Nama : Alfari Sidnan Ghilmana

NPM : 140810180011

Kelas : A

Studi Kasus 1: MERGE SORT

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

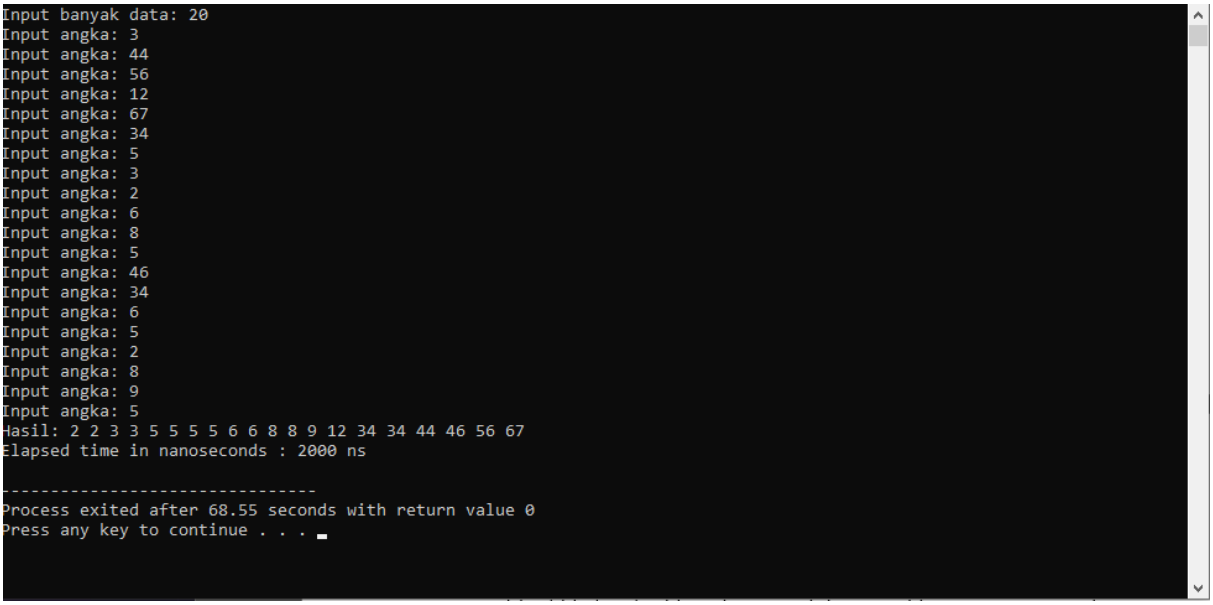
1. Buat program Merge-Sort dengan bahasa C++
2. Kompleksitas waktu algoritma merge sort adalah O(n lg n). Cari tahu kecepatan komputer Anda dalam memproses program. Hitung berapa running time yang dibutuhkan apabila input untuk merge sort-nya adalah 20?

Kasus 1

```
/*
Nama Program        : Merge Sort
Nama                : Alfari Sidnan Ghilmana
NPM                 : 140810180011
Tanggal Pembuatan   : 23 Maret 2020
**********************************************************************************
***********
*/

#include <iostream>

using namespace std;

// Fungsi untuk melakukan penggabungan kedua bagian menjadi data yang terurut.
void Merge(int *a, int low, int high, int mid)
{
    // Menyortir rendah ke sedang dan menengah +1 ke tinggi.
    int i, j, k, temp[high-low+1];
    i = low;
    k = 0;
    j = mid + 1;

    // Gabungkan kedua bagian menjadi temp [].
    while (i <= mid && j <= high)
    {
        if (a[i] < a[j])
        {
            temp[k] = a[i];
            k++;
            i++;
        }
```

```
        else
        {
            temp[k] = a[j];
            k++;
            j++;
        }
    }

    // Masukkan semua nilai yang tersisa dari i hingga pertengahan ke temp [].
    while (i <= mid)
    {
        temp[k] = a[i];
        k++;
        i++;
    }

    // Masukkan semua nilai yang tersisa dari j ke tinggi ke temp [].
    while (j <= high)
    {
        temp[k] = a[j];
        k++;
        j++;
    }


    // Tetapkan data yang diurutkan disimpan dalam temp [] ke a [].
    for (i = low; i <= high; i++)
    {
        a[i] = temp[i-low];
    }
}

// Suatu fungsi untuk membagi array menjadi dua bagian.
void MergeSort(int *a, int low, int high)
{
    int mid;
    if (low < high)
    {
        mid=(low+high)/2;
        // Membagi data menjadi dua bagian
        MergeSort(a, low, mid);
        MergeSort(a, mid+1, high);

        // Gabungkan mereka untuk mendapatkan hasil yang diurutkan.
        Merge(a, low, high, mid);
```

```
    }
}

int main()
{
    int n, i;
    cout<<"\nMasukkan jumlah data yang akan di urutkan: ";
    cin>>n;

    int arr[n];
    for(i = 0; i < n; i++)
    {
        cout<<"Masukkan elemen ke-"<<i+1<<": ";
        cin>>arr[i];
    }

    MergeSort(arr, 0, n-1);

    // Mencetak data terurut.
    cout<<"\nData Terurut: ";
    for (i = 0; i < n; i++)
        cout<<"->"<<arr[i];

    return 0;
}
```



Kompleksitas 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?

