

cout<< endl;

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

{

cout<<"Masukkan data ke-"<<i<<" : ";

cin>>data[i];

data2[i]=data[i];

}

selectionSort();

cout << "\n" << endl;

cout<<"Data Setelah di Sort : "<<endl;

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

{

cout<<" "<<data[i];

}

getch();

}

### **Studi Kasus 3: INSERTION SORT**

Insertion sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma insertion sort, lakukan langkah-langkah berikut:

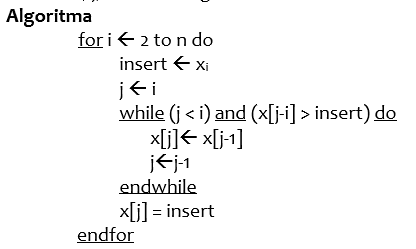
* Pelajari cara kerja algoritma insertion sort
* Tentukan T(n) dari rekurensi (pengulangan) insertion sort berdasarkan penentuan rekurensi divide & conquer:

## (1) if ≤ + ( ) + ( ) otherwise

### ( ) =

* Selesaikan persamaan rekurensi T(n) dengan **metode** **subtitusi** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
* Lakukan implementasi koding program untuk algoritma insertion sort dengan menggunakan bahasa C++

**Insertion Sort**



Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses penggabungan = n

Waktu proses pembagian = n

T(n) = cn + cn-c +cn-2c + ..... + 2c +cn <= 2cn2 + cn2

= c((n-1)(n-2)/2) + cn<= 2cn2 + cn2

= c((n2-3n+2)/2) + cn<= 2cn2 + cn2

= c(n2/2)-c(3n/2)+c+cn <= 2cn2 + cn2

=O(n2)

T(n) = cn <= cn

= Ω (n)

T(n) = (cn + cn2)/n

= Θ(n)

Source Code :

/\*

Program : Insertion sort

\*/

#include <iostream>

#include <conio.h>

using namespace std;

int data[100],data2[100],n;

void insertion\_sort()

{

int temp,i,j;

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

temp = data[i];

j = i -1;

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

data[j+1] = data[j];

j--;

}

data[j+1] = temp;

}

}

int main()

{

cout<<"Masukkan Jumlah Data : "; cin>>n;

cout<<endl;

cout << endl;

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

{

cout<<"Masukkan data ke-"<<i<<" : ";

cin>>data[i];

data2[i]=data[i];

}

cout << endl;

insertion\_sort();

cout<<"\nData Setelah di Sort : "<<endl;

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

{

cout<<data[i]<<" ";

}

}

### **Studi Kasus 4: BUBBLE SORT**

Bubble sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma bubble sort, lakukan langkah-langkah berikut:

* Pelajari cara kerja algoritma bubble sort
* Tentukan T(n) dari rekurensi (pengulangan) insertion sort berdasarkan penentuan rekurensi divide & conquer:

## (1) if ≤ + ( ) + ( ) otherwise

### ( ) =

* Selesaikan persamaan rekurensi T(n) dengan **metode** **master** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
* Lakukan implementasi koding program untuk algoritma bubble sort dengan menggunakan bahasa C++

**Bubble Sort**

Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses pembagian = n

Waktu proses penggabungan = n

T(n) = cn + cn-c +cn-2c + ..... + 2c +c <= 2cn2 + cn2

= c((n-1)(n-2)/2) + c<= 2cn2 + cn2

= c((n2-3n+2)/2) + c<= 2cn2 + cn2

= c(n2/2)-c(3n/2)+2c <= 2cn2 + cn2

=O(n2)

T(n) = cn + cn-c +cn-2c + ..... + 2c +c <= 2cn2 + cn2

= c((n-1)(n-2)/2) + c<= 2cn2 + cn2

= c((n2-3n+2)/2) + c<= 2cn2 + cn2

= c(n2/2)-c(3n/2)+2c <= 2cn2 + cn2

= Ω (n2)

T(n) = cn2 + cn2

= Θ(n2)

Source Code :

/\*

Program : Bubble Sort

\*/

#include <iostream>

#include <conio.h>

using namespace std;

int main(){

int arr[100],n,temp;

cout << "\n================================================"<<endl;

cout<<"Massukan banyak elemen yang akan diinputkan : ";cin>>n;

cout << "\n------------------------------------------------" << endl;

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

cout<<"Masukkan Elemen ke-"<<i+1<<" : ";cin>>arr[i];

}

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

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

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

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

cout << "------------------------------------------------" << endl;

cout<<"\nHasil dari Bubble Sort : "<<endl;

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

cout<<" "<<arr[i];

}

cout << "\n================================================"<<endl;

}



T(n) = cn + cn-c +cn-2c + ..... + 2c +cn

= c((n-1)(n-2)/2) + cn

= c((n2-3n+2)/2) + cn

= c(n2/2)-(3n/2)+1 + cn

=O(n2)

T(n) = cn + cn-c +cn-2c + ..... + 2c +cn

= c((n-1)(n-2)/2) + cn

= c((n2-3n+2)/2) + cn

= c(n2/2)-(3n/2)+1 + cn

= Ω (n2)

T(n) = cn2

= Θ(n2)