Untuk di program hasilnya : 2 microseconds

Tapi jika sesuai dengan O -> T(20 $\log_{10}20$)= 26

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:

$$T(n) = \begin{cases} \theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- 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++

Kasus 2 – Selection Sort

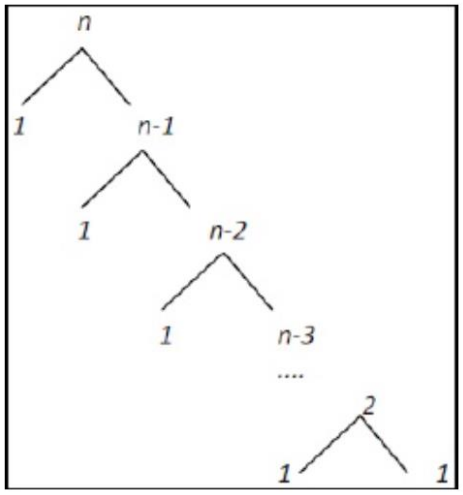
```
for i ← n downto 2 do {pass sebanyak n-1 kali}
    imaks ← 1
    for j ← 2 to i do
      if x_j > x_imaks then
        imaks ← j
      endif
    endfor
    {pertukarkan x_imaks dengan x_i}
    temp ← x_i
    x_i ← x_imaks
    x_imaks ← temp
endfor
```

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((n^2-3n+2)/2) + cn

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

=O(n^2)

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

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

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

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

= Ω (n^2)

T(n) = cn^2

= Θ(n^2)

```
/*
Nama Program        : Selection sort
Nama                : Alfari Sidnan Ghilmana
NPM                 : 140810180011
Tanggal Pembuatan   : 23 Maret 2020
*********************************************************************************
***********
*/
```

```
#include <iostream>

using namespace std;

int data[10],data2[10];
int n;

void tukar(int a, int b)
{
 int t;
 t = data[b];
 data[b] = data[a];
 data[a] = t;
}
void selection_sort()
{
 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) tukar(pos,i);
 }
}

main()
{
 cout<<"Masukkan Jumlah Data : ";
 cin>>n;
 for(int i=1;i<=n;i++)
 {
  cout<<"Masukkan data ke "<<i<<" : ";
  cin>>data[i];
  data2[i]=data[i];
 }

 selection_sort();
 cout<<"Data Setelah di Sort : ";
 for(int i=1; i<=n; i++)
 {
  cout<<" "<<data[i];
 }
```

```
 cout<<"\n\nSorting dengan selection sort selesai";
}
```

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:

$$T(n) = \begin{cases} \theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- 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++

Algoritma

```
for i ← 2 to n do
    insert ← xi
    j ← i
    while (j < i) and (x[j-i] > insert) do
        x[j]← x[j-1]
        j←j-1
    endwhile
    x[j] = insert
endfor
```

Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses penggabungan = n

Waktu proses pembagian = n

T(n)={Θ(1) T(n-1)+Θ(n)

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

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

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

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

=O(n^2)

T(n) = cn <= cn

= Ω (n)

T(n) = (cn + cn^2)/n

= Θ(n)

```
/*
Nama Program        : Insertion sort
Nama                : Alfari Sidnan Ghilmana
NPM                 : 140810180011
Tanggal Pembuatan   : 23 Maret 2020
**********************************************************************************
***********
*/

#include <iostream>
using namespace std;

int data[10],data2[10];
int n;

void tukar(int a, int b)
{
 int t;
 t = data[b];
 data[b] = data[a];
 data[a] = t;
}

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;
 }
}

main()
{
 cout<<"Masukkan Jumlah Data : ";
 cin>>n;
 for(int i=1;i<=n;i++)
 {
  cout<<"Masukkan data ke "<<i<<" : ";
  cin>>data[i];
  data2[i]=data[i];
 }

 insertion_sort();
 cout<<"Data Setelah di Sort : ";
 for(int i=1; i<=n; i++)
 {
  cout<<" "<<data[i];
 }
 cout<<"\n\nSorting dengan insertion sort selesai";
}
```

Kasus 4 Bubble Sort:

Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses pembagian = n

Waktu proses penggabungan = n

T(n)={Θ(1) T(n-1)+Θ(n)

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

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

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

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

=O(n^2)

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

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

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

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

= Ω (n^2)

T(n) = cn^2 + cn^2

= Θ(n^2)

```
/*
Nama Program        : Bubble Sort
Nama                : Alfari Sidnan Ghilmana
NPM                 : 140810180011
Tanggal Pembuatan   : 23 Maret 2020
**********************************************************************
**********
*/

#include<iostream>
using namespace std;

main()
{
    int n, i, arr[50], j, temp;
    cout<<"Masukkan total elemen yang akan diurutkan: ";
    cin>>n;
    cout<<"Masukan "<<n<<" angka:\n";
    for(i=0; i<n; i++){
        cout<<"Masukan angka ke-"<<i+1<<": ";
        cin>>arr[i];
    }

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

```
            if(arr[j]>arr[j+1])
            {
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
        }
    }

    cout<<"Data terurut dari hasil Bubble Sort::\n";
    for(i=0; i<n; i++){
        cout<<arr[i]<<" ";
    }
}
```